財 団 法 人 Ｊ Ｋ Ａ
平成 21 年度デジタルコンテンツの保護・活用に関する調査研究等補助事業

デジタルコンテンツ制作の
先端技術応用に関する調査研究

報　告　書

平成 22 年 3 月

財団法人 デジタルコンテンツ協会

KEIRIN

この事業は、競輪の補助金を受けて実施したものです。

URL : http://ringring-keirin.jp/

目　　次

1. はじめに（DCAJ）.....1
1.1 調査研究の目的（DCAJ）.....1
1.2 本年度の活動（DCAJ）.....1
1.2.1 第 1 回委員会.....1
1.2.2 第 2 回委員会.....2
1.2.3 第 3 回委員会.....2
1.2.4 第 4 回委員会.....2
1.2.5 第 5 回委員会.....2
1.2.6 第 6 回委員会.....3
1.2.7 第 7 回委員会.....3
1.2.7 セミナー「わが国ゲーム産業の未来を切り拓く　―最新技術動向と産業分析の報告―」.....3
1.3 推進体制（DCAJ）.....4
2. ゲーム開発技術ロードマップ.....5
2.1 はじめに.....5
2.2 シェーダの発展.....8
2.2.1 はじめに.....8
2.2.2 BRDF に関する進化.....10
2.2.3 光源に関する進化.....12
2.2.4 遮蔽に関する進化.....14
2.2.5 集光手法.....16
2.2.6 照明モデル.....17
2.2.7 その他のシェーディング.....18
2.2.8 この先のシェーダについて.....21
2.3 ビデオゲームにおける物理シミュレーションの歴史について.....24
2.3.1 はじめに.....24
2.4 ミドルウェア（佐藤委員）.....32
2.4.1 ミドルウェアとは.....32
2.4.2 ミドルウェアの役割.....32
2.4.3 ミドルウェアの種別.....34
2.4.4 ミドルウェア・カタログ.....36
2.5 シナリオ.....39
2.5.1 ゲームにおけるシナリオの意味について.....39
2.5.2 ゲームシナリオの現場とその変遷.....42
2.5.3 ゲームシナリオのスクリプト.....48
2.5.4 これからのゲームシナリオ.....51
2.6 ユーザーインターフェイス（ソフトウェア）.....53
2.6.1 本節について.....53

2.6.2　ゲーム UI の特性 53
2.6.3　入力デバイスの種類 53
2.6.4　UI から見たゲームの歴史 53
2.6.5　ゲーム UI の未来 78
2.6　周辺機器メーカーから見たハードウェアインターフェースの変遷 82
2.6.1　ファミリーコンピュータ時代 83
2.6.2　スーパーファミコン時代 86
2.6.3　アナログ機能時代 87
2.6.4　無線式・傾き検知機能時代 89
2.6.5　製造コスト 90
2.6.6　製造工程 92
2.6.7　社内教育 93
2.6.8　今後の変化 94
2.6.9　ユーザー層の変化 96
2.7　開発工程管理 98
2.7.1　開発工程管理とは 98
2.7.2　ビルドツール 103
2.7.3　バージョン管理システム 105
2.7.4　バグトラッキングシステム 109
2.7.5　継続的インテグレーションツール 111
2.7.6　軽量な開発プロセス 113
2.7.7　まとめ 115
2.8　ゲームローカライズの歴史とこれから 121
2.8.1　はじめに 121
2.8.2　ローカライズの各作業内容 121
2.8.3　ローカライズ作業の変遷 126
2.8.4　未来 130
2.8.5　まとめ 132
2.9　アセンブラ 133
2.9.1　プラットフォームの設計思想の違い。 133
2.9.2　PC ゲーム 136
2.9.3　家庭用ゲーム 140
2.9.4　今後の課題 143
2.10　日本のゲーム開発技術の全体像とこれからの課題 146
2.10.1　概論 146
2.10.2　ゲーム開発技術の発展の歴史 148
2.10.3　データ、プログラム、スクリプトの形成 153
2.10.4　技術導入・流入経路 155
2.10.5　仮想化の歴史 161
2.10.6　固定コンテンツの展開とプロシージャル 165

2.10.7　技術からマネージメントへ 167
2.10.8　ゲームデザインと技術 170
2.10.9　これからのゲーム開発技術の発展のために 173
2.10.10　ゲーム開発技術ロードマップ調査の終わりに 178
3.　様々な切り口から見たゲーム開発 180
3.1　技術革新と家庭用ゲーム産業 180
3.1.1　家庭用ゲーム機の地位の変化と、技術的な収斂 180
3.1.2　据置ハード価格の上昇と端末普及重視型ビジネスモデルの危機 183
3.1.3　ソフトウェア開発費の爆発 185
3.1.4　家庭用ゲーム機の存続の危機？ 187
3.1.5　ゲームのためのイノベーション？ 188
3.1.6　まとめに代えて：選択と集中 191
3.2　ゲームと価格のイノベーション 193
3.2.1　1983 年にファミリーコンピュータが作ったモデル 193
3.2.2　ファミコン時代から 1250 万倍にまで進んだ技術革新 193
3.2.3　過剰なイノベーションが準備したソーシャルゲームブーム 194
3.2.4　アイテム課金モデルのルーツは韓国から 195
3.2.5　家庭用ゲーム機のコストでは開発できないソーシャルゲーム 196
3.2.6　日本でも登場してきたソーシャルゲーム 196
3.2.7　前払いモデルから後払いモデルへとユーザーが移行 197
3.2.8　現在の変化の予兆は 5 年あまり前から存在していた 197
3.2.9　ゲームは無形の財としての性質が強くなる 198
3.3　ゲーム開発者教育 199
3.3.1　概要 199
3.3.2　大学におけるゲーム開発者教育の課題 199
3.3.3　教育の実例 202
3.3.4 展望 207
3.4　集合的なユーザーの振舞いを解釈するための手法について（―ユーザーの偏り（文化差）を考える―） 209
3.4.1　ユーザー間ネットワークの自律性 209
3.4.2　ユーザー／開発／マーケティングの相互作用 215
3.4.3　想定ターゲットユーザーの反応を、どのように推定するか 217
3.4.4　むすび 222
3.5　現場から見たゲーム開発の変遷 224
3.5.1　アセンブラの時代 224
3.5.2　プラットフォームの相違 224
3.5.3　アセンブラ時代のプログラマー雇用 226
3.5.4　2D ゲームを作る技法 226
3.5.5　C 言語の時代 229
3.5.6　C 言語普及によるプログラマーの変化 231

3.5.7　3D 表示に伴なう追加要素 231
3.5.8　現在のプラットフォーム 234
3.5.9　ゲームとは何か？ 234
3.5.10　開発者として 235
4.　ゲーム開発者に対するヒアリング調査 236
4.1　近藤敏信氏ヒアリング 236
5.　ゲーム開発者の就労意識とキャリア形成の課題 253
5.1　はじめに―問題意識と目的 253
5.2　調査の方法 254
5.2.1　調査対象 254
5.2.2　調査形式 254
5.2.3　調査実施期間 255
5.2.4　有効回答数 255
5.3　回答者・回答者の所属企業の概要 255
5.3.1　回答者の属性 255
5.3.2　回答者の所属企業の概要 259
5.4　開発者の勤続・給与 263
5.4.1　開発者の勤続年数 263
5.4.2　開発者の給与 264
5.5　開発者の求職・転職行動 268
5.5.1　開発者の求職行動 268
5.5.2　開発者の転職行動 273
5.6　開発者の働き方 277
5.6.1　勤務形態 277
5.6.2　労働時間 278
5.6.3　繁忙期 280
5.7　開発者の育成 283
5.7.1　人材育成の実施状況 283
5.7.2　Off-JT の実施状況 284
5.7.3　キャリア開発支援の実施状況 286
5.8　開発者に対する評価・処遇 287
5.8.1　開発者に対する評価 287
5.8.2　開発者に対する処遇 288
5.9　開発者の満足度 289
5.9.1　仕事の満足度 289
5.9.2　ストレス要因 291
5.9.3　ゲーム産業でのキャリアチェンジ 292
5.9.4　ゲーム産業で働くプラン 293
5.10　おわりに―開発者の声･･･離脱・発言？ 294
5.10.1　要約と結論 297

5.10.2　残された課題 299
6.　海外におけるゲーム関連技術の現状調査 301
6.1　GDC2010 における海外産業動向　～ソーシャルゲームについての側面を中心に～ 301
6.1.1　はじめに 301
6.1.2　ハイペースで 2 億人のゲームユーザーを集めた Facebook 301
6.1.3　デバイスを問わないサービスへと広がる Facebook 302
6.1.4　機能拡張によって競合関係化が進む 303
6.1.5　2007 年に考えられたソーシャルゲームに必要な要素 304
6.1.6　スーパーマーケット登場の歴史とソーシャルゲームの発展は似ている 304
6.1.7　家庭用ゲーム機が抱えていた「不便」を解消するソーシャルゲーム 305
6.1.8　ソーシャルゲームでもフランチャイズタイトルが鍵に 306
6.1.9　ソーシャルゲームでは存在感の薄い日本 307
6.1.10　終わりに 307
6.2　GDC2010 における海外のゲーム関連技術についての調査 308
6.2.1　概論 308
6.2.2　Killzone 2 の AI 技術 308
6.2.3　The Sims の AI の技術 326
6.2.4　FPS(戦闘) ⇔ 日常系の AI の比較 338
6.2.5　AI における階層構造 341
6.2.6　00 年代から 10 年代のゲーム AI へ 343
7.　日本と世界のゲーム産業支援の現状と課題 349
7.1　調査委員会の成果 349
7.1.1　2007 年度・2008 年度を振り返って 349
7.1.2　2009 年度の調査を振り返って 350
7.2　世界のゲーム産業振興政策 351
7.2.1　ゲーム産業力の公表 351
7.2.2　カナダのゲーム産業振興政策 352
7.2.3　イギリスのゲーム産業振興政策 353
7.2.4 韓国のゲーム産業振興政策 354
7.2.5　ドバイのゲーム産業振興政策 356
7.3　日本の新ゲーム産業戦略の策定に向けて 357
7.3.1　ゲーム産業戦略（2006 年）の検証の必要性 357
7.3.2　ゲーム産業の経済波及効果への期待 359
7.3.3 新ゲーム産業戦略の視点 361
附録：「ゲーム開発者の実態調査」単純集計 364

1. はじめに（DCAJ）

1.1 調査研究の目的（DCAJ）

映像コンテンツの高精細化に伴い、特にインタラクティブなコンテンツにおいては、より高度な表現を求めて、物理シミュレーションや AI 処理などの先端技術の活用が高まっている。そこで先端技術のインタラクティブ映像制作への応用を図るため、内外の関係情報を調査すると共に、新世代のデジタル制作技術基盤の方向性を追求し、核となる緊要な先端技術についての活用方策の提言を行い映像産業の振興に寄与することを目的とした調査を実施した。

これまでの調査により、我が国での映像産業の技術分野の牽引役は、ゲーム産業が担っていると考えられる。そこで、21 世紀のデジタルコンテンツを念頭におき、我が国のゲーム産業を中心としたインタラクティブなコンテンツに関連する研究の、映像制作への応用を図るため、ゲームプラットフォームやゲームを開発するための技術の変遷や教育について、内外の関連する動向・事例を調査し、その方策を検討した。

1.2 本年度の活動（DCAJ）

具体的な活動状況として、ゲームを中心としたデジタルコンテンツ制作の先端技術動向及びその適用における諸問題に関する内外の情報を収集し、学識経験者及び産業界の専門家により構成される研究委員会を計 7 回開催し、調査研究を行った。

海外の技術動向調査としては、米国におけるゲーム産業技術動向の実態について、2010 年 3 月 9 日～13 日に米国サンフランシスコにて開催される世界最大のゲーム開発者向けカンファレンス GDC10 に参加し、調査を実施した。

また、「わが国ゲーム産業の未来を切り拓く　―最新技術動向と産業分析の報告―」と題し、「デジタルコンテンツ制作の先端技術応用に関する調査研究委員会」で調査を行った「ゲーム開発者の実態調査」の分析や、日本のゲーム開発技術基盤の調査結果の報告、そうした技術革新が家庭用ゲーム産業にどのような影響を与えるかについて講師の方より紹介いただき、結果としてゲーム産業の今後の展望がどのようなものであるのかパネルディスカッションにて討論をおこなった。

1.2.1 第 1 回委員会

日時：平成 21 年 7 月 23 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 B
主な議題

- ゲーム技術の他産業への応用について
- 昨年度の報告書、および今年度追加する内容について
- 開発者教育について
- アーケードゲームについて

1.2.2 第 2 回委員会

日時：平成 21 年 8 月 20 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 A
主な議題

- 開発者教育について

1.2.3 第 3 回委員会

日時：平成 21 年 10 月 1 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 A
主な議題

- 「ゲーム開発者の実態調査」について
- ゲーム開発ロードマップへの追加項目案
- ゲーム開発の歴史の総合的な切り口について
- ヒアリング対象者について

1.2.4 第 4 回委員会

日時：平成 21 年 10 月 29 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 A
主な議題

- 「ゲーム開発者の実態調査」について
- 報告書執筆内容検討

1.2.5 第 5 回委員会

日時：平成 21 年 12 月 3 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 A
主な議題

- ゲーム開発者の実態調査アンケート進捗報告
- 報告書目次案検討

1.2.6　第 6 回委員会

日時：平成 22 年 1 月 21 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 A
主な議題

- 報告書進捗確認
- セミナーについて

1.2.7　第 7 回委員会

日時：平成 22 年 3 月 4 日（木）18：00～20：00
場所：財団法人デジタルコンテンツ協会　会議室 A
主な議題

- 報告書進捗確認
- セミナーについて

1.2.7　セミナー「わが国ゲーム産業の未来を切り拓く　―最新技術動向と産業分析の報告―」

主催：財団法人デジタルコンテンツ協会（JKA 補助事業）
開催日時：平成 21 年 3 月 19 日（金）　18：00～20：45
会場：スター貸会議室　四谷第 2　東京都新宿区四谷 1-8-6 ホリナカビル 302 号室
参加者数：40 名（申込者 65 名）
スケジュール：

17:30～	開場
18:00～18:30 18:30～19:00 19:00～19:30	司　会：馬場章氏（東京大学大学院　教授） １．ゲーム開発者の就労意識とキャリア形成の課題 講　師：藤原正仁氏（東京大学大学院情報学環） 2. 技術革新と家庭用ゲーム産業 講　師：小山友介氏（芝浦工業大学　システム理工学部　准教授） 3. ゲーム開発技術の歴史と展望 講　師：三宅陽一郎氏（フロム・ソフトウェア　技術部）
19:30～20:45	パネルディスカッション：ゲーム産業の現状分析と今後の展望 モデレーター：馬場章氏（東京大学大学院　教授） パネリスト：小山友介氏（芝浦工業大学　システム理工学部　准教授） パネリスト：三宅陽一郎氏（フロム・ソフトウェア　技術部） パネリスト：藤原正仁氏（東京大学大学院情報学環）

1.3　推進体制（DCAJ）

本研究委員会は、(財) デジタルコンテンツ協会 (DCAj) における事業開発事業として、事業開発政策委員会のもと推進体制を組んでいる。

委員会メンバーは下記の通りで、東京大学大学院　情報学環　馬場章教授の下、推進する体制とした。

事務局は、DCAj 事業開発本部先導的事業推進部がこれを担当する。

表 1.3-01　平成 21 年度デジタルコンテンツ制作の先端技術応用に関する調査研究委員会名簿

	氏名	社名・団体名	所属	役職
委員長	馬場　章	東京大学大学院	情報学環	教授
委員	新　清士	IGDA 日本		代表
	三上　浩司	東京工科大学	片柳研究所　クリエイティブラボ	プロデューサー
	藤原　正仁	東京大学大学院	情報学環	
	斎藤　直宏	株式会社バンダイナムコゲームス	コンテンツ制作本部　制作統括ディビジョン　技術部	ゼネラルマネージャー
	三宅　陽一郎	株式会社フロム・ソフトウェア	技術部	
	細江　慎治	株式会社スーパースィープ		代表取締役
	井上　明人	国際大学 GLOCOM		研究員
	長久　勝	ハイパーコンテンツ株式会社	開発部	マネージャー
	小久保　啓三	専門学校 HAL 東京		講師
	佐藤カフジ			フリーライター
	田村　祐樹	株式会社ネバーランドカンパニー		
	小山　友介	芝浦工業大学	システム理工学部	准教授
事務局	須藤　智明	財団法人デジタルコンテンツ協会	事業開発本部	課長代理
オブザーバー	松井　悠	株式会社グルーブシンク		代表取締役

2. ゲーム開発技術ロードマップ

2.1 はじめに

三宅　陽一郎（株式会社フロム・ソフトウェア）

ゲーム開発技術ロードマップは、各技術分野のエキスパートがその分野の歴史を記述したレポートを編纂したものである。ゲーム開発技術史は、ゲームタイトルの歴史や、デザインの歴史と異なり、これまで編纂されることや文献化されることが極めて稀な分野であり、昨年の報告書と合わせて計 21 分野に渡る開発技術の歴史の本編纂は初めての大規模な調査の試みである[報告書 2008]。各分野の報告は実際の開発・研究の経験を踏まえて丁寧に記述されており、用意された年表と合わせて、これから長くゲーム産業における貴重な資料として活用されるべきものである。しかし、あらゆる他の歴史分野の研究同様、本調査は過去 30 年の長期に渡る歴史を対象としているため、情報・裏付・数値的統計の検証の隙間が存在し、幾らか不完全さを含むものである。しかし、ゲーム開発技術の歴史の編纂は、現段階では、完全さを期して為せる分野からは遠く、これからも継続して調査され精度を上げ複数の裏づけを取りながら改善が繰り返されるべき対象であって、まず、最初の大きな青写真としてここに提示するものである。

現在という一瞬でさえ、ゲーム開発技術者が、世界の技術趨勢の中で、自分たちが何処に立っているかを明確に認識するためには、毎日のようにアップデートされて行く技術情報を敏感に集めておかなければならない。ある時はゆっくりと、ある時は急激に進化する技術の発展の最先端を走り続けることは技術者に必要な姿勢の一つでもある。しかし、同時に技術者は自分たちでその分野の未来を切り拓いて行く存在でもある。そして、現在のゲーム開発技術者が自分たちの仕事を為そうとするときに、一体、自分の仕事が技術の歴史の中でどこに位置づけられるか、これからどういう価値があるかを推測するためには、その分野の大きな歴史的流れに対する感覚が必要である。本ロードマップは、開発者にそういった技術の歴史的感覚を提供し､未来の仕事の役立てて貰う目的を持つ調査でもある。

また、毎年新しくゲーム産業に入って来る新人が、ゲーム産業における自分の立ち位置を知るには 2、3 年かかるであろう。そして、さらに、ゲーム産業の技術史の中で、どういう流れの中にあるかを知ろうとする時には、幸運にも周囲にその道のベテランがいる場合を除けば、全くそういった情報を知る術がないというのが現状であり、本調査は、そういったゲーム産業における活躍を新しく目指される方に役立てて貰いたいと考えている。

現在、ゲームデザインで世界を席巻できる実力がありながら、日本のゲーム開発技術の存在感は、残念ながら非常に小さい。そこで、我々が日本のゲーム開発技術の足元を見直し、根源から新しい流れを導こうとするとき、本調査が持つ歴史的内容が必要になることであろう。過去と現在、そして未来をつなぐ基礎となる足場としてご活用頂ければ幸いで

ある。

表 2.1-01　ゲーム開発技術ロードマップ執筆者一覧(2008 年度報告書) [報告書 2008]

章	開発技術分野	執筆者（敬称略）
3.2.1	ゲームデザイン	井上明人（国際大学 GLOCOM 研究員）
3.2.2	3D CG 製作	川島 基展（東京工科大学 片柳研究所　クリエイティブ・ラボ）
3.2.3	サウンド製作	細江　慎治（株式会社スーパースィープ）
3.2.4	アニメーション製作	金久保　哲也（株式会社バンダイナムコゲームス）
3.2.5	プログラミング AI	三宅　陽一郎（株式会社フロム・ソフトウェア）
3.2.6	プログラミング グラフィックス描画	宮澤 篤・大久保 明（株式会社バンダイナムコゲームス）
3.2.7	プログラミング 物理・衝突判定	長谷川　晶一（電気通信大学）
3.2.8	タスクシステム技術	田村　祐樹（株式会社ネバーランドカンパニー） 大野　功二（オープランニング） 黒須一雄（株式会社モバイル＆ゲームスタジオ） ※第 4 章に黒須氏へのインタビューを掲載
3.2.9	プログラミング スクリプト	小久保　啓三（専門学校 HAL 東京）
3.2.10	ネットワーク通信	佐藤カフジ（フリーライター）
3.2.11	開発方法論	長久　勝（ハイパーコンテンツ(株)/早稲田大学メディアネットワークセンタ）

表 2.1-02　ゲーム開発技術ロードマップ執筆者一覧(本報告書)

章	開発技術分野	執筆者（敬称略）
2.2	シェーダ	今給黎　隆（株式会社バンダイナムコゲームス）
2.3	物理シミュレーション	今給黎　隆（株式会社バンダイナムコゲームス）
2.4	ミドルウェア	佐藤カフジ（フリーライター）
2.5	シナリオ	すずき　こういち（コンテンツデザイナー）
2.6	ユーザーインターフェイス（ソフトウェア）	サイトウ・アキヒロ（立命館大学映像学部教授） 小野憲史（ゲームジャーナリスト）
2.6	ユーザーインターフェイス（ハードウェア）	インタビュー：大神佳人（株式会社 HORI 営業部） （聞き手：三宅委員・オブザーバー松井悠）
2.7	開発工程管理	田村　祐樹（株式会社ネバーランドカンパニー）
2.8	ゲームローカライズ	長谷川　亮一（株式会社セガ）
2.9	アセンブラ	山口　誠（株式会社グッド・フィール）
2.10	日本のゲーム開発技術の全体像とこれからの課題	三宅　陽一郎（株式会社フロム・ソフトウェア）

参考文献：

[報告書 2006]「デジタルコンテンツの次世代基盤技術に関する調査研究報告書」(2006年度), 2007　http://www.dcaj.org/report/2006/data/dc07_07.pdf

[報告書 2007]「デジタルコンテンツ制作の先端技術応用に関する調査研究報告書」(2007年度), 2008　http://www.dcaj.org/report/2007/data/dc08_07.pdf

[報告書 2008]「デジタルコンテンツ制作の先端技術応用に関する調査研究報告書」(2008年度),　2009　http://www.dcaj.org/report/2008/data/dc_08_03.pdf

2.2　シェーダの発展

今給黎　隆（株式会社バンダイナムコゲームス）

2.2.1　はじめに

シェーダという言葉がコンシューマーの世界で語られるようになってきたのは、ここ 10 年ぐらいの話である。シェーダが流行ったのは、GPU の年々の進化がゲームで表現できる幅を広げ、見た目にも性能の向上がわかるように進歩してきたからであろう。本稿では、その変化の歩みを一端でも振り返ればと思う。

(1)　シェーダの歴史

シェーダ(Shader)という言葉は、シェーディング(Shading)をするモノという意味がこめられている。シェーディングというのは、日本語では「陰影付け」という意味で、光の当たり具合で陰となる部分ができ、立体感を感じるための要因である。シェーダが注目されているのは、開発者の自由に陰影を付け、他との差別化が図れるようになってきたからです。近年では、この自由な陰影を計算するための仕組みであるプログラマブルシェーダを使った効果をシェーダという者も多い。

陰影付けが自由に行えるというのは最近の考えではない。1984 年に Cook は、”Shade trees”という論文で、陰影付けをする要素を組み合わせることで、幅広い表現が可能になることを発表している[Cook84]。また、実際にシェーダを使ってレンダリングするための仕組みである”Reyes Architecture”は、1980 年代前半から映画制作に使用されている。特に Reyes が使われている PhotoRealistic RenderMan は、Pixar の映画の制作で使われており[Cook87]、CG 映画の世界ではシェーダという言葉は古くから知られていた。

ゲームの世界にシェーダが登場してきたのは、2000 年 11 月 9 日にリリースされた DirectX 8.0 からであろう。DirectX 8.0 は、マイクロソフトが 2001 年 11 月 15 日に米国で発売を開始した Xbox の描画 API と近いものであり、DirectX 8.0 に搭載されたプログラマブルシェーダがゲーム業界に浸透し始めたきっかけとなった。DirectX 8.0 対応以前のグラフィックスハードウェアは、座標変換とライティングをハードウェア化していたため(ハードウェア T&L)と呼ばれていた。しかし、DirectX 8.0 リリース後は、差別化のために固定機能シェーダと呼ばれるようになった。

コンシューマーにおけるシェーダの進化は、ビデオカードの進化といっても差し支えない。3D グラフィックスの処理を行うビデオカードは、1996 年に 3Dfx 社が発売した Voodoo Graphics や NEC 社が発売した PowerVR 搭載のものがある。このころのハードウェアは、Z バッファを持ち、テクスチャーを張ったり、アルファブレンディングを行うことが

できた。それ以外にもミップマップやバイリニアフィルタ、フォグ、アンチエイリアシングの機能も搭載している。その後、1998 年の NVIDIA 社の RIVA TNT でマルチテクスチャリングや環境マップが採用されたり、1999 年の RIVA TNT2 でステンシルバッファが搭載されるなど、ポリゴンを描く枚数を増やす機能強化とともに品質を向上させるための機能が追加されていった。特に、マルチテクスチャ機能は、複数のテクスチャーを重ね合わせて一度に描画する方法で、このときの重ね合わせ方は、アルファブレンディングを指定したり、ライティングの計算を加算したりと幾つかの方法を組み合わせることができる。これは、自由度は低いがシェーディングを切り替えることを行っており、プログラマブルシェーダの原点といえる。

大々的にプログラマブルシェーダが導入された DirectX 8.0 と、その初めての対応ハードウェアである 2001 年の 2 月に発売された NVIDIA 社の GeForce3 は、幅の広がったグラフィックスプログラミングが可能になった。バンプマッピング(この技術自体は、それ以前のハードウェアでも実現できていたが、高速化のための工夫をすることができるようになった)や、ツゥーンレンダリング、モーションブラー、エッジ抽出をゲーム開発者が創意工夫でゲームに組み込むことができるようになった。しかし、ピクセルシェーダ（画素毎の陰影付け）では、テクスチャーは 4 回しか読み込みができなかったり、算術命令が 8 回しか使えないなど、お世辞にも自由度が高いとはいえなかった。しかし、それ以前の固定機能シェーダと比較すると、かなり多彩な表現が可能になってきた。

その後、2002 年 12 月 20 日にリリースされた DirectX 9.0 では、これら制限が大幅に緩和され、パフォーマンスさえ気にしなければ、かなり自由なシェーディングのプログラミングが可能になった。さらに 2006 年 11 月 30 日にリリースされた DirectX 10.0 では、ジオメトリシェーダと呼ばれる描画するポリゴンを GPU 内で自由に増減するための仕組みが搭載され、2009 年 10 月 22 日にリリースされた DirectX 11 では、曲面描画のプログラマブル化や GPU による汎用計算のための Compute Shader が追加されてきた。

(2) リアリズムの追求

コンピューター上で真に写実的な表現を行うのは難しい。リアルタイムに実現しなければならないとなるとなおさらである。シェーダは、コンピューターの中でいかにリアルな画像を生成するかということを目的に発展してきた面が多い。具体的にどのように発展してきたか見ることにする。

物体表面の陰影は、BSSRDF(双方向表面下散乱反射分布関数)を使ったモデル：

$$L(\vec{x}_o,\vec{\omega}_o) = L_e(\vec{x}_o,\vec{\omega}_o) + \int_A\int_{\Omega+} S(\vec{x}_o,\vec{\omega}_o;\vec{x}_i,\vec{\omega}_i)\max((\vec{n}\cdot\vec{\omega}_i),0)L_i(\vec{x}_i,\vec{\omega}_i)d\vec{\omega}_i d\vec{x}_i$$

が、最も一般的なモデルとして知られている[Nicodemus77]。$S(\vec{x}_o,\vec{\omega}_o;\vec{x}_i,\vec{\omega}_i)$が、BSSRDF で、物体の表面 $\vec{x}_i$ の位置から $\vec{\omega}_i$ 方向に入射した光が、別の物体の表面 $\vec{x}_o$ において $\vec{\omega}_o$ の方

向出て行く割合を表している。ここで、$L_i(\vec{x}_i,\vec{\omega}_i)$は、表面$\vec{x}_i$の位置から$\vec{\omega}_i$方向に入射した光の輝度であり、$L_e(\vec{x}_o,\vec{\omega}_o)$は、表面$\vec{x}_o$の位置から$\vec{\omega}_o$方向に射出された発光による光の輝度である。入射位置の積分は、物体表面全体で行い、入射方向の積分は、物体の法線ベクトルを中心とした2πステラジアンの半球で計算をする。$\vec{n}$は、物体表面の位置$\vec{x}_i$における法線ベクトルである。

BSSRDF を使った反射モデルでは各画素の計算に物体表面全体と、表面の各点における 2 次元の方向積分を必要とするために、非常に負荷が高い。よく知られる反射モデルは、BRDF(双方向反射率分布関数)を使った陰影の計算モデルである。

$$L(\vec{x}_o,\vec{\omega}_o)=L_e(\vec{x}_o,\vec{\omega}_o)+\int_{\Omega+} f_r(\vec{x}_i;\vec{\omega}_o,\vec{\omega}_i)\max((\vec{n}\cdot\vec{\omega}_i),0)L_i(\vec{x}_i,\vec{\omega}_i)d\vec{\omega}_i$$

この式は、光は放出されるのは、光が入射する場所とほとんど同じ場所で行われるような材質の物体に当てはまる BSSRDF 反射モデルの近似で、金属などに良く当てはまる特性である。この式は、レンダリング方程式と呼ばれる。また、入射光$L_i(\vec{x}_i,\vec{\omega}_i)$が十分遠方から入射される、物体の質感が場所に寄らない状況が良く発生する。この場合、レンダリング方程式は、さらに簡便に記述することができ：

$$L(\vec{x}_o,\vec{\omega}_o)=\int_{\Omega+} f_r(\vec{\omega}_o,\vec{\omega}_i)\max((\vec{n}\cdot\vec{\omega}_i),0)V(\vec{x}_i;\vec{\omega}_i)L_i(\vec{\omega}_i)d\vec{\omega}_i$$

と記述することができる。ここで、$f_r(\vec{\omega}_o,\vec{\omega}_i)$は、BRDF(双方向反射率分布関数)と呼ばれる。ただし、物体は発光していないものとした。また、$V(\vec{x}_i;\vec{\omega}_i)$は、無限遠から入射される光$L_i(\vec{\omega}_i)$は、位置$\vec{x}_i$に到達するまでにどのくらい遮られるか示す量で遮蔽項と呼ばれる。この方程式でも、2 次元の積分が入るため、容易に計算することができない。この計算能力の限界を打ち破るために、いろいろな近似法が編み出されてきた。そして、この近似は、計算能力の向上に伴い、より精度の高い計算が実現できるように進化してきた。

2.2.2 BRDF に関する進化

BRDF は、物体表面での反射率を表す項である。

(1) 定数

最も簡単な BRDF は、物体表面を与えられた 1 色で塗りつぶしたり、テクスチャー画像をそのまま張り込む計算である。通常、現実の計算から拡散反射光などの視点や入射方向に依存する項を取り除いた場合の色付けを行うために使用され、アンビエント項と呼ばれる。ゲームでは、遠方に存在したりする物体には計算負荷をかけられないために、この非常にシンプルな計算が多用される。

(2)　拡散反射

拡散反射は、光が物体の表面で様々な方向に反射する現象である。コンピューターグラフィックスの世界では、通常、拡散反射というと物体表面の色が見る方向に寄らず、BRDF が定数の反射成分を指す。法線ベクトルと光の入射方向の内積の項が入っているため、物体表面の色は光の正面の面が明るく、側面より向こうを向いている面は拡散反射の色は着かない。

(3)　完全反射

完全反射は鏡などのように物体の表面に入射してきた光を入射角と同じ反射角で放出する散乱の計算である。周囲の映り込みを表現するために使われる。

(4)　鏡面反射

鏡面反射も完全反射同様、入射角と反対の方向に光を反射させる計算を指す。しかし、コンピューターグラフィックスで鏡面反射といった場合には、入射角と反射角が等しい完全反射だけではなく、少しずれた角度に対しても反射光の成分が存在する計算を行う場合が多い。鏡面反射を取り込んだ計算としては、Phong の反射モデル[Phong75]がよく知られている。鏡面反射項が存在する場合には、大きさがほぼ 0 とみなせるような小さな光源に対しても反射の効果がハイライトとして描画される。

(5)　フレネル係数

海で水を見る時、足元の水面は透明に見えるが、遠方の水面は空の色が移って見える。このように反射光に関して、物体表面の法線と視線が平行に近いときに反射光の成分が弱まる効果を電磁気学で知られるフレネルの式を使って計算すると、反射光がよりリアルに感じられることが知られている。通常は、フレネルの式そのものではなく、簡略化した Schlick の近似や、さらに簡略化した計算が用いられる。

(6)　法線マッピング

BRDF の関数形を変えるのではないが、照明効果を変化させるものして、法線マッピングがある。これは、法線マップと呼ばれる、法線の傾きを摂動させる画像を画素毎に割り当てることによって、物質の素材感を表現する方法である。鏡面マップをそのまま適用しただけでは、物体の表面はつるつるした質感であるが、法線マッピングを行うことで、目で見る凹凸が再現される。

(7)　物理ベースの BRDF

表面の見えない微妙な凹凸を考慮し、それらによって生じる微細な遮蔽情報を考慮したり、フレネル係数を考慮することで、より精密な BRDF を計算する試みも行われている。代表例としては、Cook-Torrance 金属反射モデルや Ashikhmin の異方性反射モデル、粘土などをよく再現する Oren-Nayar のモデルが存在する。Cook-Torrance の金属反射モデルは、HALO3 で実装されていることが GDC 08 で報告されている[Chen08]。

(8)　物理的に正しい BRDF

現実の物質の BRDF は、いくつかの性質を満たしている。1. 入射方向と反射方向を入れ替えても BRDF は変わらない。2. ヘルムホルツの相反性（入射方向と出射方向を入れ替えても BRDF は不変)。3. エネルギー保存則（射出するエネルギーは入射するエネルギー以下でなければならない)。これらを満たすものを物理的に正しい BRDF と呼ぶ。ゲーム開発者でも、物理的に不正確な BRDF と正確な BRDF では、表現される質感が違うという者がいる。

(9)　測定ベースの BRDF

BRDF は光の入射方向と反射方向の関数になっている。従って、当てた光と反射した光がわかれば、BRDF は逆算できることになる。具体的には、ある物体に対して、さまざまな方向から 1 回ずつ光を当て、その物体の写真を様々な方向から撮る事で、BRDF を測定できる。BRDF のデータは、4 次元のデータなので、実際には幾つかの鏡面反射の組み合わせに分解するなど、軽量な手法にフィッティングすることが行われている。

2.2.3　光源に関する進化

入射する光についても、いかに低負荷でもっともらしい照明効果が得られるか研究が行われてきた。

(1)　アンビエントライト

光の強さを方向に寄らず一定にする計算がもっとも負荷が低い。全体的な雰囲気を与えるだけなら、これで十分である。

(2)　平行光源

位置に依らない方向だけの光を平行光源と言う。無限遠にライトがひとつおいてある状態を考えると良い。拡散反射の場合には、物体の面の法線と光源の方向のベクトルの内積だけで面に当たる光の強さが求まるので、非常に頻繁に用いられている。複雑な光源は、複数の平行光源をさまざまな方向から照らすことで再現できる。

(3)　点光源

方向に寄らない位置だけの光が点光源である。（非常に良く作られた）裸電球と思うのが良い。エネルギーの保存から、光源の位置からの距離の 2 乗に反比例して光の強さは弱まる。最も簡単な光の強さが位置に応じて変化する光源と思っても良い。爆発の際の照明などに使われる。

(4)　スポットライト

現実のコンサートや芝居で使われるスポットライトを模倣した光源も考えられている。ライトの中心位置とライトの向きを持ち、照明を当てる物体がライトの向きから逸れている角度で光の強さを弱める計算を行う。

(5)　半球ライティング

1 つの平行光源とその逆方向からの光源を 1 セットとして入射光を計算する手法を半球ライティングと呼ぶ。1 つの平行光源とは、太陽光のような強い光で、反対側の光は太陽光の反射光を疑似的に表現した効果を与える。半球ライティングは、計算負荷が効果と比して低いため、『ヴァルキリープロファイル 2 -シルメリア-』(2006、スクウェアエニックス)や『METAL GEAR SOLID 4 GUNS OF THE PATRIOTS』(2008、コナミデジタルエンタテインメント)等のゲームで使われている[Zenji06, Zenji08]。

(6)　環境光源

あらゆる方向からの光を考慮するように見せる手法が環境光源である。無限遠法から入射する光を書き込んだ画像を光源として利用する。あらゆる方向の光といっても、光の方向に関する分解能は、画像の解像度に依存する。環境マップは、視線の反射方向に対応する環境光源の画像のテクセルの値を読み込んで反射光として使う環境光源の最も簡単な使い方である。それ以外にも、ぼかした環境光源を用いて同様の計算を行えば、鏡面反射が実現出来たり、環境光源を何点か読み込むことで、たくさんの方向からの光を取り込むこ

とができる。環境光源を使った描画は、IBL(イメージ・ベースド・ライティング)とも呼ばれる。

(7) 光伝搬ボリューム

空間をグリッドに分割し、グリッドの各頂点に光の情報を格納して描画する方法が知られている。もともとは、放射輝度ボリューム[Greger98]としてコンピューターグラフィックス業界で知られた手法であったが、各方向に入射する光を集めた放射輝度だけでなく、入射する輝度を球面調和関数で展開する手法や[Kaplanyan10] 、8 方向の情報を持たせる手法が実装されている[Zenji09] 。

2.2.4 遮蔽に関する進化

(1) 定数

遮蔽に関する最も簡単な処理は、物体を一様に明るくしたり暗くしたりすることである。具体的には、キャラクターが木陰に入った際に暗くなるように処理を行う。この手法は、キャラクターが小さかったり、動きが速い場合には有効である。

(2) 影

ゲーム開発では、物体が影を落とすことに研究が続けられてきた。丸いテクスチャーをキャラクターの足元に置く方法から、光源からの距離をテクスチャーに格納して物体が影に隠れるか計算を行うシャドウマップの手法まで進化してきた。

(3) ライトマップ

ライトマップは、スポットライトの前にセロファンを張って色を変えるように、テクスチャーを与えることで、ステンドグラスのような、複雑な色、形状の光を落とす仕組みである。テクスチャーの模様はアーティストが自由に制御することができるため、面白い模様の光を当たることができる。たとえば、リッジレーサー6 で、パックマン型に光を照らすヘッドライトを持った車を実装した。

(4) アンビエントオクルージョン

影を落とす手法では、平行光源などの一点から射出される光を対象としており、環境光源に対する影を生成することができない。アンビエントオクルージョンは、無限遠から一

様な強さの光が来ると仮定して、物体の各点に入射する（遮蔽されない）光を集めた結果を物体の陰影計算に使用する手法である。各方向からの光の強さを考慮しないため、光源が動くような場合に対応できないが、事前に計算され、実行時の負荷が低いため、古くから行われている。近年では、レンダリングした結果の深度情報から陰影を付加するSSAO(スクリーン・スペース・アンビエント・オクルージョン)の技法を発展させる研究が進められている。ゲームでは、2007 年の『Crysis』(エレクトロニック・アーツ)で実装されている[Martin07]。

(5) PRT

アンビエントオクルージョンでは、光源の動きに追従できない。そこで、遮蔽項を球面調和関数展開して光源を動かした時に物体に落とす影も動くように拡張したものがPRT(Pre-computed Radiance Transfer:事前計算済み放射輝度伝搬)である。球面調和関数展開した遮蔽項と球面調和関数展開した環境光源は展開係数を積和演算するだけで物体に入射する光を計算することができる。実際には、球面調和関数の低次の項のみしか計算に使用しないので、鋭い光源を表現するには向いていない。

(6) シャドウレイ

物体表面の各点から、陰影を推測する方法がシャドウレイである。物体表面から光源方向にレイを飛ばし、レイが光源に達した場合には、その値を使用し、光源に達しない場合にはその光源の寄与を考慮しないことで遮蔽計算を行う。レイトレーシングで良く用いられる手法であるが、高さテクスチャーを利用した凹凸計算を行うレリーフテクスチャマッピングで影を計算する場合にも用いられる。

(7) 高次反射

遮蔽項は、通常は物体に落とす影を計算し、物体を暗く表現するのにつかわれるが、シャドウレイの計算において、物体に衝突した際に、物体表面の色を用いて色付けする手法も知られている。必ずしも遮蔽項のみが影響するわけではないが、他の点における反射光をライティングに使用した効果を高次反射と呼ぶ。通常は、一点からの光やそれによる影の効果、環境マップなどのシンプルな環境光源の寄与以外の照明効果を大域照明と呼ぶ。

2.2.5　集光手法

(1)　マルチライト

平行光源や点光源は、1 つの光源による効果を表している。より現実的な結果を得るためには､ライトを複数置く必要がある｡ライトの本数はそのまま計算負荷に比例するため、

(2)　遅延レンダリング（Deferred Rendering）

近年、注目を集めているのが、遅延レンダリングである。これは、画面と同じ大きさのバッファ（G バッファ）を複数用意しておき、ここに深度や法線などの情報を書き出し、最終的に画像処理として画面を完成させる手法である。遅延レンダリングは、陰影計算を処理する回数が画素のサイズと上限が決まっているため、複雑な形状をした物体の場合でも計算負荷が低い。また、光源の数を増やしても、光源の数に比例するよりも低い負荷にすることができる。ただし、遅延レンダリングは、半透明の物体の裏にある物体の陰影計算を行うことができないため、不透明の物体に関するレンダリングに用いられるのが普通である。遅延レンダリングは､『Killzone2』(2009、ソニー・コンピュータエンタテインメントジャパン)で実装されていることが発表されている[Valient07]｡遅延レンダリングの研究は現在も進んでおり、G バッファに何を書き出すのか？レンダリングをどのように設計するのか、アンビエントオクルージョンも統合すべきかなど、まだ最終的に効率が良い方法は定まっていない。

(3)　球面調和関数ライティング

球面調和関数は、球面を定義域とする関数の近似方法である。最低次の近似では、全方向からの光の平均が、次の次数の近似では、X,Y,Z 軸に関する光の強さの平均が、次数をよりためた場合には、より細かな光の空間分布を取り込むことができる。球面調和関数ライティングは、後述する PRT と共に用いられ、半球ライティングよりもより複雑な周辺の入射光を考慮することができる。無限に次数が高い計算ができれば、正確に入射光を考慮することができるが、コンピューターのメモリ、計算能力は有限であるので、実際にはそれほど高い次数の計算はできない。

(4)　ファイナルギャザリング

BRDF を正確に計算するには、画素毎に入射する全方向からの光の強さを調べなくてはならない。BRDF が光の入射方向に依存する場合には、この計算は、画素の対象とする物体表面から、様々な方向に入射光の強さを調べることになる。これは、完全反射が正反射方向だけに対する入射光を考慮したことと比較すると、全方向の入射光を考慮することに対応する。精確な計算をするには多くの方向の入射光を考慮する必要があり、精確さを求

めるほど計算時間を必要とする。

2.2.6　照明モデル

以上、それぞれの個別要素について、複雑さについての歴史をたどってみた。実際には、要素を組み合わせた照明モデルが重要になってくる。遮蔽項も含めて考えると、シェーディングの計算を分解することは難しい作業であり、ある陰影の効果を多重に計算する場合が多く見受けられる。処理能力が有限のゲーム開発では、いかに気にならないようなモデルを使うかということが重要になってくる。

(1)　頂点色のみ

ポリゴンゲームの初期のころは、計算能力がほとんど無かったため、ポリゴンの頂点や面にあらかじめ色を付けておき照明計算を用いない手法が使われていた。

(2)　テクスチャー貼り付けのみ

次の複雑さとして、テクスチャーを張るだけで陰影の計算が行われてきた。

(3)　ベイク(焼付け)

頂点色やテクスチャーに事前に精確に計算したレンダリングの結果を使用することをベイクという。拡散光の結果を頂点データにベイクすることで、アンビエントオクルージョンなどの効果を再現することは、1998 年の『R4 -RIDGE RACER TYPE 4-』(バンダイナムコゲームス)の頃から行っている。プレイステーションでは、能力を最大限に引き出すために、16 色のテクスチャーが多用されたが、その際に（線形の補間で綺麗にグラデーションが掛かる）頂点色を乗算することで表示される色数を増やすような工夫が行われていた。

(4)　Lambert の反射モデル

アンビエントライトによる定数の反射項と平行光源による拡散反射を組み合わせた計算式が、Lambert の反射モデルである。ハイライトを生じない物体には、このモデルが多く使用されている。

(5)　Phong の反射モデル

Lambert の反射モデルに光の反射ベクトルと視線の内積のべき乗による鏡面反射項を追加したモデルが、Phong の反射モデルである。Phong の反射モデルや Lambert の反射モデルは、プログラマブルシェーダ以前から使用されてきた。

(6)　Lambert の反射モデル＋環境マッピング

Phong の反射モデルは、鏡面反射項の計算が追加される分だけ処理負荷は高い。環境マップとして光源をぼかしたテクスチャーを使うことで、環境マッピングと鏡面反射を同時に実現することが可能である。

(7)　法線マッピング付き反射モデル

2004 年に発売された『DOOM3』(id software)や『ハーフライフ 2』(Valve)から人体に対する適用を含めた本格的な法線マップを使った処理が用いられてきた。ほとんどが法線ベクトルを摂動させ、形状の変化を伴わないバンプマッピングでの実装であるが、これからは、細分割処理を伴うディスプレースメントマッピングがいろいろなソフトウェアで実現されることが予想されている。また、このようなレンダリングが実現されてきた頃から、シャドウマップやそれを拡張した影の計算を行うゲームが増えてきている。

(8)　球面調和関数を用いた広いレンジのライティング

『Halo3』では、球面調和関数展開した環境光源による拡散項と、同じく球面調和関数展開した低い周波数の Cook-Torrance 金属反射、ぼかした環境マップピングおよび点光源に関する鏡面反射の計算が行われている[Chen08]。このように色々な種類の光を適切な負荷で混合する照明処理に関するバランスが現在のゲーム作りに求められている。

2.2.7　その他のシェーディング

(1)　BSSRDF の効果の取り込み

BRDF では、入射した光は、入射した場所から放出される。しかし、ひと肌などは、よりやわらかに見える反射が起きており、これは入射した光が他の点から放出された結果生じている。この効果を再現する試みはいくつか行われているが、簡易的な手法としては、リムライトが知られている。リムライトは、逆光時に側面の輪郭部分から光が滲み出るような現象で、シェーダ処理では、本来のライティングよりも輪郭部分に仮想的に光を強く与えることでこの効果を実現することができる。リムライトの処理は『バーチャファイタ

ー5』(2006、セガ) や『METAL GEAR SOLID 4』で実装されていることが知られている[Zenji062, Zenji08]。

(2) ポストエフェクト

プログラマブルシェーダの応用例の 1 つに画像処理がある。レンダリングされた画像に対してシェーダを使って処理することで、よりリアルな画像を生成したり、効果的な見せ方をすることができる。ゲーム開発では、この処理はポストエフェクトと呼ばれ多用されている。

(a) グレア

人が現実にものを見ている時には、まつ毛などの効果により、星形に光が漏れて見えたり、まわりがぼやっと広がって見えたりする。ゲームでは、ポストエフェクトで輝度の高い領域を抽出し、その領域に対して光を広げるなどの実装によってグレアを表現する。グレアは、『DOUBLE-S.T.E.A.L』(2002、ぶんか社)で実装された手法が広く知られている[Kawase02] 。

(b) トーンマッピング

ディスプレイで表現できる色情報は、赤・緑・青が各 8bit の制限された情報である。似たような色は画面では同じ色に表示され、非常に強い光は全て白く表示される。物理的な正確さではなく、見た目をよくするために、陰影の結果を 8bit に上手におさめる手法が研究されている。これらの手法を一般にトーンマッピングと呼ぶ。トーンマッピングは、人間の目のシミュレーションにも使われる。人の目は明るいところでは明るさになれ、暗いところでは次第に暗いところでもものが見えるようになる。この対応は、明順応・暗順応と呼ばれる。トーンマッピングのパラメーターを状況に応じて変化させることで、この効果を再現することができる。『ワンダと巨像』(2005、ソニー・コンピュータエンタテインメント)や『ヴァルキリープロファイル 2』でトーンマッピングが実装されていることが知られている[Zenji05, Zenji06]。

(c) ガンマ補正

ディスプレイで表示される明るさは、0 が黒、1 を白とした場合に、0.5 の強さが黒と白の丁度中間の明るさに見えるわけではない。これを補正するための処理がガンマ補正である。ライティングの計算は、エネルギーの数値計算を行っていることに対応するため、シェーディングの結果はガンマ補正を施さないと正確な値にはならない。この効果は通常の

レンダリングでは気になることは少ないが、グレアや被写界深度の計算に対してガンマエフェクトを考慮するかどうかによって、最終結果は大きく変わってくる[Kawase08]。

(3) NPR

必ずしもリアルな陰影が良いというわけではない。アニメーションを題材としたゲームでは、アニメーションを再現することやイラストの画風を再現することが重要な場合もある。このような現実を超えた陰影処理を行うことは Non-Photo Realistic Rendering と呼ばれる。

(a) トゥーンレンダリング

アニメ的な絵を再現することはトゥーンレンダリングと呼ばれる。具体的には、陰影の結果を、2～3 種類の段階的な色分けを行うことと、輪郭線を抽出する処理がトゥーンレンダリングの処理として知られている。

(b) ハーフランバート

通常の拡散反射の計算では、光の裏を向いている面には光の寄与は与えられない。この結果、日が当らない部分の陰影はのっぺらとした単調なものとなる。この問題を回避した陰影方法が Half-Life で開発されたハーフランバートである [Mitchell06]。これは、拡散反射の計算で法線と光源方向の内積を取った後に、負の値を 0 に丸め込まないで、-1 から 1 の間の値の結果を 0 から 1 に正規化して拡散反射の計算に用いる手法である。したがって、最終的な結果は物理的に正確ではないが、「それらしく見えるのが大事」というゲーム開発の現場では有効な手法といえる。

(4) モーションブラー

カメラで写真を撮るときは、シャッターが開いている時間はある有限な時間である。また、ビデオでも連続的な時間をある間隔で区切って、その間の情景を 1 つのフレームに格納している。この結果、物体やカメラが動いていると、得られる絵は動いたものが線を引いているかのように広がったように見られる。これを画面上で再現したものがモーションブラーである。モーションブラーは単にビデオの絵を再現するというだけではなく、現実的な時間の連続性を認識するのに有効である。つまり、動きが見えやすくなる。モーションブラーは、カプコンの MT フレームワークで実装されていることが知られている[Ishida06]。なお、残像として網膜に光が残る現象のシミュレーションは、レースゲームにおいてテールランプが尾を引いたり、剣劇ゲーム等で刀の通った道筋に軌跡を残すなど

の処理が古くからおこなわれている。

2.2.8　この先のシェーダについて

以上、不十分な点は多くみられると思うが、陰影処理を中心にシェーダで扱われていることがらを列挙させていただいた。ここでは、現在の状態を見すえて、将来の道筋を探ってみたい。

まず、現在の画像をよりリアルにするという方向性は続いていくであろう。現実的な結果を求めるゲームでは、それしか道がないからである。近い将来、我々はリアルタイムの完全な大域照明が実現するように頑張らないといけない。また、必ずしもリアルでなくても、CG 映画を目標とした品質向上の試みは続いていくと思われる。

ハードウェアとしては、現世代のビデオカードでは、細分割が強化された。今までは、『Halo WARS』(2009、Bungie)における地形表現のような限られた場所だけでハードウェアを用いた細分割曲面が使われてきた[McAnlis09]が、これからは、より滑らかなポリゴンが見られるようになるであろう。

デバイスとしては、現在、映画業界、TV 業界を中心に 3D（立体視）が注目を集めている。ゲームは、基本的には、3 次元計算をしているので、とりあえずの対応は難しくはないが完璧な対応をするには処理を追加しなくてはならない。例えば、ビルボード処理している物体は、ビルボードのポリゴンの Z 値が深度バッファに記録されるため、凹凸がある場合には、その分の情報を付与しなくてはならない。

他に、TV との関係として、現在、TV の明るさはまちまちである。グレアの強さは、限界がある TV というデバイスに関して、脳に入る光の強さを TV に表示される前に補正するという考えであるが、TV の明るさが分からないと正確な補正ができない。今後、TV 側からの情報フィードバックを元に、グレアの強さなど、より正確な絵を出力できる時代が来るかもしれない。もしくは、高輝度ディスプレイが普及することで、グレア処理が不必要になる可能性もある。

参考文献：

[Chen08] Chen, H. “Lighting and materials of Halo 3,” Game Developers Conference, 2008.

[Cook84] Robert L. Cook. "Shade trees," Computer Graphics (SIGGRAPH '84 Proceedings), pp. 223-231.

[Cook87] Robert L. Cook., Loren Carpenter, and Edwin Catmull. "The Reyes Image Rendering Architecture." Computer Graphics (SIGGRAPH '87 Proceedings), pp. 95-102.

[Greger98] The Irradiance Volume, Greger, G., Shirley, P., Hubbard, P., and Greenberg,

D., IEEE Computer Graphics & Applications, 18(2):32-43, 1998
[Ishida06] 石田智史, "次世代機に向けたゲームエンジンの設計," CEDEC 2006, 2006.
[Kaplanyan10] Anton Kaplanyan, Carsten Dachsbacher, "Cascaded Light Propagation Volumes for Real-Time Indirect Illumination," ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2010.
[Kawase02] Masaki Kawase, " DOUBLE-S.T.E.A.L におけるリアルタイム CG 表現技法," CEDEC 2002, 2002.
[Kawase08] Masaki Kawase, " IMAGIRE DAY (1) レンダリスト養成講座 2.0," CEDEC 2008, 2008.
[Martin07] Mittring, Martin, "Finding next gen: CryEngine 2," SIGGRAPH '07: ACM SIGGRAPH 2007 courses, 2007.
[McAnlis09] Colt McAnlis, "HALO WARS: The Terrain of Next-Gen," Game Developers Conference, 2009.
[Mitchell06] Jason Mitchell, Gary McTaggart and Chris Green,"Shading in Valve's Source Engine," SIGGRAPH Course on Advanced Real-Time Rendering in 3D Graphics and Games, 2006.
[Nicodemus77]F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric considerations and nomenclature for reflectance. Monograph 161, National Bureau of Standards (US), October 1977.
[Phong75] Bui Tuong Phong, "Illumination for Computer Generated Pictures," Communications of the ACM, 18(6), pp.311-317, June 1975.
[Valient07] Michal Valient, "Deferred Rendering in Killzone2," Presentation at Develop Conference in Brighton, July 2007.
[Zenji05] 西川善司, "3D ゲームファンのための「ワンダと巨像」グラフィックス講座", http://game.watch.impress.co.jp/docs/20051207/3dwa.htm, GAME Watch, 2005.
[Zenji06] 西川善司, "3D ゲームファンのための「ヴァルキリープロファイル 2」グラフィックス講座", http://game.watch.impress.co.jp/docs/20060911/vp2.htm, GAME Watch, 2006.
[Zenji062] 西川善司, "3D ゲームファンのための「バーチャファイター5」グラフィックス講座", http://game.watch.impress.co.jp/docs/20061025/3dvf5.htm, GAME Watch, 2006.
[Zenji08] 西川善司, "西川善司の 3D ゲームファンのための「METAL GEAR SOLID 4」グラフィックス講座", http://game.watch.impress.co.jp/docs/20081203/3dmg4.htm, GAME Watch, 2008.
[Zenji09] 西川善司, "西川善司の 3D ゲームファンのための「ソニック・ワールド・アドベンチャー」グラフィックス講座", http://game.watch.impress.co.jp/docs/series/3dcg/20090410_110682.html, GAME Watch, 2009.

表 2.2-01　シェーダの年表

年代	名称等	種類
1984	Shade trees	論文
1986	PhotoRealistic RenderMan	レンダラー
1996	Voodoo Graphics, PowerVR	ビデオカード
1998	RIVA TNT	ビデオカード
1998	R4 -RIDGE RACER TYPE 4-(ベイク)	タイトル
1998	Half-Life(ハーフランバート)	タイトル
1999	RIVA TNT2	ビデオカード
2000	DirectX 8.0	API
2001	GeForce3	ビデオカード
2002	DOUBLE-S.T.E.A.L(グレア)	タイトル
2002	DirectX 9.0	API
2004	ハーフライフ 2(法線マッピング、トーンマッピング)	タイトル
2005	ワンダと巨像（トーンマッピング）	タイトル
2006	ヴァルキリープロファイル 2（半球ライティング）	タイトル
2006	バーチャファイター5（リムライトシェーダ）	タイトル
2006	MT フレームワーク（モーションブラー）	ゲームエンジン
2006	DirectX 10.0	API
2007	Halo3(球面調和関数展開した金属反射など)	タイトル
2007	Killzone2(遅延レンダリング)	タイトル
2007	Cyrsis(SSAO)	タイトル
2008	ソニックワールドアドベンチャー(光伝搬ボリューム)	タイトル
2009	Halo WARS(細分割曲面)	タイトル
2009	DirectX 11	API

2.3　ビデオゲームにおける物理シミュレーションの歴史について

今給黎　隆（株式会社バンダイナムコゲームス）

2.3.1　はじめに

ビデオゲームにおける物理シミュレーションの歴史であるが、古くから振り返ると大きく分けて以下の 2 つの潮流が感じられる。

(1)　現実に知られた遊びをコンピューターの中で再現する試み

(2)　ゲーム内の動きを、より現実に近づけるための試み

　(a)　エフェクト物理：演出としての物理シミュレーション

　(b)　ゲーム物理 : 技術としての物理シミュレーションが遊びの感覚にも影響を及ぼす

1 つめの現実の遊びを再現する試みは、スポーツのゲーム化として多くの事例が見られる。2 つ目のゲームをより現実に近づける試みは、ゲームの演出として用いられることが多い。その中を細かく見ると、ゲームの難しさにほとんど影響を及ぼさないエフェクト物理と、積極的にプレイ感覚に盛り込もうとするゲーム物理に分かれる。もちろん、両者は密接に関係している。より現実に近いシミュレーションが可能になったために、今まで不可能だった遊びが模倣できるようになったという場合も多い。以下では、それぞれについて実例を挙げて、より深く掘り下げていく。

(1)　現実の遊びを模倣するためのシミュレーション

スポーツゲームなどの現実の遊びに物理シミュレーションを入れた試みについては、ビデオゲームの初期から始まり、多くのジャンルが知られている。1958 年に発表されたコンピュータゲームの元祖的存在であるブルックヘブン国立研究所の『Tennis for Two』は、テニスをモチーフとしていた。Tennis for Two は、地面とネットのラインがオシロスコープのモニターに横向きで描かれ、ボタンを押してボールを打ち返すことを楽しむゲームである。実際のテニスでは、ラケットで打ち返すときの腕の動き等に応じて、球の飛び方は複雑に変化するが、そこまでのシミュレーションは行われていない。現実の遊びを模倣するシミュレーションの歴史は、コンピューターの計算性能に応じて、詳細な計算を取り入れるという発展をしてきた。この分野に関しては、古くは計算性能が家庭用ゲーム機よりも高い PC のゲームやゲームセンターにおいてあるアーケードゲームにおいて色々な試みが行われてきた。

ピンボールは、古くからコンピューター上で遊ばれてきた。1981 年に BudgeCo 社からリリースされた『Raster Blaster』は、Apple II 向けの最初のピンボールのゲームのよう

である。この BudgeCo 社がその後開発し、1983 年に Electronic Arts 社から複数のコンピューター用に発売された Pinball Construction Set は、盤面をユーザーがデザインすることが出来た。これは、コンピューターによるシミュレーションならではの自由度を活かした遊びの提案であるといえる。

ゴルフゲームも、物理シミュレーションによってビデオゲームとして早くから再現されたジャンルの 1 つである。1983 年に T&E ソフトから発売された『3-D ゴルフシミュレーション』は、MSX や FM-7 などの 8bit 機の PC 向けのゲームでありながら、物理計算によるゴルフボールの転がり方の計算や、3D ポリゴンによる画面表示が行われていた。それ以外の球技では、ボーリングやビリヤードが、古くから PC などのゲームとして遊ばれてきた。しかし、どの程度正確に計算されていたかは不明である。

スポーツゲームの 1 分野として、乗り物を操作するゲームにもシミュレーションが用いられてきた。フライトシミュレーションは、そのうちの 1 つである。現実と同じような操作方法によって飛行機を操縦し、大空を自由に飛ぶゲームが作成されてきた。マイクロソフトが開発した『Microsoft Flight Simulator』は、そのようなゲームの 1 つで、元々は、1979 年に SubLOGIC 社が Apple II 向けに開発したゲームである。ただ飛行する以外にも、1986 年の Microprose 社による『Gunship』の様に、ヘリコプターというテーマで戦闘をするようなゲームも発売されている。

乗り物のゲームとしては、車も多くシミュレートされている題材である。3D ポリゴンを用いた車のシミュレーションゲームとしては、1988 年にアタリ社からリリースされた『ハードドライビング』やナムコの『ウイニングラン』が存在する。以上は、アーケードゲームであるが、家庭用ゲームでも、1997 年にソニー・コンピュータエンタテインメントから発売された『グランツーリスモ』は、リアルドライビングシミュレーターと銘打っており、より本物の車に近い挙動の計算を行い、プレイステーション本体の売り上げを牽引した。

(2) 現実に近づけるシミュレーション

「ゲーム内の動きを、より現実に近づけるための試み」は、主に自然現象を題材としている。

最も簡単な物理シミュレーションは重力のシミュレーションといえる。物体は、一定の重力加速度によって下方に押される、他の力が掛からない場合には、ある初速度で投げ出された物体は放物線の軌跡を描く。この動きは、Tennis for Two において既に実装されていた。Tennis for Two は、テニスというには球の挙動がリアルであるとはいえない。しかし、現実世界を反映させるものとして、重力計算が最もシンプルで効果が高いということがビデオゲームの初期から認識されていたということは、注目すべき事柄のように感じる。但し、ゲームにおける放物運動の計算では、「ジャンプテーブル」と、呼ばれる、ジャンプ

アクションゲーム等のジャンプした時の高さの時間変化等をあらかじめ計算しておき、配列変数に格納することで、重力を考慮する手法が知られている。これを、シミュレーションというかは微妙な問題である。しかし、ゲーム機の計算能力が低い時期でも、運動方程式を位置 X と速度 V に関する連立一階微分方程式、

$$V \leftarrow V + h * F$$
$$X \leftarrow X + h * V$$

（h は動きを制御するための時間差分的な係数。細かな係数は省略）

として計算している場合もあり(力 F は実際には重力)、これは現在、広く知られている物理エンジンの草分け的なものと言えそうである。

物理シミュレーションという点では、慣性も非常に重要なものといえる。1979 年にアタリ社がリリースした『Lunar Lander』は、月面着陸をモチーフとしたゲームであり、自由に向きを変えられるロケットを、進行方向の勢いについて重力を考慮して調整することで特定のポイントに静かに着陸することを目指すゲームである。ロケットには慣性が付いており、慣性による移動量と速度を考えないとクリアできないのがゲームを難しくも面白いものとしている。また、1984 年にアタリゲームズが発売した『マーブルマッドネス』は、ボールを転がして制限時間以内にゴールを目指すゲームであるが、このボールの動きにも強い慣性が働き、簡単な物理シミュレーションがゲームを面白くする一例として知られている。

次の段階としてゲームで見られる物理シミュレーションとしては、糸や髪の毛のような細い紐のシミュレーションが挙げられる。2 次元的なゲームとしては、1992 年にサイベル社より発売されたパズルゲームソフトの『インクレディブル・マシーン』において、紐のシミュレーションが行われていたり、1994 年に株式会社 TNN から発売された『海腹川背』においてゴムロープのシミュレーションが行われ、ゲーム物理として楽しまれてきた。3 次元の物理としては、1993 年にリリースされたセガの『バーチャファイター』で、女性キャラクターの髪の毛などが、重力や髪の付け根の動きの力から動力学的な運動を行っている。髪の毛のシミュレーションは、キャラクターを動かす関節の骨構造に対して、髪の毛の複数の代表点に関して関節を付与し、関節を質点とし、関節間を固定長で関節において自由に曲がるモデルとして計算することが多い。このモデルは、髪の毛の動きにとどまらず、草木のゆれなどに使われる場合もある。

また、1994 年にリリースされた続編の『バーチャファイター2』では、衣類が部分的にはためくクロスシミュレーションが実装されている。この時期の手法は、衣類は、関節が縦軸に対して横軸が繋がるモデルとして計算しており、関節の角度を一定の順番で計算すれば全て求められる髪の毛よりも計算は複雑である。布の計算は、CG 業界では古くから行われてきたが、色々なゲームで使えるようにライブラリ化された物は少ない。Ageia(現 NVIDIA)の SDK では、2006 年のバージョン 2.4.0 から Cloth の計算が入っている。Havok 社の Havok Cloth は、2008 年に発表されているが、まだ発展途中のライブラリといえる。

他に物理シミュレーションが取り入れられた表現は水面である。特に、1996 年に任天堂

から発売された『ウエーブレース 64』は、波の動きが美しいゲームとして知られている。水面を表現する方法としては、水面を 2 次元の四角形のメッシュに区切り、メッシュの各頂点において、周囲の頂点との高低差を緩和する方向に修正する高さ場的な方法が広く知られている。2002 年にぶんか社から発売された『DOUBLE-S.T.E.A.L.』では、テクスチャーのテクセルを水面のメッシュの頂点に対応させた 2 次元の波動シミュレーションに関する GPU を用いた実装を行っている[Kawase02]。水面のシミュレーションには、物理シミュレーションではなく、ノイズ関数やトロコイド曲線を使う方法も知られているが、近年では、流体の運動方程式であるナビエ-ストークス方程式を使った水の表現の挑戦が行われている。2008 年にハドソンからリリースされた『AQUA FOREST Powered by OctaveEngine Casual』では、水の内部まで考慮した流体シミュレーションに関して、水槽を横から見たような 2 次元の場合について粒子法を用いて計算を行っている。3 次元のナビエ・ストークス方程式に基づく流体シミュレーションは、計算力やメモリについて現在のゲーム機にとって大量のリソースを必要とするため、リアルな水の物理シミュレーションを実現するには、まだ時間が掛かるものと思われる。また、流体に近い軟体も、精力的に挑戦されているテーマの 1 つである[Zenji09]。

それ以外に汎用的に導入されるようになってきたのが、剛体物理である。ゲーム機がポリゴンを表示できるようになると、本格的な剛体シミュレーションを行うようなソフトウェアが現れてきた。1998 年に Electronic Arts 社から発売された『Jurassic Park: Trespasser』は、1 人称視点のゲームであるが、独自の物理シミュレーションを搭載し、3 次元のフィールドの中を物体を動かしながら冒険できていた。1999 年にソニー・コンピュータエンタテインメントが発売した『パネキット』は、剛体シミュレーションで動くパーツを 100 個まで組み合わせて、乗り物などのモデルを作り、動かして遊ぶようなゲーム物理としての物理シミュレーションゲームである。パネキットは摩擦や空気抵抗という物理法則を主とした遊びでも十分にエンターテインメントとなりうることを示した好例である。このようなゲーム物理の例としては、2008 年にソニー・コンピュータエンタテインメントが発売した『リトルビッグプラネット』がある。リトルビッグプラネットでは、ユーザーが自由にステージを作成することが可能である。ゲーム物理の難しい点は、シミュレーションでは不安定な状況が発生することで、ゲームが進まなくなる可能性が発生しがちな点であるが、それをレベルデザインによって逃げることなく自由に制作できるようにして提供しているのは非常に挑戦的である。

近年、剛体シミュレーションは、ライブラリ化という独立した物理シミュレーションとして確立してきた。通常、「物理エンジン」と呼ばれるものは、剛体シミュレーションのライブラリのことである。2002 年に設立された Ageia（2008 年に NVIDIA が買収。Ageia 自身もスイス連邦工科大学のスピンオフ企業である NovodeX を買収して、物理エンジンの提供を始めている）の NovodeX（現 Nvidia PhysX）や、1998 年に設立された Havok 社（2007 年に Intel が買収）の Havok のように商業的に成功したライブラリも現れてい

る。物理エンジンの開発会社が CPU や GPU の会社に買われるのは理由がある。今後、CPU のアーキテクチャは、計算ユニットのコア数が増えて来て、剛体物理は挑戦的な問題ではあるが並列化は不可能ではなく、将来の CPU のトレンドに合致している。また、GPU は逆に自由度が増すことで CPU に近づいてくることが予想され、同様の計算が高速に行えることが見込まれる。それゆえ、将来の CPU/GPU が有効に使えるアプリケーションとして物理エンジンがチップメーカーに注目されたのである。非商用のライブラリについても、Bullet が Rockstar Games 社の『Grand Theft Auto 4』(2008)に使われる[Erwin09]など、オープンソースのライブラリが市販されているゲームに組み込まれている。

日本のゲーム会社は、全てのプログラミングを独自で開発しようとするといわれている。物理エンジンも例外ではない。幾つかの会社で、独自エンジンと呼ばれる内製の物理エンジンが作られている。カプコンの MT フレームワーク 2.0[Zenji09]、コナミのメタルギアチーム[Zenji08]やコーエーテクモ[Tsuda08]は独自で物理エンジンを実装していることが知られている。また、ソニー・コンピュータエンタテインメントの内製の物理エンジンである PhysicsEffects は、PlayStation 3 SDK の一部として提供されていた物理シミュレーションエンジンであるが、オープンソース物理エンジン「Bullet」内部で利用されるコンポネントとして公開されている。なお、内製でない物理エンジンである Havok の日本での使用例としては、2004 年に発売されたセガ社の『ASTRO BOY 鉄腕アトム』が以前から知られている[Sega03]。

現在、ビデオゲームで精力的に取り入れられつつあるのが破壊現象である。物理シミュレーションは、近年の CPU/GPU の発展に伴って大量の物体に関する複雑な計算が可能になってきた。物体を複数に分割する破壊シミュレーションは、破壊のための判定の計算の負荷も大きいが、細かく分割された後の破砕物は、物体の数が多くなるために壊れないとしてもシミュレーションの負荷が高くなる。破壊の処理は、以前は壊れる前の大きな物体と、複数の物体を用意し、破壊が起きた時点で、物体を差し替えることで実現されてきた。破片の動きも、破片の形状を考慮しない簡易的な粒子シミュレーションが多く行われていた。しかし、現在では、2008 年にルーカスアーツから発売された『Star Wars: The Force Unleashed』の様に、有限要素法による動的分割を用いてより物理的に正確な破壊を計算する試みも行われている[Eric09]。今後は物理シミュレーションの発展に伴って、より多くのゲームで物体が大量に破壊されるゲームが増えることが見込まれる。Havok 社は、Havok Destruction というライブラリを 2008 年に発表しており、このライブラリが普及すれば、さらに破壊は身近なものになることが推測される。

それ以外では、『人間』が重要なテーマになりつつある。人間の動きは、アニメーションの分野で広く研究されているが、物理的に正確な動きをするために、物理シミュレーションを行うようになってきた。人間に関する物理シミュレーションとして、剛体を角度制限された関節でつないで表現するラグドールが知られている（Jurassic Park: Trespasser が、最初にラグドールが導入されたゲームのようである)。ラグドールでは死体のような動

きしか再現することが出来ないが、Havok の Havok Animation では、アニメーションと物理を自然に見える形でブレンドするための機能が提供されていたり、Nvidia 社の PhysX は Natural Motion 社の人体挙動をシミュレートするミドルウェアの中の物理演算部分に組み込まれ、利用されていたりと、人体に関して、より環境に応じつつ自然な動きをする研究が進められている。

(3) シミュレーションの将来

破壊の説明でも述べたように、CPU/GPU のトレンドを考えると、将来の物理エンジンはより並列化に適した多数のオブジェクトを同時にシミュレートするようなゲームが増えることが推測される。多くの物体という点では、髪の毛のシミュレーションが飛躍しそうである。学術レベルでは、現在、髪の毛一本、一本をシミュレーションする研究が行われているが、これが数年後には、ゲーム機で実装されている可能性が高い。無論、それに応じて、布の動きなどもよりリアルになってくるであろう。他に現在学術系で積極的に研究されているテーマとしては、流体や爆発、燃焼があり、この分野もさらに進化すると思われる。これらは総じて「捕らえるのが難しいもの」という特徴があり、シミュレーションは、より「やわらかいもの」へ向かうことが見て取れる。

人体の動きに関しても発展が望まれる。現在は、モーションキャプチャーを使って、モーションデータを作成するのが主であるが、将来的には、シミュレーションによって動きを生成することが増えるであろう。他にも、筋肉シミュレーション等を導入することで、さらに現実との見分けが付きにくい動きが実現する可能性は高い。

(4) さいごに

以上、駆け足となってしまったが、物理シミュレーションの遍歴について紹介させていただいた。筆者の未熟さのため、日本のビデオゲームが主な対象になってしまったり、抜け落ちていた情報があると思うが、ご了承いただきたい。

執筆に際してソニー・コンピュータエンタテインメントの堀川様、松生裕史様、高橋律視様、櫻井亮介様に協力をいただいた。また、バンダイナムコゲームスの大森 靖、中西 健祐、原田 英樹、松本 正士、手塚 忠孝、大友 健司、谷 史郎、辛 孝宗、髙橋 誠史、福田 強、森井 進の各氏にもアドバイスをいただいた。この場を借りてお礼を申し上げたい。

表 2.3-01　現実の遊びを模倣するシミュレーション年表

現実の遊びを模倣するシミュレーション	
1958 年	『Tennis for Two』での簡易テニスシミュレーション
1979 年	『Flight Simulator』でのフライトシミュレーション
1981 年	『Raster Blaster』でのピンボールシミュレーション
1983 年	『3-D ゴルフシミュレーション』での 3D のゴルフシミュレーション
1988 年	『ハードドライビング』や『ウイニングラン』等のドライブシミュレーション

表 2.3-02　現実に近づけるシミュレーション年表

現実に近づけるシミュレーション	
慣性	
1958 年	『Tennis for Two』での重力加速度シミュレーション
1979 年	『Lunar Lander』での慣性シミュレーション
揺れ物	
1992 年	『インクレディブル・マシーン』での紐シミュレーション
1993 年	『バーチャファイター』での髪の毛シミュレーション
1994 年	『海腹川背』でのゴムロープシミュレーション
1994 年	『バーチャファイター2』でのクロスシミュレーション
流体	
1996 年	『ウエーブレース 64』での高さ場シミュレーション
2002 年	『DOUBLE-S.T.E.A.L.』でのピクセル単位のシミュレーション
2008 年	『AQUA FOREST Powered by OctaveEngine Casual』での 2 次元流体シミュレーション
剛体	
1998 年	『Jurassic Park: Trespasser』での剛体シミュレーション、ラグドール
2008 年	『Star Wars: The Force Unleashed』での破壊シミュレーション

参考文献：

[Erwin09] Erwin Coumans, "AAA Titles using Bullet ?", BULLET PHYSICS LIBRARY, http://www.bulletphysics.org/Bullet/phpBB3/viewtopic.php?f=17&t=3025&start=0&sid=125872fa979e45e5017943b68cf8b62a, 2009.

[Eric09] Eric G. Parker and James F. O'Brien. "Real-Time Deformation and Fracture in a Game Environment". In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 156-166, 2009.

[Kawase02]川瀬正樹, "DOUBLE-S.T.E.A.L.におけるリアルタイム CG 表現技法", CEDEC2002, 2002.
[Sega03]. 株式会社 SEGA "ニュースリリース、『ASTRO BOY 鉄腕アトム』セガ、HAVOK 社の物理シミュレーションエンジンを採用", http://sega.jp/corp/release/2003/1201_2/, 2003.
[Tsuda08] 津田順平, “物理エンジンの作り方”, CEDEC 2008, 2008.
[Zenji08] 西川善司, “西川善司の 3D ゲームファンのための「MT フレームワーク 2.0」講座”, http://game.watch.impress.co.jp/docs/20081204/3dmg4.htm, 2008.
[Zenji09] 西川善司, “西川善司の 3D ゲームファンのための「MT フレームワーク 2.0」講座”, http://game.watch.impress.co.jp/docs/series/3dcg/20090613_283112.html, 2009.

2.4　ミドルウェア（佐藤委員）

佐藤　カフジ（フリーライター）

2.4.1　ミドルウェアとは

現在、ゲーム開発現場においては多種多様なミドルウェアを活用することが常識となっている。ゲームプラットフォームが高性能化し、ひとつのゲームを構成する技術要素が高度化・複雑化した結果、単一の企業だけで全ての技術要素を開発することが困難になったことがその背景である。

本稿でいうミドルウェアとは、ゲーム開発の技術基盤・技術要素を提供する様々な形態のソフトウェアモジュールを指す。その実態は、ゲームエンジン、開発フレームワーク、あるいは特定のサービスを利用するための組み込みモジュールまで幅広く考えられる。

本稿ではメインストリームのゲーム開発に用いられる商用のミドルウェアについて説明する。ミドルウェアが何故必要で(あるいはどのようなときに必要ないのか)、どのような役割があり、どのようなものが利用できるのかといった現状を理解することは、現代的なゲーム開発を知るために有用である。

2.4.2　ミドルウェアの役割

(1)　プラットフォームの抽象化

ミドルウェアの多くは、ゲーム開発における個別の技術要素、あるいはゲーム全体の開発フレームワークを提供するものである。国内外の様々な企業がミドルウェアベンダーとして、ゲーム開発企業を対象にビジネスを行なっている。

近年多くのゲーム開発現場でミドルウェアを必要とする主目的を 2 つ挙げることができる。ひとつはより高度で堅牢なソフトウェアを構築するためであり、もうひとつは開発コストの圧縮である。

最新世代のゲームプラットフォームにおいては､かつての 8 ビットゲーム機のように｢直接ハードウェアを叩いて」ゲームを開発する方法はコストが掛かりすぎ、もはやほとんどナンセンスである。

現実的なゲーム開発を行なうためには、サウンド、グラフィックスといった基礎機能から、AI、物理といったゲーム上の様々なメカニクスにいたるまで、開発対象が何らかの形で抽象化されている必要がある。例えば、マイクロソフトの Direct 3D は、グラフィックスプロセッサーの複雑な機能を「パイプライン」､「ジオメトリ」や「シェーダ」といった体系に抽象化するミドルウェアである。このほか様々な技術要素について、高度に抽象化された環境を提供して開発目的の達成を容易にすることがミドルウェアのソフトウェア工

学上の役割と考えられる。

(2) マルチプラットフォーム化

ミドルウェアビジネスが本格的に立ち上がり始めたのは、2000 年代初頭、プレイステーション 2 世代のゲーム開発が盛んになった時期と重なる。この時期、多くのゲーム会社が 3D グラフィックスの基盤として旧 Criterion Software(2004 年に Electronic Arts が買収)の RenderWare を使用した。マルチプラットフォーム開発に対応した RenderWare は、プレイステーション 2、Xbox、GameCube といった同世代機間のゲーム開発に好都合であった。

この種のミドルウェアは、開発フレームワークと呼ぶこともできる。プラットフォームの抽象化を推し進め、ターゲットプラットフォーム毎の些細な違いを吸収し、容易に他のゲームプラットフォームで動作する版を作成できるようになることが大きなメリットとなっている。

近年ではこの傾向がさらに進み、Windows PC、プレイステーション 3、Xbox 360、Wii、PSP、DS といった性能の全く異なるゲーム機間で、開発リソースを可能な限り共通化しようという試みが行なわれている。この点で突出しているのが米 Vicarious Visions 社の開発プラットフォーム「Alchemy」である。米 Activision はこのミドルウェアを使い、同一タイトルの 7 機種同時開発・同時リリースという実績を残している。

(3) 技術要素の高度化

高度な 3D 表現や物理処理、サウンド、ネットワーク機能といった個別の技術要素については、より早くからミドルウェアの活用が進んでいる。特に本格的な物理処理は大手の開発企業でもインハウス開発が難しい分野であり、本格的な物理処理を導入したほとんどのゲームタイトルが米 Nvidia「PhysX」と、米 Intel「Havok」のどちらかのミドルウェアに依存している状況である。

ミドルウェアビジネスの発展は、こういった専門的な技術がより高いレベルで蓄積されていくためにも良い影響を与えている。潜在的には全てのゲーム開発企業が顧客という状況のなか、各ミドルウェア企業が個別の得意分野にリソースを集中できることがその理由である。この結果、各ミドルウェア企業はますますその得意分野に特化し、後発企業に立ち入る隙すら与えない、という傾向を見ることができる。

またこの結果、極めて洗練された技術要素が広く利用されるようになり、エンドユーザーはその成果を様々なゲームタイトルで見ることができるようになった。マクロ的にみれば、技術発展の成果が極めて安価に流通するようになったといえる。物理や AI といった分野でこれが顕著である。

(4) インハウスのゲーム技術開発との関係

現代的なゲーム開発シーンにおいては、多くの技術要素をアウトソースに頼らざるを得ないが、それと並行して様々な分野でインハウスのゲーム技術開発を持つことも重要である。

特に大手のゲーム企業にとって、基盤となる開発技術の社内蓄積は戦略的に重要と考えられている。例えば、2004 年に米 Electronic Arts が行なった Criterion Software の買収は、Criterion のミドルウェアである RenderWawre を利用していた多くのゲーム開発企業に、ミドルウェアに依存することの潜在的なリスクを突きつけた。大方の予想通り、RenderWare はその後外販が行なわれなくなり、それに依存していた企業は別の手段を見つける必要が生じた。

このような駆け引きもあり、現在、体力のある大手ゲーム企業の多くは、ゲームエンジン技術、3D グラフィックス技術、サウンド技術といった各主要分野において、インハウスの独自技術を持つよう勤めている。その上で、個別の要求に応じて社外のミドルウェアを利用するといったスタイルが一般的である。好例は、米 Electronic Arts の「Frostbite」、英 Codemasters の「Ego」、カプコンの「MT フレームワーク」のような内製ゲームエンジンである。

2.4.3 ミドルウェアの種別

(1) ゲームエンジン型ミドルウェア

ミドルウェアビジネスが現在のように成熟する以前、ゲームにおけるテクノロジーベンダーとして初の大きな成功を収めた例として米 id software を挙げることができる。id software が 1992 年に開発したゲーム『Wolfenstein 3D』は、当時ゲーム技術の傑作と評価され、彼らはいくつかの競合他社に対してそのソースコードを数万ドルで販売することができた。

その後 1993 年の『DOOM』を経て、1996 年『Quake』ではさらにモジュール化されたゲームエンジンが開発され、「Quake Engine」として多数のゲーム企業にライセンシングが行なわれた。これによりゲーム業界におけるミドルウェアビジネスの存在と効果が認識され、すぐさま米 Epic Games による「Unreal Engine」のような追従者を生んでいる。

このように 1996 年から 1990 年代末にかけてのミドルウェアビジネスの中心は、「Quake Engine」に代表されるようなゲームエンジンであった。これはいわば「動作するゲームの一式を提供する」というスタイルであり、ライセンスを受けたスタジオは、まずそのゲームを改造するところからゲームの開発をスタートするような形であった。

(2) 開発フレームワーク型ミドルウェア

ゲームエンジン型ミドルウェアは、その応用が元となるゲームの特長に縛られすぎるという欠点がある。例えば、『Quake』を改造して FPS を製作することは可能であるが、ストラテジーゲームを作ることは難しい。

それでは多くのソフトウェア資産が無駄になってしまうので、もうひとつの方向性として 1990 年代末から主流となったのが RenderWare、Alchemy、Gamebryo といったフレームワーク型のミドルウェアである。広義にはこれらをゲームエンジンと呼ぶこともあるが、開発フレームワークという呼び方が実態をよりよく表す。

開発フレームワーク型のミドルウェアは、ゲームの基本ロジックをベースに 3D グラフィックス、ユーザーインターフェイス、サウンド、物理などの技術要素を総合的に提供し、あらゆるジャンルのゲーム開発を支援する。

実際のゲーム開発においては、個性を出すために個々の技術要素をインハウス技術や他の単機能ミドルウェアに置き換えるという使われ方が多く見られる。このため、開発フレームワークはより柔軟な設計のものが好まれる傾向があり、Emergent Technologies の「Gamebryo Lightspeed」のように、各技術要素をプラグイン式に置き換えられる機能がウリとなっているミドルウェアも出てきている。

(3) 単機能特化型のミドルウェア

3D グラフィックス、アニメーション、動画、ユーザーインターフェイス、サウンド、物理、AI、その他もろもろの技術要素について、単一の機能に特化したミドルウェアは、近年のゲーム開発においてより重要性を増している。

例えば、物理処理の分野では「Havok」、「PhysX」の 2 強が市場を支配しているが、そのいずれも、版を重ねるごとに物理に関する幅広い現象を再現できるようになり、また、最新のハードウェアで効率的に動作する実装を実現するなど、他の企業が 1 から開発して追いつくことが考えられないほどの機能を持つようになっている。

このように単機能に高度化したミドルウェアは多数の企業が利用することになるため、ライセンス単価はそう高額なものとならない傾向がある。このため、比較的小規模なゲーム企業でも、ある特定の部分に高度な機能を用いて競争力のあるゲームタイトルを実現することが可能になってきている。

(4) 教育目的に使われるミドルウェア

近年の傾向として、より多くのユーザーを獲得したいミドルウェア企業は、大学や専門学校などの教育機関向けに無料バージョンのミドルウェア製品を提供することが多くなっている。

例として EPIC Games の「Unreal Engine」は、無料バージョンを配布しており、教育や学習目的の使用について自由に使用できる。また他の例として「Gamebryo」は、北米や中国の教育機関に無料バージョンを提供し、教育カリキュラムへの導入を進めている。

こうしてミドルウェア企業が学生を囲い込むことにより、各ゲーム開発企業がミドルウェアを選定する際、無視できない材料が与えられることになる。しかし本格的な教育機関への導入は始まったばかりであり、その効果は今後数年のうちに明らかになると考えられる。

2.4.4 ミドルウェア・カタログ

ここでは、現時点で一般の国内ゲーム開発企業が使用ライセンスを獲得可能なミドルウェアのうち、代表的なものを種類別に列挙する。各社が独占的に使用している内製ゲームエンジンは含まれない。

(1) ゲームエンジン

表 2.4-01　代表的なゲームエンジン

名称	提供元	対応機種	特徴	主なジャンル
Unreal Engine 3	EPIC Games	Windows / Xbox 360 / PlayStation 3 /	高品質グラフィックス / 高度なレベルエディタ / スクリプトエディタ	FPS 3D アクション
Source	Valve	Windows / Xbox 360 /	レベルエディタ等周辺ツールがオープン/ 拡張性・柔軟性が極めて高い	FPS 3D アクション
CryEngine 2	Crytek	Windows / Xbox 360 / PlayStation 3	高品質グラフィックス / ゲームのランタイム編集 / 独自物理エンジン	FPS

(2) 開発フレームワーク

表 2.4-02　代表的な開発フレームワーク

名称	提供元	対応機種	特徴	主なジャンル
Alchemy	Activision / シリコンスタジオ	Windows / Xbox 360 / PS2 / PS3 / PSP Wii	マルチプラットフォーム開発の高い実績 / ライセンス費が安価で小規模開発にも向く	3D ゲーム
Gamebryo	Emergent Technologies	Windows Xbox 360 PS2 / PS3 / Wii	高い柔軟性 / レベルエディタ / ランタイム編集 / PhysX ビルトイン	3D ゲーム
Unity	Unity Technologies	Windows / Mac Wii / iPhone	Web ブラウザで動作するモジュールを出力可 / 統合開発環境 / 安価	ブラウザゲーム カジュアルゲーム モバイルゲーム
Torque Game Engine	GarageGames	Windows / Mac / Xbox 360 / Wii / iPhone	Web ブラウザで動作するモジュールを出力可 / 極めて安価	ブラウザゲーム カジュアルゲーム モバイルゲーム
BigWorld	BigWorld Technology	Windows	MMOG 開発・運営のための完全なテクノロジースイート	MMOG

(3) 単機能ミドルウェア

表 2.4-03　代表的な単機能ミドルウェア

名称	提供元	機能	特徴	主なジャンル
PhysX	NVIDIA	物理処理	NVIDIA CUDA 対応 / 各プラットフォーム最適化 / 無料版あり	3D アクション 各種シミュレーター
Havok	Intel	物理処理	極めて豊富な採用実績 / 姉妹 AI エンジン、アニメーションエンジンと協調	3D アクション 各種シミュレーター
Kynapse	Autodesk	AI(経路探索)	経路探索を高速に処理 / 環境の変化にリアルタイム適応	3D アクション ビジュアルシミュレーション
HumanIK	Autodesk	アニメーション	ボーンアニメーションのリターゲッティング / UE3 に統合可能	3D ゲーム
BINK Video	RAD Game Tools	動画コーデック	高速 / 高画質 / 低ビットレート / 無数の採用実績	ジャンル問わず
YEBIS	シリコンスタジオ	リアルタイム・ポストエフェクト	プログラマーなしにグラフィックス品質を調整 / Alchemy との親和性高	3D ゲーム
VCE	コミュニティーエンジン	ネットワーク	サーバー/クライアント型ネットワークゲームのフレームワーク	MMORPG
SpeedTree	IDV	植生エンジン	植生自動生成/ 手動編集 / ランタイム表示エンジン / 高品質な映像表現	3D ゲーム
ScaleForm GFx	ScaleForm	UI エンジン	Adobe Flash で作成した UI 要素をゲームにインポート / 動画再生支援	3D ゲーム 2D ゲーム
ファイルマジック PRO	CRI ミドルウェア	ファイルシステム	透過的にゲームデータの圧縮・復号を支援 / ファイルシステム最適化	ジャンル問わず
Dolby Axon	Dolby	ボイスチャット	3D 空間の角度、距離、環境要素を考慮してボイスチャット音声を再現	3D マルチプレーヤーゲーム

2.5　シナリオ

すずき　こういち（コンテンツデザイナー）

2.5.1　ゲームにおけるシナリオの意味について

シナリオとは、元々舞台で上演される演劇の脚本に対して、テレビや映画など映像で表現される劇の脚本の事を、区別するために使われてきた用語である。従って一般的にシナリオという呼称を用いた場合、そこには映像表現を前提とした脚本という含意がある。

ゲームシナリオという新しい用語は、ゲーム中における物語的な要素を広義に捉えており、さらに一般的なシナリオと大きく異なる点として、その役割が物語を表現するための脚本の域に留まらず、広くゲームデザインに係る領域まで及んでいる事が挙げられる。この章では、まずそれらについて述べてみたい。

(1)　感情移入を促すためのシナリオ

(a)　ゲームの背景や状況設定を理解させるためのバックストーリー

プレイヤーがゲーム世界に感情移入する事を助けるために用意されたバックストーリーや、それを感じさせる環境要素としてのゲームシナリオは、ゲームのジャンルを問わず古くから用いられてきた手法である。

例えば下記に挙げるタイトルは、ビデオゲーム黎明期に登場した単純な反射神経型作品ではあるが、そこには既にプレイヤーがゲーム世界に没入するための仕組みとしてのバックストーリーが作成されている。

①　スペースインベーダー(1978 年 タイトー)

侵略してくる宇宙人を砲台で迎え撃つ、ビデオゲーム黎明期の代表的なシューティングゲームである。

敵キャラクターが「侵略者」である、というバックストーリーがあり、そのストーリーを補強する演出として、一定のルールに基づいて画面上を横切るボーナスキャラクターは侵略者(インベーダー)の母船とも言える UFO として描かれている。

これらのバックストーリーの存在は、極めてシンプルな物語要素であるとはいえ、ブロック崩しをはじめとする従来のビデオゲームとスペースインベーダーとを隔てる、大きな違いの一つと言える。

② ドンキーコング(1981 年 任天堂)

マリオを操作して、ドンキーコングに掠われた恋人を助けに向かうアクションゲームである。

この作品においては、明確なキャラクター性を有した主人公（マリオ）と、恋人、悪役、というプリミティブな人間関係を持った登場人物達がゲームを支えるバックストーリーとして組み込まれている。掠われた恋人を取り戻す、というストーリー上の主人公の動機がゲームの目的と合致しており、プレイヤーはそのバックストーリーを意識する事で主人公キャラクターに容易く感情移入する事ができる。

これらの場合、ゲームシナリオの役割はあくまでもプレイヤーの置かれた状況と目的を物語に沿う形で明確にする事であり、

- 自分（プレイヤー）はなぜ、この場所にいて
- なぜ、クリア条件を満たさなければならないのか

といったゲーム世界の前提をプレイヤーに納得させ、プレイヤーのモチベーションを高める役割を担っている。

ここでのゲームシナリオとは、ゲーム世界の状況設定とプレイヤーの目的に説得力を与えるための補助的な存在に過ぎない。だが単純なアクションゲームであったとしても、そこに適切なバックストーリーが加えられることによって、ゲーム世界に没入するための世界観を具体的に想像させ、プレイヤーのゲームクリアに対するモチベーションを高める事に成功したという点において、ゲームシナリオの果たした役割は大きいものであったといえるだろう。

(b) バックストーリーの複雑化

ゲーム性に直接作用しないバックストーリーとしてのゲームシナリオは、その内容や精度によってゲームバランスが影響を受けるものではないため、当初ゲーム開発のコストとしては極めて低く、導入も容易と見られていた。

そのため、バックストーリーはパブリシティ材料として利用される事が多く、1980 年代前半の PC ゲーム初期に於いては、単純なアクションゲームに著名な映画タイトルを冠しただけの作品も少なからず存在していた。

綿密に構築されたバックストーリーは、ゲームの世界観を深め、プレイヤーにとって、自分の操作するキャラクターへの愛着を高める効果がある。そのため、キャラクター性そのものが重要視される傾向にあるアクションゲーム、格闘ゲームなどのジャンルにおいては、世界設定の複雑化、バックストーリーの肥大化が進む傾向があった。

『ストリートファイター』(1987 年 カプコン)、『バーチャファイター』(1993 年 セガ)

などは息の長いシリーズ作品であるが、キャラクターや世界観が魅力的に描かれ、シリーズにおいては原則としてその世界観が継承されている。

『アテナ』(1986、 SNK)ではアーケードゲーム版の翌年にリリースされたファミコン版において、ゲームソフトにアテナのテーマソングのカセットテープを付属させるなど、主人公キャラクターと、その世界観を全面的に押し出す試みも行われた。同タイトルはその後、プレイステーションで『ATHENA -Awakening From the Ordinary Life-』(1999、SNK)としてアドベンチャーゲーム化、同時にテレビドラマ化もされている。

また、『ゼビウス』(1983、namco)は縦スクロール型のシューティングゲームでありながら、ファードラウト・サーガと呼ばれる本格 SF 調のバックグラウンドストーリーを小説として発表(電波新聞社マイコンベーシックマガジン 1983 年 12 月号に掲載)、さらに世界観に存在感を与える工夫として、作中世界の独自言語を設定するなど本格的な世界設定を試みたタイトルとして知られている。ファードラウト・サーガは後に単行本として出版され、その重厚なバックグラウンドストーリーはアニメ映画としても公開された。

この様に、複雑化したバックストーリーが本来のゲームから独立した形で発展し、他メディアに展開されるというケースは 80 年代から少なからず存在していた。コンシューマーゲーム機において、ゲームデザインとは直接関係のない世界設定やバックストーリーとしてのシナリオが複雑化する傾向は、80 年代後半に家庭用ゲーム機に CD-ROM が搭載され、音声や映像による表現の幅が広がった事で、一層加速したと考えられる。

(2) ゲームデザインとしてのシナリオ

ゲームにおいて、プレイヤーキャラクターの取り得る行動のすべては、そのゲーム世界の根幹を成すルールに基づいて決定されている。ゲームのルールに対してストーリー上の理由を付与する事で、プレイヤーはルールを物語の文脈として理解する事が可能となり、ルールの説得力もまたストーリーをエビデンスとして持つことで、プレイヤーにとって納得のしやすい形で補強される事になるのである。

この手法においては、ゲームのクリア条件やプレイヤーの行動規範等をストーリー上の理由として規定する事が可能なため、シミュレーションゲームやストラテジーゲームなどに多く採用される方法論といえるだろう。

(3) 目的としてのシナリオ

昨今のアドベンチャーゲームやロールプレイングゲームにおいてゲームシナリオの果たす役割は大きく、そのシナリオを読み解き、物語を体験する事自体がゲームプレイの目的となるケースも少なくない。換言すれば、ゲームの目的としてのシナリオであり、概ね現在一般的に言われるところゲームシナリオとはこのパターンを指す場合が多い。

殊にテキストで綴られた物語を読む事を主軸とし、謎解きやパラメーター上げといった所謂ゲーム的な操作を求めない作品群は通称ノベルゲームと呼ばれ、国内黎明期の代表的な作品としては『DOME』(1988、システムサコム)、『弟切草』(1992、チュンソフト)、雫(1996 年 リーフ)などが挙げられる。

謎解きよりもストーリーを楽しむ、といったプレイスタイルは、美少女アニメブームと相乗する形で流行したアダルト系ゲームを中心として広がり、やがて一般向けの ADVG、RPG にも取り入れられるようになった。特に、謎解きが基本のゲーム性となる ADVG ジャンルでは、トリッキーなギミックや理不尽とも成りかねない難度競争の過渡期にあり、そうしたヘビーユーザー向けのゲームバランスに付いていけないライトユーザーにとって、「ボタンを押すだけでストーリーが進む」ノベルゲームは、大いに歓迎されたのである。

2.5.2 ゲームシナリオの現場とその変遷

この章では、ゲームシナリオがゲーム開発の現場に於いてどのように扱われてきたか、どのような位置づけであったのかを、ADVG、RPG のジャンルを中心に時代を追って解説する。

(1) 舞台設定としてのシナリオ(80 年代以前)

ゲームにシナリオという概念が持ち込まれた当初に於いて、その役割の殆どは第 1 章で述べた「感情移入を促すためのシナリオ」に類するものであり、ゲームの舞台やプレイヤーキャラクターの演じる役割に意味づけするための「設定」であったと言えるだろう。厳密に考えた場合、これらのゲームシナリオは脚本としての体を成していないため「シナリオ」と呼ぶのは適切ではない。ゲームシナリオの黎明期において、それらはあくまでも「舞台設定」、「キャラクター設定」の域を出るものではなかったのである。

ADVG、RPG では、ジャンルの性質上、初期の頃より物語性をゲーム内に取り込む事が主眼とされてきた。しかしながら、ADVG 初期の代表作品とも言える ZORK(1980 年 Infocom)シリーズにおいても、そのゲームシナリオは主人公の置かれた状況を巧みに描出しているとはいえ、「ト書きによる状況説明」以上のものではなく、緻密に描かれた舞台設定に過ぎなかった。

それは RPG においても同様であり、

- どのような世界で
- どのような NPC(non player characters)が存在し
- プレイヤーキャラクターは何を目的とするのか

といった大枠のゴールの設定と、舞台設定のみがゲームシナリオの担う役割であったと言えるだろう。

これは、80 年代前半における ADVG のゲーム性がパズル的な要素を多分に含んだ謎解きに主眼を置いていた為であり、さらに RPG のゲーム性がプレイヤーの自由度の高さを目指していた（RPG はそもそもテーブルトーク RPG の模倣から始まっているため、プレイヤーがキャラクターの役柄=role を自由に演じる=play 事がベースデザインとなっていた）事に加え、ハードウェアの記憶容量の制限からゲーム中に表示できるメッセージ量にも自ずと限界があり、その限られたメッセージ量の中では複雑な物語を描くこと自体が不可能であった事が要因として挙げられる。

従ってこの段階のゲームシナリオのライティングにおいては、所謂作劇法のようなシナリオ技術は必要とされず、むしろキャッチーなアイデアやテーマ性が重要視されていたと言えるだろう。

(2)　シナリオによるドラマの表現(80 年代)

やがて、ハードウェアの性能向上に伴いゲームの表現力が増すに従って、ゲームシナリオの中にドラマ性を取り入れようとする流れが盛んになる。

ADVG において、『ポートピア連続殺人事件』(1983 年、エニックス)、『惑星メフィウス』(1983、T&E ソフト)などの作品は十分な物語性を有しており、読み物としても完成度の高いストーリーをゲームシナリオとして制作している。

また、RPG では『ハイドライド』(1984 年、T&E ソフト)がアクション RPG という従来の RPG とは全く異なるゲーム性を提示すると同時に、プレイヤーの自由度が限られるものの、ストーリー性の強いドラマチックな RPG という新しいメソッドを打ち出した。

前項で述べたとおり、RPG というジャンルにおいては、その出自から自由度の高さを謳う事が主流であったが、その反面ドラマ性の強いシナリオを実装する事が困難でもあった。ハイドライドが提示したスタイルは通称「一本道 RPG」とも呼ばれ、その後代表的な RPG のスタイルの一つとなって行く。

これらの水準に至るゲームシナリオを制作するためには、開発者にはある程度の物語を構築する作劇法と、実際にシナリオをライティングする技術が求められる。従って、前述したポートピア連続殺人事件は制作者である堀井雄二氏がシナリオを含み一人で開発を行っており、惑星メフィウスを開発した T&E ソフトでは開発の様子を「本筋だけ（ディレクターである横山英二氏が）決めといて、あとはディスカッションで細かい部分を煮詰めていくんです」[1] とインタビューで述べているとおり、同時期における開発現場においては、メインの開発者が一通りそれらのスキルを身につけている必要があったのである。

(3)　原作付きシナリオと、オリジナルシナリオ

ゲームシーンにおいてドラマ性を持ったシナリオが求められるようになり、開発者にはある程度のシナリオライティング技術が求められるようになった。さらに、前項で参考例としたポートピア連続殺人事件や惑星メフィウスも開発者によって書き起こされたオリジナルストーリーであったが、このように物語を最初から組み立て得るストーリーメイキングの技術を持ったスタッフを抱えるメーカーは、一部に限られていたと言えるだろう。

そこで、ストーリーメイキングの負荷を減らしつつドラマ性のあるシナリオを謳うために、アニメ、映画、コミック、小説など、他メディアで実績のあるストーリーを原作とし、アレンジを加えた作品が制作された。

原作付きのアドベンチャーゲームとしては、『ゴルゴ 13』(1983、PONYCA)、『機動戦士ガンダム』(1983、ラポート)、『めぞん一刻』(1986、マイクロキャビン)などの人気アニメ、コミック作品や、『さよならジュピター』(1983、PONYCA)、『原田知世の時をかける少女』(1984、PHOENIX)など、映画作品を題材にしたものが数多くリリースされた。

ベースとなるストーリーを原作に依存する事で、メーカーはゼロベースで物語を組み立てるプロセスから解放され、ゲームデザインの設計に注力する事ができる。つまり、ストーリーメイキングの技能を習得していない、あるいはシナリオ技術の未熟なプランナーであっても、ある程度ストーリーオリエンテッドなゲームシナリオを書くことが可能となったのである。

その一方でドラマ性を持ったゲームシナリオを志向し「ストーリーの書けるプランナー」の所属するメーカーは、オリジナルストーリーによる ADVG、RPG を意欲的にリリースし、「シナリオに強い会社」というポジションを作り出して行った。

鍵穴殺人事件(1983 年)をはじめとする推理 ADVG を得意としたシンキングラビットや、ザ・デストラップ(1984 年)やウィル(1985 年)などで映画的な演出を好んで取り入れたスクウェアなど、メーカー毎にいわゆる「作家性」とでも言うべき特色が打ち出されていた事が、この時代の傾向として挙げられるだろう。

(4)　ゲームボリュームの増大に伴うドラマツルギーの複雑化

ゲーム機のハードウェア性能が向上しゲームのボリュームが肥大化するに伴って、ゲームシナリオに求められるストーリーは、より大きく、複雑なものとなって行った。また、映像や音声の表現力が格段に進化したことで、演出面においてもより深みのある人間ドラマや、説得力のあるストーリーが必要とされるようになった。

家庭用ゲーム機に初めて CD-ROM を搭載した PC Engine CD-ROM²(1988 年)の登場を境に、従来の「シナリオの書けるプランナー」の技術水準では書ききれない規模のシナリオ、複雑なドラマツルギーを必要とするシナリオが要求されるようになり、専門的なシナリオ技能を習得したプロの作家の参加が広く求められるようになったのである。

他方、「シナリオの書けるプランナー」ではなく「ゲームのシナリオを書けるシナリオライター」の重要性に早くから気づき、人材確保に積極的であった開発会社も存在した。一例を挙げれば 1985 年の月刊ドラマ(映人社)には、エニックスによるゲームシナリオライター募集の公告が掲載されており、シナリオライター畑の人材からゲームシナリオのスタッフを採用しようとしていた事が窺い知れる。

(5)　ストーリーのためのゲーム(90 年代)

従来、あくまでもゲーム世界の雰囲気作りのための舞台設定に過ぎなかったゲームシナリオが複雑なドラマツルギーを持ち、物語としての完成度が高まるにつれて、やがてシナリオを楽しむことをゲームの目的そのものとした、新しいスタイルの ADVG が誕生する。

1990 年代に台頭するノベルゲームの先駆けともいえる、システムサコムのノベルウェアシリーズでは、原作者に夏樹静子氏、多摩豊氏といった SF 界の重鎮を起用し、ゲームデザインにおいてもゲームオーバーにつながるような選択肢を極力廃し、簡単な選択肢を進めていくだけで物語を楽しめるような設計が成されていた。このような系譜はその後、エニックスのサウンドノベルや、アクアのビジュアルノベルといった作品群に受け継がれ、物語を楽しむ事を主眼としたゲームとして、独自のスタイルを確立して行く。

ノベルウェアシリーズは、そうしたストーリーオリエンテッドなゲームデザインに適した開発スタイルを早期に採用していた事も特徴の一つとして挙げられる。それまでのゲームシナリオは画面上に表示されるテキストをプログラム内に直接埋め込む方式が主流であり、シナリオを担当するプランナーがプログラマーの指示に従う形でテキストをライティングするケースが多かった。しかしノベルウェアシリーズにおいては、社外のライターをシナリオライターとして迎え、彼らの執筆するストーリーを社内のプランナーがシナリオスクリプトと呼ばれる中間言語でシステムに記述して行く、という方法をシリーズの最初から採択し、専用のスクリプト言語を設計、実装していたのである。

(6)　プランナーに新たに求められたもの(90 年代後期～)

シナリオスクリプトの登場によって、ゲームシナリオの自由度は開発の行程において飛躍的に高まった。その結果、外部の作家を起用するタイトルも格段に増えたものの、開発現場は新たに「ゲーム畑の外のクリエーター」とコミュニケーションを取るスキルを求められる事となった。従来は、いわば「学生サークル気分」で開発が可能であったゲーム開発の現場に、全く異なる次元でのチームプレイが要求される事になったのだ。

同時期、ゲームという存在に興味を抱いたゲーム畑以外のクリエーター達が、積極的にゲームとの融合を図ろうとした事もその傾向に拍車を掛ける。90 年代後期以降のゲーム開発現場において、社外のスタッフを全く使用しないタイトル（完全内作タイトル）は、一

定規模以上のタイトルであれば一部の例外を除きほぼ存在しなくなったと言っても過言ではないだろう。近代ゲームタイトルを制作するためには、そうした様々な外部クリエーターとの円滑なコミュニケーションと、コラボレーションを図れるスキルが必要とされるようになったのである。

(7)　メディアの大容量化とゲームシナリオの変化

上に述べたゲームシナリオの時代変化は、それぞれの転換点においてゲームメディアの大容量化を伴っている。メディアの大容量化がゲームシナリオにとってどのような影響を及ぼしたのかを、この項では述べてみたい。

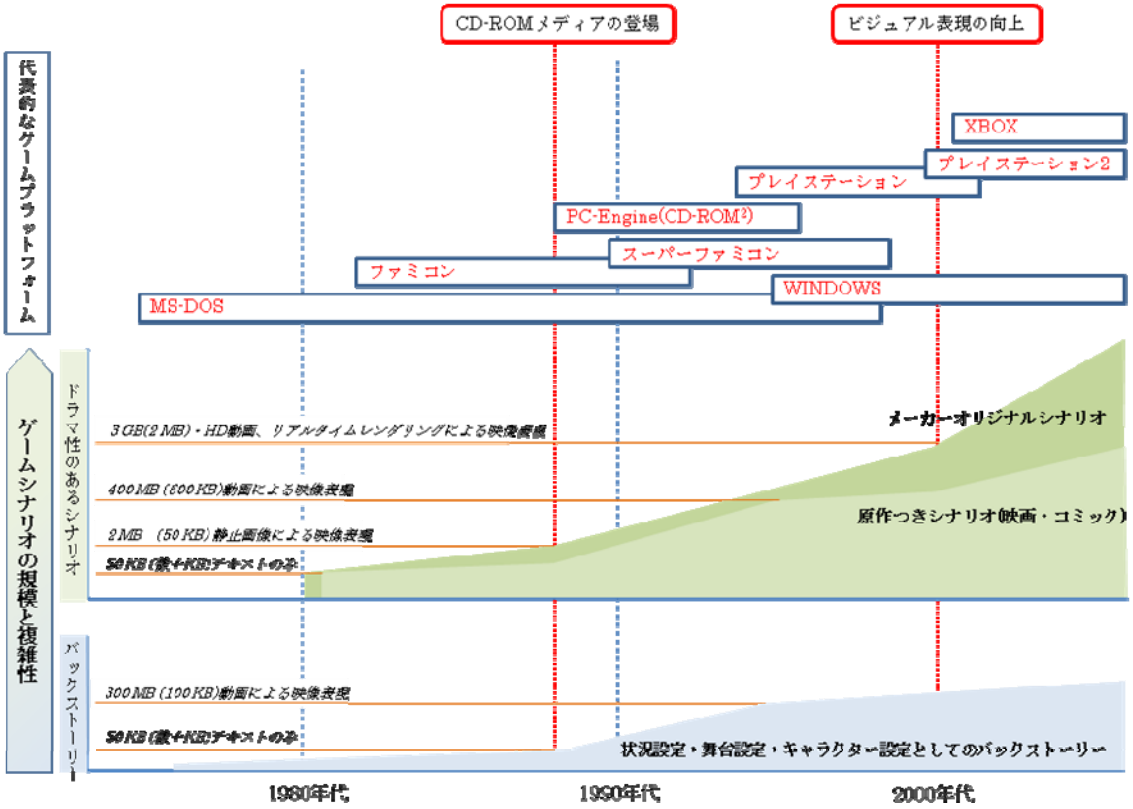


図 2.5-01 ゲーム中におけるゲームシナリオの規模と複雑性の変遷

容量表記は、「全体シナリオ量（画面に見えるテキスト量）」とした。

「全体シナリオ量」とは、ここでは「デモ用の素材やムービーなどを含む大凡のゲームシナリオ用に割り当てられた容量」を指す。「画面に見えるテキスト量」とは、実際に画面に表記されユーザーから読み取られるテキストの量である。両者とも、タイトル依存が大きいが、その時代の大型のシナリオを擁するタイトルの代表的な容量を目安として表記した。

図 2.5-01 では、メディアの進化とシナリオボリュームの関係を図示した。縦軸には、デモ用の素材やムービーなどを含む大凡のゲームシナリオ用に割り当てられた容量と、シナリオ実装の一例として画面上に表示されるテキスト量を併記している。(ムービーの圧縮レートやイベントシーンの実装方法によってゲームシナリオの表現に必要とされる容量は変化するため、ゲームシナリオを演出するイベントの総尺と、上記の容量とは必ずしも一致しない事に留意して頂きたい)

上図に見られるように、80 年代以前まで主にバックストーリーの表現として用いられてきたゲームシナリオは、80 年代前半からドラマ性のあるゲームシナリオを楽しむ形態が発生し、80 年代後半の CD-ROM メディアの登場を契機に、爆発的な表現力の向上と、それに伴うゲームシナリオの複雑化へと推移する事となる(バックストーリーについてはゲーム外で語られるシナリオは除外し、ゲームプレイ中にプレイヤーの目に触れる範囲でのゲームシナリオとした)。

ゲーム中にバックストーリーを語る場合、それらはゲームのオープニングやステージクリア時のデモに含まれる事が多く、容量的にはチープなものであった。 数十キロバイトからせいぜい数百キロバイトのテキストと、数枚の静止画での演出が大半であったといえる。

それが、CD-ROM メディアの登場によってオープニングムービー、ステージクリアムービーという形で表現の幅が一気に膨らんだことで、各メーカーは挙って有名声優を起用した OP ムービー、イベントムービー（ステージクリアムービー)、ED ムービーの制作に力を注ぎ始める。しかし、ゲーム本体に組み込む事ができるバックストーリーにはそれ以上の大きな進化が見られることもなく、バックストーリーのためだけに CD-ROM 数枚分の容量を割くようなタイトルも寡聞にして聞かない。DVD メディアの台頭により動画品質の向上は見られるとはいえ、シナリオの複雑化、肥大化に劇的な変化を与えたとは考えづらい。

一方、ゲームシナリオ自体がゲーム性に直結するジャンルでは、CD-ROM メディアの登場による情報量の増加に伴い、ゲームシナリオで扱えるシーン数、例えば分岐ルートの数や、深さ、マルチエンディングの種類なども劇的に増大した。その結果、ゲームシナリオの構造は複雑化し、ゲームシナリオに求められるドラマも複雑なものとなっていった。

より多くのストーリーバリエーションが好まれ、より長いプレイ時間が要求された RPG、ADVG のジャンルにおいて、記憶メディアの容量増加は、そのままゲームシナリオの規模拡大を促したと考えてよいだろう。

CD-ROM、DVD-ROM の登場による影響を考察するに、メーカーオリジナルのゲームシナリオはその都度用意されたメディアの最大限を活かしきろうと、より長大なストーリーを生む傾向に進んだと言える。一方、原作付きのゲームシナリオでは、ルート分岐、エンディングの分岐などストーリーのアレンジに制約がある事から、膨大なメディア容量はもっぱらムービー画質の向上といった「原作とゲームの差を埋めるための要素」に割り当

てられる場合が多かった。従って原作シナリオ重視の作品においてゲームシナリオの複雑性やシナリオの規模は決してオリジナルのそれを大幅に超えることはなく、大容量メディアの普及に伴い頭打ちとなっていった。

いずれにしても、ゲームシナリオの表現をムービー再生に依存していた 2000 年代以前は、どれだけのテキストと映像と音声をゲームに搭載できるか、といった素材のボリュームにおいてゲームシナリオの規模と複雑性が決定されるケースが多かったと言えるだろう。

2000 年以降、リアルタイムレンダリングによる非ムービー再生型のドラマ表現が可能になった事で、ゲームシナリオのボリュームがメディア容量のみによって制限される事例は極めて少なくなってきたと考えられる。将来的には、音声合成技術による動的なキャラクターボイスの生成や、プロシージャル生成された舞台背景、さらには自動生成された台詞やストーリーなどが登場する事も十分あり得るだろう。それらプロシージャルコンテンツに一層のリアリティが加わることで、いずれはプレイヤーの行動をインタラクティブに反映して動的に生成されるルート分岐や登場人物との対話といった、即興劇のような無限の深さをもつゲームシナリオが実現される可能性もある筈だ。

2.5.3　ゲームシナリオのスクリプト

この項では、実際にゲームシナリオがどのように実装されてきたのかを時代を追って解説する。

(1)　シナリオスクリプト以前(80 年代)

1980 年代における初期の RPG や ADVG では、プログラム中のデータ領域に直接テキストデータを埋め込む事でシナリオを実装していた。

例えば RPG におけるシナリオデータの制作プロセスとしては、

①プランナーによるイベントの決定
②プログラマーによるイベントの実装
③プランナー(ライター)によるイベント用テキストの作成
④プログラマーによるイベント用テキストの実装

といった手順を踏むことになる。

イベントの発生条件などを仕様上策定するのはプランナーの役割だが、実際の実装や細かいタイミングの制御はプログラマーにゆだねられており、そこで表示するテキストの分量や内容なども原則としてプログラマーからの指定に基づいてライターがデータ化するケースが多かった。

シナリオライターからのテキストは、プレーンテキストや、場合によってはワードプロセッサで記述した原稿を紙にプリントアウトしてプログラマーに引き渡す、といった極めて原始的な方法で行われる場合もあった。

また 1980 年代前半のテキストアドベンチャーでは、数十 KB というメモリ容量の厳しさから、ゲーム中に表示されるテキストデータをそのままの形で実装する事ができず、辞書方式と呼ばれるスタイルが採用される事もあった。

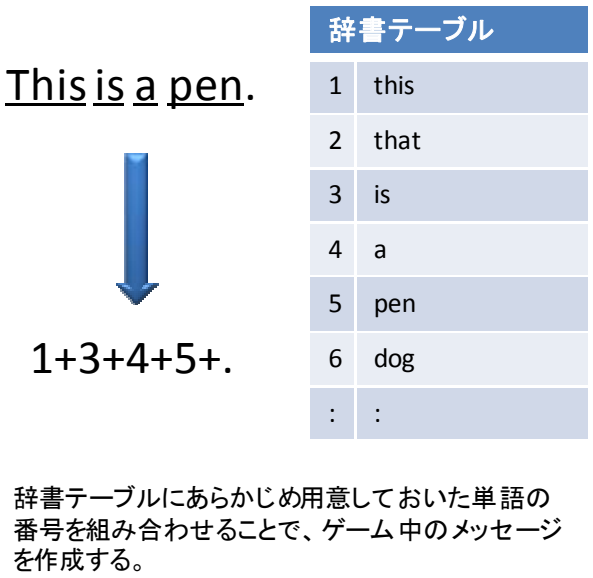


図 2.5-02　辞書方式によるメッセージの格納例

こうした実装スタイルにおいて、シナリオデータとプログラムは不可分の関係にあり、開発終盤においてデータの修正（文字の修正）などを行う場合には、プログラマーに作業を戻す必要があった。そのためシナリオデータの繊細な調整は容易いものではなく、ある程度のプログラマーとの共同作業をやむなくされていたと言っても良いだろう。

(2)　シナリオスクリプトの登場(80 年代中期～)

ゲームボリュームの増大に伴って、ゲーム中のすべてのシナリオデータをメインプログラマーが一括して管理する事はやがて困難となる。複雑化するシナリオと、膨れあがるシナリオデータをスムーズに実装するためには、大きく二つのアプローチがある。

(a) シナリオ専属のプログラマー（イベントプログラマー）を用意する
(b) プランナーがコーディング可能なマクロ言語（シナリオスクリプト）を最初に開発し、プランナーがシナリオスクリプトを使ってゲームシナリオをコーディングできるようにする

後者は、イニシャルコストが非常に高く、デバッグ環境を含めたスクリプト用の開発環境を構築しなくてはならないというデメリットがあったが、メインプログラマーが低レベルな開発に専念できること、また、マルチプラットフォーム化が容易い事などから、開発力のあるメーカーでは積極的に採用される事となった。また、開発環境の構築に関しては、後述する既存開発言語の流用などで効率化を図る試みも行われた。

(3) シナリオスクリプトの実際

シナリオスクリプトの形式は各メーカー独自のスタイルを持ち、その具体的な実態は詳らかではない。しかし基本的な考え方としては、

(a) プランナー（あるいはシナリオライター）の学習が容易い事（複雑なシンタックスを持たず、非プログラマーに理解しやすいこと）
(b) 独自コンパイラによって中間言語にコンパイルされ、シナリオ駆動モジュールはその中間言語を逐次解釈し実行すること
(c) テキストのデコレーション、背景画像やキャラクター画像の制御、サウンドの制御、選択肢の表示、プレイヤーの入力による分岐など、ゲームシナリオに係る一通りの動作を記述可能で、プランナー（あるいはシナリオライター）はデバッグ環境上で実際の動作の確認、および修正が簡単にできること

などが、共通した設計思想にあったのではないかと考えられる。特に項目 c においては、ゲームの演出やプレイヤーの操作に伴う分岐処理など、ゲームの進行に係る制御部分をプログラマーから切り離し、プランナーがゲームシナリオに対して大幅な主導権を持つことを可能にしたという点において、従来のワークフローを劇的に変化させる契機となったと言える。

各メーカーが自社のラインナップに相応しい独自言語を開発する事が多かったが、80 年代後半から 90 年代初頭、MS-DOS 上で開発されていた RPG のゲームシナリオ等において、スクリプトとしてより複雑な構文が必要とされる場合には、MASM(Microsoft Macro Assembler)等のアセンブリ言語のマクロ定義を利用してシナリオスクリプトを設計し、アセンブラによって出力されたバイナリーコードを中間言語として用いるといった省力化が採り入れられる場合もあった。MASM のような既存のアプリケーション開発言語をスクリプト記述のプラットフォームに採用する事で、デバッガーなどの開発支援ツールを流用し、開発効率の改善を図ることが可能となったのである。また、ごく一部の大手メーカーを除いて少人数での開発を余儀なくされていた開発現場においてスクリプトシステムをフレームワークから設計、開発することは困難であったため、プログラマーが低レベルでの実装に専念するためにも、既存開発言語の流用は効果的であったと考えられる。

スクリプトによるゲームシナリオの実装という概念は、90 年代以降のいわゆる同人 PC ゲームの台頭とともに広く一般にも認知されるようになったと思われる。代表的なものでは、HSP(ONION software)、NScripter(高橋直樹氏)、吉里吉里(W.Dee)などがあり、それらはノベルゲームを中心とした同人 PC ゲームや、一部商業ソフトでも採用されている。

(4) スクリプターとライターの分業

前項では、スクリプトの登場によるプログラマーとプランナー（シナリオライター）の分業について解説した。しかし、シナリオスクリプトによってプログラマーとプランナーの分業が可能になったとはいえ､あくまでもプログラム言語であるシナリオスクリプトは、外部のシナリオライターや作家にとって扱いやすいものではなかった。また社内のプランナーとプログラマーの間では容易な情報共有も、外部スタッフとの間で同等の連携を図る事は当時の情報インフラの未熟さも加え非常に困難であった。

そこで、外部の作家が書き下ろしたテキストや紙ベースのフローを社内のプランナーがスクリプト言語に書き起こす、という行程が登場した。そのようにスクリプトを専門に取り扱うプランナーの事は、スクリプターと呼ばれる。

外部の作家は、細かい台詞の分岐や繰り返し処理、例外処理などについては原則として関与せず、ゲームシナリオの大筋の部分、大きな分岐部分に限ってライティングを行う。

社内のスクリプターは必要に応じて作家と打ち合わせを行いながら、より具体的なテキストデータにシナリオを落とし込んでいく、といった分担である。外部の作家は物語に対して責任を持ち､スクリプターはそれをゲームシナリオとして演出するというスタイルは、スクリプトを介してゲームシナリオを実装する現場に於いて、もっとも多く見られたのではないだろうか。

それまでも外部作家を起用した作品は存在していたが、スクリプター登場以前はプログラマーが外部作家の意向を汲み取って作品に反映させる必要があったため、まったくプログラム的な素地のない作家との間では、適切なコミュニケーションを期待することは難しく、外部の作家が最終的なゲームシナリオに深く関わる例は極めて少なかったと考えられる。プログラマーとライターの間にスクリプターというポジションが置かれたことで、最終的なゲームシナリオに作家の趣向をより細密に反映させる事が可能になったのである。

2.5.4 これからのゲームシナリオ

2010 年現在、家庭用ゲーム機ではハイデフ化が当然となり、もはやプリレンダリングによる高画質ムービーの再生も陳腐化しつつある。現状のハードウェア性能を見るに、ゲームシナリオと通常のシナリオにおいて、映像と音による表現力の差違は殆どないと言って良いだろう。映像脚本としてのシナリオのサブセットであったゲームシナリオは、いよい

よ本来のシナリオの意味を超えて、新しい表現へと進化する必要がある。それは、ゲームシナリオでなくては表現できない物語の誕生であり、新しいメソッドによるドラマツルギーの発明であるはずだ。

人がドラマに要求するものは、極言すれば「感動」である。高解像度の映像表現や外連味だけの演出といった技法に終始するのではなく、ゲームシナリオでしか表現し得ない感性とそれによる新しいドラマが、今まで誰も体験したことのない「感動」を生み出すことを期待して止まない。

参考文献：

[1] T&E ソフト:　"惑星メフィウスはこうして作られた" pp.52-53, (1984)

2.6　ユーザーインターフェイス（ソフトウェア）

サイトウ・アキヒロ（立命館大学映像学部教授）
小野憲史（ゲームジャーナリスト）

2.6.1　本節について

本節の考察は、著者の独自の調査研究によるものです。掲載されている各製品発売メーカーの公式の見解とは異なる場合がありますが、ご容赦ください。

2.6.2　ゲーム UI の特性

テレビゲームのプログラムは、入力・内部処理・出力が 60 分の 1 秒、または 30 分の 1 秒単位で循環するという、非常に高速なループ構造をとる。この時にプレイヤーとゲーム機の間で、入力と媒介の手段となるのがゲーム UI（ユーザー・インターフェース）である。

ゲームのおもしろさは、ループ構造の過程で発生する。この過程でプレイヤーに心理的な情動をかきたてる仕組みがゲームデザインである。ゲーム UI はゲームデザインの邪魔をしてはならない。そのためにはゲーム UI は空気のような存在であることが求められる。

これを実現するには、ハードウェアとソフトウェアのチームが密接な協力体制で開発する必要がある。既存のハードウェア上で開発するには、ハードウェアの特性を十二分に生かしたソフトウェア開発が重要である。

2.6.3　入力デバイスの種類

テレビゲーム機においては、ボタン・パドルコントローラ・ジョイスティック・アナログスティック・キーボード・マウス・十字ボタン・マイク・カメラ・タッチペン・モーションデバイスなど、多種多様な入力デバイスが存在する。UI デザインにおいては、これらの特性を十分に理解し、画面デザインなどを行うことが重要である。

2.6.4　UI から見たゲームの歴史

ゲーム機の歴史は、アーケードゲームや PC ゲームで培われた UI を、コンシューマゲームが取り込んできた過程だと捉えられる。

アーケードゲームや PC ゲームでは、コンシューマゲームより多彩な入力デバイスや、より高速な CPU、大容量なメモリを搭載でき、さまざまなチャレンジができた。ここで培われた UI は、標準コントローラーとロースペックという制約を抱えたコンシューマゲー

ム機に移植される過程で、主にソフトウェア面から、さまざまな工夫がなされた。

コンシューマゲーム機もモデルチェンジを繰り返しながら、新しい遊びを生む UI を創造すべく、ハードウェアとソフトウェアの双方で、さまざまな工夫がなされていった（企業名は原則として当時のもの）。

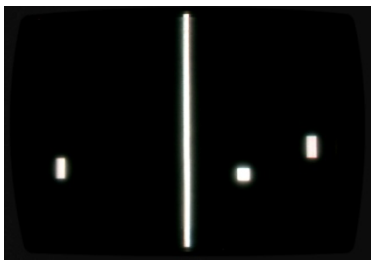
(1) 黎明期

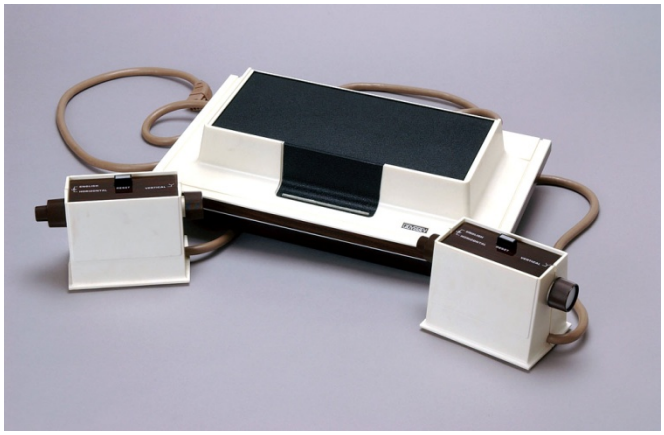
黎明期はテレビゲームの誕生からファミコン発売までの期間に相当する。テレビゲームはアーケードゲームから誕生し、コンソールゲームに広がった。初期のコンソールゲームはコントローラー一体型が中心だったが、やがてコントローラー分離型が生まれ、カートリッジ交換式の普及と共に、標準コントローラーという概念が生まれた。

(a) ポンの成功

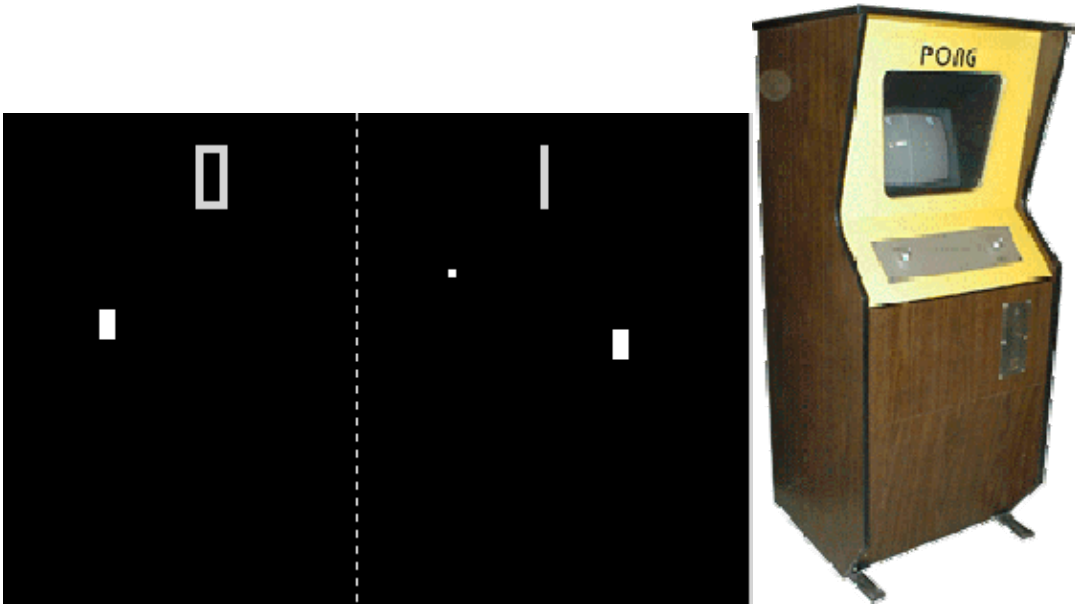
それまで大学などで遊ばれていたテレビゲームが、はじめて大衆娯楽となり、テレビゲーム産業の礎となった作品が、アーケードゲームの『ポン』（1972、アタリ）である。

『ポン』は同時期に開発された、世界初の家庭用ゲーム機「オデュッセイ」（1972、マグナボックス）に収録された『テーブルテニス』の影響を受けて開発された。





左、「テーブルテニス画面」、右、「オデュッセイ」



左、『ポン』画面、右、『ポン』筐体

© 1995-2010 by The International Arcade Museum

「オデュッセイ」のコントローラーには左に 2 つ、右に 1 つのパドル(ダイアル型のコントローラー)があり、左の外周のパドルでラケット(正式名称は「プレイヤー」)を左右に、右のパドルでラケットを上下に移動させ、さらに左の小型のパドルで跳ね返したボールも上下にコントロールするという複雑な操作性を持っていた。しかし『ポン』ではパドルが 1 個に整理され、ラケットのも上下移動のみに制限し単純化している。

その一方で、ボールの反射時に高低 2 つのブリッツ音を鳴らすことができた。またボールがラケットに当たる位置で反射角度が変わるなど、腕前を競う要素もあった。

『ポン』は、それまでのゲームと違い、画面の画像情報と筐体に付いているパドルを見ただけですぐに遊びの方法論がわかり、次第に引き込まれるという、おもしろいゲームに必要なゲーム UI の要素を備えていた。これが『ポン』がテレビゲームで初めて成功する大きな要因となった。

(b) 初期アーケードゲームでの実験

『ポン』が登場する以前から、アミューズメント場にはピンボールやエレメカなどの遊具機器が配置されていた。初期のアーケードゲームは、これらで培われた技術を背景に、さまざまに進化していった。[1]

アナログ技術のエレメカに対して、電子回路で作られたアーケードゲームは、故障率が低いというメリットもあった。エレメカの中身をテレビゲームに移し替えただけのものも見られたが、次第にあきられ、テレビゲームならではの遊びの要素を持ったゲームが人気

を集めるようになった。

こうして 70 年代にかけて、アミューズメント場ではエレメカからテレビゲームへの世代交代が進んだ。また 77 年にはアップルから世界初の家庭用パーソナルコンピュータ、アップル II が発売され、さまざまな PC ゲームがリリースされていった。

『Space Race スペースレース」（1973、アタリ）はじめてジョイスティックを採用

© 1995-2010 by The International Arcade Museum

『Gran Trak 10 グラントラック 10』（1974、アタリ）はじめてハンドル類を採用

© 1995-2010 by The International Arcade Museum

「Qwak ! クワック」（1974､アタリ）アーケードではじめてガンコントローラを採用

© 1995-2010 by The International Arcade Museum

「Tank タンク」（1974、アタリ）はじめて 2 基のジョイスティックを採用）

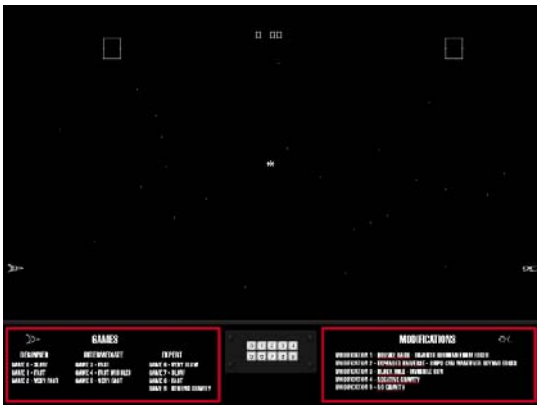
© 1995-2010 by The International Arcade Museum

「Night Driver ナイトドライバー」（1976、アタリ）はじめて疑似 3D 方式でコースを再現

© 1995-2010 by The International Arcade Museum

『Breakout ブレイクアウト』（1976、アタリ）ボールとパドル方式の完成系

© 1995-2010 by The International Arcade Museum



『Space Wars スペースウォーズ』（1977、シネマトロニクス）はじめてベクタースキャンを採用

© 1995-2010 by The International Arcade Museum

(c)　ATARI2600 の成功と標準コントローラーの誕生

『ポン』を皮切りにアーケードゲームで成功したアタリはコンシューマゲームにも進出し、『ポン』が家庭で遊べる据え置き型ゲーム機「ホームポン」（1975、アタリ）を発売した。これはコントローラー一体型で、ゲーム内容も『ポン』のみだった。

さらにカートリッジ交換式の据え置き型ゲーム機「チャンネル F」（1976、フェアチャイルド）が発売されると、続いて「ATARI2600」（1977、アタリ）が発売された。

左、「チャンネル F」、右、「ATARI2600」

両機種の特徴はカートリッジを交換して、さまざまなタイプのゲームが遊べることと、コントローラーがパドル式から脱して、汎用性を志向していたことである。両者は表裏一体の関係にある。汎用コントローラーがなければ、多様なゲームに対応できないからである。

ただし「チャンネル F」のコントローラーは現在の目から見ればいささか特異なデバイスである。片手でスティックを握り、もう一方の手でスティック上部に付いている方向入力バーを動かすものである。一方、ATARI2600 のコントローラーは、アーケードと同じ 8 方向＋1 ボタン式のジョイスティック型をしており(パドルタイプも付属)、コントローラーとしての汎用性は「ATARI2600」の方が優れていると判断せざるを得ない。これはアーケ

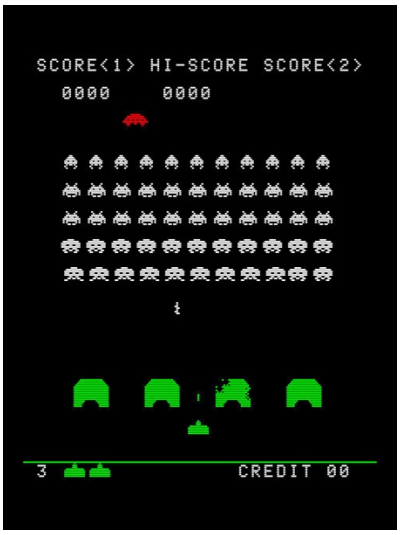
ードゲームでさまざまなタイプのゲームを開発してきたアタリ社の方が、多様なゲームを受け入れることのできるコントローラーとはどういったものかという事をその経験から理解していた(また当初はアーケードゲームの移植が中心であったという必然的な要求の)結果といえるだろう。

ATARI2600 に触発されて、アメリカでは 70 年代後半から 80 年代前半にかけて、据え置き型ゲーム機の発売が続いた。コントローラーの形状は様々で、縦長でテンキーを備えたものや、側面にボタンをつけたもの、パドルを備えたものもあった。

当初は苦戦した ATARI2600 も、『スペースインベーダー』（1978、タイトー）の移植でブレイクし、最終的に全米の 1/4～1/3 の家庭に広まった。

(d)　国産アーケードの興隆

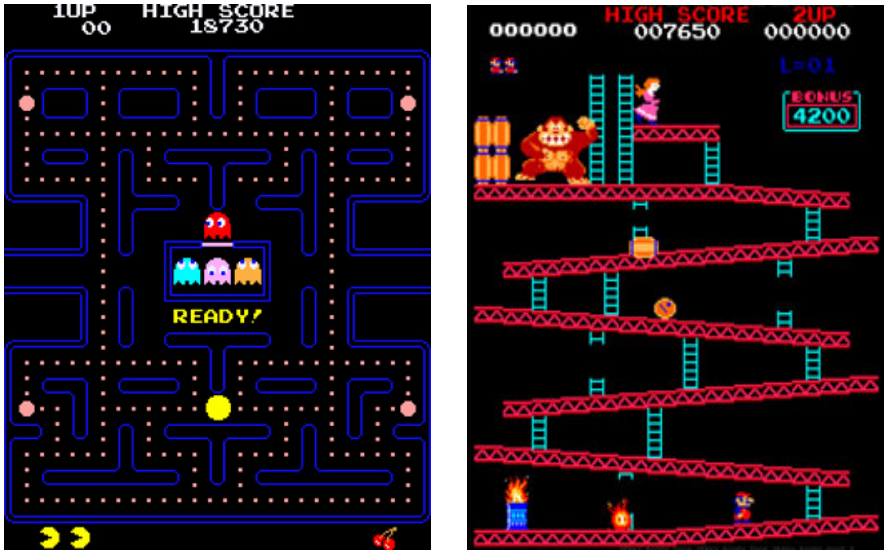
アメリカでブームとなったテレビゲームは、すぐに日本にも輸入された。並行輸入からコピー品レベルの製造を経て、次第にオリジナルタイトルが開発されていった。国産アーケードゲームもエレメカを受け継ぐ形で進化し、78 年に『スペースインベーダー』が大ヒットしたことで、アーケードゲーム産業が成立した。



©TAITO CORPORATION 1978 ALL RIGHTS RESERVED.

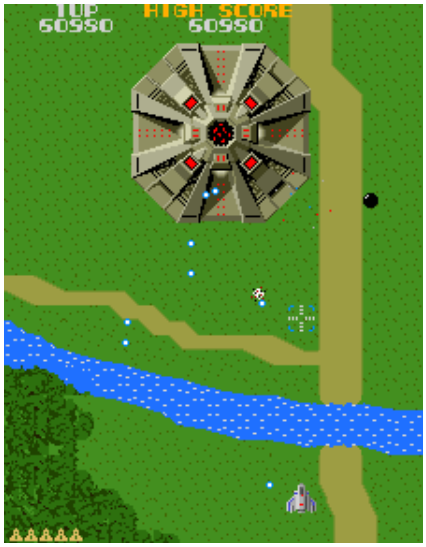
ジョイスティックとボタンの組みあわせでも、さまざまな実験が見られた。初期の『スペースインベーダー』は、左右移動と発射の 3 ボタン方式だったが、すぐに 2 方向ジョイ

スティック＋発射ボタンに変更された。『パックマン」（1980、バンダイナムコゲームス）は 4 方向ジョイスティックのみで、片手で操作できた。『ドンキーコング』（1981、任天堂）では 4 方向ジョイスティック＋1 ボタンだった。



左、『パックマン』©NBGI、右、『ドンキーコング』

8 方向ジョイスティック＋2 ボタンスタイルの代表作が『ゼビウス』（1983、バンダイナムコゲームス）である。空中、地上向けの 2 種類の発射ボタンを備えており、状況に応じて押し分けて遊ぶスタイルをとった。



©NBGI

81 年にはマージャンコントローラをはじめて採用した麻雀ゲーム『ジャンピュータ』（1981、アルファ電子）もリリースされた。

こうしたアーケードゲームの入力デバイスや UI は、ゲームデザインと対をなす形で両方が常に融合する形で開発されていった。そして、さまざまな実験を繰り返しながら、アーケードゲームの宿命でもある「簡単な説明書で誰でもすぐに遊べる」という命題をクリアすべく、メーカーはそのゲームデザインのノウハウを蓄積させていった。

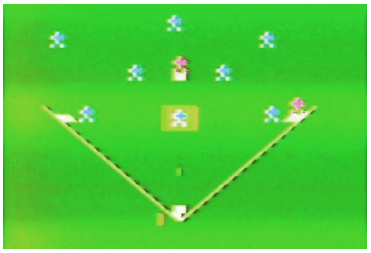
(e)　国産コンソールゲームの興隆

アーケードゲームに続いて、コンソールゲームでも国産ゲームがすぐに発売された。「ホームポン」発売の同年には、コントローラー一体型の据え置き型ゲーム機「テレビテニス」（1975、エポック社）が発売されている。

この「テレビテニス」もそうだが、初期の国産据え置き型ゲーム機は『ポン』タイプのゲームが中心で、「システム 10」(1977、エポック社)や、「ColorTV-GAME 6/15」(1977、任天堂)など、本体に複数のパドルが内蔵／接続されたものが多かった。1977 年に日本国内で販売された膨大な数のテレビゲーム機の大半がこのパドル式(内容もポンタイプ)であることからも、当初のコンソールゲームはそのスタイルがテレビゲームの UI そのものであったことがうかがえる。

左、「システム 10」、右、「ColorTV-GAME 15」

しかし翌 78 年には、パドル形式でありながらポンタイプではないゲームの模索も始まっている。この年に発売されたゲーム機「テレビ野球ゲーム」（1978、エポック社）はその代表的なもので、パドルを使用しながらも、プッシュボタンを追加することで新しい遊びを作ろうとしている事が判る。



翌 79 年以降になると、『スペースインベーダー』と『パックマン』の流行によって、UI の主流はパドルからジョイスティックに移行していく。

その後、アメリカのカートリッジ式汎用ゲーム機の輸入販売などの経緯を経て、日本でもアメリカと同じく、カートリッジ交換式のゲーム機が増えると共に、汎用コントローラーを備えたゲーム機が主流となった。

また 80 年代初頭は、日本でもマイコンブームが巻き起こり、「ぴゅう太」（1982、トミー工業）「M5」（1982、ソード／タカラ）といった、ゲームパソコンが発売された。当然のことながらここではキーボードが中心となるが、ゲームも遊べるように円形の方向キーとボタンが付いたマウス兼コントローラーが付いていた。しかしコントローラーとマウスの両立は難しく、特にマウスとしての操作性の悪さが目立つものとなっている。

左、「ぴゅう太」、右、「M5」

こうした中で、狭間の位置にあったゲーム機が「カセットビジョン」（1981、エポック社）と「ゲーム＆ウオッチ」（1980、任天堂）だった。

「カセットビジョン」はカートリッジ交換式のゲーム機で、ファミコンが発売されるまで最も普及したゲーム機だったが、パドル主体の UI で、しかもコントローラーが一体型であったため汎用性に乏しく、そのためにゲームデザインの範囲が狭かった。

「ゲーム＆ウオッチ」は液晶による携帯ゲーム機で、カセット交換式ではなかった。しかしゲーム内容に応じてボタンの数や形状、配置を変えるなど、ゲーム内容と UI の整合性を作品ごとに追及していった。任天堂にとって「ゲーム＆ウオッチ」は、携帯性というコンパクトさを要求されるデバイスでのゲームデザインを実験・実証していくことで、パドルやジョイスティックといった大型筐体のデバイスを引きずることない入力デバイスと、それに伴うゲームデザインの UI を昇華させていくメディアとなった。その結果として、コンパクトでありながら、手のひらに収まりやすく、遊びやすくて、かつ汎用性の高い「十字ボタン」が発明され、後にファミコンで採用されることとなる。

以下におおまかなゲーム＆ウオッチにおける入力デバイスを思考錯誤の流れを時系列で取り上げてみたが、ボタンの数や左右の配置配分、縦長ボタンと横長ボタン、ボタンと

横長ボタン、十字ボタンではなく十字に配したボタンなど、さまざまな試みがされていることが判る。

上段左『ボール』(1980/4)、上段中『マンホール』(1981/1)、上段右『ドンキーコング』(1982/6)
中段左『ドンキーコング JR.』(1982/10)、中段右『マリオブラザーズ』(1983/3)
下段左『マリオズセメントファクトリー』(1983/6)、下段中『スピットボールスパーキー』(1984/2)、下段右『ボクシング』(1984/7)

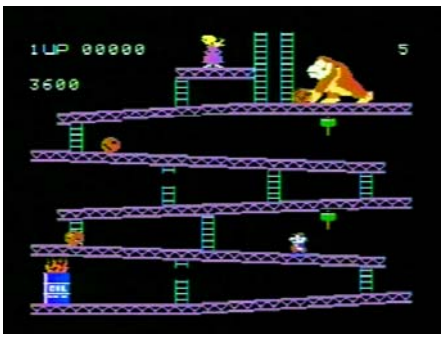
「ゲーム＆ウオッチ」は大ヒットし、ファミコン開発の原資となったうえ、上記のようにゲームデザインのノウハウ蓄積も進んだ。「ゲーム＆ウオッチ」は国内販売が終了した後も、海外向けに 91 年まで開発が続いた。

(2) 2D ゲーム期

2D ゲーム機はファミコンからスーパーファミコンまでの期間に相当する。アーケードゲームや PC ゲームの影響を受けながら、コンソールゲームにおけるソフトウェア面でのゲーム UI の成熟が進んだ。

(a) 標準コントローラーの確立

「ファミリーコンピュータ」（1983、任天堂）が発売された 1983 年は、他に十種類以上ものテレビゲーム機が発売され、この時点では、各社さまざまなタイプの UI へとアプローチされていることが判る。
それらの中でファミコンが成功した理由の一つに、そのコントローラーが非常に優れていた点がある。ファミコンはアメリカの「コレコビジョン」（1982，Coleco）や ATARI2600(1977、ATARI）を含むアメリカのテレビゲーム機を参考に作られたが、コレコビジョンのコントローラーは当時アメリカのゲーム機で主流になりつつあったテンキー付属のものであった。このテンキーそれぞれに対して、ゲーム毎に独自の操作を割り振るというもので、当時すでに複雑化するゲーム内容に対応するためボタンを増やすという方向性に向かっていたといえる。



左、「コレコビジョン」、右、「コレコビジョンのドンキーコング」

ファミコンと同時期に他社が発売したゲーム機はこのテンキー付きのものも含め、それまでのパドルやジョイスティックといった入力デバイスの概念から抜けきれないでいたが、ゲーム＆ウオッチの経験を生かして採用した、コンパクトかつ指の感覚だけで直感的に操作できる十字ボタンと A,B ボタンという独自のコントローラーは、これまでのデバイスを駆逐し、以後のコンソールゲームにおける標準デザインとなったことからもその画期性が理解できるだろう。また制作コストがかからない点もポイントで、「ファミリーコンピュータ」が他社のゲーム機よりも安価であったことも普及に大きく貢献した。

上段左、「TV ボーイ」(1983、学研)、上段右「ぴゅう太 Jr」(1983、トミー)
中段左、「ARCADIA」(1983、バンダイ)、中段右「SG-1000」(1983、セガ・エンタープライゼス©SEGA)
下段左「PV-1000」(1983、カシオ)、下段右「ファミリーコンピュータ」(1983、任天堂)

初期のファミコンソフトはアーケードゲームからの移植が中心で、ゲーム内容もシンプルだった。しかし、ファミコンがロングランを続けるにつれ、ゲーム開発者は、より複雑

でバラエティに富んだゲームソフトを作り出す必要性に迫られた。

そこで、ボタン数の少ないコントローラーで複雑な操作を直感的に行わせるため、さまざまなゲーム UI の革新がソフトウェアの面で行われていった。

(b) ボタンの役割固定

100 円で遊べるアーケードゲームに比べて、数千円もするカセットを買わせるコンソールゲームでは、1 本のゲームのプレイ時間が格段に長い。そのためファミコン初の家庭用アクションゲームとして開発された『スーパーマリオブラザーズ』(1985、任天堂) では、アーケードゲームと比べて、はるかに大量のアイデアが盛り込まれた。

ゲーム中でマリオは歩く、走る、しゃがむ、すべる、泳ぐ、ジャンプする、蹴る、アイテムを取る、つたや梯子を昇降する、ファイアを発射するなどのアクションがとれる。後のシリーズになると、これに空を飛ぶ、回転する、つかむ、モノを投げる、話す、看板を読む、選択する、使用するといったアクションも加えられている。

これを直感的に操作させるために、まずマリオのアクションを体系化し、移動関係を十字ボタン、メインのアクションを A ボタン、サブのアクションを B ボタンに割りふった。なお、この際に十字ボタンの下を押すと「しゃがむ」など、操作とアクションが関連性を持つように設計されている。

次に、ボタンが押されたときのマリオの環境に応じて、最適なアクションが自動的に実行されるようにプログラムされた。地上で A ボタンを押すとジャンプするが、水中だと浮かび上がる。B ボタンを押すとダッシュするが、ファイアマリオ状態ではファイアを発射する、といった具合である。

この 2 つを組みあわせることで、プレイヤーは最小限のボタン操作で、複雑なアクションを自由自在に繰り出せるようになった。また作り手側も、さまざまな仕掛けを盛り込むことが可能になった。

ボタンの役割固定と、最適なコマンドの自動選択は、ゲーム UI の基本的な考え方で、

幅広く見られる。麻雀ゲームで当たり牌を自動で選択するなどは、その一例である。

(c) メニュー操作

ファミコンではじめて発売された RPG『ドラゴンクエスト』(1986、エニックス）は、当時アップル II で流行していた『ウルティマ』(1980、オリジン)『ウィザードリィ』(1981、Sir-Tech）シリーズをベースに、日本的なアレンジを加えて開発されている。中でもパソコンに比べて圧倒的に狭い画面で、いかに優れた UI を設計するかが鍵となった。

©1986 ARMOR PROJECT/BIRD STUDIO/CHUNSOFT/SQUARE ENIX CO., LTD. All Rights Reserved.

そこで PC のアドベンチャーゲームなどで使用されていたコマンド選択システムを参考に、メニューウィンドウと組みあわせて使う UI が考案された。これは普段はメニューウィンドウを閉じておき、必要に応じて開いて表示させ、十字ボタンでコマンドを選択して実行するというものである。

ゲーム中のさまざまな行為は、グループごとにツリー化され、次々にウィンドウを開いて選択するようになっている。ウィンドウを開いたり、行為を選択するのは A ボタン。行為をキャンセルしたり、一つ前のメニューに戻るなどの場合は B ボタンだ。メニュー配列も十字ボタンで選びやすいように、縦横でレイアウトされている。

またキャラクターレベルなどのサブ情報ウィンドウは、一定時間操作を中断すると自動的に開き、何かボタンを押すと消えるようにプログラムされた。

「ドラゴンクエスト」シリーズは大ヒットし、以後ほとんどのゲームで「十字ボタンに相当する選択ボタンでメニュー項目を選択し、A に相当するボタンで決定、B に相当するボタンでキャンセル」という操作が定着した。

(d) SE によるリズム感の付与

『ドラゴンクエスト』でもう 1 つ重要な点は、メニュー操作が決定音などの SE（サウンド・エフェクト）によって補強された点である。

項目選択時に SE を鳴らす演出は、それまでのゲームでも一般的に見られたが、「ドラゴンクエスト」では、これが意図的に作成され、割り振られている。その結果、何万回にも及ぶメニュー操作のストレスを軽減する役割を果たしている。

今日ではメニューの選択音、決定音、キャンセル音、それ以上行為がキャンセルできない場合に鳴らす強決定音、アイテム合成音など、メニュー操作だけで十数種類の SE が用意され、メニューを操作するだけで心地よく感じられるような配慮がなされている。メニューが開いて十字ボタン選択で「決定」「キャンセル」するばあい、「キャンセル」を選んで A 決定するときは、決定でありながらキャンセル SE を鳴らすことで、無意識の操作互換を行っていることなども効果的な SE の使用方法といえだろう。

(e) コントローラーの複雑化

据え置き型ゲーム機はモデルチェンジを続けながら、コントローラーを複雑化させていった。これには内的要因と外的要因がある。

内的要因には、コントローラーのボタン数を増やすことで、より複雑なゲームが作れる点である。ファミコンのボタン数では複雑なゲーム操作に限界があると考えた任天堂は、スーパーファミコン（1990、任天堂）の開発で操作ボタンを 4 つ増やした。そして以前のものとあわせて、A,B,X,Y を前面に配置し、いままであまり使用していなかった左右の人差し指を効果的に使用するアイデアとして、L,R ボタンを上部に配置した。

X ボタンはメニューボタン、Y ボタンは補助ボタンとして使われる例が多い。L,R ボタンはドライブゲームなどで、より左右への動きを加味する際などに多用された。このボタン配置のコンセプトは以後のゲームコントローラにも共通して見られる。

外的要因としては、アーケードゲームの移植を容易にする点が上げられる。セガはメガドライブ（1988、セガ・エンタープライゼス）設計にあたり、コントローラーに A、B、C の 3 ボタンを配置した。これは当時アーケードゲームでジョイスティック＋3 ボタンの操

作パネルが主流になっていたからである。

©SEGA

さらにメガドライブ 2（1993、セガ・エンタープライゼス）では A,B,C,X,Y,Z の 6 ボタンを配置した。この背景には当時アーケードゲームで人気のあった格闘ゲーム「ストリートファイターII」（1991、カプコン）が、ジョイスティック＋6 ボタンの操作系だったことがある。

©SEGA

逆に現世代機においては、アーケードゲームでアナログスティック＋8 ボタンの操作パネルが主流になっている。これはアーケードゲームで据え置き型ゲーム機の互換基盤が用いられており、相互の移植を容易にする点が背景にある。

(3)　3D ゲーム期

プレイステーションなどを経て、NINTENDO64 から本格的な 3D ゲーム期が始まり、今日まで続いている。アーケードゲームや PC ゲームに続いて、コンソールゲームでも本格的にポリゴンが使えるようになり、ゲーム画面が 3D になった。入力操作も十字ボタンからアナログスティックとなり、箱庭的なゲームデザインが可能になった。

(a)　3D ゲームの誕生

2 次元のテレビモニタで奥行きのある画面を擬似的に再現しようとする試みは、テレビゲームの初期から見られた。ポリゴンを用いた 3D ゲームの第 1 号は、アーケードゲームの「I, Robot」（1981、アタリ）となっている。

その後アーケードゲームで『バーチャファイター』（1993、セガ・エンタープライゼス）『リッジレーサー』（1994、バンダイナムコゲームス）など、さまざまな 3D ゲームがリリースされるようになった。これらはセガサターン（1994、セガ・エンタープライゼス）、プレイステーション（1994、ソニー・コンピュータエンタテインメント）に移植され、コンソールゲームでも 3D ゲームが楽しめるようになった。

これらはグラフィックが 3D になったが、コントローラーの形状は十字ボタン方式を引き継いでいた。これに対して NINTENDO64（1996、任天堂）は、3D 空間を直感的に操作できる入力デバイスを独自に考案する。コントローラーにアナログスティック（3D スティック）を搭載し、スティックの傾きと奥行きを同期させることで、立体的で奥行きのある箱庭世界をストレスなく探索できるようになった。

また 3D 空間内で衝突判定を効果的に伝えるために、コントローラーに振動機能が加わった。まず NINTENDO64 の拡張デバイスとして「振動パック」（1997、任天堂）が発売された。

アナログスティックと振動機能の組み合わせは、プレイステーションで拡張コントローラー「デュアルショック」（1997、ソニー・コンピュータエンタテインメント）が発売さ

れるなど他社も追随し、これ以降のゲームコントローラの標準となった。

またアナログスティックの数もプレイステーション向けの「アナログジョイスティック」(1996、ソニー・コンピュータエンタテインメント) から 2 個搭載されるようになり、これ以降のゲームコントローラの標準となっている。これにより、フライトシミュレータや FPS など、主に PC ゲームで進化した、より複雑な操作が必要なゲームの移植が容易になった。

(b) 3D ゲームとゲームデザイン

ゲーム UI が入力側も出力側も 3D に対応したことで、ゲームデザインにも大きな影響を及ぼした。

3D 空間は 2D 空間に比べて、より正確な状況がつかみにくい。特に立体的な表現が可能になった一方で、空間内におけるオブジェクトの「高さ」の概念がわかりにくくなる。

そこで『スーパーマリオ 64』(1996、任天堂) では、マリオがジャンプすると地面に擬似的な影を表示して、着地点の目安にさせている。この「影による高さの表現」は、これ以外にもさまざまなゲームで用いられている。

また十字ボタンに比べてアナログスティックでは、入力の自由度が増した一方で、微細な操作が難しくなった。そのため 3D ゲームでは、より「曖昧な操作」を前提としたゲームデザインが必要になった。

一例としてテニスゲームなどでは、3D ゲーム化で操作の正確さよりも、読み合いの要素が強まった。逆にシューティングゲームなど、操作の正確さを重要視するゲームでは、グラフィックは 3D でも操作は十字ボタンやジョイスティックに留まる例も見られた。

(c) マウス操作の取り込み

マウスとキーボードは PC ゲームに特有の入力デバイスで、80 年代中盤から普及した。これによりマウス操作を前提としたゲームが作られるようになり、コンソールゲームへの移植時に障害が生じた。

初期の代表例として、画面上の任意の箇所をクリックして進めていく AVG の『MYST』(1993、Cyan)、都市計画 SLG の『シムシティ』(1989、マクシス)、神の視点でゲームを進める『ポピュラス』(1989、ブルフロッグス) などがある。

また 90 年代以降は、FPS、RTS、オンライン RPG などが PC ゲームならではのジャンルとしてヒットした。

これらはコンソールゲームへの移植時に、十字ボタンやアナログスティックでカーソルを移動させたほか、ゲーム機によっては専用マウスが発売されることもあった。コントローラーのボタン操作でカーソル移動を支援するなどの取り組みもあったが、総じて大きな成功には至らなかった。

こうした中で、ゲーム内容をコントローラーでの操作向けに調整し、成功した例として、FPS の『Halo』（2001、マイクロソフト）などがある。

それまでの FPS は立体的なステージ構造の中で、ジャンプなどを繰り返して移動・攻撃するものが多かった。これが『Halo』では平面的なレベルデザインが多用された。また敵キャラクターを異星人とし、より丸みを帯びたフォルムにした。キャラクターの動きもそれほど素早くなく、総じてアナログスティックでも狙いがつけやすくなっている。その一方で AI アルゴリズムの進化で、敵キャラクターの動きを複雑化させており、ゲームに深みをもたらしていた。

© 2010 Microsoft Corporation. All Rights Reserved.

RTS ではマウスドラッグによる範囲指定を、ボタンの押し込みで代用した『ピクミン』（2001、任天堂）の例がある。オンライン RPG ではアイコンを並べて簡易チャットを行なわせた『ファンタシースターオンライン』（2000、セガ・エンタープライゼス）などの例がある。

(d) ホームボタンとヘルプ

コンソールゲームはゲーム以外の機能をそぎ落とすことで競争力を高めてきたが、現行機の Wii、PS3、Xbox 360 では常時接続を前提とした設計となっており、情報端末の側面が強化されている。

それに伴い従来の UI 操作だけでは、多彩な要求に追いつかなくなった。そこでコントローラーに追加されたのが「ホームボタン」で、これを押すことで、どのような状態からでも一定の箇所（デスクトップ画面など）に戻ることが可能になった。

またゲームの複雑化に伴い、ゲーム内ヘルプがさまざまな形で発展していった。コントローラーの特定のボタンにヘルプ機能を割り当てたり、ゲーム画面上にソフトウェアボタンとして割り当てるだけでなく、ゲーム内にヘルプメッセージを配置したり、ロード時間に TIPS を表示させたり、プレイヤーを誘導するヘルプキャラクターを盛り込んだりと、さまざまなバリエーションがある。

(4)　モーションデバイス期

2000 年以降、複雑化が進んだコントローラーの反動として、タッチペンで入力したり、プレイヤーの腕や体の動きで操作するタイプのゲーム機が登場した。この流れを総称してモーションデバイス期と呼ぶ。Wii とニンテンドーDS、PS3 などから始まり、他社も追随しつつある。

(a)　モーションデバイスの登場

ボタン操作ではなく、プレイヤーの動作自体がコントローラーになるという発想は、90 年代後半からまず、アーケードゲームの大型筐体で見られた。スキーゲームの『アルペンレーサー』（1995、バンダイナムコゲームス）や、音楽ゲームの『ダンスダンスレボリューション』（1998、コナミ）などは、その一例である。

これをコンシューマゲームで実現したのが、PS2 の拡張コントローラーで発売された、Web カメラコントローラの『EyeToy』（2004、ソニー・コンピュータエンタテインメント）である。これはプレイヤーの全身をカメラでテレビモニタに映し、画面上のオブジェクトとインタラクションがとれるというもので、欧州を中心にヒットした。ただし、この取り込まれた体の動きと、ソフト内で再現される動きが感覚的に気持ち良いレベルには達しておらず、大きく広まることはなかった。

本格的なモーションコントローラ時代を切り開いたのが「Wii」（2006、任天堂）である。コントローラーに CMOS イメージセンサーと MEMS3 軸加速度センサーを備え、コントローラーを剣やテニスのラケットに見立てて振ったり、ハンドルに見立てて回転させたり、ビリヤードのキューに見立てて突いたり、拳銃などに見立てて画面の 1 点を狙う、などの操作ができる。

PS3（2006、ソニー・コンピュータエンタテインメント）の初期標準コントローラー「SIXAXIS」にも 6 軸検出システムが搭載され、コントローラーを傾けて操作ができた。「SIXAXIS」は後に振動機能を搭載した「デュアルショック 3」（2007、ソニー・コンピュータエンタテインメント）へと引き継がれた。

2010 年春には PS3 で「PlayStation 専用モーションコントローラ(仮称)」が発売される予定で、Xbox 360 でも RGB カメラやマイク機能などを活用し、コントローラーを使わずに体の動きと声で操作する「Project Natal」の提供が予定されている。

(b) タッチデバイスの登場

タッチデバイスを搭載した初めてのゲーム機が、携帯型ゲーム機の「ニンテンドーDS」（2004、任天堂）である。タッチペンを用いて、つつく、なでる、こする、押すなどの入力ができる。

その後、携帯電話の iPhone（2007、アップル）が登場した。こちらは指で直接タッチ操作でき、DS と異なり 5 点までのマルチタッチ操作ができる。これによりフリック（はじく）、タップ（軽く叩く）、ピンチ（つまむ）などの操作を可能にしている。また Android、WindowsPhone でもタッチ操作の UI が採用されているが、こちらは DS と同じシングルタッチの操作である。

左、「ニンテンドーDS」、右、「iPhone」

タッチ操作を生かしたゲームには『東北大学未来科学技術共同研究センター川島隆太教授監修 脳を鍛える大人の DS トレーニング』（2005、任天堂）などがある。本作はまた、ボイス操作によるゲームの代表作でもある。

(c) モーションデバイスの影響

モーションコントローラであれ、タッチデバイスであれ、アナログスティックよりも、操作感がさらにアバウトになる点は避けられない。

たとえば文字入力では、同じ文字でも人によって筆跡の癖が大きい。同じ「振り下ろす」という動作でも、大人と子どもでは振り下ろす距離も、加速度も異なっている。誤操作がおきる可能性も、より大きくなる。

そこで必要なのは「ゲームの特定箇所で必要な入力操作を限定させる」（特定の文字や記号を書かせる、特定の Wii リモコンの操作をさせる、など）ことと、「プレイヤーの入力行為を補完し、その意図をくみ取って、適切なアクションを起こす」という 2 点になる。これは 2D ゲーム期の「ボタンの役割固定」で説明した「入力アクションを体系化し、ボタンにわりふる」「プレイヤーの状況に応じて、最適なコマンドを自動選択する」ことの発展系となる。また、これはゲーム以外の UI でも広く応用可能な考え方である。

このほかモーションコントローラ、タッチペンは共に、画面上の正確な点の特定には不向きであるため、ボタンやアイコン類は大きくデザインする必要がある。オプション画面などでも、1 枚の画面内で選択させる項目はできるだけ少なくし、複雑な設定が必要な場合でも何枚ものウィンドウに分けて、段階的に設定させるなどの配慮が必要だ。

このように入力デバイスによるアバウトな入力をソフト側で補完して、感覚的に気持ち良く使用させるのは、容易なことではない。こういった感覚的なものはシステムとして仕様書に書けるようなものではないからで、その実現のためには、ハード側とソフト側がそれぞれ歩み寄って開発しなければならないからである。今後このような人間の感覚に寄り添う UI 実現のために、今のようなハードとソフトの縦割り構造を根本から変えなければいけない時期に来ているといえよう。いずれにせよ、モーションデバイスによる UI デザインは、従来のノウハウを土台としつつも、さらなる発展が必要である。まだ答えが出尽くしているとはいえず、さらなる研究が必要だろう。

2.6.5 ゲーム UI の未来

これまでゲーム UI の歴史について論じてきた。本章ではまとめとして、近い将来におけるゲーム UI の可能性について論じていく。大きなトピックとして立体視テレビへの対応があげられる。

(1) 立体視テレビ

テレビゲームの誕生以来、ゲーム UI はさまざまな形で進化してきた。しかし、出力媒体であるテレビモニタは、この数十年の間で SD から HD へと解像度が変わった程度で、大きな変化がない。

しかし、ここにきて立体視テレビが登場し、普及のきざしを見せ始めている。ハリウッド映画「アバター」は本格的な立体視映画として大きな話題を呼んだが、E3 2009 でも UBI が「アバター」を題材に、FPS「アバター THE GAME」（2009、UBI）の PC による立体視版を試作し、一部の関係者に公開していた。

筆者は直接体験できなかったが、体験談によると体力ゲージやアイコンなどが目の前に浮き上がって配置されたり、奥から手前に起き上がるようにアニメーションするなど、立体視ならではの UI が工夫されていたという[3]。

またグラフィックチップベンダーの Nvidia では 2009 年より、家庭用立体視システム「NVIDIA 3D Vision for GeForce」を発売しており、300 タイトル以上の PC ゲームが立体視で楽しめる。コンソールゲームでも立体視対応のアクションゲーム「インビジブルタイガー」（2009、バンダイバンダイナムコゲームスゲームス）が Xbox 360 の Xbox LIVE アーケード向けに配信を始めるなど、立体視ゲームが身近な存在になってきた。

このことはゲーム UI に、さらなる革新が求められることを意味している。これまでの歴史を振り返ればわかるように、2D ゲームでは十字ボタン、3D ゲームではアナログスティックというように、ゲーム UI は出力側と入力側で共に進化してきた。

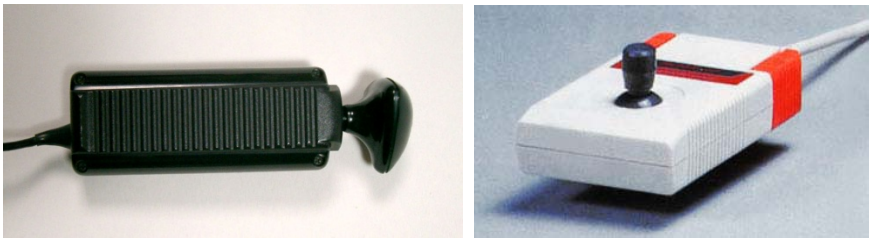
しかしモーションデバイス期においては、入力デバイスは拡張されたが、画面表示は従来のままに留まっている。これが立体視テレビが普及することで、真の意味でのモーションデバイス操作が必要になると考えられる。

一例として三次元メニューを考えてみよう。メニュー項目を従来の二次元配列ではなく、奥行きも含めた三次元表示として、立体的に表示させられるようになる。これによってメニュー自体を回転させたり、奥の方のアイコンに腕を伸ばして直接選択したり、手前に持ってくるなどの動作が可能になる。二次元から三次元にアイコン配列が増加することで、e コマースサイトなどで、より多くの情報を直感的に扱えるようになるだろう。

(2) かつてのデバイスの再考

ここまで駆け足でゲーム機における UI の歴史を追ってきたわけだが、その流れの中で消えていったコントローラーのデザインもさまざまである。そういった中には、いくつかの特異な入力デバイスもあった。これらは当時あまり活用されることなく消えていったものではあるが、現在のテクノロジーとソフト開発のノウハウがあれば、面白い活用法があるかもしれない。たとえば「チャンネル F」のコントローラーは左手でスティック部を握り、もう片方の手でスティック上部に付いたジョイ部で方向入力をするわけだが、方向入力だけでなく、上から叩くという入力も可能である。ここだというタイミングで、手のひらで「ポンッ」と叩いて入力する行為は意外に楽しく、こういった面白さの可能性を再考することも重要と思える。これ以外にも、「スーパーカセットビジョン」(1984、エポック社)のように(ジョイスティックは別にして)片手で握りながら人差し指と親指でボタンを

入力するのも最近のコントローラーにはない快感がある。ソフトの交換と一緒にインターフェースも交換してしまう「Microvision」(1979、Milton Bradley Company)などもユニークな発想といえるだろう。またゲーム機ではなくても、ソニーの PDA「クリエ」等に付いていたジョグダイアルは、親指で回転させて必要に応じて押し込むといった操作が非常に快適であった。ゲームソフトでもかつて消えていったゲームシステムを現代によみがえらせて成功している例がある。このように「Wii」や「DS」の登場で、十字ボタン＋ボタンという操作性からの脱却を試みている現在、かつての UI を再考してみればおおいに参考になる部分があるのではないだろうか。



左、「チャンネル F」のコントローラー、右、「スーパーカセットビジョン」のコントローラー

(3)　変わるものと変わらないもの

立体視テレビなどでゲーム UI の可能性が広がる一方で、普遍的なものもある。それは先にも述べたように、ハードウェアとソフトウェアの双方が、互いに連携を取りながら「使い勝手」という目に見えない、不確かな価値を作り上げていくことである。

これはゲーム開発の本質な部分である。ゲームデザインによって生み出される「おもしろさ」は、ゲーム UI を通してプレイヤーに伝えられる。そのためゲーム UI の善し悪しはゲームの評価に直結するといって過言ではない。

このハードウェアとソフトウェアの連携は、80 年代から 90 年代にかけてアーケードゲームの開発部門で普遍的に見られ、日本のゲーム開発を支えてきた。任天堂がファミコン以降、今日に至るまでコンソールゲームで成功を収めてきた要因も、ここにある。

またゲーム業界以外では、iPhone があるだろう。iPhone が成功した要因の一つは、タッチした感触が抜群に良かったからで、これはハードウェアとソフトウェアの高度な連携なしには実現不可能である。そして世界中の企業が未だに iPhone を超える操作性を持つ携帯電話をリリースできない主な原因の一つに、縦割り方式の開発体制の弊害があることはすでに述べたとおりである。

今後ゲーム UI がどのような進化を遂げるかは不明だが、この一見非効率とも感じられる、異なる部門の連携なしには、広く浸透する製品作りは不可能だろう。

参考文献：

[1]赤木真澄：“それは『ポン』から始まった”，アミューズメント通信社 (2005)

[2]赤木真澄：“アーケード TV ゲームリスト 国内・海外編”，アミューズメント通信社 (2006)

[3]西田宗千佳：“西田宗千佳の― RandomTracking ―E3 2009 特別編 UbiSoft が「3D 版 Avatar:The Game」を極秘デモ～パナソニックと共同で。103 型フル HD 3D プラズマ＋ Xbox 360～”，http://av.watch.impress.co.jp/docs/series/rt/20090605_212603.html

ゲーム＆ウオッチ写真提供：山崎功/オトナファミ

2.6　周辺機器メーカーから見たハードウェアインターフェースの変遷

ユーザーインターフェイスのハードウェア面における変遷について、周辺機器メーカーとしての立場から、株式会社 HORI 営業部大神佳人氏にお話を伺った。（聞き手：三宅委員・オブザーバー松井）ヒアリングの参考資料として、昨年の「ハードウェアの変遷」をコントローラーデバイスに重ね合わせた表を用いている。

大神佳人氏プロフィール：
2001 年 株式会社ホリ入社。以降様々なゲーム周辺機器の企画・開発に携る。代表的な製品には、ファイティングスティック、ファイティングコマンダー、コマンドスティック、グリップコントローラ、ゼロテックなどがあり、リアルアーケード Pro.シリーズも手がけている。現在は製品企画の他、広報、自社ホームページ・インターネットストア「HORISTORE..com」運営、ファンクラブ会長なども兼務している。

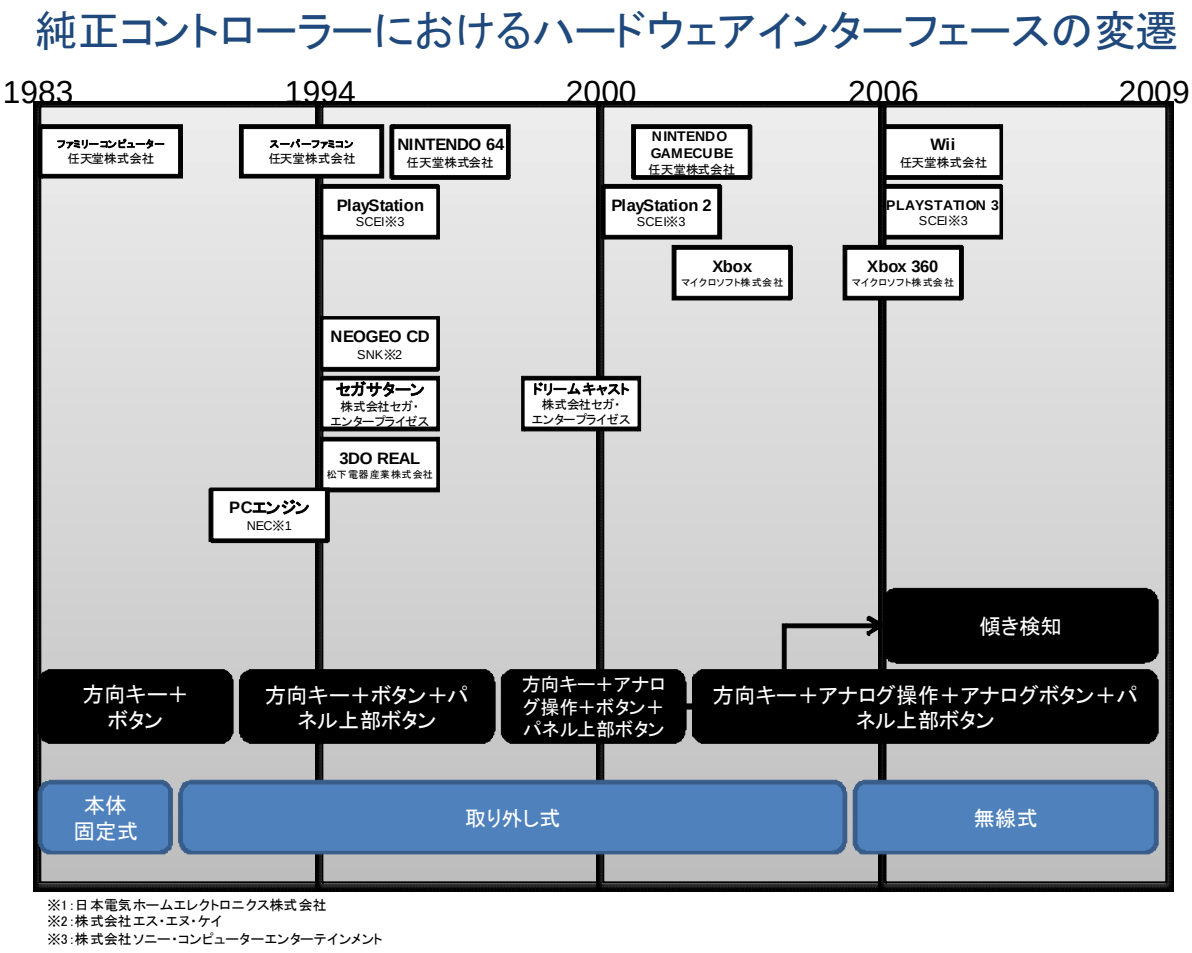


図 2.6-01 純正コントローラーにおけるハードウェアインターフェースの変遷

2.6.1　ファミリーコンピュータ時代

--こちらの図表を見ながらお話を伺っていきたいと思います。まず今お渡ししているもので、「ファミリーコンピュータ」時代からペリフェラル（周辺機器）メーカーとしてのコントローラーを作られていますね。大神さんはそのころから HORI におられるのでしょうか。

大神：私がこの会社に入ったのは、ちょうどファミリーコンピュータからスーパーファミコンに切り替わる時期ですから、ファミコンの時代というのはリアルタイムに私は体感していません。
ただし、そこの流れはある程度私が認識している事の他に、色々な話を聴いている部分もあるので、お話ができると思います。

--最初にコントローラーデバイスを大きく「本体固定式」から「取り外し式」、「無線式」というふうに大きく分けて考えています。この、取り外し式は「PC エンジン」あたりからではないか、と考えていますが、いかがでしょうか。

大神： PC 関連のものは別ですけれども、家庭用のゲーム機に関してはそうなっていると思います。

--御社の製品は、当然ながらサードパーティーですから、取り外し式の……ファミリーコンピュータでいう前面ポートの位置に差し込む形式でしたね。

大神：そうですね、外部入力端子に、いわゆるカスタムコントローラーとして接続するという形をとっていました。4 方向の方向キーと AB ボタン、それからセレクトボタンとスタートボタンで構成されていました。

--その頃から、コントローラーパッド型と、スティック型のモデルを作られていたんでしょうか。

大神：はい、ファミコン時代から両方ともやっています。

--その部分で、なぜ……というと変かもしれませんが、すでにゲーム機を買うと 2 個コントローラーが付属している状態で、さらにコントローラーパッドを販売された理由はどこにあったのでしょうか。市場として、売れると判断されたという部分ですが。

大神：弊社が最初にオリジナルでやらせていただいたコントローラーは「ジョイスティックセブン」というファミコン向けのジョイスティックですが、実はこれは任天堂さんの純正品として作る予定だったんです。それが途中で、任天堂さんが売らないという話になって「それじゃホリさんの方でいかがでしょう」というところから始まっているんです。

それまで任天堂さん以外のメーカーともコントローラーを作らせていただいていましたし、自社製品として「やってみたい」というのはありましたから、渡りに舟というか、やってみようということで始めたのがジョイスティックセブンです。ですからジョイスティックが弊社の最初のオリジナル製品、いわゆるホリのブランドとしてやらせていただいたのがこれですね。

写真 2.7-01　ジョイスティック 7

--まだ、当時は現在のように大手量販店も無かったでしょうし、当然インターネットの直販もできませんでした。流通面はどうされていたのでしょうか。

大神：あの当時は、おもちゃ問屋さんというのがゲームの流通のメインでした。そのうち任天堂さんの製品を取り扱う問屋さんの、当時初心会という流通の組合がありましたので、そういうところと主にお取引していたようです。

--ファミリーコンピュータ時代の御社の製品で一番売れたものというのは？

大神：コントローラーですね。当時、連射の機能がついたコントローラー、「ホリコマ

ンダー」を出しまして。それは爆発的に売れていました。これはスーパーファミコンの時代に入っても、ファミコンのコントローラーなのにも関わらず売れ続けたという逸話があります。

--購入層はシューティングゲームがメインでしたか。

大神：そうですね、シューティングゲームでご使用されている方が多かったと思われます。ただ連射の機能に関してはゲームの中の要素として、必須ではないんですけれどある瞬間にどうしても必要となったりすることがあります。

早くボタンを押す連打というものは、多分当時、ゲームをおもしろくする要素の大きな部分で、シューティングゲームに限らずある種、連打を必要とする場面と言うのは結構多かったですし。単純にゲームを進めていく上で、惰性としてボタンを押していけばなんとなく進む、ゲームの進行を助けられる。そういうことの補助のために使われているところもありましたので、使い方は限定されてなかったようです。ユーザーさんがそれぞれ、これに使えるなということを見出して使っていただいているような、そんな形だったと思います。

写真 2.7-02　ホリコマンダー

--ちなみに、連射機能について特許を取られたりはしていますか？

大神：無いですね。非常に単純な行為でもありますので、特許の取得は非常に難しいですね。

--そもそも、この連射機能は何がきっかけで入れることになったのでしょうか。ユーザーさんからの声とか開発者とか……。

大神：そのきっかけについては、私のほうではっきりはつかんでいないですが、当時、そういう連打を必要とするようなゲームがありましたので、おのずと必要な機能だという事になったんだと思います。

2.6.2　スーパーファミコン時代

--その後、スーパーファミコンに移行するにあたって、大きな変化として、ボタンが増えたことが挙げられます。

大神：劇的に増えています。いきなり 4 つ増えています。それも、今まで使わなかった指を使うようなボタンまで出現していますので。おそらくその時点では、「こんなにボタンは必要ないだろう」と見方が多かったんじゃないでしょうか。

--スーパーファミコンが新しく出る際に、すでにペリフェラルの契約であるとか技術の開示っていうのは任天堂からはありましたか？

大神：あまり詳しくは申し上げられませんが、当然技術的な開示を頂かないと我々は作れないですから。ある程度事前にお話はいただいております。

--出来上がったものを買ってきて分解して、といったことはされていないですか？

大神：そういうことではないですね。
--ファミリーコンピュータからスーパーファミコンになって、ボタンが増えた以外にハードウェアインターフェースとして変わったなというものはありましたか？

大神：そこの時点で大きく変わっているものというのはおそらく無いと思います。

写真 2.7-03 スーパーホリコマンダー

2.6.3　アナログ機能時代

--先ほどの図表でいいますと、インターフェース面の変化として、大きなタイミングとして挙げられるのは、アナログ操作の要素が追加された、という時代、PlayStation の中期頃でしょうか。

大神：アナログの要素が入ったことは大きいですね。

--御社でも、アナログコントローラーを販売されましたが、製品化までには時間がかかっていたのでしょうか？

大神：そうですね。ただ、すでにお手本になるものがあるわけで、一から全ての要素を開発しなくてはいけないわけではありませんでしたから、開発自体に膨大な時間を掛けたということはありません。

--技術的に、アナログコントローラーを作るというのは難しかったのでしょうか。

大神：その時点で一番大変だったのは、そのアナログ的な要素が入ってきたことよりも、アナログコントロールの部分を動かすために、どういう形にするとか、使いやすくするためにどうするとか。当然純正コントローラーがありますので、そこをベースにして考えれ

ばそんなに難しくはないんですしょうけれども。オリジナルの要素を盛り込んでどう昇華していくっかという部分に力を入れています。

--そのままではなくて、HORI のいいところを出すという。

大神：そうですね。一番困るところというと、例えば弊社の製品ですと、連射の機能をつけなければいけない。それを動かすためのスイッチですとか。そういったものをコントローラーの中に収めなければいけない。ですが、例えばその時入ってきたアナログの要素が入っている上で、なおかつそういったものを付けなければいけないですから、スペースが限られている中で、どこにそれを付けていこうかといった調整は非常に苦労した記憶があります。

写真 2.7-04　アナログ振動パッド

--PlayStation の時代は、御社では、コントローラーのバリエーションが非常に豊富でした。片手用コントローラー、プロポ型コントローラー、小型ジョイスティックですとか。あの頃はハードウェアとしてプレイステーションが沢山売れていたので、色々な種類を出されていたのでしょうか。

大神：もちろんそれはありますね。ある種のゲームジャンルに特化しているようなコントローラーを作るのであれば、そのゲームがどれくらい売れているかが非常に重要になってくるわけですが、それ以前にハードが十分に普及していることが前提になりますので、

その時一番売れているハードというのは、そういうことをやりやすいということになります。

--PlayStation では初期のコントローラーにアナログスティックが入ってきて、その後振動機能が入ってきました。そういう意味では最初とアナログ機能と振動機能が追加されて、同じハードの中で 2 回も変わっています。振動機能のデュアルショックについても先ほどのアナログと同様に最初に技術の開示があって、なのでしょうか。

大神：確かその辺は結構突然にやってきたような記憶がありますね。ただ、その時点で弊社として緊急に追従していかなくてはいけないという背景はありませんでした。当然コントローラーに新しい要素がつけば、それに対してソフトメーカーさんがゲームにその要素を取り込んでいくかどうかを検討されて、ゲームに反映されていくのですけども、まだ、アナログの要素の無いコントローラーの方がはるかに沢山売れていた時代ですので。まだ必要性が十分に認知されていないというか……現実的にそういうものを必要とするゲームが出てきていない状態ですので。

--「コントローラーがゲームのおもしろさを引き出す」という認知が一般にされた段階で売れ始めるということでしょうか。

大神：そうですね。その後もアナログ要素の無いコントローラーというものが売れ続けていますので。そういうことを考えると、実際に遊ぶソフトにその機能が必要かということが判断材料になっていることは間違いないですね。

--ちなみに初代 PlayStation の時代で、一番売れたコントローラーの形状、ジョイスティック型あるいはコントローラー型、片手型とか含めると、どういったものなのでしょうか。

大神：やっぱり通常のデジタルのタイプ、アナログ要素の無いコントローラーですね。これが一番売れています。弊社のコントローラーの全体の歴史の中でも相当上のほうを占めていると思います。

2.6.4　無線式・傾き検知機能時代

--現世代機では、傾き検知の機能が PlayStation3 と Wii で入ってきました。

大神：技術的なトレンドがデジタルなボタンからアナログに変わって、振動が入って、

現在は傾きもありますね。

--コントローラーとして、大きな変化というところでは、ワイヤレス化も挙げられます。ワイヤレス化は、任天堂のゲームキューブでちょっとでて、純正コントローラーとして販売されたもので、ワイヤレスになったのは Xbox 360 からですね。

大神：そうですね。標準のコントローラーとしてワイヤレスの機能が付くというのがある種衝撃的でした。

--御社でも、PlayStation2 時代にワイヤレスコントローラーを出されています。

大神：弊社の PlayStation2 のワイヤレスコントローラーもそれを作るのに結構苦労しました。こういう流れになるんだとそこではっきり認識しましたね。

--御社の PlayStation2 ワイヤレスコントローラーはどんな規格を使われていたんでしょうか。

大神：あれは普通に無線の規格を使っています。今のものと基本的には同じですけど、ただ周波数帯がちょっと違うという感じですね。

2.6.5 製造コスト

--コスト面についてお話をお伺いします。製造原価といいますと、もちろん大量生産を行うことでコストは下がっていくものですが、ハードウェアとして製造していくにあたって、コストの波はありましたか？

大神：もちろん、弊社としてもコストダウン等を重ねてきていますので、その時々では一番原価を抑えてやっているつもりではいますが、現在の製品が一番コストはかかっています。

--それは、ハードウェアメーカーへのロイヤリティを計算しなくても、でしょうか？

大神：それとはあまり関係ないですね。単純に使用している部品の数も違いますし。使っている要素も全く違いますので。

--部品点数でいうと、最初のジョイスティックの時代から何倍、何十倍くらいになっているということですか。

大神：そうですね、ケタが変わっていますね。

--今ですと、例えば Xbox 360 ではボイスチャットのユニットが入っていますから、かなり多くなっているわけですね。

大神：そうですね、Xbox 360 に関してはそういう機能がスタンダードで付いていますので、部品は非常に多いです。

--ちなみに Wii と PlayStation3 と Xbox 360 の中で一番コストが低いのは Wii のコントローラーになるのでしょうか？

大神：残念ながら詳しくは申し上げられません。部品の数だけでは一概にコストを判断できませんね。

--ライセンスを受けた製品を発売する前に、当然ライセンス元の企業さんに提出するわけですね。

大神：もちろんです、それは。ライセンスを頂くイコール、ご承認を頂かなければいけないですから。単純に動けばいいというだけではなく、色々求められている基準がございますので、その基準をクリアしているのかということを証明して、それをチェックしていただいています。基準のハードルは年々上がってきているというのが現状です。昨今の安全性への関心の高まりに対して各ハードメーカーさんがご自身の製品に対してのクオリティを高めておられますし、我々サードパーティーも同じように答えていく必要があると考えています。

--海外向けの商品では、国内のレギュレーションとは異なるものでしょうか。

大神：地域によっては国内と異なる規定があります。ですが、国内で出しているものに関しても同じような基準で作ることもあります。

--その地域特性によってパーツを変えたり、方向キーの部分の強度を変えたりはされていますか？

大神：基本的に共通です。

2.6.6　製造工程

--御社の製作工程は、回路設計から最後のモックアップまで全部専門スタッフが作業にあたっているのでしょうか。

大神：はい。製作工程は、まず企画ありきでスタートします。それから、それが実際に売れるかどうかというマーケティングをして、その上で設計を行いまして、あとは製作、その製造の準備ができて製造に入ります。全体の流れとしては平均的なに 6 ヶ月くらいの時間がかかっているのではないかと。最近は短縮していく傾向にありますけれど。

--最初の企画のときは技術者でない方からも提案があるのでしょうか。

大神：そうですね、企画がどこから上がっていくかというのは弊社に関していえばどこからでもよいという形になっています。当然、開発をやっている人間からも出ますし、営業を行っている人間からも出ます。それ以外の事務職を行っている方から出てきてもそれは企画として成立しないわけでないですから。誰でも提案できるような体制になっています。

--回路部分の進化というというのは大まかにどういった形で進んでいったのでしょうか。昔は回路チップだったのか、最初から IC だったのか。

大神：かなり早い段階から IC で処理する段階に移行する形にしています。それがないと作れないものもあります。

--IC は独自で設計開発されている？

大神：そうですね。

--それ以降の集積回路の変遷といったものはございますか？　複雑になったのは IC に入れ込んでしまう感じでしょうか。

大神：そうですね、いわゆるセンシングをする部分が増えてきていますので、最近でいうと傾き検知ですとか。それは単純に IC がもともと持っている機能として使うものも必要でしょうし、それをやるための専用のものも必要ですね。

--基板上にいろんなセンサーに対応するモジュールが増えると。

大神 : そうですね。IC の中でやることというのは複雑にはなってきています。それが急激に変わってきているということはあまり無いと思いますが、要素が増えるたびにその分を追加していきます。当然 IC 自体の性能もあがってきていますので。そういったものにどんどん切り替えて設計しなおしています。

--PlayStation 時代に入力を記録できるタイプのスティックを出されていましたね。あれは操作部分の基盤の部分とコマンドの記憶装置、入力装置は別のものだったのでしょうか。

大神 : 同じですね。1 つの IC で確かやっていました。当時記憶できる容量というのは相当少なかったと思いますが、現状でもしやるとすればそういうものが多分数十分単位で登録できるのではないかと思います。もしリアルタイムに動きを記憶していくことができるのであれば、ある程度ゲームをクリアするまでの工程を全部とっていくことも技術的には可能になっていると思います。

2.6.7 社内教育

--技術レベルがどんどん上がっている現在では、採用される人材とかも高度な大学教育とかを受けられた方をになっていくのでしょうか。

大神 : 弊社では、もともとその技術を持っている方を採用してという流れもありますが、内部で育てていくような形や、新しい技術があったら内部で新しい技術を取り入れてといくことが多いですね。あとは単純に社内ではできない部分ですと、そういう技術を持っているところにお願いしたり。技術的な部分をすべて内部でまかなっていこうという流れではないです。

--徒弟制度のような、後輩の方が先輩を覗きながら覚えていくスタイルが多いのでしょうか？

大神 : そうですね。そういった部分はまだあると思います。

--やはり皆さん HORI のコントローラーを触られてきて、作りたいという方が。

大神 : そういう者もおりますし、まったく興味が無くていきなり入ってくる者もおります。そこは多種多様ですね。ただ、好きでないと現実的にはそういう仕事できませんので。たいていの場合はそういったものがある程度興味があってやっていると認識しています。

2.6.8　今後の変化

--今コントローラーがどんどんアナログ化していって、ワイヤレスになって傾き検知が入って、と徐々に進化している中で、今後技術分野から見た「ハードウェアインターフェースの予測」は可能でしょうか？

大神：難しいですね。正直に申し上げまして、ファミコンの時代から今に至るまで劇的に変わったという要素は恐らく無いんです。我々の想像できる範疇で動いていますので。そこがいきなり大きく切り替わるということは考えにくいなと正直に思っています。少しずつ色々な要素が取り入れられて変わっていくだろうなとは考えていますけれども。それはイコール我々の方からご提案していかなくてはいけない部分ではあります。

ソフトを作っているメーカーさんから「こんな要素があったら面白いです」という形でのご提案があっても面白いですし。後は実際にハードを作られているメーカーさんが戦略的に新しい要素を取り入れられている部分もありますので……そういうことを非常によくやってらっしゃるのが任天堂さんですけど。そこに関しては我々は若干流されている部分でもあります。ただ今後に関してはできるだけ色んなことをご提案していきたいなと考えています。

--2009 年の E3 で、Xbox 360 の Project Natal や、PlayStation3 のリモコン型コントローラーなど、脱コントローラーパッドの流れがありました。

大神：面白い試みだとは思います。当然ハードウェアの性能が上がってくることによって、ソフトウェアができることの幅が広がってきていますので。本来、3D のキャラクターがこんなにバリバリ動かせるようになるというのはその当時は考えていなかったでしょうし。そういったものがもっと簡単に作れるようになってくれば色々なセンシングの要素を使ってゲームをおもしろくしていこうという流れは増えていくと思います。

--Project Natal の様なコントロールデバイスが普及してくるような時代、カメラ型デバイス、あるいはリモコン型デバイスに対してどうアプローチしていこうかとお考えですか？

大神：難しいですね、まったく新しい仕事を始めなければならないかもしれないですね。形は違いますけれど紛れも無くコントローラーで、インターフェースですから。

ご提案できることは色々あるんじゃないかなと思います。

--ジョイスティックとはもう完成されたプロダクツと思っていいのでしょうか。ボタンが増えた、あるいはスティック部分の検知方向が増えた、ということを除けば、大きな変

更はほとんど加えられていないイメージがあります。

大神：日本のアーケードゲームの市場が存在している限りあのスタイルはほぼ半永久的に続いていきそうな気がします。

パーツ単位での変化はありますけれど、基本的には同じものですから。インベーダーの時代から何が変わったかというと、中身が変わっているところは基板上でやっていることが変わっているだけで。いわゆる我々が手を触れる部分、レバーにせよボタンにせよそこの部分が変わっているということは無いです。これは一度作ってしまって、それをベースに動いているということだけだと思います。どうしても今度そこにアナログ的な要素が必須になってくるということであればまた形は変わっていくかもしれませんね。

--アーケードゲームメーカーさんから御社に何かアプローチはありませんか？

大神：実はあまりないんです。今はあのスタイルで完成されている部分が大きいと思います。弊社として、ジョイスティックという市場に対してどういう風な提案を出していくのかは今後の課題だとは思います。

どちらにしても我々はそこの市場にあるものを使ってそこの資産を使ってお仕事させて頂いておりますので。そこを作り出す側にあがっていく必要性はあると思いますね。

写真 2.7-05　リアルアーケードプロ

2.6.9　ユーザー層の変化

--御社からの視点として、ゲームユーザーの変化は感じられたことはありますか？

大神：ゲームユーザーが変化していると感じている部分は確かにあります。本当にそのゲームに没頭してやられるユーザーさんというのは少し人口的には少なくなってきているかなと。年齢も上がってきていますし。ゲーム以外の要素というのも生活の中に入ってきていますので。楽しみ方の幅が広がってきているのがひとつの要因だと思いますけれど。そういった意味ではゲームで楽しめることというのをもっと幅を広げていかないといけないかもしれませんね。

--ゲームをプレイする人の中で、一番下の年齢層の方っていうのはどういう風に入ってくるのでしょうか。

大神：これはご家庭の環境によって全く違うと思いますが、昨今はもともとゲームのある環境で生まれてこられる方が多分ほとんどだと思いますので、多分「ゲームがそこにあること」による刺激っていうのはそんなに強くないですよね。

我々は途中から「凄い刺激的なもの」としてゲームが入ってきましたので。そういった意味合いではゲームに対する価値観は全然違うと思うんですよ。今の時代の子どもたちっていうのはある種「それがもともと楽しいものなんだ」という認識が無いものから始める。例えば「お父さんお母さんがやっているから自分もやってみたい」ですとか。もしくは自分の近い世代の人たち、兄弟、友達が始めることによって巻き込まれていく流れだと思います。

一番正常であってほしいのは何か核となる面白いゲームがあって、そこに子どもたちが集まっていってコミュニケーションが生まれて、広がっていくという流れ、これが一番自然だと思います。そういったところに非常に力を入れているのが任天堂さんだと思うんですが、導入の部分で「ゲームってちょっととっつきにくいな」とか、「難しくて向いていないんじゃないか」とか、そういう壁を作ってしまった人がたくさんいるんではないかなと思いますね。

我々の世代では「ゲームがへたくそでもとりあえず付いていかなきゃ」という部分がありましたけど、最初からゲームと自分を切り離してしまう人たちがもしかしたらいるような気がしていますね。

--ジョイスティックを買う方はある程度上手い、年齢の高い方が買われているのでしょうか。

大神：そうですね。ある程度操作がうまくなってこないと必要性が感じられないものではないでしょうか。

--ゲームにおけるインターフェースについては、昔からそうですが、最近は特にインターフェース黄金時代といいますか、ゲームの中でインターフェースが占める割合というのは飛躍的に大きくなったと思います。ゲームの楽しみそのものにインターフェースの性能が直接関わっているような気がします。そういった変化というのは感じられますか。

大神：顕著な例で言えば、Wii のコントローラーがそうですね。センシングの部分がきちんとゲームのほうに反映されないと、ゲームが成立しないことになりますので。そういった意味ではそれを開発する上では相当苦労されていると思います。

また、ハードメーカーさんがそういったことを積極的にやられているということがゲームの幅を広げている一つの要因になっていると思いますので。我々はそこに助けられていると言うのもあります。

昨今は、一つの製品のライフサイクルがどんどん短くなっています。以前は一つコントローラーを作れば､数年間そのままスタイルを変えずに売れるという時代がありましたが、携帯ゲーム機が入って、アクセサリがメインになってきたことによってさらにそれが加速して。出した製品が数ヶ月すると古いものになってしまう。おそらく流通さんや、お店さんはその点について、もっとも強く感じられているのではないかと思います。

2.7　開発工程管理

田村　祐樹（株式会社ネバーランドカンパニー）

2.7.1　開発工程管理とは

(1)　開発工程管理とは？

開発工程管理とはソフトウェア開発における開発の構造管理のことである。広義な意味でいえば開発において必要となるあらゆる作業、ないしは人材を管理することを意味するが、ここでは開発における基礎的な流れを管理することを焦点において解説する。

開発工程はよく開発プロセスとも呼ばれる。そもそも開発プロセスといった姿の見えないものを考える必要があるのだろうか？

まだゲームが一人で開発できる時代にはこうした開発工程管理などはほぼ必要なかった。

だが、現在では開発規模が大規模になるにつれ、膨大な量のソースコードや、データ、様々な工程が発生することにより、これらをうまく処理できなければ開発効率の低下、ひいてはプロジェクトの失敗を招くこととなる。

開発工程を改善するための取り組みはゲーム業界だけでなく、様々なソフトウェア開発において問題となり、その問題を解決するために数多くの方法論が考え出され、ツールが作られた。

故にツールは方法論が実践的な形となったものであることが多い。様々なツールにはそれぞれ思想があり、どのように使われるべきか考え抜かれている。

これらツールは個人での開発にも役立つものであり、現在では個人であっても開発工程を管理することに利点がある。その知識や経験は集団で作業する際に役に立つものであり、かつまた、個人の効率の改善は全体での効率の改善に寄与するためである。

この章では主に開発工程管理における

- ビルドツール
- バージョン管理システム
- バグ管理システム
- 継続的インテグレーションツール
- 軽量な開発プロセス

に対して言及する。

これらはこれからの全てのゲーム開発、ないしは全てのソフトウェア開発において不可

欠なものである。

実際の開発においてはこれらと盲目的に使用するのではなく役割と成り立ちを考え、柔軟に組み合わせて使用する必要がある。ツールに使われるのではなく、効果的にツールを使用できるようになることが正しく開発工程を管理することにつながる。

(2)　なぜ開発工程管理は必要とされたか？

ゲームが個人、ないしは二、三人で開発されていた時代、プログラムのソースコードも短く、グラフィックやサウンドといったデータリソースもそれほど膨大ではなかった。

プログラムに関してはそもそも個人が一つのファイルに収まるように書き、数千行あるファイルが数個といった開発が一般的であったし、グラフィックデータに関しても色数が少なく、画面も小さかったため、画面を埋め尽くすとしても 16Kb～32Kb 程度のサイズしか消費しなかった。そもそも画面解像度が今の 35 倍以上小さかったためだ。そもそも販売形態である記憶媒体に入りきらないため、個数も要領も少なかった。

ゆえにこれらはフロッピーディスクなどの記憶媒体を介した手渡しで行き来し、組み込みも手作業で行われるような時代があった。

しかし、現在のゲーム開発ではある規模のゲームを開発しようとしたとき、ソースファイルが数百個、グラフィックリソースはモデルやモーション、画像データなどを合わせて数万個といったこともおかしくはない。

画面を埋め尽くす画像データを用意するだけでも 16Mb～32Mb といった感じで K から M へと約 1000 倍のデータサイズが必要となっている。もちろん、種類も幅広くなっているためプロジェクト全体のデータサイズでみれば 1Gb～数 10Gb といったようなサイズであることも多い。また、動画のようなデータを扱うことも増えデータサイズは増大する一方である。記憶媒体も DVD や BD といったように記憶容量は増大する一途を辿る。

これらを手作業で受け渡しすることは最早不可能であり、何らかの方法で正しく管理する必要がある。

これらを改善するために作られたのがこれから紹介するツール類である。おそらくは今現在においてもこうした便利なツールを使わずに作業を行っている環境があることは想像に難くない。この章がそれら改善に僅かでも寄与すれば幸いである。

(3)　そもそも開発工程管理に正解はあるのか？

答えは「ない」。

開発工程は様々な組織、ないしはプロジェクトによって大きく異なる様相を持つ。故にそれらすべてに当てはまる万能な工程管理は存在しない。なので、ここでスタンダードとなる開発工程を紹介することもしない。

「銀の弾などない」

1986 年、フレデリック・ブルックスが著したソフトウェア開発における論文の中で述べられた有名な言葉である。

この言葉の意味することは、「ソフトウェア開発という狼男」を「銀の弾丸」として一発で仕留められるようなものは存在しないということだ([1])。

魔法のようにすぐに役に立ち、プログラマーの生産性を倍増させるような技術や実践方法は今後 10 年は現れないだろうという主張なのである。

そして、1986 年から 24 年経過した 2010 年の今でさえ、そういったソフトウェア開発における特効薬は開発されていない。

しかしながら、開発工程管理をよりよくしていくことはできる。過去に比べ、実に多くのツールが生まれ、これらは開発に適応することができるようになった。我々に「銀の弾丸」は与えられていないが、我々は歴史から育まれたツールという武器を手に取ることはできる。

(4) ゲームにおける開発工程

環境によって様々な違いはあるだろうが、多くの場合、ゲームを構築するために必要なのは

- データ作成、プログラム作成における仕様策定
- 各種データの生成（グラフィック、サウンド、スクリプト、イベントデータなど）
- ソースコードにおける実装
- 各種データの最新版取得
- 各種ソースコードの最新版取得
- 取得したデータのビルド
- ソースコードのビルド
- 実行ファイルの生成
- 開発環境での実行

となるだろう。

開発者の日々はこれらの繰り返しであり、こうした繰り返しを効率化するためにツールが存在する。各種素材、及びソースコードの工程を管理することは他のソフトウェア開発と大きく違わないため、開発工程管理のためのツールはゲーム開発においても実によく働く。開発工程の一例を図にしたものを図 2.7-01 に示す。あくまで一例であり、これが絶対ではないが、多くの場合の基本形となる。また、図 2.7-02 には各ツールの分類と代表的なツール、及びその関係を示した図をのせた。

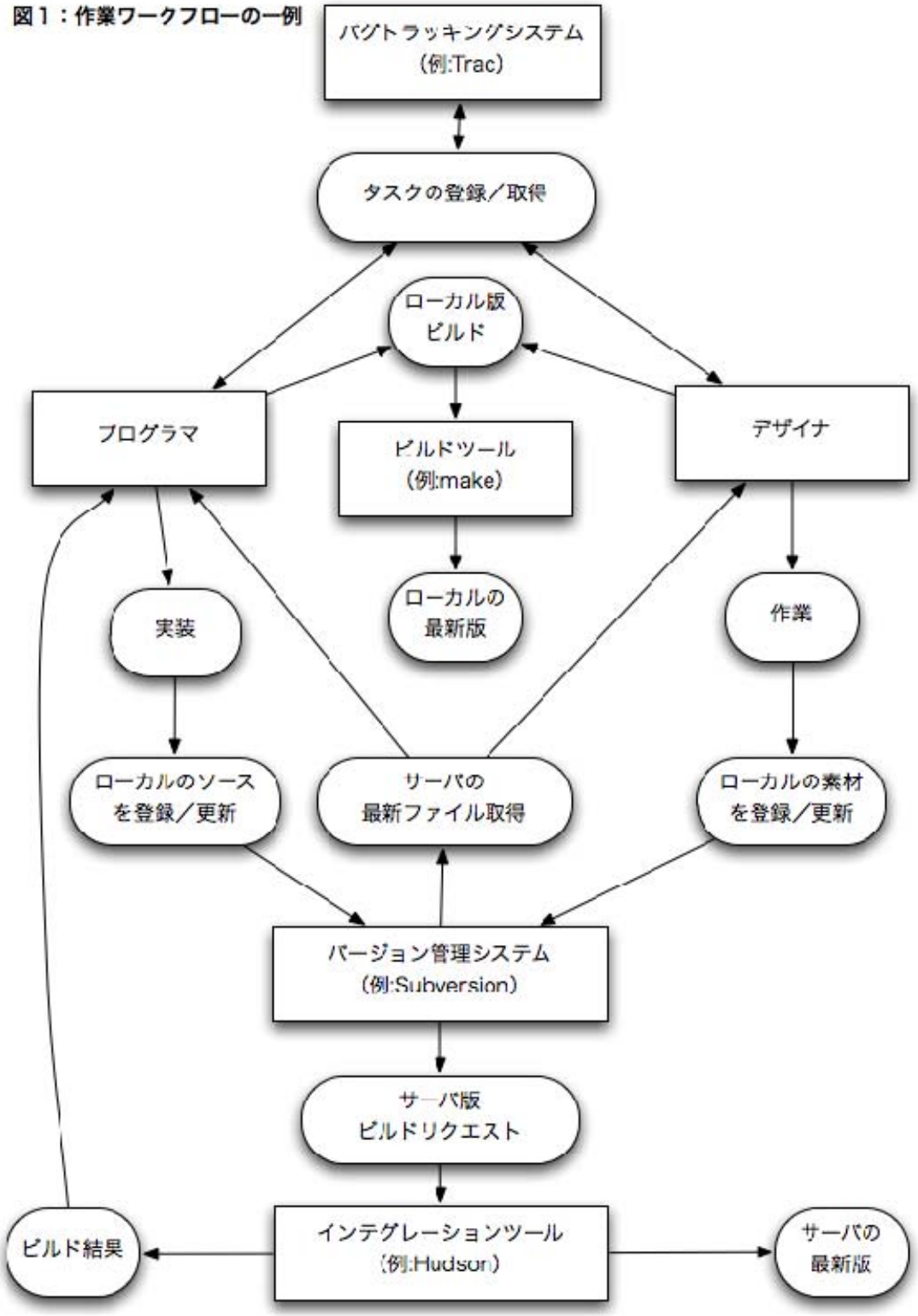


図 2.7-01　作業ワークフローの例

図２：ツール分類および関係表

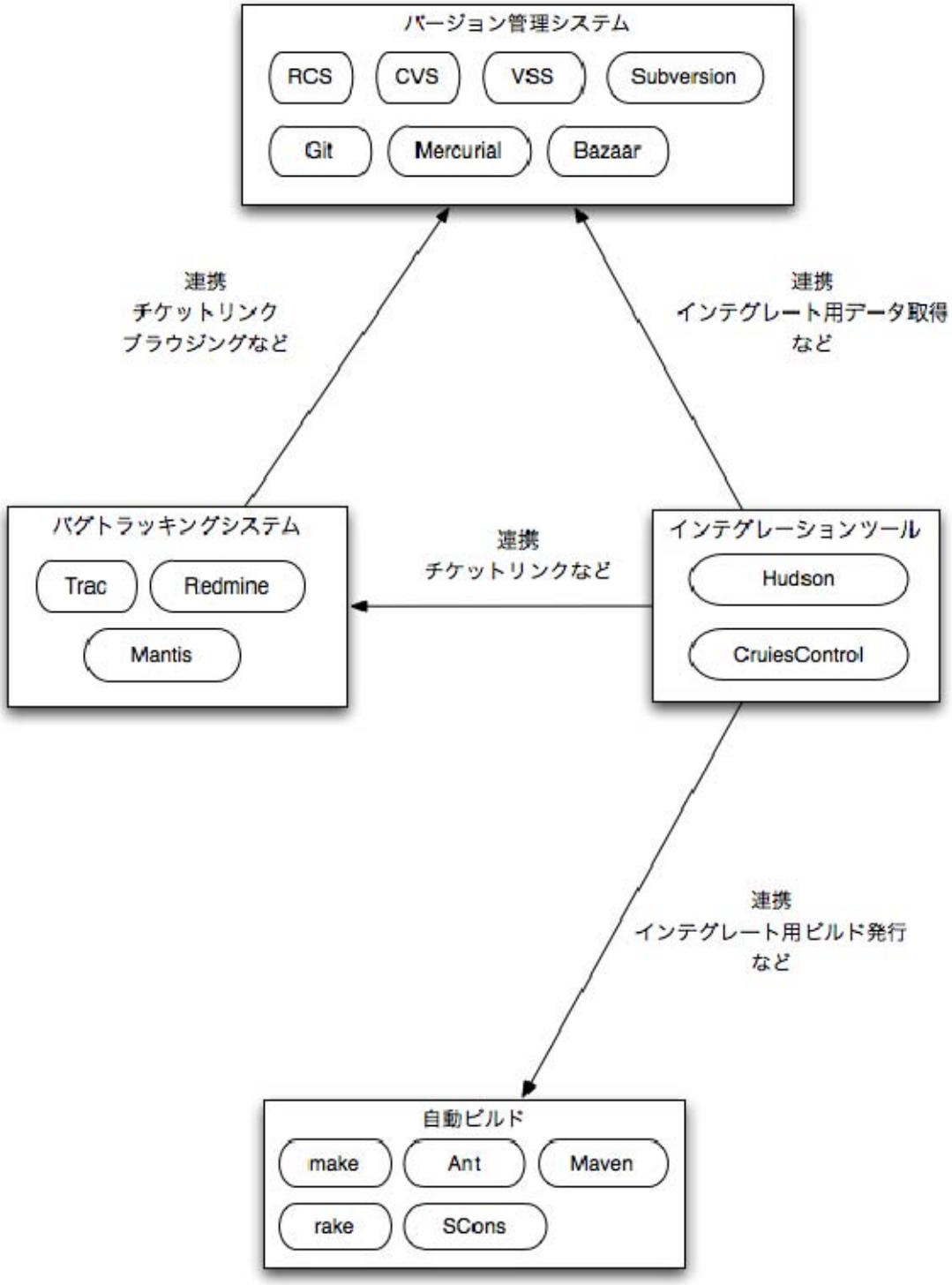


図 2.7-02　ツール分類および関係表

2.7.2　ビルドツール

前章では主に開発工程管理について必要な工程について述べた。この章では開発工程における「ビルド」を自動化するためのツールについて述べる。

(1)　ビルドとは何か？

ここでのビルドは様々なデータをゲーム上で動作するデータに変換すること、及びソースコードを実行できる形にすることを意図している。

たとえば、3D のモデルデータは多くの場合そのままではゲームに利用できる形にはならない。これらを描画可能な状態に変換したり、またシナリオやイベントを構築するためのデータを実際のゲーム上で動作するようにすることを含んでいる。

(2)　そもそもビルドを自動化する利点とは何か？

ビルドを自動化する、といっても様々な意味合いがこめられている。

- ソースコードを実行ファイルにすること
- ゲームの実行に必要なデータを最新の状態にまとめること
- 様々なチェックやテストを実行すること
- ドキュメンテーションを生成すること

これらを自動化することによって製品の品質は向上する。なぜか？

- 所要時間を短縮できる
- 冗長で無駄なタスクを排除できる
- 手順ミスなどのヒューマンエラーを排除できる
- 特定の人物に依存しないで各個が実行するファイルを生成できる

たとえば、ビルドを手作業で行えばミスが起こり原因不明のエラーによってゲームが起動しなくなるかもしれない。こうした原因不明のエラーが何らかの手順違いなどで引き起こされたとしたら追跡するのは困難であり、それを誰かに問い合わせることはお互いにとって時間の無駄でもある。

これらを自動化することで、すべての環境で手軽で同じビルドを行うことは開発工程管理において必須なのである。

下記のツールを利用することによって、これらは簡単に自動化できる。

(3) make

make は 1977 年に作られ、今なお現役で活躍するツールである。

make 自体は何かを生成するものではない。Makefile という書式に従い、データを生成するためのルールを記述すると、そのルールに従ってファイル生成を行ってくれる。いわばファイル生成のためのルールを処理するツールである。我々は成果物を得るためのルールを記述すればよい。

指定したファイルを指定したルール（例えば指定のツール）で生成してくれるだけでなく、複雑に関連しあったファイルの依存関係を解決してくれる。

たとえば、A というファイルを生成するために B というファイルが必要である場合、A というファイルが存在しないか、B というファイルが更新されたときにだけ A から B を生成する手順を踏んでくれる。

これは、A から B、B から C といった多段に依存が存在し、順番に生成しなければならないケースや、A から B、C、D、といった複数のファイルを生成する際にもうまく働く。逆に B、C、D から A を生成するといった場合にもだ。

UNIX 系のソフトウェアやオープンソース系のソフトウェアビルドにおいても現役であり、未だ FreeBSD といったサーバーOS のソフトウェアパッケージ管理にも用いられている。様々な言語で書かれたソフトウェアをビルドするのによく働くが、とりわけ C/C++で開発するプロジェクトなら make を無視することはできないだろう。

(4) make 以外のビルドツール

make のようなビルドツールは各プログラミング言語上において適性にあった機能を持って用意されることも多い。Java であれば Ant や Maven、.NET であれば NAnt や MSBuild、Ruby であれば rake、Python であれば Scons といったようなものが代表的なものとして上げられる。

Ant や Maven は Java で構造管理をしやすいように xml で記述されており、Maven にいたっては必要なライブラリを自動的に指定ネットワークからダウンロードする機能もある。rake や Scons は設定ファイルの中に Ruby や Python のコードをそのまま埋め込むことができ、より柔軟な記述を可能としている。また、rake や Scons は C/C++のコードをビルドするためのサポート機能も備えている。

場合によっては make だけが適切な選択肢ではない。

(5) Makefile のルールを自動生成するツール

あまりにも膨大なファイルを扱ったり、様々な環境でうまく動くことをしなければならないときに、Makefile に記述されたルール自体があまりにも複雑化してしまうケースがあ

る。この場合に、Makefile のルールを自動生成するツールといったものも作られている。
これは Autotools などがあり、多種多様な環境に対応するために使われている。

(6) IDE

統合開発環境である IDE(Integrated Development Environment)でもソースコードの自動ビルドが行える。ただし、この場合はソースコードだけしかサポートしないことが殆どである。
ビルドの前後に何らかの実行プロセスを立ち上げることは可能であるので、make などと併用されることがある。

2.7.3 バージョン管理システム

この章ではバージョン管理システムについて述べる。ソフトウェアを構成する個々のファイルをバージョンを管理することで我々はいつでも過去に対するどの状態にでもアクセスすることができるようになる。
バージョン管理システムは SCM と呼ばれることもあるが、これはもともとは「ソフトウェア構成管理（Software Configuration Management）のことである。

(1) バージョン管理とは？

「どのファイル」が「誰によって」「いつ」「どのように」変更されたかを管理することである。ソフトウェアの変更の履歴を適切に管理することはデータの流れを管理することにもつながるため、工程管理における重要な要素となる。

(2) そもそもバージョンを管理することの利点は何か？

筆者は実際に LAN 上のネットワークに各々が自分のソースファイルをコピーし、担当者がそれをまとめた後に圧縮ファイルにて皆に配布するという環境で作業を行った経験がある。
このようなケースで常に問題になりうるのは「何が変わったのかわからない」ということだ。他人のソースに依存した処理を記述していた場合に動作が変わった場合もどのように追跡すべきかわからない。もちろん、一つ前の状態をとっておき、それとの差分を見ることも可能ではある。
が、結合をするたびにその履歴をとっていたのであればその労力は無駄である。バージョン管理システムを用いるだけで、「どのファイルを」「誰が」「いつ」「どのように」変更したかを容易に追跡できるし、加えてすべてのユーザーが「なぜ」を履歴にコメントとし

て残すことができる。

また、複雑な作業工程の中で先祖がえり（新しいデータを古いデータで上書きしてしまうこと）が発生してしまうこともあれば、新しいデータにしてみたがやはり昔のデータの方がよかった、ということもある。こうした問題を解決するのに「いつでも」過去のバージョンを取り出せれば大変便利だ。

よって、バージョンを適切に管理する意義がある。

(3) バージョン管理システム

バージョン管理システムはソースコードやその他のリソースの履歴を管理し、いつでも過去の状態に戻せるようにするために必須となるツールである。

また、「いつ」「誰が」「なぜ」「どんな変更を行ったか？」を管理することで、問題の追跡が容易になり、不具合さえいつでも再現できるようになり、また、いつでも前のバージョンに戻すことができる。バージョン管理ツールの目的は開発工程管理にとどまらず、原稿の管理などにも用いることができる。

ここでは、代表的なツールである RCS と CVS、Microsoft Visual SourceSafe、Subversion、Git、その他、について述べるが、まず簡単な用いられ方について述べ、次に特性として重要となる「ロック式とマージ式」「集中型と分散型」について述べる。

ここでは管理ツールのデータベースを「リポジトリ」と呼称し、これにファイルを預ける行為を「コミット」、引き出す行為を「アップデート」と呼称する。

(4) リポジトリとコミットとアップデート

軽く前述したが、管理ツールがファイルを管理するデータベースを「リポジトリ」と呼ぶ。

全ての履歴はここに蓄えられ、開発者はここに向かってファイルの変更などを通達することになる。

「リポジトリ」からファイルを受け取ることを「チェックアウト」と呼び、これは多くの場合、最初に一度だけ行う。

以後は、「リポジトリ」に加わった変更を取り出すことを「アップデート」と呼ぶ。これにより、ローカルのファイルはリポジトリと同じ状態になる。

そして、ローカルのファイルを変更し、皆に反映するために「リポジトリ」にファイルを預けることを「コミット」と呼ぶ。

こうした仕組みによって、開発者はローカルのファイルを編集しつつ、全員のソースコードやデータを最新に保つのである。

ただしシステムによってはコミットをチェックイン、アップデートをリフレッシュなど

と呼称することもある。

(5)　ロック式とマージ式

ファイル管理の方法について、バージョン管理システムには二通りの方法がある。ひとつがロック式であり、もうひとつがマージ式である。

(a)　ロック式

ロック式は管理されているファイルを「ロック」することができる。このファイルはロックした人がそのロックをはずすまで誰も触ることができない。一つのファイルを複数の人間が触るケースがある場合に一人の人間がそのファイルを長時間ロックするとそれが作業効率に対するボトルネックとなるため多人数の作業には向いていない。

(b)　マージ式

マージ式は管理されているファイルをいつでも誰でも自由に編集することができる。ロック式と異なり、編集に制限はない。ローカルで変更したファイルをサーバーに対して戻す際にマージを行い、変更を適切に反映しようとする。一つのファイルを複数の人間が触る際に効率よく編集できるため、多人数の作業に向いている。ただし、マージ可能なのはテキストファイルのような差分が意味を持つファイルに限られる。バイナリデータのようなマージが不可能なものに関しては擬似的なロックで対応することもある。

(6)　集中型と分散型

リポジトリというファイル履歴管理情報の持ち方について、バージョン管理システムには二通りの方法がある。一つが集中型であり、もう一つが分散型である

(a)　集中型

集中型は単一の場所に一つだけリポジトリを有する。すべてのユーザーはそのリポジトリに対してコミットやアップデートを行う。

(b)　分散型

分散型は単一の場所にあるリポジトリのコピーをすべてのユーザーがローカルに保有する。コミットはそのローカルに対するリポジトリに対して行い、その他のリポジトリに対するコミットをプッシュと呼び、アップデートをプルと呼ぶ。

普段のコミットが全員に対して反映されないため、気軽に行うことができるようになる。集中型と比べ柔軟な管理が行えるのが分散型の利点である。

(7) 代表的なバージョン管理システム

(a) RCS と CVS

RCS は非常に初期に作られたバージョン管理システムである。未だに保守はされており、多くの Wiki エンジンの履歴管理などに使われているが、大規模のプロジェクトを管理するのには向いていない。CVS は RCS を元にして作られたものであるが、幾つかの欠点を有するため、今日では多くの新しいバージョン管理システムに置き換えられてしまった。

ただし、この欠点を元に多くの改良点が生まれた。

(b) Microsoft Visual SourceSafe

ロック式で集中型。マイクロソフト社が提供するバージョン管理システムである。同社の Visual Studio と連携して機能する。

以前はリポジトリサイズの制限があったり、HTTP を介した更新や取得ができず、また Unicode の扱いに問題があったため、柔軟な扱いが不可能であったが最新のバージョンでは修正されている。Visual Studio との連携時に好んで使われるケースが多い。Windows 上でしか動作しない。

(c) Subversion

マージ式で集中型。過去に広く使われていた CVS の問題点を解決すべく作られた直系とも呼べるバージョン管理システム。その経歴から CVS と置き換えられて使われることも多い。

CVS では対応していなかったディレクトリの移動や削除に対応したため、柔軟な運用が可能になっている。専用の書籍が幾つもあり、様々な OS の上で動作するために非常に扱いやすい。サーバー側でも様々なカスタマイズを行うことができ、コミットに反応して何らかの処理を走らせる、コミットしてはいけないファイルをサーバー側ではじく、などといった処理を記述できる利点がある。詳しくは書籍を参照されたい。([2,3])

(d) Git

マージ式で分散型。大規模になった Linux カーネルを開発していくなかで、これらを効率よく管理できるバージョン管理システムが必要とされ Linux カーネルの開発者でもあるリーナス・トーバルズが開発した。

機能としてはバージョン管理システム BitKeeper と Monotone が元となっている。大規模開発において高速でスケーラブルになるように設計されており、注目が集められている。

近年では書籍も発行され、多くの OS 上で動作するため非常にとっつき易い。2005 年より開発を受け継いだ濱野純氏が書かれた“入門 Git”が有名である。([4])

分散型の代表として述べたが類似の生い立ちを持つものに Mercurial が存在する。加えて、分散型と集中型を選べるものとして Bazaar といったものも存在する。

(e) その他

ゲーム開発では、Alienbrain、Perforce といったバージョン管理システムが使われることもあるが、非常に高価である。ただし、大手の会社などでも採用実績があり、注目されている。

2.7.4 バグトラッキングシステム

この章ではバグトラッキングシステムを紹介する。BTS やバグ管理システムと呼ばれることもある。プロジェクトのバグを登録し、修正状況を追跡、管理するシステムだが、バグだけでなくタスクを登録することで開発の初期から開発の後期まで非常に役に立つ。前述したバージョン管理システムなどとも連携することができるためその役割は非常に大きい。

(1) バグトラッキングシステムをどのように使うのか？

バグトラッキングシステムは多くの場合、チケットという単位でタスクやバグを処理する。多くの場合は必要なタスクや発見されたバグを「チケット」に登録することで、それらが投稿され、どのような経緯をたどり修正されたかが履歴として残る。

これらの履歴はデータベースで管理されるため、検索や関連付けが可能で、修正された場合に関係者にメールを送信する機能などもある。

タスクやバグなどを管理する理由は記憶によって管理しようとすると人はすぐにそれを忘れてしまうからだ。「いつ、どんなバグがどういった理由で発生し、それをどう直したのか」ということを管理するのがバグトラッキングシステムである。類似のバグがでた場合にそれを追跡することを容易にする。また、それがタスクでも同様である。

バグトラッキングシステムの扱いはプロジェクトによってカスタマイズすることができるため、「これ」といった使い方はないが、もし最初であれば一般的な使い方に従うのがよいだろう。問題があればそのつど変更するのがよい。

(2) 一般的なワークフロー

バグトラッキングシステムには様々な種類があり、それぞれにそれぞれのワークフローが用意されたり、また自由にカスタマイズできるものも存在するが、ここに一般的なワークフローを示す。

- 発見されたバグ、必要なタスクの登録をする。チケットは「新規」となる。
- 担当者を決定。チケットの状態は「担当者決定」となる。
- 担当者は修正および実装を行いチケットの状態を「確認待ち」にする。
- チケットを登録した人間がその修正、実装を確認し「確認済み」にする。
- 管理担当者が「確認済み」チケットを確認し、「終了」とする。

(3) 一般的なバグトラッキングシステム

バグトラッキングシステムには商用、非商用を含め多くの種類があるが、ここでは広く使われ、導入も簡単である Trac と Redmine について紹介する。

なじみがあるものとして Mantis などといったものもよく使われており、日本語化もされているが、書籍がないためここでは割愛する。

(a) Trac

Trac はバグデータベースのほか、バージョン管理システムの Web インターフェース、及び Wiki を内臓しており、これらを連携することができる。

日本ではインタアクト社が日本語化を担当しており、日本語化されたものを配布しているほか、Trac Lightning と呼ばれる Windows 向けのインストールパッケージも存在する。

幾つかの近代的なバージョン管理システムとの連携に対応しており、プラグインなども豊富で柔軟な運用が行えるようになっている。日本でも書籍が幾つか出版されており、運用などの説明やヒントはそちらに詳しい。見た目はシンプルだが、プラグインでテーマを追加したり、自前で見た目に手を加えることもできる。([5], [6])

(b) Redmine

Trac に大きく影響を受けたバグトラッキングシステムである。Trac とほぼ同等の機能を持つほか、Trac がプラグインで提供している機能を最初から備える点が異なる。また、Trac がどちらかといえば素っ気無いインターフェースをしているのに対し、Redmine はリッチなインターフェースを提供する。カスタマイズや諸所設定に関しては日本語書籍なども存在するため、そちらを参照されたい。([7], [8])

2.7.5　継続的インテグレーションツール

この章では継続的インテグレーションツールを紹介する。継続的インテグレーションシステムは上記のツールに比べて「必須」と呼べる程ではなく、導入していないプロジェクトも多々ある。しかし、これを組み合わせることで上記のツールはさらに力を発揮するようになる。

インテグレーションとは「統合」のことであり、「統合」を自動化するツールである。

(1)　継続的インテグレーションとは？

チームメンバーが各自の成果物を頻繁に統合するソフトウェア開発のプラクティス（手法）である。通常、最低でも一日一度は各自でインテグレーションを行う。インテグレーション時のエラーはできるだけ早く検出できるよう、全てのインテグレーションはテストを含めて自動化されたビルドによって検証される。多くのチームが、このアプローチを用いることでインテグレーション上の問題を大幅に削減でき、動作するソフトウェアの開発をより短時間で行えると考えられている。([9])

(2)　継続的インテグレーションの利点とは？

- リスクを軽減する
- 繰り返しの手作業を削減する
- いつでも、誰にでも動作するファイルを生成できる
- プロジェクトの見える化を実現する
- 開発チームでの自信を向上させる

(a)　リスクを軽減する

統合を一日に何度も行うことで、プロジェクトに欠陥が入り込んだ事を即座に知ることができる。欠陥は誰もチェックしない場合に長く見過ごされがちで、それがやがて積もり積もって解決に時間のかかる問題となることも多々ある。

継続的に統合することは早期にリスクを解決し、ソフトウェアの健康を保つ秘訣である。

(b)　繰り返しの手作業を削減する

人間が統合を一日に何回もやっていたらそれだけで一日が終わってしまう。なので、ツールで自動化し、常に同じ方法でビルドを行う。

バージョン管理システムへのコミット毎に統合を行うこともできるし、誰でも好きなときにボタン一つで統合を試すことができる。

開発者は無駄な手作業を減らし、思考に集中した仕事を継続できる。

(c) いつでも、誰にでも動作するファイルを生成できる

統合担当者がいなくても、必要なタイミングで必要なファイルを生成することができる。これはリリースを頻繁にしなければならないソフトウェアである場合にも役立つし、そうでない場合でも単に結合のためだけに無駄に人を待機させる必要がない。

(d) プロジェクトの見える化

ツールは統合に関しても履歴を残してくれるため、統計に基づいた判断を行うことができるようになる。例えば何でビルドが失敗しているかは人間の記憶に頼るとあいまいだが、統計化されたデータならば確かな理由を見つけ出し、それを改善することができる。

(e) 開発チームの自信を向上させる

統合が自動化されることによって、問題が即座に知らされるため、そうでない場合よりも修正がしやすく、拡張や修正に対して恐れなくなる。

(3) 代表的な継続的インテグレーションツール

ここで幾つかの代表的なツールを示す。継続的インテグレーションツールは CI ツールとも呼称されることもあるので注意されたい。

(a) Hudson

Java で実装され、Servlet コンテナ上で動作するツールである。プラグインによって幅広いバージョン管理システムに対応する。日本語化されており、非常に使いやすい。

(b) CruiseControl / CruiseControl.NET

Java で実装され、Java 用としてはもっとも採用実績のあるツールである。.NET 版も経歴は浅いが、Java 版を基にしており信頼性のあるシステムとして名高い。

(c) その他

その他では AnthillPro、Continuum、Draco.NET、Luntbuild、といったようなツールが有名である。様々なものがあるので、環境によって適切なものを選択するべきである。ただし、迷うくらいならば Hudson を使うのが今なら最適な選択になるだろう。

2.7.6　軽量な開発プロセス

この章ではツールであってツールでないものを紹介する。

(1)　軽量な開発プロセスとは何か？

ゲーム開発に限らず、開発中に変化が訪れる事が常であるソフトウェア開発において、スケジュールの欠陥、要求の増大や変更、人員の離脱、仕様の崩壊や欠如、個人の生産性の差など様々な問題点が従来より存在した。

これらを改善するための絶対的な方法はないものの、無策で開発を行うことは容易にデスマーチと呼ばれる状態を作り出してしまう。

（デスマーチとは長時間の残業や、徹夜、休日出勤の常態化といった開発者に対して極端な負荷を強いるプロジェクトのことである）

これらを改善するために、幾つかの変化に強い開発プロセスが近年考案され、さらに例は少ないながらも実績を作りつつある。

軽量な開発プロセスは特に「変化に強い」ことが利点としてあげられる。

(2)　アジャイルソフトウェア開発

アジャイルソフトウェア開発はソフトウェア工学において迅速で柔軟な形で開発を進めるための軽量な適応的開発手法の総称である。

アジャイル開発は反復（イテレーション）と呼ばれる短い期間単位を計画に用いることでリスクを最小化しようとしている。一つのイテレーションは一週間から長くても四週間で構成されることが多い。この 1 イテレーションの中にチームは 1 つの機能だけを実装する。それが計画的になされなかった場合、チームは新たに機能を評価しなおすなどとして、反復を繰り返す。

広く知られたアジャイルソフトウェア開発の手法として代表的な「エクストリーム・プログラミング」と「スクラム」を次に示す。

(a)　エクストリーム・プログラミング

エクストリーム・プログラミングは 1999 年にケント・ベックらによって書籍の形で発表されたものである。こちらは日本語化されており手軽に読むことができる。([10])

Extreme Programming, 略称として XP などとも呼ばれる。

大元のエクストリーム・プログラミングは 10 から 12 個程度の実践的な手法によって構成されている。これらの手法を「プラクティス」と呼び、これらを実行していくことで開発体制を軽量化しようという手法である。

エクストリーム・プログラミングは四つの価値を重んじる。

- コミュニケーション
- シンプルさ
- フィードバック
- 勇気

である。これらは各プラクティスの根底原理とされ、開発者は決められたプラクティスをチームにあわせて柔軟に適応する。

しかし、これらは現状においてすべてを取り入れることは大変困難であるため、現段階では部分的な採用に留まることが殆どである。

ただし、それはエクストリーム・プログラミングの想定内のことであり、無闇に全てのプラクティスを押し付けることは逆にしてはいけないこととされる。

「変化を抱擁せよ」というキーワードと共に非常に関心の高まっている手法の一つである。

今までの価値観とは異なった方法論であるため、導入の際は書籍を参照されたい。プラクティスも時代や環境によって増えつつあるため、オライリー・ジャパンから出版されている“アート・オブ・アジャイル デベロップメント ―組織を成功に導くエクストリームプログラミング”が非常によい手引書になる。([11])

(b) スクラム

1993 年に登場した開発手法である。当初はオブジェクト指向設計・分析ツールの開発に用いられた。ラグビーのスクラムにちなんでいる。

開発チームはスポーツチームのように動かなければならないとされ、各個が適切な役割を担いながら同じゴールを目指す。上記で説明したエクストリーム・プログラミングの管理的な弱さを補う手法として採用されることも多い。

「自己組織化」をキーワードとしており、自発的に秩序が形成されることを目的とする。

「スプリント」「日次スクラム」「プロダクトバックログ」「スプリントバックログ」「スプリント計画ミーティング」「スプリントレビュー」といったプラクティスを持つ。

また、「スクラムマスター」といったスクラムを推進し、プレイングマネージャのような役割を持った人間を置く。

スクラムの価値は

- コミットすること
- 自分の役割に集中すること
- 隠し事はなくオープンであること

- チームメンバーに敬意を払うこと
- 勇気を出すこと

であり、

こちらもエクストリーム・プログラミングと同じように人間を主体としたプロセスの一種と呼べる。ここに全てを書くには膨大なスペースを必要とするため、XP 同様に書籍が出版されているので参照されたい。([12,13])

2.7.7 まとめ

(1) 今の開発工程管理をどうすべきか？

開発工程管理にこれ、という正解はない。

ただし、ビルドツールはソースコードやデータを適切に管理し、成果物を生成してくれる。

バージョン管理システムはプロジェクトが生まれたばかりの過去に遡ってプロジェクトのデータを全て管理してくれる。

バグトラッキングシステムは過去から現在にかけて存在しているバグとタスクを管理してくれるし、継続的インテグレーションは頻繁に結合を行い、ビルドツールとバージョン管理ツールが適切な状態にあるかどうかをいつも管理してくれる。

軽量な開発プロセスはこうしたら変化に強くなるという指針を提供してくれる。様々な現代的なプラクティスを用いて開発効率を改善することができる。

開発者はこれらを適切に利用し、残された思考しなければならない作業に集中するべきだ。

改善できる箇所を改善することが開発工程管理の前進につながる。

(2) ツールやプラクティスの導入に反対されたが？

ツールが便利だと知っているのなら反対されても導入すべきだ。本当に使えないツールなら導入後に廃れてしまうだろうし、そうでないなら以後役に立つはずだ。XP のようなプラクティスも同様で、導入せずにそれが当てはまるのかどうかを推し量ることはできない。

(3) 未来の開発工程管理はどうなるか？

「銀の弾丸はない」が示すようにこれからも突然ソフトウェア開発の工程管理が改善されることはない。ただし、こうしたツールは進化し続けるであろうし、新しい開発手法も

現れるだろう。それをどのようにしていくかはそのプロジェクト次第であり、そこに属する開発者次第である。

(4)　開発規模の増大とツールの登場

ツールはゲーム開発を目的として作られたわけではないので密接な相関関係はない。しかし、年代が進み新たなプラットフォームの登場とともにそれらが使われるようになっていることは事実であり、これはハードウェアとソフトウェアの進歩の軌跡とも呼べるだろう。代表的なコンシューマー機の登場時期と各種ツールの登場時期を図 2.7-03 に示した。

開発規模の大きさによってはおおよそであり、現在の PS3 の開発規模はファミコンの 100 倍であることもあるだろうし、Wii の開発規模は PS3 の半分であることもあるので了承されたい。それらはプロジェクトに依存するため、一概に語ることができない。もちろん、現在でも iPhone のような小規模開発もあることを留意されたい。

ツールに関しては登場時期が明確でないものは 1.0 リリースの時期を基にしている。加えて、各ツールの代表的なものとして、Subversion、Trac、Hudson のスクリーンショットを末尾につけておいたので参照されたい。

(5)　理想の開発体制に関する展望

開発工程管理は小規模開発であれ、大規模開発であれ、開発を効率よくするために使われる。これは、プログラマーを含めた開発者の気持ちを高め、開発に対する意欲を高める。ツールが何かを生み出すのではなく、ツールはプログラマーやデザイナが何かを生み出す意欲をブーストする。年代を経るごとに様々なツールができていく中、それがもっともっと身近でよりよくなればストレスは減り開発は楽しくなる。開発が楽しくなれば、作っているものをよくすることがもっと楽しくなる。

現在存在するツールをうまく使うだけで、飛行機のエコノミークラスからファーストクラスに乗り換えるくらいの快適さが得られるが、いずれはさらに至れり尽くせりの環境がやってくることは想像に難くない。それは想像するならば、行き先を指定すれば自動かつ無事故でその場所まで移動してくれるような自動車のように、自動で排除できる面倒や心配事をすべて排斥してくれるだろう。プロジェクトに対するどんな小さな変更も安全かつ迅速に成果物に反映され、それは容易に皆が知ることができるようになる。

そして、開発者は自動では排除できない面倒や心配事に注力できるはずだ。全自動自動車の例で示すならば、目的地に対する思いを存分に巡らせたり、同乗者との会話に没頭できるようになる。

(6) 最後に

いかに優れたツールが存在してもそれを使いこなすことができなければ意味がない。それと同様に、いかに優れた手法が現れようとそれを使いこなせるかどうかは人次第でしかない。

多くの近代的な開発プロセスが人を重視するのは開発工程とは人がいて初めて適切に機能するものだからである。

便利なツールが存在するのにもかかわらず、それを利用しないのはより大規模でより複雑になりつつある開発工程管理を何時までも困難にするだけだ。

ツールやプラクティスといった方法論と共に人が過去から学び成長することが開発工程管理を改善する最善の一歩である。

参考文献：

［1］Jr.,フレデリック・P. ブルックス： “人月の神話—狼人間を撃つ銀の弾はない”, ピアソンエデュケーション (2002)

［2］C. Michael Pilato, Ben Collins-Sussman, Brian W. Fitzpatrick： “実用 Subversion 第 2 版”, オライリージャパン (2009)

［3］Mike Mason： “Subversion 実践入門:達人プログラマーに学ぶバージョン管理(第 2 版)”, オーム社 (2007)

［4］濱野 純： “入門 Git”, 秀和システム (2009)

［5］菅野 裕, 今田 忠博, 近藤 正裕, 杉本 琢磨： “Trac 入門 ——ソフトウェア開発・プロジェクト管理活用ガイド”, 技術評論社 (2008)

［6］高山 恭介： “入門 Trac with Subversion—Linux/Windows 対応”, 秀和システム (2008)

［7］倉貫 義人, 栗栖 義臣, 並河 祐貴, 前田 直樹： “Redmine -もっと手軽にプロジェクト管理!”, インプレスジャパン (2009)

［8］前田 剛： “入門 Redmine Linux/Windows 対応”, 秀和システム (2008)

［9］ポール・M・デュバル, スティーブ・M・マティアス, アンドリュー・グローバー： “継続的インテグレーション入門 開発プロセスを自動化する 47 の作法”, 日経 BP 社 (2009)

［10］ケント ベック： “XP エクストリーム・プログラミング入門—変化を受け入れる”, ピアソンエデュケーション (2005)

［11］James Shore, Shane Warden： “アート・オブ・アジャイル デベロップメント —組織を成功に導くエクストリームプログラミング”, オライリージャパン (2009)

［12］ケン シュエイバー, マイク ビードル: “アジャイルソフトウェア開発スクラム”,

ピアソンエデュケーション （2003）
［13］ケン シュエイバー： “スクラム入門-アジャイルプロジェクトマネジメント”，日経 BP ソフトプレス （2004）

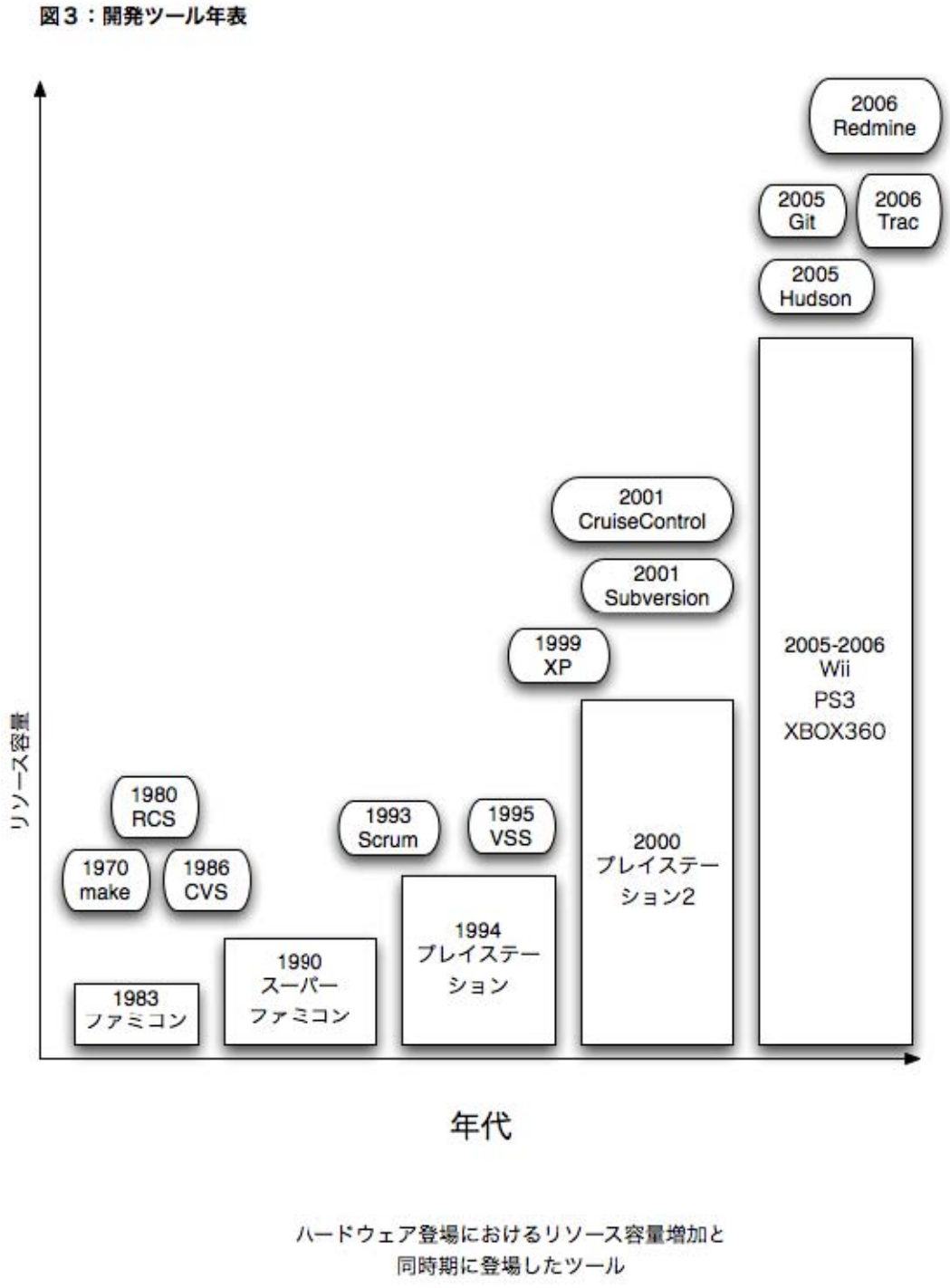


図 2.7-03　開発ツール年表

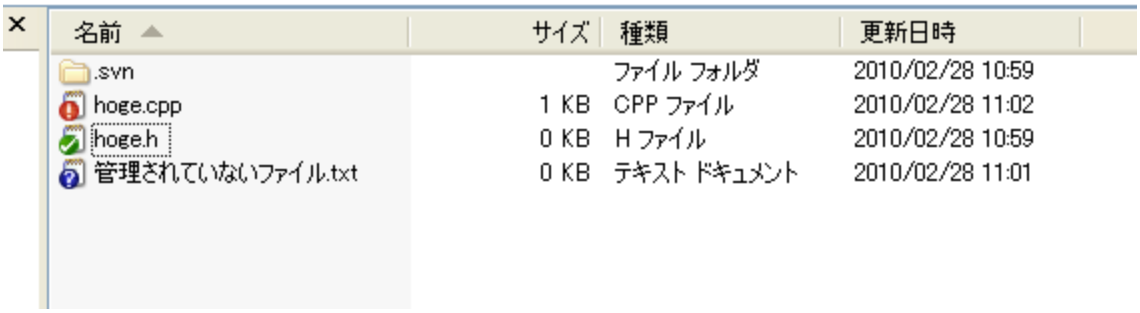


図 2.7-04　Subversion（TortoiseSVN）によって、バージョン管理が行われている状態
アイコンセットによって直感的にそのファイルがどのような状態であるかがわかる

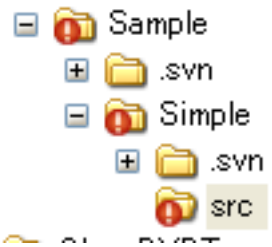


図 2.7-05　ディレクトリが管理されている状態（！アイコンは変更されていることを示す）

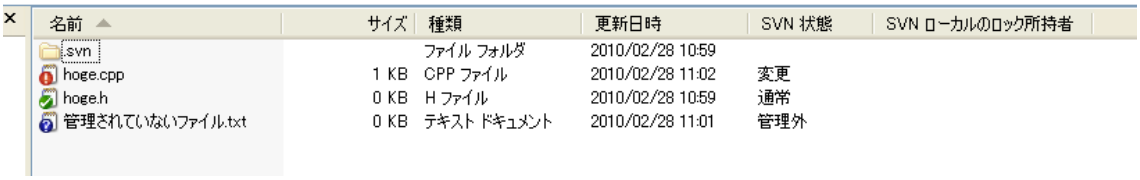


図 2.7-06　このように拡張情報を表示することもできる（状態とロックの所持者を表示）

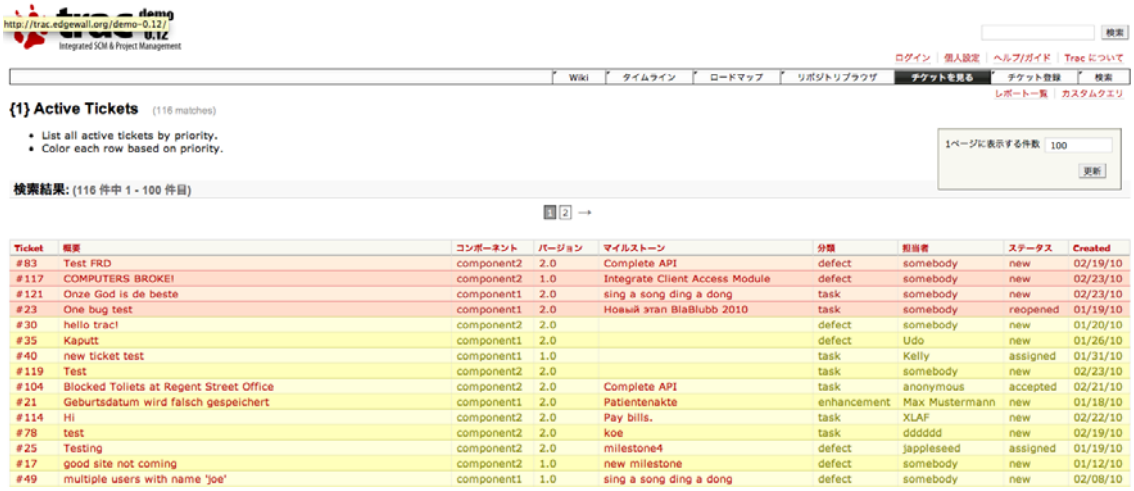


図 2.7-06　Trac によるチケット管理画面（公式サイトの Demo サイト）

日本語化されたものがインタアクト社より公開されているので参照されたい

図 2.7-07　Hudson によるビルド管理画面

様々なビルド状態を定義でき、さらに外部から URL でビルドスタートすることもできる

2.8　ゲームローカライズの歴史とこれから

長谷川　亮一（株式会社セガ）

2.8.1　はじめに

そもそも「ローカライズ」とは何か。簡単に言うなら、ある地域で作られたゲームを、他の地域向けに作り変える作業である。しかしながら、それは単に言語の翻訳にとどまらず、文化、国民性、宗教など様々な要素を考慮し、それぞれの市場に合わせたチューニングを行う必要がある。その重要性、そして不適切に行われた場合の危険性について今まで大きく議論される事は無かったが、ゲーム市場のグローバル化により避けて通ることのできない問題となりつつある。その作業自体は 80 年代から行われていたものの、時代によりその中身も大きく変化してきた「ローカライズ」というプロセスについて、各時代ごとの作業内容と今後の展望についてまとめてみる。

2.8.2　ローカライズの各作業内容

(1)　テキスト

当然ながら、ローカライズの中で最も重要な位置を占めるのが翻訳作業である。90 年代半ばまでは日本語⇔英語への対応がほとんどであったが、家庭用ゲーム機が世界中に普及していく中で多言語化への需要が高まり、現在では最低でも英語以外にも FIGS(フランス、イタリア、ドイツ、スペイン)、タイトルによっては十数カ国語に対応しているものもある。

テキストの翻訳で一番難しいのは、ある言語での表現やニュアンスが直訳では他の国で伝わらない場合がある、という事である。特にスラングやジョーク、その国独自の生活習慣や風習などは、それぞれの国の似たようなものに置き換えてみたり、あるいは全く違うものにしてしまったりと様々な試行錯誤があった。以前はそれらの翻訳作業を行うのが必ずしも両方の言語や文化に精通している人間ではなかった事もあり、アタリの業務用ゲームが日本語化された際の「コインいっこいれる」や「タフすぎてそんはない」、東亜プランのメガドライブ欧州版『ゼロウィング』のオープニングデモに登場する「ALL YOUR BASE ARE BELONG TO US」など、ローカライズ史に残る名（？）誤訳が登場したりもしたが、ゲームをきちんと理解しているバイリンガルのスタッフが翻訳を行うようになりそのような事も少なくなってきた。

一方、欧米の言語で使われるシングルバイト文字を日本語などダブルバイト文字にする場合のプログラム対応、フォントデータを持つための VRAM の容量問題（携帯機で起きやすい問題)、欧米の言語ではスペースの位置で自動的に行われる改行(ワードラッピング)が日本語では句読点以外の区切りがつけづらいために改行コードの手打ちで対応したり

（4:3 と 16:9 でテキストウィンドウのサイズが変わるゲームではさらに細かい対応とチェックが要求される）、さらに行頭、行末の禁則文字処理なども必要になってくる。逆に、表意文字である漢字なら数文字ですむ情報が、表音文字であるアルファベットでは長くなり、特にドイツ語では数十文字必要になってしまう場合もあるため、メニューやオプション画面などの大幅なレイアウト変更が必要になるなど、ローカライズにはそれなりの知識と経験が必要になってくる。

最近ではゲームの題材も専門的になり、例えば軍事、歴史、スポーツなどでは通常の翻訳作業とは別に、その分野の専門知識を有する人による監修が必要になってくる事も増えてきた。また、大作ロールプレイングゲームなどゲーム内で使われているテキスト量が数十万～百万ワードを超えるような場合、社内スタッフだけでは対応しきれないケースもあるが、現在ではそういったサービスを提供する翻訳会社（ローカライズベンダー）がいくつも存在するので、それらのリソースを有効活用するのが良いだろう。

(2) ボイス

テキスト同様、ゲーム中のキャラクターが話すボイス対応もローカライズの中で極めて重要な作業である。ゲームがカートリッジで供給されていた時代にはほとんど無かったが、メディアが CD-ROM や DVD-ROM に移行する中で声の演技はゲームの中でも重要な位置を占めるようになり、当然それらに対するローカライズの需要も高まってきた。さらに、吹き替えの際に著名な声優や俳優を使う事でマーケティング的な話題性を狙うケースも少なくない。

テキストとは逆に、同じ長さの日本語と英語では英語の方が多くの情報量を含んでいるケースが多い。そのため、必要な情報を残し装飾語を削るなどといった、読むテキストの翻訳とは異なる台本向けの翻訳スキルが要求される。日本では元々洋画や海外ドラマの吹き替えという文化があるため、そういった現場のプロフェッショナルの協力を仰ぐことが多かった。一方海外でも、特に DVD が主流となって以降日本のアニメが欧米向けに吹き替えられるケースが急激に増えてきており、その結果として翻訳や声優の演技のクオリティも伸びてきている。アニメやゲームに慣れ親しんだ演者が増えてきたこともその大きな要因である。

ゲームの海外版音声収録にもその好影響が見られるようになってきたと感じるが、注意すべき点もある。海外では声優が組合を結成しており（非組合員もいるが）、労働時間や休暇、声の権利に関して厳密なガイドラインが設けられている。特に夏場は 1 ヶ月程度の休みがあるため、音声収録がそのタイミングになる場合にはスケジュールに余裕を持たせておく必要がある。また、販促用のビデオやネット上の動画など、成果物の 2 次利用については別途使用料が発生するケースも多く、それらも考慮に入れておくべきである。

(3) グラフィック（キャラクター）

任天堂「ファミリーコンピュータ」（以下ファミコン）の時代には画面の解像度がそれほど高くなく、またハード的な制約もあり、結果として国産のキャラクターアクションゲームなどでは 2 頭身のキャラクターが主流であった。顔を大きく描くことでキャラクター性を持たせやすく、また感情表現もしやすかった。そういった理由もあり、キャラクターも必然的にアニメ的な手法で描かれる事が多かった。

一方、海外では現在ほどアニメに対する親和性が高くなく、さらにアニメは子供の見るもの、という認識もあり、結果として販売側は高い年齢層のプレイヤーにも訴求するためにパッケージイラストを大胆に、リアル志向に描き直すケースが多かった（筆者も、当時海外販社のマーケティング担当から「アニメ的スタイルはネガティブファクターだ」とハッキリ言い切られた経験がある）。

90 年代後半になると海外でも日本のアニメが放送されたり、マンガの単行本やビデオが売られるようになることでアニメ的スタイルも浸透していったが、それでも海外開発タイトル、特に人間の描写はリアル志向のままだった。近年になるとハードの進化につれて表現力も向上し、海外タイトルではさらにリアルな描写のキャラクターが増えてきたが、それがそれまでのアニメ的スタイルになれている日本のプレイヤーの目には異質なものとして映り、拒絶されることが多かった。テキストやボイスに比べグラフィック、特にキャラクターのスタイルは異なる市場ごとの好みの溝を埋めるのが困難であり、世界共通で好まれるキャラクターを生み出すことは全世界の開発会社の夢ともなった。マリオやソニック、ピカチュウなどの成功例もあるが、極めて稀であると言えるだろう。

(4) レベル（ステージ）デザイン

かつて海外タイトルが「洋ゲー」と呼ばれて一部のファン以外に敬遠されていた時代、プレイヤーを段階的に上達させるために考えられた敵の配置、普通には届かないところにアイテムを配置しその取り方を考えさせる、などといったレベルデザインは国産タイトルの得意とするところで、海外タイトルでは単にステージ中にオブジェクトを適当に配置していくのみ、というように確かに作りの荒さが目立つものが多かった。

昨今では海外でもステージデザインの方法論はかなり研究され、以前のようなアバウトさは見られなくなったが、それでもなお海外タイトルをプレイしていて「不親切だ」と感じることのあるプレイヤーも少なくないようである。

個人的には「自由度の高さ」がその原因ではないか、と考えている。海外のプレイヤーはこの「自由度の高さ」に慣れており、ヒントも無い状態で何度もトライ＆エラーを繰り返し、その結果正解を見つける、という「探索」行為そのものをゲーム性として捉えている。その一方、プレイヤーが詰まって進めなくならないように、とゲーム内をあちこち調べればほぼ必ずヒントが与えられる国産タイトルに慣れている日本の、特にライトユーザ

ーの場合、この「自由度の高さ」を「何をして良いか分からない → 不親切」と感じてしまうようである。

そういった海外タイトルを日本向けに移植する場合にはどうしたら良いか。例えばセリフの中にオリジナルには無いヒントを付け加える、マップの情報量を増やす、目印となるような視覚的ヘルプ（矢印、空まで伸びる光の柱、特徴的な地形や建物など）を追加する、などのサポートが望ましい。ヒントの無い状態、迷っている状態は多くの日本のプレイヤーにとってはストレスであり、ゲームのプレイを止めさせてしまう可能性もある。あまり誘導されていることを意識させずに上手くプレイヤーを誘導する、というのが理想的である。

(5) レーティング

ゲーム中に登場する暴力的、性的な表現や犯罪、ギャンブル、飲酒、麻薬などの扱いに関し、それまでいくつかのゲームでの過激な描写が社会的な問題となったことを受け、それまでのメーカーによる独自規制から第三者機関による内容のチェックが行われるようになった。北米では ESRB(Entertainment Software Rating Board)が 1994 年に発足、日本でも CERO(コンピュータエンターテインメントレーティング機構、Computer Entertainment Rating Organization)が 2002 年に設立された。現在では他にも欧州、ドイツ、オーストラリア、韓国などにそれぞれの審査機関が設けられ、対象地域でタイトルを発売する場合はそれぞれの機関のチェックを受ける必要がある。

基本的にどの機関も必要以上に過激、反社会的な表現を抑えるための基準を設けているが、やはり多少の差異がある。暴力表現に関しては欧米が、性的表現に関しては日本がやや寛容、と言われており、それらの要素の含まれたタイトルを相手地域にローカライズする際には、単なる翻訳のみならず、映像やアニメーションの手直しや削除などの作業が発生する場合がある。審査期間、費用も含め、プロジェクトの当初から意識しておく必要があるだろう。

海外タイトルの場合、逆にレーティングシステムが存在している事で 18 歳以上向けの過激な表現が盛り込まれたものも見られるが（当然ながら、正しい対象年齢層に対して販売されている限りは問題にはならない）、暴力表現に関してはドイツの基準がかなり厳しく、同じタイトルでもドイツ版のみ血液の量や色、身体部位欠損の表現などが抑えられている場合が多いため、海外タイトルを日本向けにローカライズする際はドイツ版をベースにするケースが多い。

(6) 宗教的要素

日本人は基本的に宗教に関する感心がそれほど高くないと言われており、日常生活の中で自分の信仰する宗教を強烈に意識する機会も多くない。しかしそれはあくまでも日本国内の話で、世界に目を向けるとそれぞれの信仰する宗教の様々な戒律の中で暮らしている人々の方が多い。ゲームの中で宗教的なモチーフに触れることは、仮にこちらに何の悪意が無くとも相手に不快な思いをさせてしまう場合があり、特に気をつけなくてはいけない要素である。

ロールプレイングゲームではよく回復やセーブを教会で行うが、以前は建物や中で働く人々の衣服などに十字架が描かれている事が多かった。これらは海外版では消されたり、違うシンボルに変えられている（もっとも、十字架の扱いの難しさに関しては比較的広く知られており、最近では日本版でも直接十字架が用いられることは少なくなった)。「ダビデの星」とも言われる六芒星はイスラエルの国旗に描かれ、ユダヤ民族の象徴とされている。日本のゲームではあくまでも幾何学的模様として魔力を秘めた装飾品や魔物を召還する魔方陣などに使われている事が多いが、これも海外版では五芒星に描き変えられるなどしている。

世界中で人気となったカードゲームの日本版で、あるカードの背景に「卍（まんじ)」のマークが描かれていたことがあった。海外版ではこのマークは消されていたのだが、熱心なファンが独自に日本版を取り寄せてこの事実が発覚、現地メディアに大きく取り上げられたこともあった。ハーケンクロイツやスワスティカと呼ばれ、ナチスのシンボルとなっている「逆卍」とは異なるものであるし、海外版では正しく対応されていたのだが、あまりに人気が高かったことが仇となり、センセーショナルに騒がれてしまった。

また、イスラム教は特に神の扱いが厳格であるとされ、ゲーム業界に限らず問題になった事例がいくつもある。ある大手スポーツ用品メーカーが 1997 年にリリースしたバスケットシューズでは、かかとの部分に燃えている炎のようなフォントで「Air」という文字をいれたところ、イスラム教で神を意味する「Allah」のアラビア文字に見える、という理由で発売中止、商品回収になった例がある。また、あるタイヤメーカーが販売したタイヤのトレッドパターンにも同じようにアラビア文字で「Allah」と読めてしまう部分が見つかり、商品回収、生産中止になったケースがあった。

ゲーム業界でも、2003 年に発売された格闘ゲームの BGM の 1 つにイスラム教の聖典であるコーランの一節が引用されていることが判明し、商品回収、販売中止となる騒ぎがあった。このニュースは世界中のゲーム業界に衝撃を与えたのだが、2008 年、別のゲームでやはり BGM の中にコーランからの引用が見つかり、修正のために発売が延期されることとなった。2007 年に発売された FPS（一人称視点のシューティングゲーム）でも、ゲーム中にリアルに描かれたマンチェスター大聖堂が登場し、英国教会が法的措置を求める騒ぎに発展した（これは大聖堂がゲーム中に登場することそのものではなく、教会の中で銃

を使った殺戮がなされることが問題とされている)。

ゲームというメディアはまだ歴史が浅く、しかし特に若年層における影響力が強いこともあり、世界中で何かと批判の対象になりやすい。最近のゲーム機では国内版のソフトがそのまま海外版ハードで動くことも多く、国内向けのタイトルだから、という言い訳が通用しない事も考えられる。商品の回収や発売中止ともなれば損害額は莫大であり、そのリスク回避のためにもこれらのモチーフの扱いには必要以上に慎重になるべきであろう。

2.8.3 ローカライズ作業の変遷

(1) 1980 年代初頭

まだ家庭用コンピューターが「マイコン」「パソコン」などと呼ばれていた時代、当時は日本でも大手電器メーカーから各種のハードウェアが発売されていたが、本体や周辺機器の価格が極めて高かったこと、そして使うのにかなりの専門知識が必要とされていたことからユーザー層は今よりもかなり限られていた。

その頃アメリカでは既に IBM-PC や Apple II 向けに優れたゲームソフトがリリースされていたが（業務用、つまりゲームセンターで遊べるゲームと比べるとハードの処理速度が遥かに劣るため、シミュレーション、ロールプレイング、パズル、テキストアドベンチャーなどが主流だった)､画面に表示されるテキストはシンプルな英語がほとんどであったため、ハードとソフトを扱う輸入代理店がマニュアルのみ日本語に翻訳、ゲームをプレイするには英和辞書が必須、などという時代であった。

その後海外タイトルへの需要が高まっていく中、一部のソフトハウスがライセンスを取得して国内ハード向けの移植を行いはじめた。とは言え、この当時は欧米諸国と比べ日本の地位が低く、また海外のメーカーと交渉して国内向けローカライズ、販売権を得ることの出来るようなスキルのあるスタッフが個々のソフトハウスにいる事は珍しかった。そんな中、取材活動を通じて海外メーカーとのパイプを持っていたコンピューター専門誌が仲介役となり、国内のソフトハウスを集めて海外営業ツアーを行いライセンス取得のヘルプをする、という事もあった。同様に、そのようにしてローカライズされたテキストの監修を専門誌の編集部員が行うこともあったらしい。全てにおいて試行錯誤の時代だった。

(2) 1980 年代中期～後期

ファミコンは国内のみならず海外でも「Nintendo Entertainment System」として爆発的なヒットとなった（最終的には全世界で 6100 万台以上の出荷)。それまでは PC 向けなどの海外から国内向けに移植されるタイトルが多かったが、ここにきてその需要が逆転、日本製タイトルが海外で多く販売されるようになる。

初期のラインナップはアクションやレースなどシンプルなルールのタイトルが多く、ま

たテキスト量も少なかったことから必要最小限の対応で済んでいたが、ゲームのジャンルが増え、内容が複雑になるにつれ様々な対応が求められるようになってくる。テキスト翻訳や前述した宗教的モチーフへの対応はもちろんのこと、版権キャラクターや音楽を使ったタイトルではキャラクターや曲を差し替えたり、逆に海外で別のキャラクターの権利を取得し、グラフィックを差し替えて販売するケースなどもあった。

この時代の特徴的なタイトルとして『Dragon Warrior』（海外版『ドラゴンクエスト』）シリーズを挙げたい。日本版ではギリギリの容量で作られていたものが、海外版ではより大容量の ROM が使えるようになったため、タイトルロゴ、オープニングデモ、キャラクターのアニメーションパターンなどが追加され、インターフェースもブラッシュアップされるなど国内版に比べ豪華な作りになっている。使われている英語フォントも筆記体に近く、さらに文体も古語が使われており（「You shall」が「Thou shalt」になるなど）、限られたハード性能の中、中世の荘厳な雰囲気を上手く演出している。

(3)　1980 年代後期～1990 年代前期

任天堂「スーパーファミコン」やセガ・エンタープライゼス（以下セガ）「メガドライブ」の時代になると、海外の開発会社の作ったタイトルが日本市場に持ち込まれることが多くなってきたが、原色やパステルカラーなど明るくてポップな色合いの画面の多い国産タイトルに対し彩度を抑えた渋めの色使い、そしてやはりリアル志向のビジュアルなど独特な雰囲気を放っていた。そのため、日本版の販売会社がパッケージや広告用のビジュアルをより日本向けのスタイルにしたり、難易度を調整するなどのアレンジを行うようになってきた。それでもライトユーザーには敬遠されがちだったこれらのタイトルだが、全く異なる発想から生まれる独自のゲーム性や高い難易度など、国産タイトルには無い魅力にとりつかれるコアユーザーがいたのもまた事実だった。

ゲームのメディアとして CD-ROM が使われるようになったのもこの時代である。NEC「PC-Engine CD-ROM2」やセガ「メガ CD」の登場により、それまでのカートリッジとは比較にならないほどの大容量を扱うことができる CD-ROM タイトルでは、その特長を生かすべく生の音声や楽曲が積極的に使われるようになった。これにより、ローカライズ作業の中に「音声の吹き替え」というプロセスが加わる。発売当初はそのハードウェアの価格から特に海外での普及が進まなかったが、しばらくすると独自の海外タイトルがリリースされるようになってきた。

現行機よりは遥かに見劣りするものの、画像を連続して読み込んで表示するストリーミング再生も可能だった。国産タイトルのほとんどがアニメ絵の連続再生を行っていたのに対し、海外タイトルでは実写も多く使われた。動画を撮影し、減色、コマ数を落とすなど手間はかかるが、やはり市場の好みがあったのだろう。3DO、メガ CD 用に発売された『ナイト・トラップ』（1993、セガ）や、PC エンジン用『シャーロック・ホームズの探偵講座』

（1991、ビクター音楽産業）では、ハードの制約もあり画面の 1/8 程度のサイズの解像度の低い実写動画が再生される程度だったが、それでも普段はテレビで放送される洋画の吹き替えを担当するプロの声優を使い、それまでのゲームとは全く異なる、新しい時代の到来を感じさせた。

また、この時代には ESRB のような審査機関が存在しなかったため（ESRB の発足は 1994 年）、ゲーム内における表現の基準はあくまでもメーカー独自の判断に委ねられていた。そのため一部のタイトルでは過激な描写を売りにすることもあったが、これもローカライズの際にメーカーの判断により相手の市場向けに調整された。

(4) 1990 年代中期～後期

ともに 1994 年に発売されたソニー・コンピュータエンタテインメント（以下 SCE）「プレイステーション 2」とセガ「セガサターン」は、ハードウェア性能の向上により、それまでは扱うのが難しいとされていたポリゴンによる 3D 表現が可能となった。日本市場では既に業務用で 3D ゲームがトレンドとなっており、そういったタイトルの移植版や類似タイトルが積極的に作られていった。一方、海外市場では当時既にゲームセンター文化が下火となっており「業務用タイトルの移植」が日本ほどの影響力を持たなかったが、代わりに PC でヒットしたタイトルの移植が多く行われた。

この時代の海外タイトルのローカライズでは、以下のような作業が新たに発生することになった。

(a) キャラクターのモデリング

それまではスプライトやテキストウィンドウ内に表示される顔を描き換えることで異なる市場の好みに対応できていたものが、3D モデルになるとアニメーションの関係もあり変更が難しくなってしまう。このため、キャラクターのビジュアルが固まった時点での各地域での事前の調査がそれまで以上に重要になった。大きな変更が難しい場合でも、例えば瞳や髪の毛の色を変えるなどの対応が行われることが多かった。

(b) 難易度調整

フィールドが 3D になることで自由度が一気に高まり、そのためプレイヤーを誘導することに主眼を置いた国産タイトルと、自由に探索させることを目的にした海外タイトルのゲーム性の差がさらに大きくなっていったため、特に海外タイトルを日本に持ってくる場合にはさまざまな調整が必要となった。

(c) カメラ

海外タイトルでは画面がプレイヤーの動きに正確に追随するものが多く、そのためキャラクターの小さな動作でも画面が大きく動き、それが 3D ゲーム慣れしていない日本のプレイヤーの 3D 酔いを引き起こすことが多かった。まだ 3D 酔いに関する研究、分析も本格的には行われておらず、またプログラム的にも調整が難しかったため、「洋ゲーは酔いやすい」というネガティブな印象をユーザーに持たれてしまった時期でもあった。(この問題は、カメラの追従性を弱くする、カメラとキャラクターの距離を離す、マップ上で迷わないようにヒントを増やす、などといった対応や、そしてプレイヤー自身の慣れによって徐々に緩和されていった)

また、この時代の特徴的なタイトルとしてはプレイステーション用『クラッシュ・バンディクー』シリーズを挙げたい。アメリカの独立した開発会社によって作られたタイトルながら、開発チーム全体が彼らのゲームを日本でヒットさせたい、という強い意志を持ち、SCE 日米欧のプロデューサー達との綿密なブレスト、各地域で繰り返し行われたフォーカステスト（後述）を経ての難易度やキャラクタービジュアルの丁寧な調整などが行われた結果、日本を含む全世界で爆発的なヒットを記録、ローカライズの成功例と言って良いだろう。

(5) 2000 年代前期

SCE「プレイステーション 2」、セガ「ドリームキャスト」、マイクロソフト「Xbox」といったハードウェアが登場、性能はさらに向上し、またメディアも CD-ROM から DVD-ROM（ドリームキャストは独自規格の GD-ROM）へと移行、ゲーム内に収録される音声、映像、テキストの量も増大し、当然ながらローカライズの際に必要な作業量も増大することになった。

この時代に新たに発生したプロセスとして台本の翻訳精度の向上がある。1 つ前の世代では、会話時のキャラクターの口はただ開閉しているだけの場合が多く、口が動いている間に声が出る、程度の精度で良かったものが、キャラクターの顔のディテールが細かくなり、口の動きや形がハッキリと分かるようになったため、特にセリフの最後の母音まで合わせた翻訳、演技が必要になる。それまでは担当社員でも十分に出来たスクリプトの翻訳や演出も、この頃から洋画の吹き替えなどを行うプロのエージェントに依頼するケースも見られるようになった。

さらに、日米欧で発売タイミングをなるべく近づけたい場合にはアニメーションの完成を待っていては遅いため、テクスチャーも貼られておらず、ただ直立して口だけが動いているだけのムービーを音声収録のために先行で作成する、など、ゲーム全体の開発スケジュールの中にローカライズ作業もきちんと組み込まれる必要が出てきた。

(6)　2000 年代中期～現在

SCE「プレイステーション 3」、マイクロソフト「Xbox 360」といった現行機が登場、メディアも DVD、Blu-ray とさらに大容量化。テキストや音声データ量がさらに増え、また対応する言語もそれまでの英仏伊独西の 5 ヶ国語から、ヨーロッパやアジア地域を含めた 10 ヶ国語以上に増え、開発会社の負担もさらに大きくなっていく。

しかしながら、ネットの発達による情報の伝播速度を考慮し、さらには他地域からの非正規な輸入による販売機会損失を避けるためにも各テリトリーごとの発売日をなるべく近づける必要性は極めて高く、それまでは副次的な作業であったローカライズはもはやプログラムやデザインと同様開発プロセスの 1 つとなりつつある。大規模なプロジェクトでは専門のチームが組まれることも珍しくなくなったが、中堅タイトルでは内部での対応が困難になってきたため、翻訳や音声素材の作成、管理を専門で行うローカライズベンダーに作業をアウトソースするケースも増えてきた。

また、ほぼ実写レベルに近づいたキャラクターのモデリングはアニメーションの生成にも莫大なコストがかかる。当然ながら、会話シーンでのキャラクターの口の動きや表情の変化も、もはや手付けでは対応しきれなくなりつつある。最近では音声収録の際に声優の顔にマーカーを付け、顔の動きを全て取り込むという「フェイシャルモーションキャプチャ」という技術が急速に発達してきており、これによる大幅な作業の効率化が期待できる。

一方、任天堂「ニンテンドーDS」、SCE「プレイステーションポータブル」といった携帯機向けタイトルでも当然ながらローカライズ作業は発生する。

これらのハードでは様々な制約があるため、特に海外タイトルを日本向けにする場合は文字周りで問題が発生する場合が多い。アルファベットはかなり小さくても読むことは可能だが日本語はある程度以上の大きさがないと読むことが出来ないため、場合によっては文字の表示エリアやメニュー画面などのレイアウトを大きく変更する必要が生じる。また、アルファベットでは多くても 100 文字前後あれば良いのが、日本語では数百～数千の大きな文字を表示するため、フォントデータの管理方法や、文字表示プログラムを大きく修正する必要が生じる場合がある。

また、ESRB や CERO のような審査機関も世界中に増えてきているので、各機関ごとの基準を理解するのはもちろん、審査コストや期間、提出タイミングなども十分考慮に入れるべきであろう。

2.8.4　未来

これまで述べてきたローカライズの各項目については既にそれなりの認知があると思われるが、今後この分野において重要となってくるであろうと思われる 2 つの要素について簡単に説明する。

(1)　フォーカステスト

海外の開発会社では多く採用されているものの、国内では一部大手パブリッシャー以外ではそのコンセプト程度しか知られていないのが「フォーカステスト」である。「モニターテスト」、「消費者テスト」などとも呼ばれるこのテストは、発売前のゲーム、キャラクターイラスト、アイテムや舞台設定などを一般から選ばれた参加者に見てもらい、そのリアクションを集計する。開発スタッフが自ら担当するゲームに関わる時間が長くなると客観的な判断をする事が難しくなるが、それを第三者の目から評価してもらうことが目的である。

通常のフォーカステストではゲームの難易度調整がメインとなるが、ローカライズで重要なのはユーザーインターフェイスの使いやすさ、そして何より登場するキャラクターのテイストである。最近はインターネット、特に動画サイトの爆発的な普及により各国のユーザーが他国のコンテンツを閲覧しやすくなり、結果として以前あったような他国コンテンツに対する違和感、拒絶反応はかなり薄くなってきていると思われる。また、国内のコアユーザーの場合、昨今の海外の良質なタイトルをプレイする機会が増え、さらにネット上での評価が後押しをして、かつてのようにビジュアルの印象だけで敬遠する事が損である、という認識が急速に広まっているようだ。

しかしながら、やはり国内と海外で好まれるキャラクターのビジュアルには依然大きな差があり、特にキャラクターゲームなどライト層にアピールしたいタイトルであれば開発の早期段階でフォーカステストが行われるのが望ましい。服装、表情、しぐさなどについても同様である。通常のプロジェクトの場合、社内にいる外国人や海外販社のマーケティング部門の人間などがその役を担うようだが､担当者自身の感性に委ねられる部分が多く、実際のターゲット層の好みとは異なるディレクションがなされてしまうこともある。

また、メニュー画面やゲーム内で使われるアイコン、一部パラメーターの略称なども国によっては意味が通じない場合があり、これらも早い段階で確認をしておくことが望ましい。当然、開発側が意図しない、文化的、宗教的なトラブルを事前に回避するためにも重要である。

さらに、第三者からの客観的な指摘があることでチーム内での不安や疑問が払拭され、目指すべき方向性が明確に設定される大きなメリットが期待できる。それなりのコストはかかってしまうが、その意義は十分にあると思われる。

(2)　ローカライズ素材管理ツール

ゲーム制作の過程において、例えばプログラムならプログラミング言語、グラフィックなら画像作成ツールや 3D モデル作成ツール、音楽であればサウンドツール、最近では物理演算エンジンやゲーム内のイベント、スクリプトをコントロールするツールなど様々な支援ツール、ミドルウェアが存在する。しかし、ローカライズはその作業の多様性から、

現時点では包括的な支援ツールが存在していない。

もちろんテキストの翻訳自体もまだ自動翻訳に任せられるものではないのだが、ローカライズ作業では様々なデータを関連付けて管理する必要があるため、仮に包括的な支援ツールを作ろうとしても相当の手間になると思われる。例えば、十数ヶ国語のテキスト対訳と音声データをファイル番号などで管理し、さらに表示される場所によっては文字数制限が必要になり、ムービーなどのセリフであればト書き（そのセリフが言われている状況）が必要になり、一部言語では男性、女性、中性名詞を区別する必要がある。また、マルチプラットホームゲームの場合、一部の名称が機種依存となるため、例えば英語では「controller」1つで良かったものが、日本語では「コントローラ」と「コントローラー」の2種類準備する必要があったりもする。

さらに、開発中の調整やフィーチャーの追加などによってテキストデータに変更があった場合、例えば各国ごとに担当者がExcelをベースに作業をしているとその通達にラグが生じてしまい、後に送られてきたデータがそういった変更を反映させたものなのか、どのバージョンが最新なのかを管理するのが大きな手間となってくる。

一部の大手メーカーではそのような素材をウェブ上で管理、常に最新の情報が参照できるようになっているオンラインデータベースを作成、運用を始めているところもあるようだが、これが出来るのはまだごく一部に限られるだろう。

2009年からGDC、そしてCEDECにローカライズというカテゴリーが新設され、様々なパブリッシャー、デベロッパーのローカライズ担当者、ローカライズベンダーがそのノウハウをシェアしはじめた。現時点ではこれらは各社が独自に蓄積しているケースがほとんどであるが、今後積極的な議論を通じて知識として共有され、それが支援ツールの登場につながることが期待される。

2.8.5 まとめ

現在のゲーム市場を世界規模で俯瞰した場合、そのシェアの大半は欧米であり、残念ながら国内市場は大きな伸びを期待できない状態である。今後さらに増大するであろう開発コストを回収する意味でも、一部のタイトルを除き、海外市場での展開は常に意識しておく必要があるだろう。

本稿にてローカライズの意義と重要性をご理解頂き、積極的に海外市場にチャレンジする開発会社が増えれば望外の喜びである。

2.9　アセンブラ

山口　誠（株式会社グッド・フィール）

本節は、プラットフォームが、主に 8 ビット～16 ビット CPU を採用していた時代の、アセンブラによるゲーム開発技術を、著者の開発経験を元に解説する。プラットフォームとしては、PC-8801、PC-8801SR（V2 モード）、MSX-2、ファミコン、NES、スーパーファミコン、GameBoy、GameBoy Color のみを扱う。同時代であっても、これらのプラットフォーム以外で活用されていた技術に関しては、取り扱わない。また、対象とするプラットフォームについても、特定のプラットフォームの仕様に言及することが目的ではないため、具体的にプラットフォーム名を挙げて解説することは最小限に止める。

2.9.1　プラットフォームの設計思想の違い。

業務用ゲーム機器は、路上や店頭に置かれ、客目を引きコインを投入させる必要から、より画面を派手に動かせるプラットフォームが試行錯誤の上開発されていった。そうした中で、家庭用ゲーム機が登場した頃のハードウェアでは、画面全体をビットマップ方式で扱った場合、メモリ帯域が小さくて処理速度が遅く、大きな面積が扱えないので、8×8 や 16×16 ドットなどのキャラクター単位で表示領域を扱う画像処理回路が一般的だった。また、時間単位でコインを投入させる必要があり、じっくり考える物より手軽に遊べる体感型のリアルタイムゲームが好まれた。リアルタイムゲームは、プレーヤの入力に対する、プログラムの応答時間によって、面白さが大きく影響を受ける。テレビは秒間 60 フレーム表示なので、フレーム切り替えに同期させ、1 フレーム内で、コントローラーの入力を得てキャラクターを動かす事が品質的に重要となっていた。また、画像処理回路は、テレビの走査線に併せてデータの出力をしており、色やキャラクターなど CPU が表示データを切り替えるタイミングは、走査線が折り返しているVブランクやHブランクの期間に限られ、常に走査線のタイミングを意識したプログラム作りがなされていた。こうした業務用ゲーム機の設計思想は、安価なハードウェアで構成される家庭用ゲーム機にも引き継がれていた。

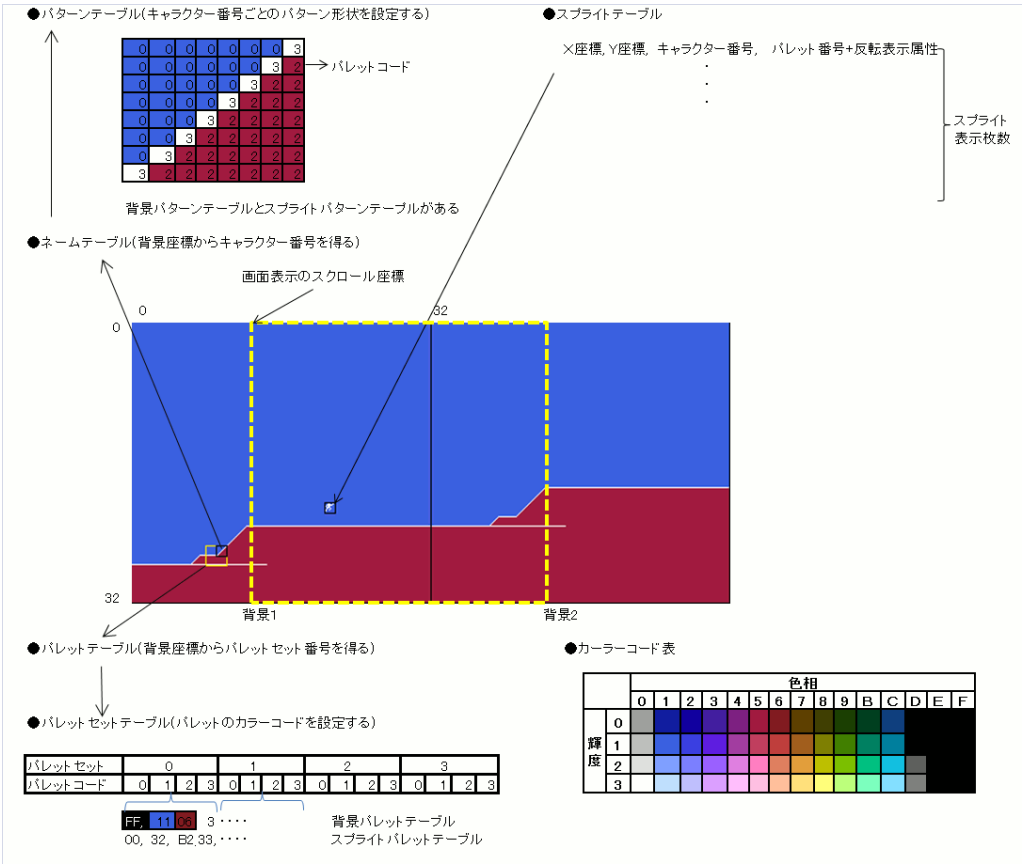


図 2.9-01　コンシューマーの画像回路フォーマット例

これに対して PC では、漢字や、表計算ソフトにおけるグラフ表示のため、高精細な画像表示が求められ、画像処理回路もドット(点)単位で表示を扱えるビットマップ方式の物が主流で、動画より静止画表示を得意とした。モニターも様々な表示周波数や解像度の物が採用されていて、走査線やフレームを意識したプログラムは一般的に実施されず、この設計思想の違いから、リアルタイム性の高いゲーム制作には不向きなプラットフォームが多かった。

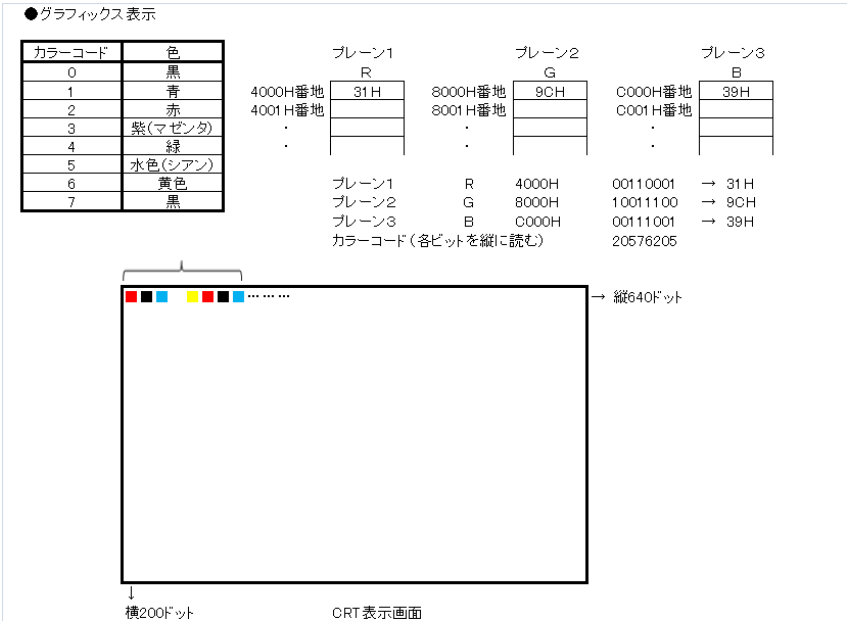


図 2.9-02　PC でのグラフィック回路フォーマット例

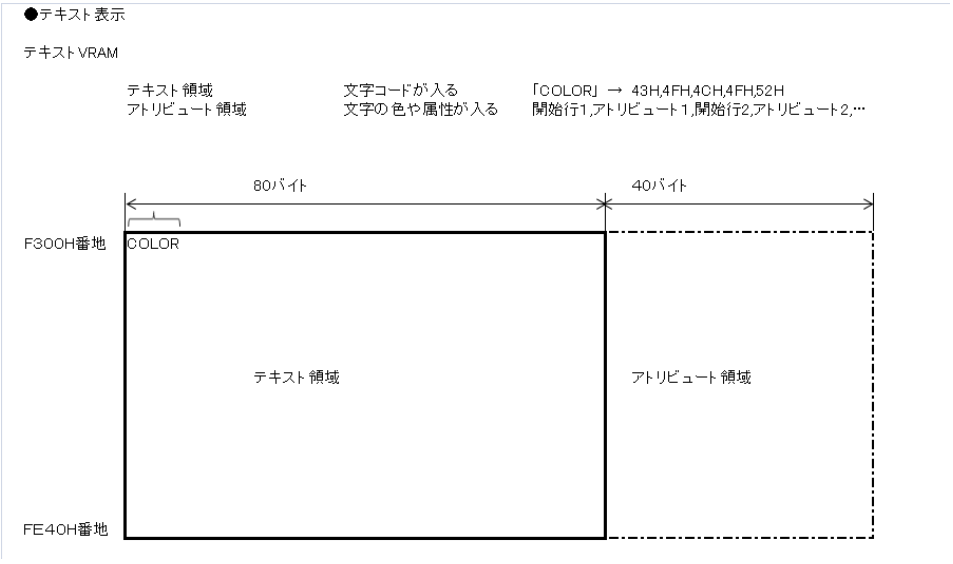


図 2.9-03　PC でのテキスト回路フォーマット例

また、当時は画像処理回路、CPU、RAM は高額だったため、ハードウェア単価の高い、業務用ゲーム機、PC、家庭用ゲーム機の順番で、性能の良いものが採用された。業務用ゲーム機や PC には、数 10MHz の動作周波数を持つ CPU が採用される一方、家庭用ゲーム機では、動作周波数が 1MHz しかなく、掛け算どころか 16 ビットの加算すら一命令では出来ない CPU が採用されていた。家庭用ゲーム機で、業務用ゲーム機と同様の表現をするためには、こうしたハードウェア間の格差を埋めるのに、多くの努力が必要だった。PC と家庭用ゲーム機で、作成出来るゲームジャンルの偏りが解消されるのは、ビットマップ方式だった PC

の表示性能が、飛躍的に向上したWindows普及期以降である。その後、有力OSベンダのマイクロソフト社が、Xboxで家庭用ゲーム機市場に参入し、PCと家庭用ゲーム機の文化の垣根がなくなってからは、作成出来るゲームジャンルの偏りが完全に解消された。

2.9.2 PCゲーム

(1) 開発環境とデバッグ手法

メモリのダンプやストア、1パスのアセンブラなど簡単なモニター環境はメーカーから提供されていたが、大規模なアセンブラプログラムを記述するには、ラベル配置やリンクの機能が無いと不便で、別途、開発環境を購入して利用していた。しかし、そもそもプログラムの動作環境自体が安定したマルチタスクOS下に無く、プログラムを展開するメモリも少ない状況で、開発環境上では極めて小規模なプログラムしか実行出来なかった。このため、開発環境上ではなく、コンパイル済みのゲームプログラムだけをメモリにロードして実行し、動作検証しなければならない場合が多かった。

ICE(インサーキットエミュレータ)を利用すれば、現在のVisualStudioに近い高機能デバッキングも可能だが、当時のPCゲームは、ホビーユーザか中小のベンチャー企業による開発が一般的で、高価なICEを自由に使える者は限られていた。このため、デバッグ作業は、実行ターゲットとなるPC上で行うことが多かった。具体的には、本体をリセットしてもクリアされないメモリ領域に何か数値を書き込み、プログラムが暴走した場合でも、リセット後にメモリの数値を確認できる、という手法が用いられた。これにより、プログラムが特定のコードを実行したのかを検出したり、タイマー割り込みがかかったコール先やレジスタの値をログ化し実行軌跡を辿る、といったことが可能であった。

(2) テキスト表示とCG表示

当時のPCは、Windowsなど現在のGUIベースのものとは異なり、グラフィクスAPIが搭載されておらず、テキストベースの利用が主だった。このため、画面表示を行うハードウェアは、テキスト描画とグラフィックスを扱うCG描画に回路が分かれており、テキスト描画を高速に扱うための専用回路を持つプラットフォームが多かった。相対的に、CG描画は遅かった。

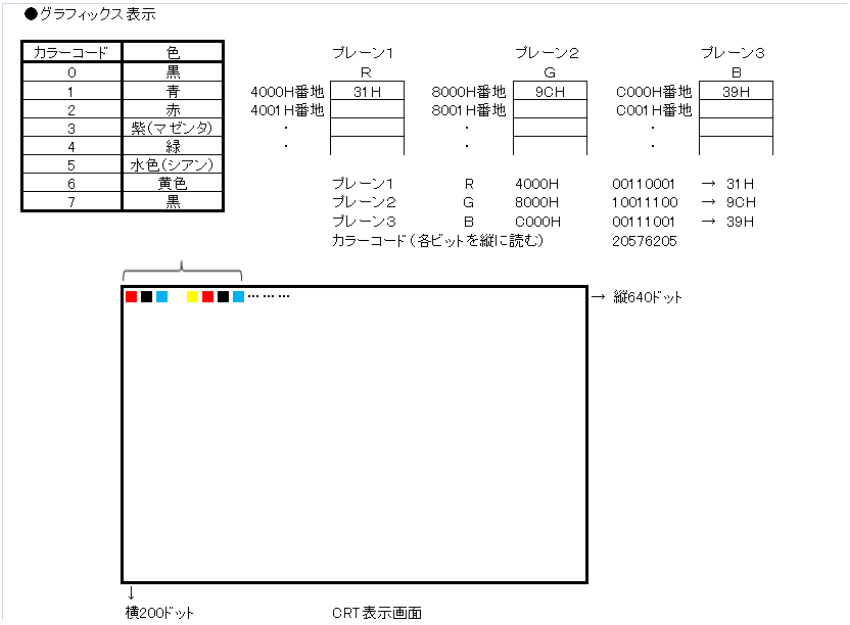


図 2.9-03　PC でのグラフィク回路フォーマット例

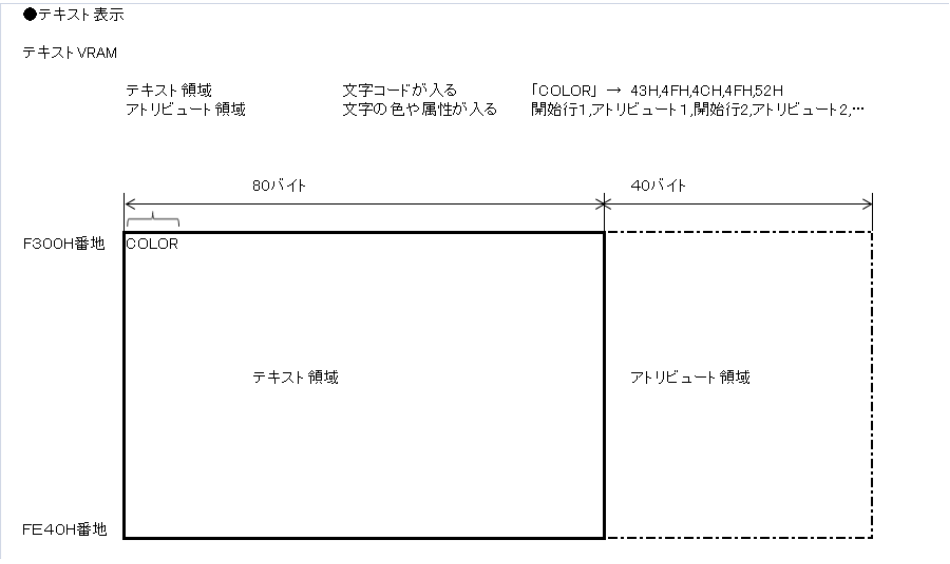


図 2.9-04　PC でのテキスト回路フォーマット例

(3)　メモリ優先での高速化

こうしたプラットフォームでは、テキストと CG を重ねて表示しており、テキスト表示機能を OFF にする事で、CG 表示を高速化できる場合があった。

また、CPU と画像処理回路の双方で、VRAM などのメモリを参照した場合、バスの取り合いが起こり、メモリアクセスが遅くなることがあった。こうしたケースでは、どちらかの回路のメモリアクセスを OFF にしたり、優先を設定する事で、高速に実行できる場合があ

った。

ただし、CPU を止めた場合、その間の計算処理が遅くなったり止まってしまったり、描画を止めると画面がブラックアウトしたり、アクセス優先を下げると画面にノイズが乗ったりする場合もあった。

ゲームの演出タイミングに沿って、こうした特性を切り替えて設定する事で高速化する事が出来た。

(4) ループ軽減、動的プログラムコードの作成

頻繁に利用するサブルーチン部分は出来るだけループを減らし展開した形にしたり、実行コードを動的に書き換えることで処理の高速化を実現していた。例えば、レジスタを使い切った状態で、更に、あるスコープで定数と見なせる値を参照したい場合、値をメモリ上に置くとアクセスに時間がかかるが、コード上に値を埋め込んだ後で実行することで、その時間を節約できる。動的書き換えのオーバーヘッドよりも、節約できる時間が大きければ、高速化となる。

(5) スタックポインタ操作によるワードデータアクセスの高速化

本来スタックポインタは、サブルーチンへの引数やアドレスの引渡しを目的に設計されたレジスタであるが、8080 や Z80 のインストラクションセットでは、16 ビットの単純なメモリアクセスはスタックポインタが最も高速であり、VRAM の書き込みなどにこれを利用する事で高速化出来た。

(6) 2 プレーンでの背景描画

初期の PC では、背景とキャラクターとの重ね合わせ処理が遅く、背景は黒一色などで表現される事が多かった。これを解消する手法として、RGB 毎に用意された背景の 3 プレーンについて、同時書き換えを断念し、2 プレーンだけの書き換えで背景を表現する方法が考案された。この手法では、4 色のカラーコードしか使えないので背景の表現に無理があると思われていたが、青と黄のカラーを一松模様に配置するとグレーの中間色が作れるなど、中間色の効果を利用する事で、2 プレーンだけで背景の描画を行えた。

(7) 一枚画の背景描画

16×16 や 24×24 など、タイル化されたパターンの組み合わせで背景を表現出来る様にはなったが、一枚画の背景については、2 プレーン描画で描画速度は確保出来ても、データ容量が大きく、RAM 上に展開出来ないという問題があった。この解決方法として、横方

向の着色を一度ドットおきにする事で、背景全体の明度は下がってしまうが、1 プレーン分のデータで 2 プレーン分の背景データを賄う手法が使われる様になった。

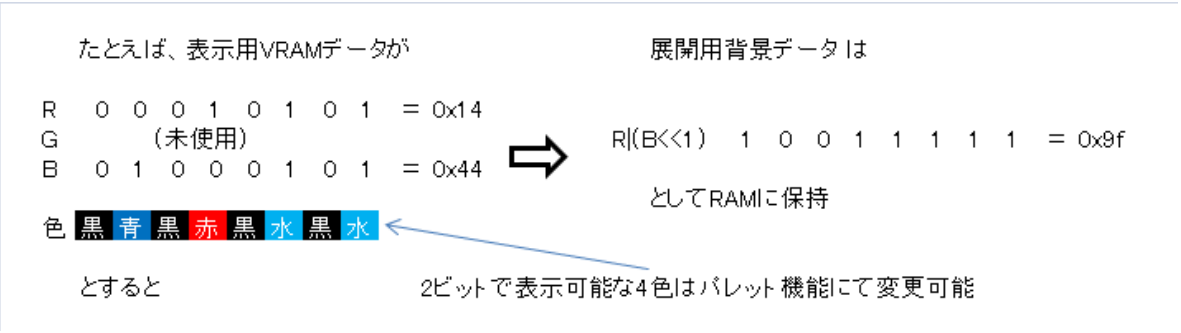


図 2.9-05　展開用背景データの例

(8)　周辺デバイスの CPU を利用

フロッピーディスクドライブなどのデバイスには CPU を積んだ物があり、メイン CPU、メモリと同じバスに接続されていた。こうしたサブ CPU を活用し、高負荷な思考ルーチンをメインの処理に対して並列実行させたり、スクロール先の背景データのキャッシングを実現できた。また、デバイスと本体 CPU 間のデータ転送速度を倍速化するなどの技術が開発された。

(9)　ホットスタート

BASIC の種別によっては、ロード機能に自動実行機能が無い物もあった。この為、プレーヤーはプログラムをロード後、実行コマンドや開始アドレスを指定して、プログラムを実行する必要があった。この解決手段としてプログラムのロードデータに BASIC やモニタプログラムなどのシステムワーク領域までを含めておき、ロードデータ内のシステムワークのロード終了フック部分にプログラム開始アドレスを入れて置くことで、ロード終了時にプログラムを自動実行させる手法が多く用いられていた。

(10)　マウスとウインドウプログラムの登場

Apple の Macintosh の登場以降は、他の PC でもマウスやウィンドウ表示などが多く採用されたが、Windows の普及以前は、共通の API が無く、アプリケーション毎にマウスドライバやウインドウマネージャが実装されていた。この為、ゲームアプリケーション内にドライバ機能を持たせる必要があった。

2.9.3　家庭用ゲーム

(1)　基本システムプログラムの共通化

家庭用ゲームの開発は、PC に比べ組織的に運営されており、開発会社の規模も大きく、ジャンルもシューティングゲームやアクションゲームなどのリアルタイムゲームに特化した傾向にあったため、コントローラーからの入力や、背景とキャラクター表示のプログラム部分など、再利用できる部分が使い回され、デザインツールで作成した画像データをゲーム機に組み込むまでのワークフローと共に、社内で共通化していく様に成り、後に、この共通化が各社のゲームエンジンとして注目される様になった。

(2)　処理棒と引きつり

家庭用のリアルタイムゲームの処理は、2.11.1 にて解説した様に、TV のフレームに同期させ 1/60 秒内で処理することが求められた。複数プレーヤのゲームなど、高負荷で殆どの処理が 1/60 秒以内で終わらない物に関しては、2 フレーム毎の 1/30 秒を処理に費やす時間とし、周期的な処理を実現させていた。この処理負荷を計る一般的な目安として、TV 走査線の V ブランクから処理を開始して終了するまでの時間を、処理棒という形でメータとして表示し調整をしていた。殆どのフレームで定期的に処理出来ているゲームの場合でも、表示物が大量に発生する場合など、一時的に高負荷になる事があり、その場合は、次のフレーム処理を遅らせて対応するといった手法が使われていた。ゲームのプレー中にこういった状況が発生すると、等速で動いていた物が一瞬止まって、カクっとして見える。こうした現象を指して、「引きつり」や「処理落ち」などの表現が使われていた。

(3)　バンク切り替え

8 ビットのゲーム機では、当初 64K バイトのメモリ空間しかアクセス出来なかったが、業務用ゲーム機で培われた手法の応用として、ROM カセット内に専用のメモリコントローラチップを搭載する事で、特定範囲のアドレス空間を動的に物理メモリに配置する事が可能となったり、64K バイト以上のメモリアクセスも可能になった。こうしたバンク切り替えの手法をパターンテーブル領域に利用すると同じキャラクター番号でも異なったイメージパターンの絵が表示可能となり、ステージ毎に違う敵キャラクターを出現させたり、アニメーションさせる事で、マップ上の水の部分を流れているように見せたり、一枚の背景で擬似的な多重スクロールを表現するなどが可能になった。また、プログラム領域に利用すると、敵の動きやステージ内の仕掛けなど、常時必要のないコードを、必要な時のみバンク切り替えでアクセス可能として使える様になり、全体のプログラムサイズを増やす事が出来た。

(4)　ブランクの活用

TV の走査線が折り返す V ブランクと H ブランク中に、画像処理回路の設定を変える事で様々な効果を出していた。

例えば、背景画面の表示座標を画面中央の H ブランクで変更すると、画面上部は固定されたままに、画面下部のみスクロールさせることが出来た。また、H ブランク毎に、細かく水平座標値を左右に揺らすことで、水面下にある背景の様な表現が出来た。H ブランクに合わせて表示位置を操作する手法を、ラスタースクロールと呼ぶ。

さらに、H ブランク中にバンク切り替えを行ったり、パレットのカラーコードを変える事で、画面に表示出来るキャラクターを倍にしたり、色数を増やす事が出来た。

こうした表現の応用として、背景として敵キャラクターを書いておけば、オブジェクト（スプライト）では表示出来ない大きなキャラクターをアニメーションをさせる事が出来る。

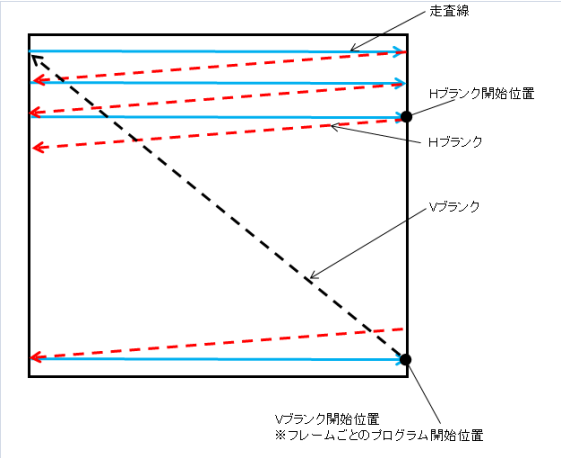


図 2.9-06　TV 走査線とブランク

H ブランクや V ブランクの開始時には割り込みが掛けられ、ブランクが終了するまでの間、画像表示回路は動作しておらず、様々な処理が可能であった。

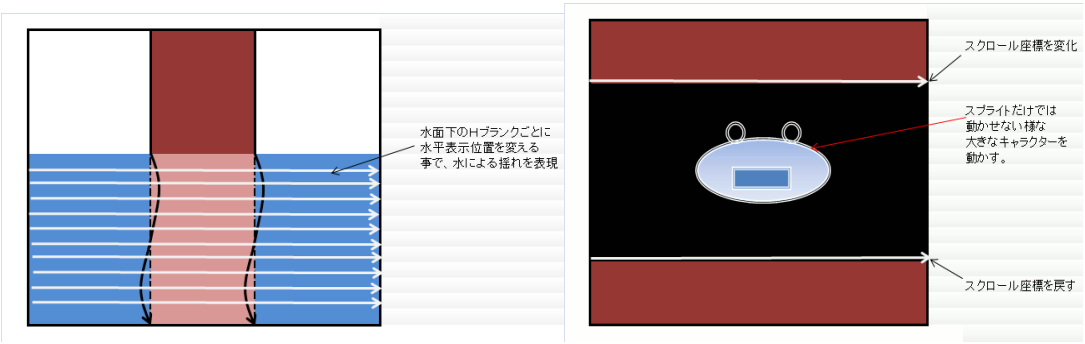


図 2.9-07　水面下のラスタライズ（左）、大きなキャラクター表現（右）

(5)　背景データのモジュール化

背景としてキャラクターを並べるデータを持つ際、スクリーンに配置するキャラクター番号をそのまま保持するのでは無く、4×4 などの単位にモジュール化したデータとして持ち、モジュール単位での再利用性を高め、データを圧縮して持つことが出来る。パレット情報や、進入の可否などの地形情報も、モジュール単位で管理出来るため、背景情報の管理が容易となった。

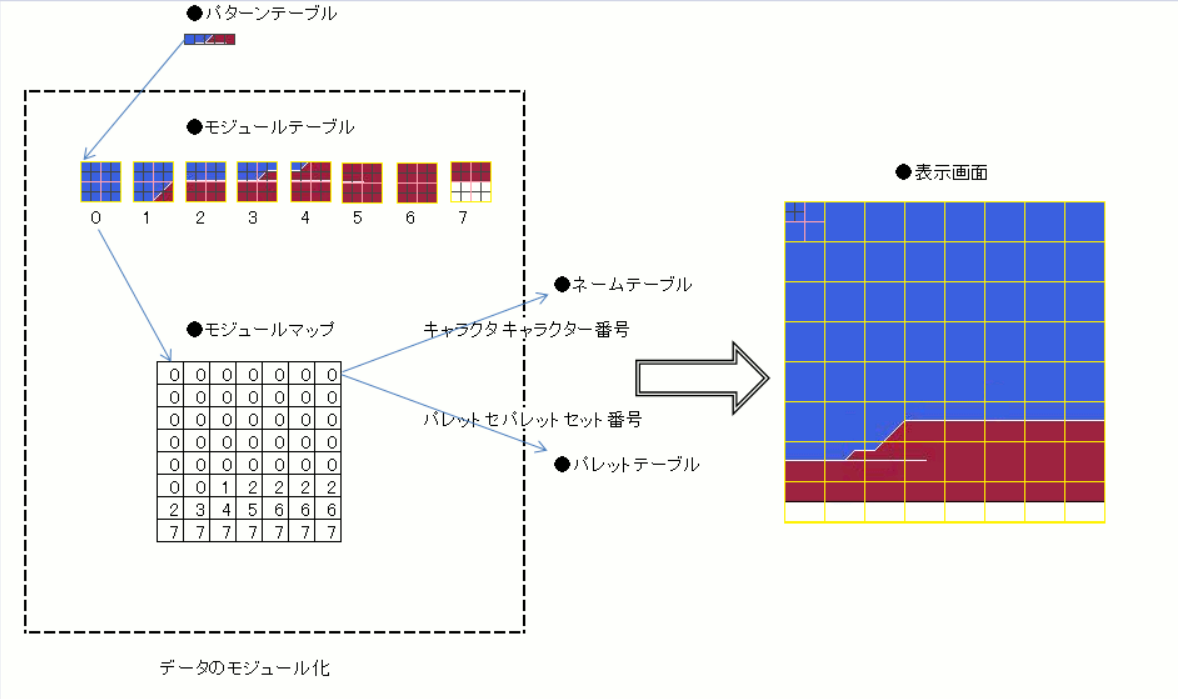


図 2.9-08　背景データのモジュール化

(6)　オブジェクトパターン

ひとつのオブジェクト（スプライト）でひとつの登場キャラクターを表示するのではなく、座標オフセットの異なる複数のオブジェクトを重ねあわせたオブジェクト群で構成する事で、形状の異なる登場キャラクターを表現していた。また、異なるパレットのオブジェクトを重ねる事で多くの色を使うことが可能であった。更にはオブジェクトだけでは扱えない様な、サイズの大きな画像や、色数の多い画像については、オブジェクトと背景を重ねあわせて扱う事で、使用できる色数や画像サイズを大きくする手法が取れた。

(7)　オブジェクト回し

当時のハードウェアには、画面中に表示できるオブジェクト数に制限があり、特に、水平方向に重複して表示できる数の制限があった。制限数を超えたオブジェクトは表示され

無くなるため、ボスキャラなどの大きなオブジェクトパターンは表示が困難だった。この解決手法の一つとして、表示を開始するスプライト番号をフレーム毎にづらす事で、点滅表現にはなるが、多数のオブジェクトを組み合わせた大きなオブジェクトパターンを表示出来た。

(8)　ワークサイズとプログラムコードの圧縮

PC と比べ、ワークエリアとして使える RAM 領域もプログラムが格納される ROM 領域も非常に少なかったため、双方とも圧縮が必要であった。例えば、jr 命令の後に ret 命令が来ると 4bytes 必要となるところを、jp 命令に置き換えて 3bytes にし 1byte 削減したり、命令の順番を入れ替えるなどして 3byte 以上同じコードに出来る部分は全てサブルーチン化し共有化して削減したり、コードを短くすることが行われた。敵の発生データを保持する際も、XY の座標値にそれぞれ 1Byte 使うと無駄が出るため、アライメントを設けて、空いたビットにワークの初期値を入れるなどしていた。

(9)　ゲーム開発用アプリケーション

PC の場合、当時の開発環境は DOS や BASIC などテキストベースが主流だったが、開発機と実行環境が分離し易いコンシューマーでは、UNIX ワークステーションなどが導入されているケースもあり、X-Window や Motif ベースのゲーム開発用の GUI アプリも導入されていた。

2.9.4　今後の課題

アセンブラにてゲームを開発していた頃を思い出し、当時の技術を何点か紹介した。当時のコンピューターは、大型コンピューター、汎用機、ワークステーションと、高価なプラットフォームほど高機能なハードウェアが採用されていた。ゲームプラットフォームも同様に、業務用と比べると家庭用はハードウェア性能も劣ったため、家庭用ゲームの表現能力を如何に業務用に近づけるかといった点で、工夫が試みられた。PC では、元来ゲーム向きの設計がなされていないハードウェア上で、如何に品質の高いゲームを実現するかという課題があった。ソフトウェア技術においても、学術的な研究成果の導入よりも、職人的な経験則や工夫により、独自の技術が培われていた。これは、高級言語が使えなかった背景の影響もあろう。しかし、ゲーム性を表現するための技術的な考え方やゲーム開発専用言語やゲーム開発用アプリケーションの利用など、今日も着目されている技術の原型は洗練度こそ異なるが当時からあった事が伺える。

今日、コンピューターのダウンサイジングが進み、論文などで発表される技術の多くも、

家庭用ゲームでも使える状況となった。しかしながら、多くの開発者が関心を持ち流行る技術が必ずしも商品価値が高いとは限らない。

今日、ゲームプラットフォームの種類は増加し、さまざまな形態の物が存在しているが、一般的な技術転用で簡単に実用化できる無個性な商品ばかりが氾濫している様にも見える。それぞれのプラットフォームでしか体験出来ない個性的なゲームでプレイヤーを楽しませようと考えた場合、アセンブラ時代の様に独創的な技術を使ってでも品質を高め、求められている物を如何に商品化するかというアプローチも必要だと考える。

参考 URL：

Enri's Home PAGE （PC-8001）

http://www2.odn.ne.jp/~haf09260/Pc80/Pccrt.htm

PC-8801 Emulator

http://www.117.ne.jp/~show/pc8801/tech88/3090.html

2004-12-09 - Classic 8-bit/16-bit Topics

http://d.hatena.ne.jp/hally/20041209

History of NEC PC-8801series

http://cdlmaster.ld.infoseek.co.jp/data88/index.htm

Books for PC-8801

http://cdlmaster.ld.infoseek.co.jp/data88/book88.htm

PPU

http://www.media.imit.chiba-u.jp/~motchy/nes/PPU.html

参考書籍

書 名	著 者	出版社
NEC PC-8001/6001 ハンドブック	朝日新聞電子計算室	朝日新聞
NEC PC シリーズ PC-9801・8801 ハンドブック プロ入門のための機能徹底解説	朝日新聞電子計算室	朝日新聞
PC-Techkow8800mkII PC-8031/PC-8032 ユーザーズマニュアル	平松達夫/八木良一共著 システムソフト監修	システムソフト

表 2.9-01　アセンブラ関連年表

年-月	製品	技術情報
1981-11	PC-8801(NEC)	
1983-6	アルフォス(エニックス)	スタックポインタによるワードデータアクセスの高速化、2 プレーンでの背景描画
1983-7	ファミリーコンピュータ(任天堂)	
1984-5	ニュートロン(エニックス)	一枚画の背景合成
1986-2	SeeNa(システムソフト)	周辺デバイスの CPU を利用
1986 ～ 1989	ファミリーコンピュータ向けゲーム	背景データのモジュール化やオブジェクト回しなど
1986-11	書籍「LodeRunner で学ぶ実践 C 言語」(BNN)	PC98 版の C 言語実装
1987-4	上海（システムソフト)	GUI、マウスドライバ実装
1990-11	スーパーファミコン(任天堂)	
1994-11 、12	セガサターン(セガ)、プレイステーション(SCE)	アセンブラによるゲーム開発が主流ではなくなる

※技術情報については、提示したタイトルが初出でない可能性を、ご了承下さい

2.10　日本のゲーム開発技術の全体像とこれからの課題

三宅　陽一郎（株式会社フロム・ソフトウェア）

2.10.1　概論

この節は、各報告書のその前置きとして、各報告から得た情報から、ゲーム開発技術の歴史全体の流れを記述する。ゲーム開発技術の歴史は複数の方向の運動が絡み合いながら進んでおり、それらを一つ一つ分解して流れを形成する主要素を示すことで、全体を捉える指針を示すことを目的とする。そういった意味で、本節を読むタイミングは、各分野のレポートを読んでからでも、読む前でも構わない。

歴史が人の活動の総体であるように、ゲーム開発技術の発展は、たくさんの開発者、関係者の活動の総体から出来ており、複雑な運動の集合である。しかし全体として見れば、ゲーム開発技術は全体としていくつかの明確な特徴を持ちながら前進している。レポート全体を合わせてゲーム開発技術の歴史の全体像を再構成しようとするとき、その運動全体の特徴を見出だすことが出来る。

ゲーム開発技術はゲーム開発で必要な、或いは導入されて来た技術の総体である。しかし「半導体技術」「車製造技術」などの産業と異なり、ゲーム（タイトル）開発技術は（周辺機器を付随する特別なタイトルを除けば）純粋にソフトウェア製作の一分野であって、技術的にそこから逸脱するわけではない。ゲーム産業に特有のゲーム開発技術は殆ど存在しないと言ってよい。ゲーム開発技術は、ハードウェア、ソフトウェアの基盤技術をデジタルゲーム用に応用したものであるから、その起源は、ソフトウェア技術そのものにあり、決してオリジナルなゲーム産業起源のものではない。勿論、ゲームを開発するにあたって、ゲーム開発特有のノウハウやコツ、ハードウェア上の特殊な構造を用いるということが存在するが、それらはノウハウや応用の極であって、そこから他の産業技術応用を発する騎士技術ではなく、逆にゲーム産業こそが、他の産業技術から先端的な技術を集めて応用した総合分野と言えるだろう。或いは、これから応用される先端技術の巨大な実験としての役割を担っている。

ゲーム開発技術は、これまで、

(A)　プラットフォーム（ハードウェア）の発展
(B)　新技術の導入（3D，ネットワーク）
(C)　新技術の導入（学術などから）

と言った要因によって発展して来た。いずれも、一般社会におけるデジタル技術の発展を

少し先取りする形で導入して来たために、先進的というイメージがデジタルゲームには付随している。しかし、プラットフェームはハードウェアとしては数年間、固定であり、後期になればなるほど、逆に世間の技術水準から遅れたものになってしまうという矛盾と宿命を持つ。

(A)　はプラットフォーム・メーカーの設計によって、
(B)　は社会インフラによって、
(C)　は開発者の継続的な努力によって、

これら諸要因が複雑に絡み合って、ゲーム開発技術全体を形成し発展して来た。一般のゲームソフト開発企業から見ると、(A)(B)は社会・プラットフォームメーカーの動向に対応する外部要因であり、導入される技術に対応（キャッチアップ）する必要が発生する。そこでは、なるべく早期に最大限の効果を引き出すことが目標とされる。そういったキャッチアップの時期が終わると、その上に独自の技術を模索して行く(C)の要素が発生する。例えば現在は、次世代機（この節では次世代機と言えば、現在のPLAYSTATION3(SCEI),XBOX360(Microsoft),Wii(Nintendo) のプラットフォームを指すものとする）ハードウェア上の製作体制が整えられた時期であり、これから独自の技術をこの上に構築することで他社との差別化を計ることが主眼となっている時期である。このように時期によって、技術導入要因のプライオリティーは変化する。特に現在のような大規模開発に対する技術導入においては、長期的な導入プランが大きくその企業の開発力を左右するため、先を読んだ技術開発・導入プランが必要である。

また、開発技術の保持という視点からゲーム開発技術を考えてみれば、ゲームにエンジンに関わる部分にせよ、タイトル依存のものにせよ、その技術が継続的にその企業の財産になって行くためには、その技術が継続的に保守され、要件に応じて改善され続けなければならない。つまり、技術を開発し保持するには、同時に、人件コストが発生し続けるのである。技術を高い水準で維持して行くには、いかに優秀な技術スタッフを集めて運用するか、そして、そのスタッフの中で、引いてはその企業の中で、いかなる形で技術を生きた形で保ち続けるかという問題が存在する。こういった事情は、あるまとまった技術を、自社で開発し保持するのか、ミドルウェア会社から購入して導入するのか、という問題を突きつける。しかし、一方で、高度化するミドルウェアは、その導入において対応する人員を自社内で割り当てる必要があり、購入経費だけでは済まないという事実もある。

こういったさまざまな要因の組み合わせによって発展するゲーム開発技術の運動は、さらに分野ごと、時代ごとにさまざまな局面を見せる。そして、その総体は、多面的に捉えることで初めて全体を捉えることができる。

本報告書におけるゲーム開発技術､計 21 分野における時間軸に沿った技術史の解説は、分野という始点からこの総体を眺めた記述されたものである。各分野は、その時々のプラットフォームの上で複雑に絡み合いながら進化して来た。そういった運動の特徴を捉えると、以下のような項目にリストアップすることが出来る。

(1) ゲーム開発技術の発展の歴史（各分野の拡大と独立（専門化)、統合など）
(2) データ、プログラム、スクリプトの形成
(3) 技術導入・流入経路
(4) 仮想化の流れ
(5) 固定コンテンツの展開とプロシージャル
(6) 技術からマネージメントへ
(7) ゲームデザインと技術

以下、各要因に沿って詳細を解説して行く。

2.10.2 ゲーム開発技術の発展の歴史

ゲーム開発技術は、現在、さまざまな分野に専門化している。しかし、80 年代初頭のゲーム開発が全体として数名程度で可能であったことを考えれば、その黎明期の開発においては、専門的に分化する前の各技術分野がその雛形の形で一つに混在していたと考えられる。現在、専門化して見られる各分野はそういった全体の中から、一つ一つの部分が急速に発展し、分化し、専門化して来たのものである。また、そういった分化と同時に、今度は、分化した各分野を統合する必要が出て来た。統合技術自身も、現在、データ管理、リソース管理、100 名以上にのぼる開発者の管理など、既に一つの技術分野となっている。

こういった､分化と統合という二つの相対する運動は､現在もなお進行中である｡また、新しい分野としては AI 分野、インターフェース分野が急速に分化・専門化しており、この分野も独立した分野になると同時に、他の分野との連携のあり方が模索され始めることになる。

このように、ゲーム開発技術の運動は、主に 3 つに分けられる。

① 発展
② 分化（専門化）
②’ 挿入（新しく独立した技術の導入。ネットワークなど）
③ 連携

①は、一つの分野が大きく発展して行き、②はそれが完全に専門分野として独立し、或いは、新しく独立した技術が導入されることであり、③は、再びその独立した分野が新しい形で他の分野と結びつき連携することである。②は例えば、近年では、タッチパネルやステレオ 3D、ポインタデバイス、カメラデバイスなど、インターフェース分野で新規技術の挿入が著しい。こういった分野の専門化も予想される（**本章「ユーザーインターフェイス（ソフトウェア）」「周辺機器メーカーから見たハードウェアインターフェースの変遷」を参照**）。

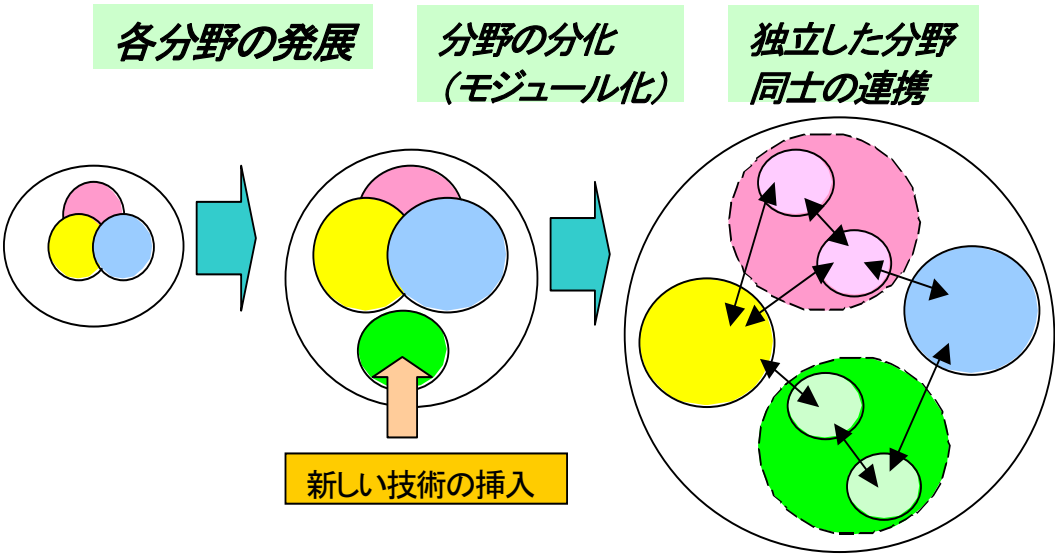


図 2.10-01　ゲーム開発技術発展・分化のイメージ図

こういった分化・専門化・統合の動きを具体的に考えてみよう。ゲームキャラクターを動作させる技術を考える。80 年代、90 年代前半までは、中規模～大規模のゲームさえ、描画、キャラクターの思考、物理運動、周囲の環境との衝突判定などのインタラクション、アニメーションは、ある程度、まとまったプログラムとして記述することが可能だった。一つのファイルに各 AI の動作が記述されているケースも多かった。モジュール化される以前では、複数の機能がキャラクター制御という名前のもとで混在していた。

ところが、ゲームの規模が大きくなり（特に 3D になり）、一つ一つの果たすべき機能が多くなり、まず、キャラクターアニメーションは、スプライトのような組み合わせからボーン単位の 3D アニメーションデータに変化し、物理運動は、物理エンジンとしてゲーム全体に統一的な物理エンジンとなり、衝突判定は、物理エンジンや独自の判定システムとなり、思考は、環境を解析して最適な運動を構築する AI システムへと発展した（**本章「物理シミュレーション」の節を参照**）。キャラクター制御は、そのメインルーチンからさまざまな処理タスクをモジュールに投げる形に分離・連携して行った。

また描画は、ドットで打たれた絵から、ポリゴンモデル、アニメーション、テクスチャ、

シェーディング、など、分化され、モジュールとなり、そういったモジュール毎に技術が高度に、そして細分化しながら発展して行った（**本章「シェーダ」の節を参照**）。

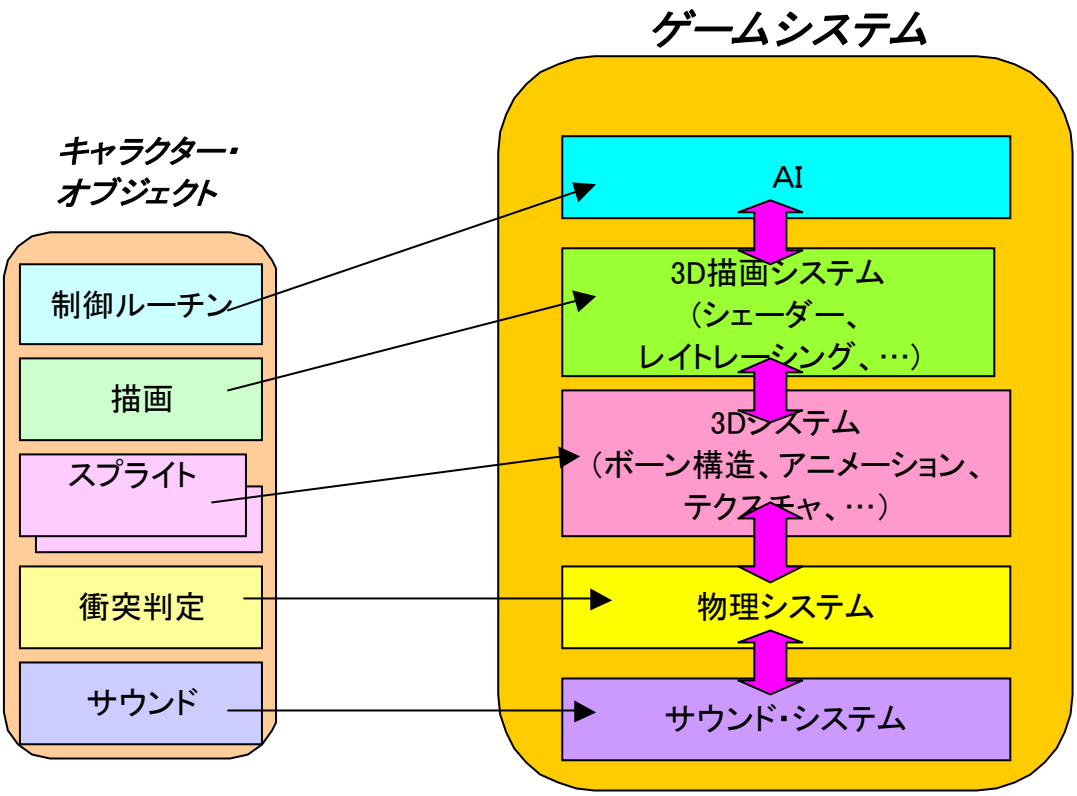


図 2.10-02　キャラクターシステムの発展・分化・統合

また、ゲームにおけるシナリオを考えてみよう。今から見れば極めてシンプルな画面を持ったゲームに対して、世界観のリアリティーを持たせるため（ゲーム設定の補完として）に付属されていた。しかし、シナリオとキャラクターは次第に増量し、本格的に練られたシナリオとなり、メディアミックスの中心として需要からも求められ、深くゲームデザインそのものと関わるようになって行った。ＲＰＧ、アドベンチャーゲームの発展は、ストーリーを語るためのゲーム、という分野の出現を促した。これは、時間的に遅れた形で米でも序々に大きな比重を占めるようになり、現代の世界的なゲームのストーリー、キャラクターへの比重のかけ方に大きく影響している（**本章「シナリオ」参照**）。

ゲームデザイナーは、かつてキャラクターデザイン、レベルデザイン、ゲームの仕組み、キャラクターアクション、スクリプティングまで、ほぼ全てのディレクションと同時に、上記の仕事の幾つかを兼ね備えていた。時には、可能ならばプログラマを兼ねることもあった。しかし、職種の分業化と共に専門化され、ゲームデザイナーは一人からチームとなり、ディレクター、プランナーなど、集団によってゲームデザインをする方向に拡大し、総合的なディレクションを必要とする時代になった。現在でも、ディレクター、プランナ

ー、ゲームデザイナー、レベルデザイナーが持つ仕事の領域は、企業、国によって曖昧である（時にゲームデザイナーは「便利屋」と自嘲する）。

こうして各分野が分化し、専門化し、職種化する中で、次に課題となったのが、それらを如何に統合するか、ということであった。もともと一つのシステムの中で、各要素を柔軟に絡み合わせていたところが、独立した各要素をしっかりと連携させなくてはならなくなった。例えば、オフラインのゲームで物理エンジンを動かすのと、オンラインゲームで物理エンジンを動かすのとでは全く事情が異なる。ネットワーク技術と物理技術をどう組み合わせるか、という問題が発生するわけである。また、オンラインゲーム上の AI を動かす場合も、新しい同期の問題をもたらす（**2008 年度報告書ゲーム開発技術ロードマップ「ネットワーク通信」を参照**）。

AI と物理、物理とサウンド、サウンドとアニメーションの関係、こういった細分化した各モジュール同士を関連付けゲーム的に無矛盾かつスムーズに協調させるために、ライブラリ、ゲームエンジン全体を精密に設計する必要が出て来た。そこで 90 年代後半から 00 年以降は大規模な共有ライブラリの整備、さらにその上のゲームエンジンの整備が目指され、特に次世代機以降ではゲームエンジンの設計に力が注がれ、実際ゲームタイトルのクオリティーはゲームエンジンの出来に大きく依存することになる。また、ゲームエンジンそのものが本格的に商品化されたのも次世代機以降である。こういった事情の中で、各ゲーム開発企業は自社でエンジンを開発するか、ミドルウェアメーカーからゲームエンジンを購入するかという選択を迫られているが、日本ではゲームエンジンは比較的自社開発に頼ることが多い。自社開発のメリットは、ゲームエンジンの実装の中身を必要に応じて自分たちの手で改善できることである。デメリットはその改善・保守にコストがかかることである。中大規模の企業では、ゲームエンジンの開発・保所用に専門部署やスタッフを当てるケースが増えている。逆に、商品化されたゲームエンジンは、ゲームエンジンに特化したスタッフを置き、利用者からバグ報告やフィードバックを受けることでノウハウを蓄積し、ますます強固なシステムを構築して行けるという利点を持つ（**本章「ミドルウェア」の節を参照**）。

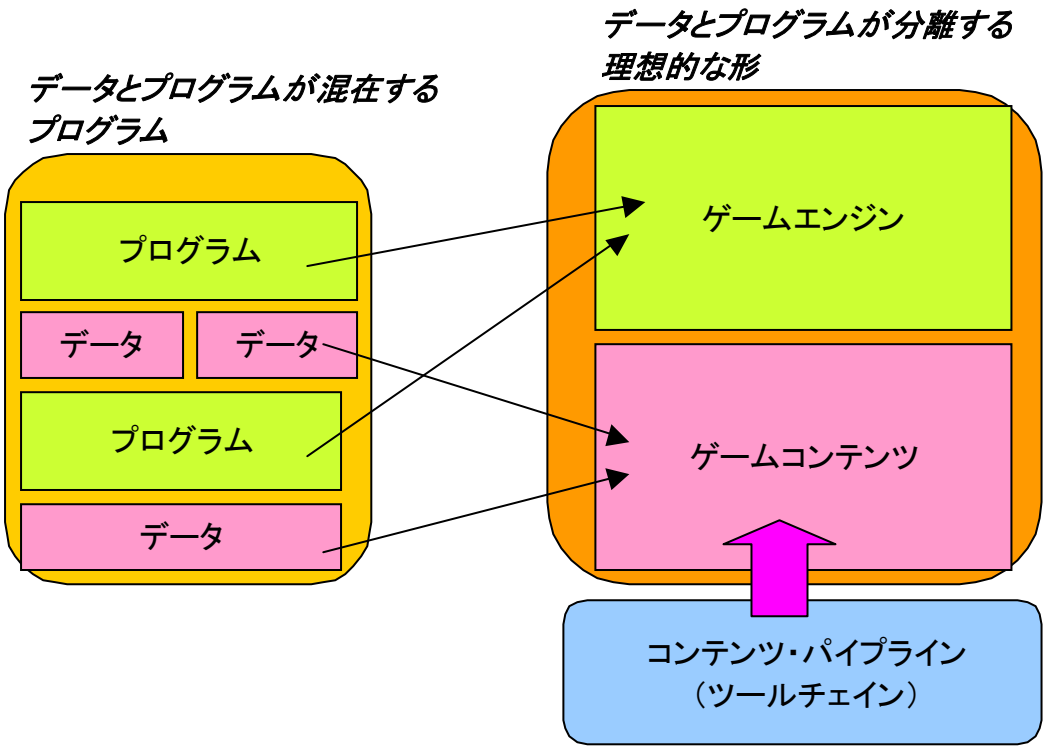


図 2.10-03　ゲームシステムの発展

ソフトウェアを製作するという感覚から、現代では管理工程まで含めた製作体制、ゲームエンジン、ゲームコンテンツパイプラインの構築がゲーム開発に求められる。

また、プログラムと同様に、増大するゲームデータの統合も課題となった。これは単に容量が大きくなるだけでなく、複数のデータ同士が互いに連携しているという点が重要である。例えば、一つのキャラクターモデルを作るには、ポリゴンモデル、各ポリゴンに貼り付けるテクスチャ、アニメーションデータ、衝突モデル、シェーダ、さらに、各パーツの衝突における衝突音（サウンド）の関連付け、さらに、それを動かす AI 思考など、各分野が製作したコンテンツを関連付けて統合する工程が必要になる。開発工程においては、ゲームコンテンツデータの形成は、複数のアーティストの手を経て統合される。こういった複数の職種、複数の工程、複数のツールによる連携は「コンテンツ・パイプライン」或いは「ツールチェイン」と呼ばれる。また、各分野の発展はこれまでにない新しいデータの必要性を促す。例えば、以前は少量のデータだけで済んでいた AI 分野も、専門的にナビゲーション用データ（移動用データ）、思考ルーチンのデータ（FSM、Production Rule…）など、特殊データを大量に準備する必要が発生しており、コンテンツ・パイプラインはこういった変化に対して、常に革新を続けて行かなければならない（**2007 年度報告書「コンテンツ管理技術」（第 5 章）を参照**）。

さて、上記のように、ゲーム開発技術は、当初内包していた分野が発展、分化、専門化、すること、3Dやネットワークのように大きなフレームの変革が行われること（挿入）、さ

らに、それらを統合する環境を構築する、という 3 つの流れによって発展して来た。

2.10.3　データ、プログラム、スクリプトの形成

(1)　データ、プログラム、スクリプトの分離

ソフトウェアはデータとプログラムからなる。ゲーム・ソフトウェアも同様である。現代のプログラミングでは、データとプログラムが明確に分化していることが理想とされている。デジタルゲームでは、特に調整に必要なパラメーターや変更可能な素材は、プログラムから分離したデータとして設計されることが望ましい。80 年代のように、プログラムの中にデータが埋め込まれていたり、さまざまな変数の値が高速化のために、そのままプログラムに書き込まれていたり（これは現代でもなくなっているわけではない。完全になくせるわけでもない）、そういった混在した状況をなるべく避け、データとコードの分離関係をシンプルな形で関係を保つことは、膨大なコードとデータの保守性を向上させるためにも必要である。

スクリプトは、80 年代後半から使用され始めたが（その前型はデータ記述である）、90 年代から本格的に使用され始めた。動的な制御ロジックをデータの形で記憶できるため、AI やエフェクト、イベントなどさまざまな制御のカスタマイズデータとして使用される。何より開発中にコンパイルなしでデータとしてゲーム内の動的なコンテンツをカスタマイズできるという点は、デバッグやコードの保守性という利点と、さらに、ゲームデザイナーとプログラマの分業を促すという利点を持っている（実際は随時スクリプトの拡張などが必要なためで完全な分業は難しい）（**2008 年度報告書ゲーム開発技術ロードマップ「プログラミング　スクリプト」を参照**）。

このように、プログラム、データ、スクリプトにおいて、ゲームソフトウェアとしての動作はプログラムに、コンテンツはデータとスクリプトの形に分離するという方向性が形成された。しかし、実際にこういった設計を実行するには、ゲームエンジンの作り込みが必要であり、完全にスクリプト・ドリブンを実現しているタイトルはそれほど多くない。

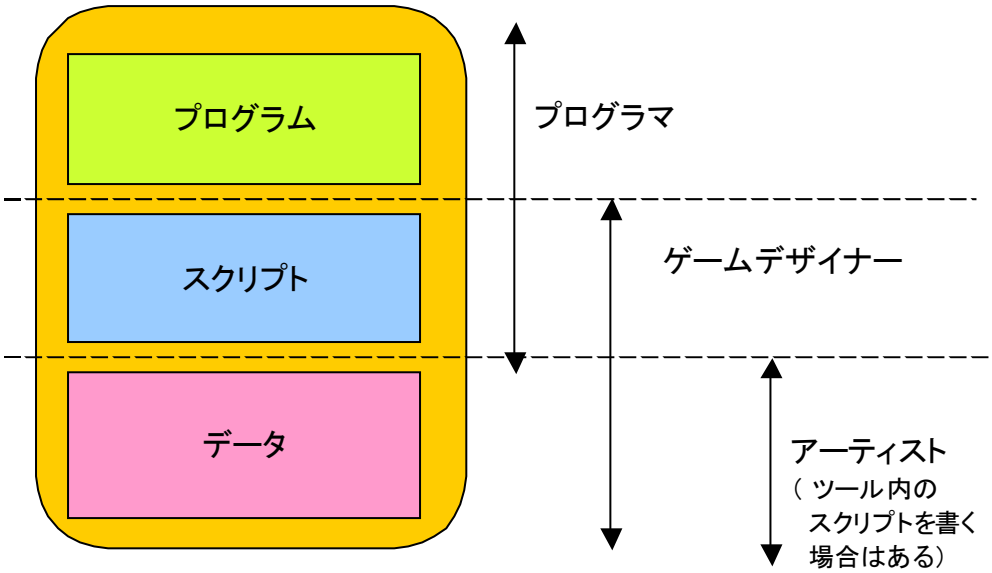


図 2.10-04　ゲームデータの分離と分業体制

(2)　データドリブン

データドリブン（データ駆動）モデルとは、デジタルゲームにおいては特に、ゲーム内のコンテンツデータとそれを動作させるコードを分離し、プログラム主導ではなく、データによってプログラムの挙動が決定される方式である。例えば、共通コードでステージ毎にデータを差し替えたデータを駆動することで、ステージが形成される仕組みである。

こういったデータドリブン型の開発は特に大規模開発におけるコードの保守性と、ゲームデザイナーとプログラマの分業を実現するのに必須である。上記のゲームデータの分離も、ゲームエンジン全体としてデータ駆動の方向に発展していると言える。しかし、現代のゲームエンジンは、データとアルゴリズムを分離する傾向にあるものの、正確な意味でデータドリブンになっているわけではない。今後、データドリブンが何処までゲームエンジンに浸透するかは、ゲームエンジンを研究する上で重要な視点である。

(3)　製作

これらのデータはプログラム、スクリプト、デザインツールなど、複数ツールから構成される開発環境をリンクする形で形成される。特にゲームデータは逐次的に製作されて行くために、新旧のファイルのバージョンや、データが要件を満たしているかを事前にチェックする機能が重要である。こういった開発環境全体をデザインすることも専門的な技術であり、各種ミドルウェアが導入されている。特に、開発の初期の段階で設計しておく必要がある。この分野の技術とノウハウは、開発体制の維持と効果に大きく効果をもたらす

ことになる（**本章「開発工程管理」、2008 年度報告書ゲーム開発技術ロードマップ「開発方法論と、その歴史」を参照**）。

(4) アーキテクチャ

ソフトウェアの構造をアーキテクチャと言う。アーキテクチャはゲームエンジンを構成する骨格であり、この構造がその後のコードの発展性・保守性といった基本的な性質を決定することになる。デジタルゲームにおけるこういったアーキテクチャの最古のものは、タスクシステムである。タスクシステムは、ソフトウェアがこなすさまざまな処理をタスクを単位として管理するシステムであり、現在のゲームエンジンでも基本設計に組み入れられていることも少なくない。また、こういったタスクシステムの源流は、商業開発においては 80 年代初頭のアセンブラの時代に既に確立されていた（**2008 年度報告書 4.10.1「タスクシステムについて黒須一雄氏インタビュー」，ゲーム開発技術ロードマップ「タスクシステム」参照**）。現代のゲームエンジンのアーキテクチャは、ゲーム内でのマルチタスクに対応する必要性や、ゲーム開発環境を含んだ PC で開発されたものがリソースを即時に実機マシン機能でテストできる環境(RMI)、製作ツールとの連携、などが要求される。ゲームエンジンのアーキテクチャの公開情報は少なく、ゲームエンジンそのものを対象とした研究も少ない。ゲームエンジンはゲーム開発の骨組みであるだけに、他のコンピューターサイエンスの成果を導入しながら発展させる本格的な研究が望まれる。

2.10.4 技術導入・流入経路

「技術がどのようにして各ゲーム開発企業にもたらされるか？」という視点は、非常に重要であり、実際、各情報経路の強さや太さが、その国全体の技術競争力に強く影響することになる。ゲーム開発の現場に新しい技術が導入・流入される経路は、次のようなものである。

① 新しいハードウェアへの対応
② ミドルウェア
③ 学術から（論文・産学連携交流から）
④ 自社独自開発技術（単なる自社開発ではなく、他社にはない新規技術の開発）
⑤ 他業界からの技術流入

などである。

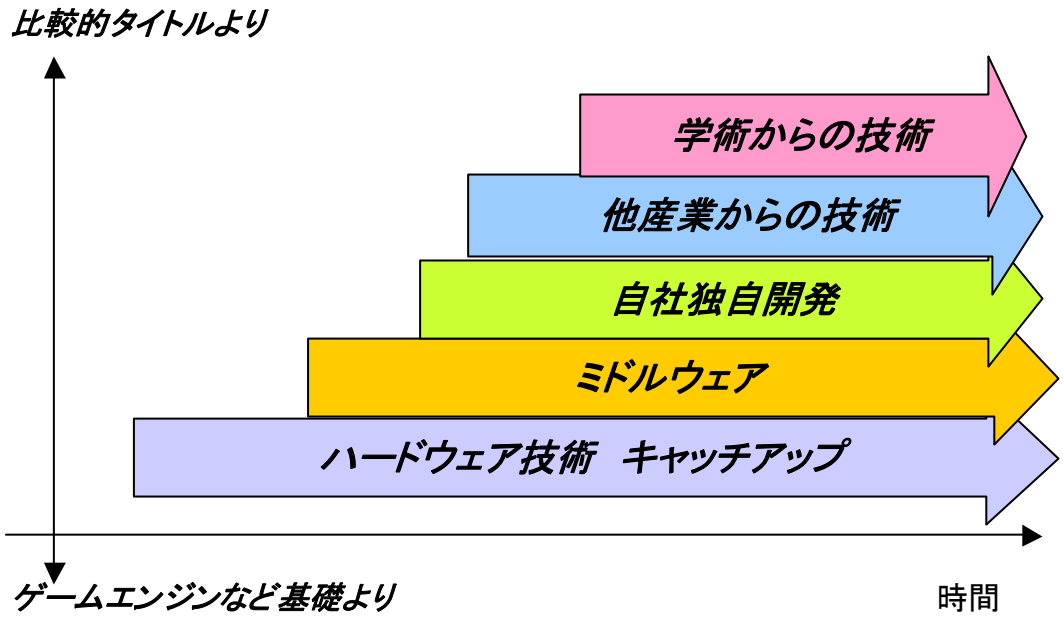


図 2.10-05　ゲーム開発企業への技術流入経路のイメージ

開発現場への技術流入は複数の経路からさまざまなタイミングでもたらされるため、そういった技術導入専門のスタッフを配置する企業もある

ゲーム開発技術の現場では、キャッチアップしなければ技術の対応と、自社の独自技術の開発に大きく分かれるが、目安としては、基幹・基礎技術ほど、キャッチアップ技術であり、タイトル特有の技術ほど自社技術である傾向が強い。現代は、ハードウェアが大きく進化したせいで、(以前から比べれば) 豊穣なメモリ・計算パワーの上で、大きく長い技術的リーチを持った力強い技術開発がタイトルに大きく反映する。即ち、コンピューターサイエンスのフィールドを深く入り組んだ技術開発が、技術競争の鍵を握り始めている。残念ながら、こういった技術開発は、この 5 年の動向を見る限り、欧米の方が強い。また、そういったコンピューターサイエンスの背景を持つ人材をよく集めている。

以下、各項目の技術流入の詳細について解説する。

(1)　新しいハードウェアへの対応

ハードウェアへの対応は、以前は 5 年置きという漠然とした法則があったが、次世代機に入り崩れ、次世代機は大幅に長期化する予想である。2005 年から始まった次世代機への移行も、2010 年になっても次の世代のプラットフォームの話題は聴こえて来ない。これは一タイトルに 2 年～5 年間の開発期間がかかる現代では、当然のことと言える。

また、プラットフォームのみならず、カメラ型、振動型インターフェースなどの対応など、断続的に変化するハードウェアに対する対応も重ねる必要がある（**本章「周辺機器メ**

ーカーから見たハードウェアインターフェースの変遷」を参照)。

(2) ミドルウェア

高い専門性をもった技術、或いは開発の補助となるような開発支援ツールをソフトウェアの形に結実させ、ゲーム産業を含む他の IT 産業に渡り提供するのが、ゲーム開発におけるミドルウェアの主な傾向である。ゲーム開発企業は、ゲーム開発ライン内部では製作する余裕のない、或いは、所持していない技術を導入することで、その技術をパッケージ、或いはツールとして吸収することが出来る。ゲーム産業で提供されるミドルウェア技術のレベルはますます高度になり、CG,AI,物理、街生成など、専門化すると同時に、その領域をますます広げている。また、AI ミドルウェアの会社では博士以外採用しないというケースもある。数年前までは、ニッチで小規模な産業だった AI ミドルウェア企業も、大手ミドルウェア会社に吸収される形でシェアを伸ばしている。また、物理ミドルウェアは CPU や GPU の企業に買収されるケースが多い（**本章「物理シミュレーション」の節を参照**)。

ミドルウェアに詰め込まれた技術は、購入することで手に入るが、利用する時点でも、それなりの技術知識が必要とされ、ミドルウェアと自社エンジンの間、或いは、ミドルウェアを社内の開発者に使いやすいインターフェースを築いたり、社内に教育する役割、アップデートに対応する役割をする人材が必要とされる。

ミドルウェアがその産業で形成される前提として、まずその技術分野のある程度共通のフォーマットが形成されること、そして、そういった共通の機能の需要があること、自社開発を始めている企業の兆しがあること、などが挙げられる。そういった漠然として形成されつつある市場に、タイミングよく完成度の高いミドルウェアを投入することが普及の鍵である。物理ミドルウェアはこの機をよく捉え、大きく飛躍した分野である。(**本章「物理シミュレーション」の節を参照**)。

(3) 学術

学術との結びつきは、80 年代～90 年代はそれほど強くなかったが、00 年代前後から、急速に強化されて行った。それまで、ゲーム開発技術は、大学の研究室の専門分野と結びつけるには、タイトル指向の実装、さらにハードウェアに依存したプログラミングなど、基盤となる技術が学問とリンクする部分が少なかった（**本章「アセンブラ」、3.5「現場から見たゲーム開発の変遷」参照**)。或いは特別なタイトルに限られていた。しかし、ゲーム業界の技術的基盤が大きくなるに従い、ＣＧ技術を契機として、さまざまな連携が可能になって来た。いち早く、その可能性に気付いたのは欧米であった（特に米)。欧米は、学術的に研究・開発された技術の応用可能性を見取ると、最初（00 年前後）はやや拒否感もあり、緩慢であったものの、2004 年前後以降は急速にその応用を前進させて来た。また、

同時に修士号・博士号を持った学生がゲーム産業に流入して来てゲーム開発技術の底上げを行っていった。

日本でも産学連携事例は決して数は多くないがいくつか実現されて来た（**2008 年度報告書 第 6 章「国内の産学連携事例の調査」参照**）。ただ、大学の研究室が直接、タイトル開発や基幹技術の開発に参与するケースは少なく、また博士号を持った人材がゲーム産業に流入する数も少なく、また、博士研究の研究背景を持つ人材を活かせるだけの技術的基盤もゲーム産業には確立していない。これから開拓される必要がある（**2007 年度報告書「産学連携」（第 5 章 3 節）を参照**）。

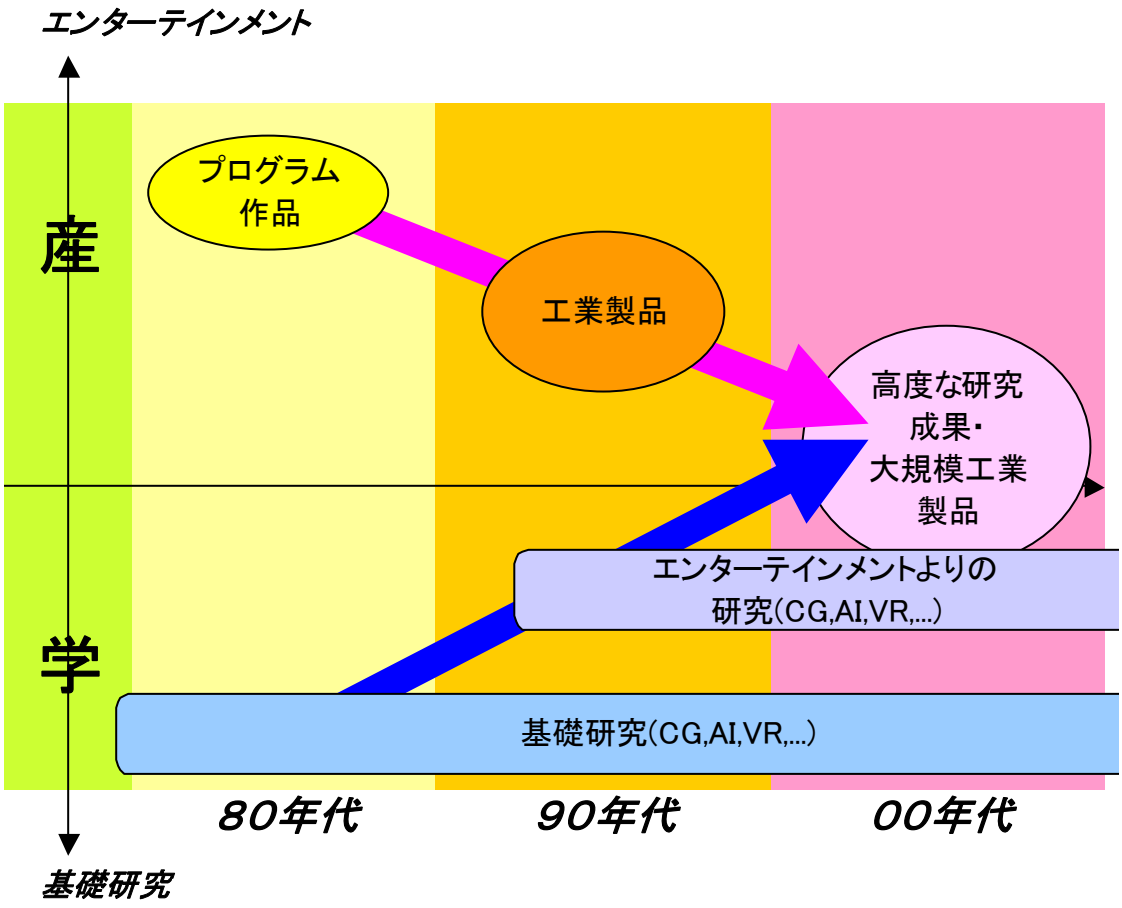


図 2.10-06　ゲーム産業と学との交差点

ゲームはプログラム作品から工業製品、そして次世代機からは高度な研究成果を投入した大規模工業製品になりつつある。そういった発展の歴史の中で、次第に学の研究分野と交差する領域が大きくなり、産と学の間が交流できるフィールドが大きくなって行った。現在は、一つのタイトルの AI の研究で学術論文が書ける時代である。

(4)　自社独自開発技術

自社内において、ある特定の技術を開発するというケースがある。これは、その企業が開発する一連のゲームにおいて広く応用可能であったり、あるゲームコンセプトを実現するためにどうしても必要であったりするケースが多い。しかし、簡単なものを除いては、実際、自社で特定の分野の技術を数ヶ月以上の一定の期間を経て開発する体制は、ある程度の規模を持つ会社でないと実現しない。つまり、

(a)　技術開発・研究に人を割くことが出来、
(b)　長期的な技術開発プランを組むことが出来
(c)　技術を必要とするタイトルを開発し
(d)　技術開発の人材を集めておくことができ、
(e)　開発ラインに人材を投入するより研究に投入した方が効果的であると判断できる

企業である。現状のところ、日本では数社に限られるが、欧米では GDC を見る限り、このケースは急激に増加している。勿論、欧米においては開発技術がタイトルのクオリティー基準と密接に結びついているという背景があってのことでもある。日本のユーザーの技術嗜好は欧米よりも弱い。

また、現在開発しているタイトルに組み込む技術を、タイトルと同時に開発しているのでは間に合わないことも多く、次の、或いは、次の次のタイトルへ向けて、事前に開発を進めておく必要がある。理想的には、そのタイトルの開発開始時には、その技術が使用できる保証が出来ていることが理想である。また、ある程度の見通しがないと、開発プランに組み込まれることさえ難しい。実際は、タイトル開発とある程度時期を重ねながら、要望を聞きつつ形成して行くことケースも多く、ゲームデザインとの刷り合わせが必要である。

(5)　他業界からの技術流入

他業界からの技術流入は、特に次世代機開発に入って重要になった。開発環境自体が、それまで、比較的、特殊な開発環境と思われていたゲーム開発環境から、他のソフトウェア産業と類似する点が多くなって来たからである(そう見えていたのはハードウェアをアセンブラを叩きつつエンターテインメントを作る、という姿勢が一般のソフトウェア開発から見ると特殊に見えたせいである。そして、そういった狭い制限の中で最大限の性能を発揮する開発こそ日本が得意な分野でもあった)。

開発の大規模化、リソース管理、XBOX360(Microsoft)のシェアの拡大、など複数の要因が重なって、ＰＣソフトウェア業界とゲーム産業の開発環境が非常に近いものになって行った。特に開発工程の技術などは、先進していた一般のソフトウェア業界からゲーム業

界に流入して来た。その情報経路は、米ＰＣ業界。米ＰＣゲーム業界、米コンシューマーゲーム業界、日本ゲーム業界、という経路であり、日本のゲーム業界はようやく開発工程技術を導入しようという流れが出来始めているところである（**本章「開発工程管理」、2008年度報告書ゲーム開発技術ロードマップ「開発方法論と、その歴史」を参照**）。

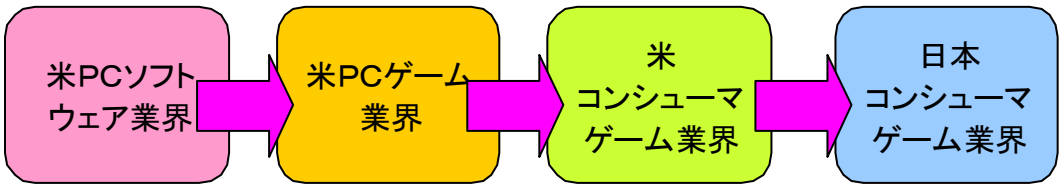


図 2.10-07　ソフトウェア工学の技術の流れ

さて、以上で①～⑤の技術導入・流入の過程を解説して来た。

① 新しいハードウェアへの対応
② ミドルウェア
③ 学術から（論文・産学連携交流から）
④ 自社独自開発技術（単なる自社開発ではなく、他社にはない新規技術の開発）
⑤ 他業界からの技術流入

このような技術流入経路のうち、①はともかく②③④⑤は圧倒的に日本の勢いが弱い部分である。

まず、②⑤が弱いの理由は IT 産業とゲーム産業の関係の弱さである。日本の IT 産業は世界からガラパゴス化（孤立特殊化）していると言われているが、さらに日本のゲーム産業は日本の IT 産業との結びつきが弱い。また、多くの企業が海外のミドルウェアを既に導入した今となっては、日本のゲーム産業相手の IT ビジネスは、そういった海外ミドルウェアメーカーとの戦いになるのである。

次に③が弱い理由は、デジタルゲームにおける技術研究の可能性が、産業・学術とも見えていないという点が大きい。それは、長い間、現在でも、日本のゲーム産業が技術情報を殆ど外に対して閉鎖して来たため、学から見ても、ゲーム開発技術が研究対象となる可能性を伺い知ることが出来なかったという事情が大きい。この部分に関してゲーム産業はもっと開発技術の現状と課題を外に向けて発信する必要がある。また、もう一つの理由は、学におけるデジタルゲームという新しい分野に対する偏見である。デジタルゲームを研究対象として扱うという提案自体が通らないことが多い。

④は、まず、深い研究が出来るだけの人材が確保できていないこと、確保できていないので、かつてそういった研究事例が公開されていないためにそういった人材が集まらない

ということ、さらに、日本のゲームの作り方が基盤技術の発展よりもタイトルに特化した開発であったため、次世代に入っても、その方向修正が出来ていないことが挙げられる。

このように、日本のゲーム産業の情報環境には問題点が多く、この点を総合的に解決することなしに、ゲーム開発技術が次なる段階に進む欧米を追い抜くことはできないだろう。

2.10.5 仮想化の歴史

(1) 仮想化とは

ここでは、ハードウェアとソフトウェアの関係に着目して、開発技術の歴史を辿って行く。この視点から見えるのは、ハードウェア、特殊環境に対する仮想化の流れである。仮想化とは、特殊環境をラップ、フィルタリングすることで全く別のシステムとして表現・構築する手法である。例えば、二つの別のシステムを、上から同一のインターフェースを重ねることで、ユーザーから見れば全く同一のシステムとして操作させることが出来る。そして、同一のシステムに見せかければ、そのシステム上で構築したソフトウェアは双方のシステムで動作させることが出来る。また、複雑なハードウェア構成も同一のインターフェースを被せることで、同じシステム、或いはより簡単な論理構造を持つシステムとして扱うことが出来る。そうした場合、仮想化されたシステムにおいては、それより下位のシステムの構造に深く干渉することが出来ない変わりに、下位構造を触れることなく、仮想化された環境の上で開発を進めることが出来る。

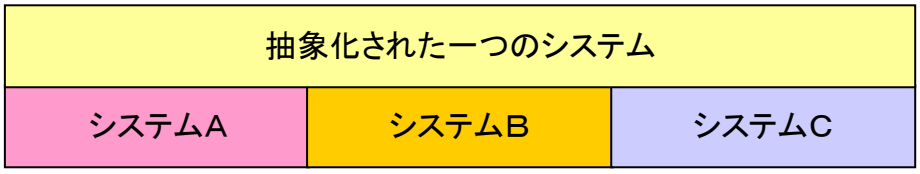


図 2.10-08　システムの仮想化

あるシステムが別の共有のシステムに仮想化されている。ユーザーから見ると一つの論理的なシステムに見える。

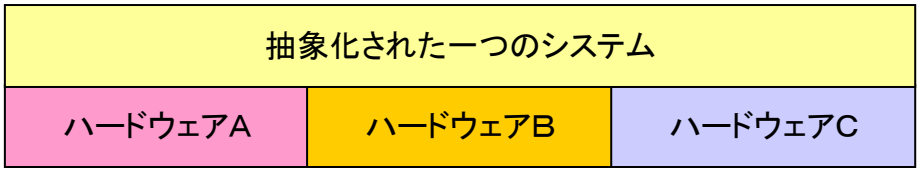


図 2.10-09　ハードウェアの仮想化

複数のハードウィアの構造が仮想化されている。ユーザーから見れば一つのシンプルな論理構造に見える。

(2)　ゲーム開発環境の仮想化

ハードウェアとソフトウェアは、最も基本的な部分では、決められた言語によって結ばれている。ハードウェア＝プロセッサーのアーキテクチャ固有の言語をアセンブラというが、プレイステーション以前までは、ゲーム開発はアセンブラ言語を用いて行うことが主流であり、プログラマは、「各プラットフォームのプロセッサーに向けて」コーディングを続けていた（**本章「アセンブラ」**）。

各プロセッサーのアーキテクチャの相違を超えて、一つの文法言語によって記述し、コンパイルによって構造の差を埋める言語を、高級言語という。プレイステーションからプレイステーション 2 の時代に多く使われるようになった C 言語は、高級言語の一つであり、これによって、全てではないが、多くのルーチンが C 言語という統一フォーマットで記述されるようになった。開発の効率化のため、各社が共通ライブラリを整備する動きが形成された(**3.5「現場から見たゲーム開発の変遷」参照**)。

次世代機（PlayStation3,Xbox3,Wii）に入り、C 言語から C++言語への移行した。C++言語の特徴は、オブジェクト指向であり、オブジェクト指向の特徴は一定の要素（メモリ領域）とそれに対する操作がパッケージングされた「オブジェクト」という単位同士の相互メッセージングシステムによって、ソフトウェアのシステム全体を構築する技術である。オブジェクト指向は大規模なプログラムを組むのに(C 言語よりは)適したシステムである。しかし、その分、設計段階における高い技術と知識、ノウハウが必要とされる。

このような大規模な基本システムの構築の必要性に対して、各開発ラインで固有のエンジンを組むという体制から、共有エンジンを構築する方向へ移行が始まった。各社は「自社ゲームエンジン」の開発に乗り出し、共有ライブラリの時代から共有ゲームエンジンの時代に本格的に移行した。

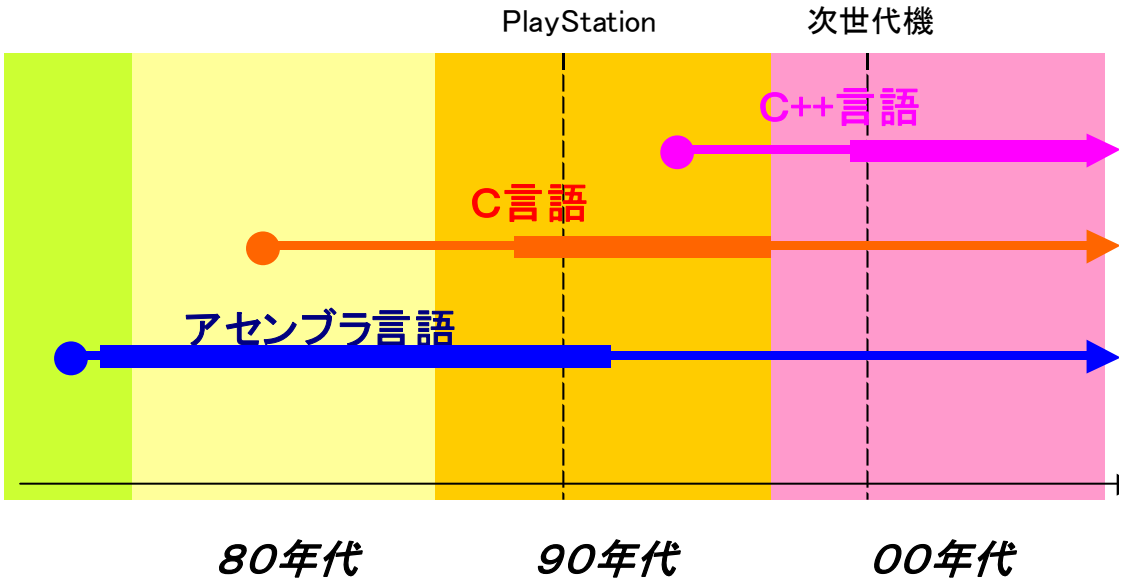


図 2.10-10　主なプラットフォーム用開発言語の変遷

その時代の最もメジャーな言語のみを表示

次世代機のゲームエンジンの特徴は主に 3 つある。マルチプラットフォーム化と、実機と連動したゲームエンジンである。マルチプラットフォーム化とは、例えば、PLAYSTATION3(SCEI)や XBOX360(Microsoft)において、開発者のレイヤーでは、なるべく、同じレイヤーに見えるように、つまり、理想的には、一つの形式で作ることで、同時に全てのプラットフォームのゲームを作ることが出来るようなシステムである。実機に連動するとは、デザイナーがツール上で作成したデータが、すぐに実機上に反映して、実際の色身や動作をチェックできる環境のことを言う。

ゲームエンジンの整備が進められる中で、もう一つの動きとして特的の分野をアウトソーシングで賄おうとする動きが出て来た。これは、物理エンジン、サウンドエンジン、AI エンジン、また、開発環境システム、データ管理システムなど、特定の分野のミドルウェアをライセンスして補おうという動きである。どの程度のミドルウェアを導入するのか、例えば、物理エンジンを購入するのか、自社開発にするのか、など、一つ一つの分野で導入が検討される。このため、各開発企業によってゲームエンジンといっても、完全な純粋な自社製のエンジンから、他社のミドルウェアを組み込んだエンジンまで多様である。さらに進んでゲームエンジンを販売するという動きも現在では盛んである。

(3)　ゲーム機、仮想化の歴史

このように、プログラマが一つのプロセッサーに対してプログラミングをしていた時代

から、プラットフォームを超えた仮想化された開発環境が開発される段階までの変化が起こって来た。これらの運動は、ハードウェア、ソフトウェア、ゲームエンジン、複数の階層を貫いて仮想化する流れである。プロセッサー依存のアセンブラはC言語でラップされ、C言語はC++オブジェクト指向によって拡張され、C++によるゲームエンジンは共通プラットフォーム化され、プラットフォームはミドルウェアによってさらに部分的に仮想化され、ゲームエンジンはそれら全てを抱え込み仮想化している。

プログラマの見る主観的な風景は、こういった仮想化の歴史の中で、プロセッサー、ライブラリの上の空間、オブジェクトが作る空間、ゲームエンジンアーキテクチャという巨大構造物の空間、と移り変わって来た。各時代によってプログラマに見えている風景はまるで違う。しかし、それは前の世代が見ていた風景が見えなくなることではない。後になればなるほど、前の世代が持っていた風景を持ちながら、新しい風景を見る必要が出て来る。例えば、ゲームエンジンという巨大なソフトウェアから見た場合、理想的にはハードウェアに依存するコードは下位に仮想化されていることが望ましい。にもかかわらず、ゲームエンジンは特定のハードウェアの性能を十分に引き出すことが必要であり、ハードウェアの性能を引き出しながら、仮想的なアーキテクチャ構造を維持するという二つの相反する要件を満たしながら開発が進められている。

次の世代のプログラマが見る風景は、コンピューターサイエンスの研究領域を含んだデジタル空間上の開発だろう。そういったコンピューターサイエンスの可能性を盛り込み、強力な基盤を構築することなしに、ハイエンドなゲームタイトルの開発の高度さについて行くことは出来ないだろうと予想する。

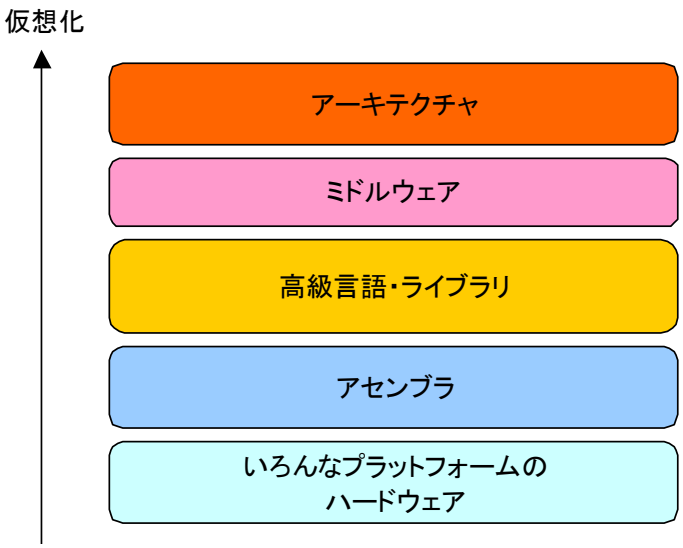


図 2.10-11　ゲームシステムの仮想化構造の簡略化した概要

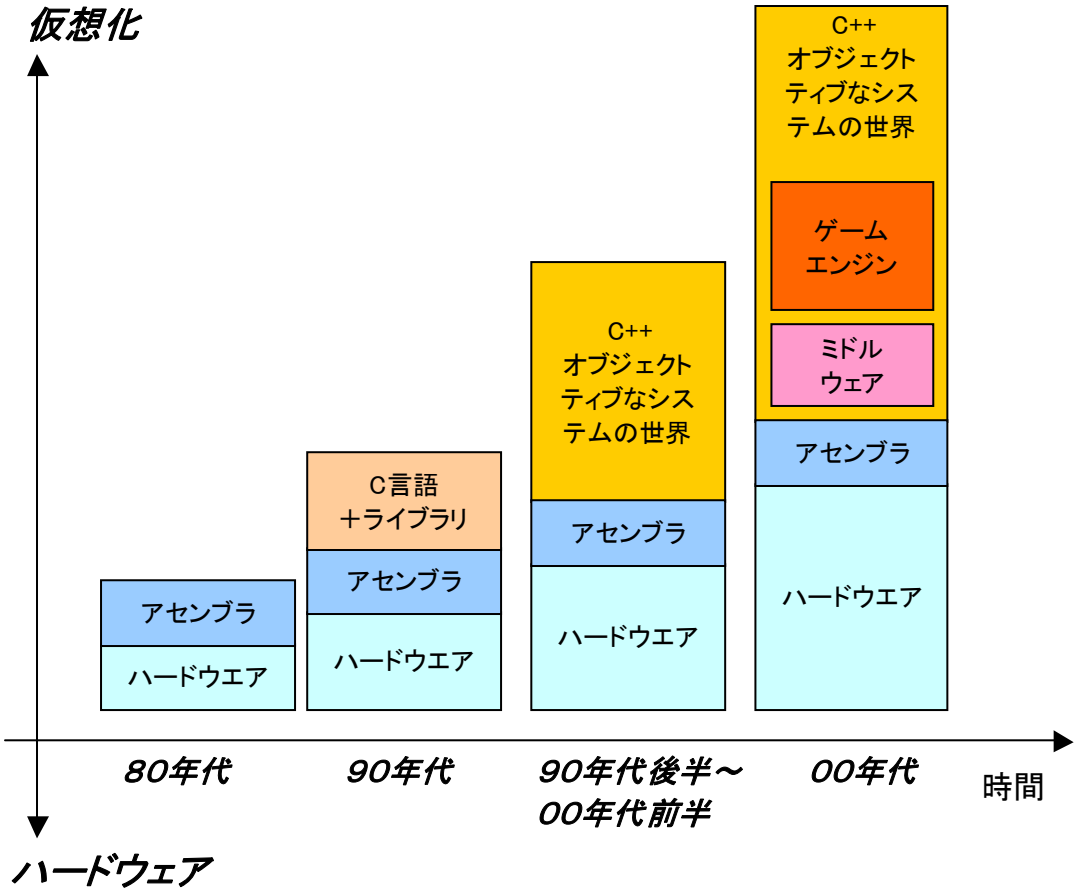


図 2.10-12　ゲームエンジンの仮想化の歴史

2.10.6　固定コンテンツの展開とプロシージャル

ソフトウェアの規模と性能は、データの大きさとプログラムのパフォーマンスを評価の目安とすることが出来る。大量のデータをチェックしながら比較的ゆっくりと変更するデータベースのようなシステムもあれば、少量のデータを高速に変化させるようなシミュレーションのプログラムもある。ゲームプログラムは、大量データの高速処理の分野である。

デジタルゲームの一つの方向として、準備したデータを適切なタイミングで高速に再生する方向で発展して来た。ますます準備できる容量が大きくなり、それを高速に処理するという方向である。特に CG、サウンドを大規模にボリューム化することで、多大な開発コストの増大をさせつつも、質の高い描画コンテンツとサウンドでユーザーを牽引して来たのが、この 10 年の動向であり、この方向は、これからもやや減速しながらも大きな勢いで継続して行くだろう。また、特に日本においては、ストーリーテリング型のゲームが大きく発展し、ストーリーに説得力を持たせる CG、ムービー、シナリオの部分が大きく増大したため、こういったデータアウトプット型の部分が大きく集中的に発展した（**本章「シナリオ」参照**）。

こういったデータアウトプット型に対して、デジタルゲームには、もう一つ、プロシージャルの方向がある。プロシージャルとは、手続き的な処理のことで、ゲームの状況に合わせてコンテンツが変化させ、状況に合わせたコンテンツの展開がされることを言う。つまり、デジタルゲームの動的動作、データ自動変形・生成機能のことである。通常、ゲームソフトウェアとして想起するイメージそのものである。しかし、敢えてプロシージャルという言葉を使う必要が出て来たのは、上記で解説した通り 90 年代後半からのデータ容量の圧倒的増大とその展開に費やされる計算コストの増大に対して、ゲーム本来が持つプロシージャルの力を復権しようとする運動があったである（**2007 年度報告書「ゲーム・プロシージャル技術分野」（第 4 章）を参照**）。

データをコントロールすることと、プロシージャルは相反する機能ではない。どんなデータもある種の状況に合わせて加工されて出力されるのであるから、プロシージャル的と言える。最近は、そういった大量のデータの上に再び、プロシージャル的な処理の比重を大きく復権しようとする動きが既に出始めている。

その理由の一つは、より一層のデータの作りこみで対応する限界が見え初めており、ある程度のプロシージャル処理を入れないと、クオリティーが頭打ちになってしまう予想されているからである。

例えば、剣の材質や振り下ろした角度で、反射音を変えたいとする。あらゆるパターンのデータを用意して場合に応じて再生するよりは、その場合に応じて処理を加えて再生するなど、プロシージャル的な処理で補うことが出来る。また AI の思考においても、大規模ゲームでは、あらゆる場合を想定して行動を基底することはもはや出来ない。プロシージャルにその場で行動のプランを作成するプロシージャル AI の方向へ発展している。

ただ、現在のプロシージャル技術の傾向は、以前のように完全に自動生成を目指したり、小さなデータを使い回したり、完全な自動処理の方向ではなく、既に構築した大量データの上に立って、それらを強力な計算リソースで利用しながら新しいコンテンツを作り出すという流れである。つまり、かつての小メモリ上の小さな足場でプロシージャル的な処理を回していた時代から、大規模データ、大メモリ上（80～90 年代に比べれば）、リッチな計算リソース上に、もう一度、大規模にプロシージャル処理を展開しようとする流れである。大規模データとプロシージャルの掛け合わせは、これからのハイエンドゲームの天井を破って行く最も有望な手段である。

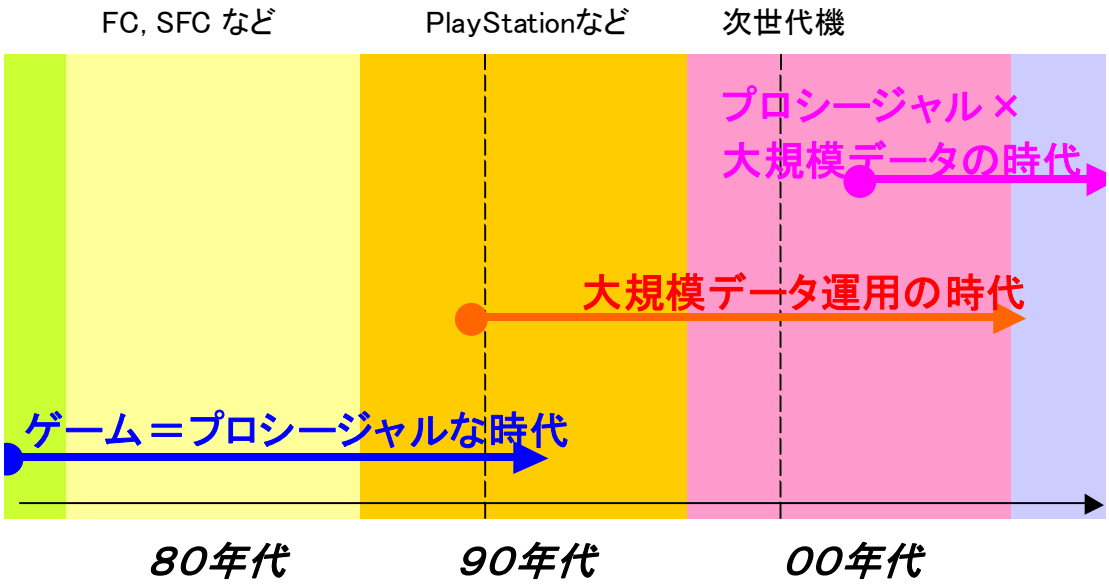


図 2.10-13　プロシージャル的な傾向の歴史

2.10.7　技術からマネージメントへ

これまでは技術の発展の流れを解説して来た。ゲーム開発技術は、複数の要因で拡大しつつ、そういった技術を構造で抑え込もうとする運動が、ライブラリやアーキテクチャ、ゲームエンジンなのであった。

そして、現在は自社で抱え込む技術に対して、チーム全体で各分野をカバーする時代である。欧米では専門化の流れが激しいが、日本でも、やはり各分野を一人でカバーする時代ではなく、チーム全体で、或いは、会社全体で広大な技術分野をカバーする、それでも不足するところは、ミドルウェアなどによって補完する時代である。そこで、こういった技術者集団をまとめて連携させる手段（マネージメント）が必要になってきた。

その手段の一つとして技術セクションの出現、強化である。各企業で、開発ラインとは独立に、「技術部」「開発サポート部」「技術支援部」などの名称で、ライブラリ、ゲームエンジン、開発環境の整備を目的に運用され、それらを通じて開発効率を間接的に上げ、さらに新しい技術や製品の導入やチェックを担当する。さらに一歩進んで、研究開発部署も実現している。このような部署は、前世代機の頃になかったわけではないが、現在、以前よりずっと大きく拡大・強化される傾向を持っている。

前世代機まではいわば、ライブラリを提供しつつも、開発は開発ラインの自律的な運動に任せる（今でもこういったところは多い。特に日本は、実際、開発現場は伝統や人材の個性に大きく依存しているのが良いコンテンツを産み出す長所であり、必ずしも一辺倒に上から合理的に変化することができない短所である）というものが殆どだったが、工程管理や、さらに一歩進んで、スクラム、アジャイル、定期ビルド、など、コーディングのマ

ネージメント、開発組織のマネージメントの手法の導入も検討されるようになって来た（**本章「開発工程管理」、2008 年度報告書ゲーム開発技術ロードマップ「開発方法論と、その歴史」を参照**）。

これらの技術支援部署の整備という点は、特に日本の企業の著しい点である。海外は主にプロジェクト単位で仕事が動くが、日本は比較的、会社という組織を通して為し得ることを目指すという体制であるから、外から必要な技術を持つ人を連れて来るよりは、社内に技術部署を設置して技術の拡大に対応するという傾向が産まれたと考えられる。また、スタジオ制の強い海外スタジオと違って、日本の中～大企業では、社内に複数のラインを抱えるケースが多く、共有化による効率化を求めてこのような傾向を産み出したとも考えられる。

ところが、キャッチアップに優れたパフォーマンスを持つこういった部署の問題は、強いインパクトを持って業界をリードする技術を投入する力が比較的弱いことである。どちらかと言うと、技術支援部署は整備・改善に力点が忙殺される傾向が強く、新規技術開発はその次のプライオリティーである。今後、技術部署から新しい技術やインパクトのある技術を出して行く、或いは近い将来、開発ラインから要望されるであろうレベルの高い技術をあらかじめ準備して行くためには、研究セクションを独立させる、技術ロードマップを描いて計画的な技術研究開発を着実に進めておくなど、支援以外の更なる仕事が必要である。こういった研究開発は、日本のゲーム産業が世界に技術的存在感をもってタイトル開発を進めて行くためには必須のことであろう。

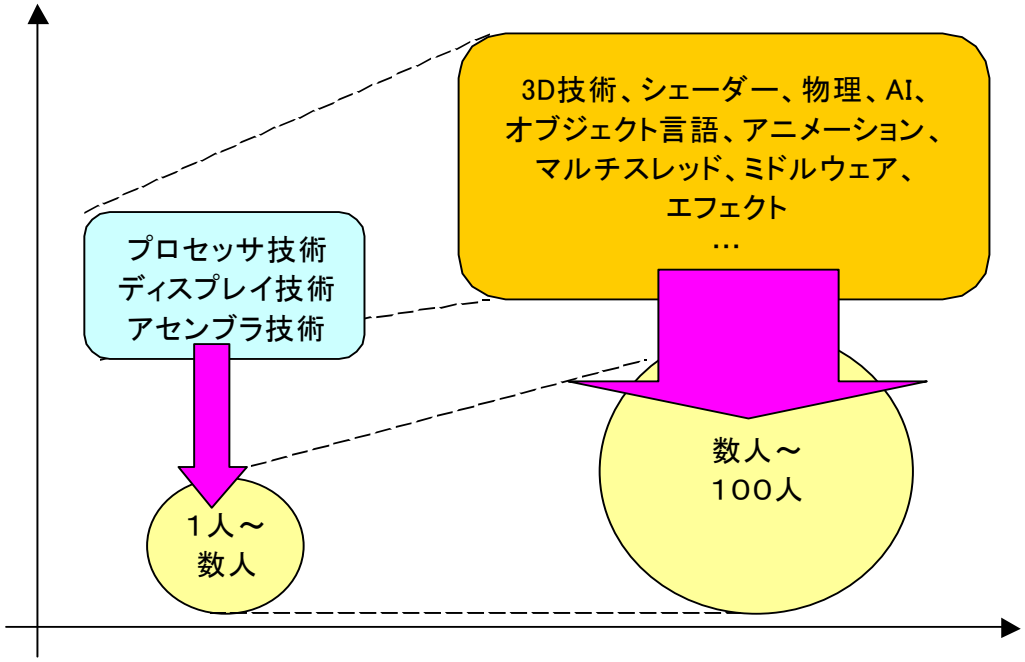


図 2.10-14　技術の拡大とそれをカバーする人材の比率

かつては数人で一定の技術分野をカバーしておけばよかった時代から、現代は、部署やその企業全体で広大な技術分野をカバーする時代である。

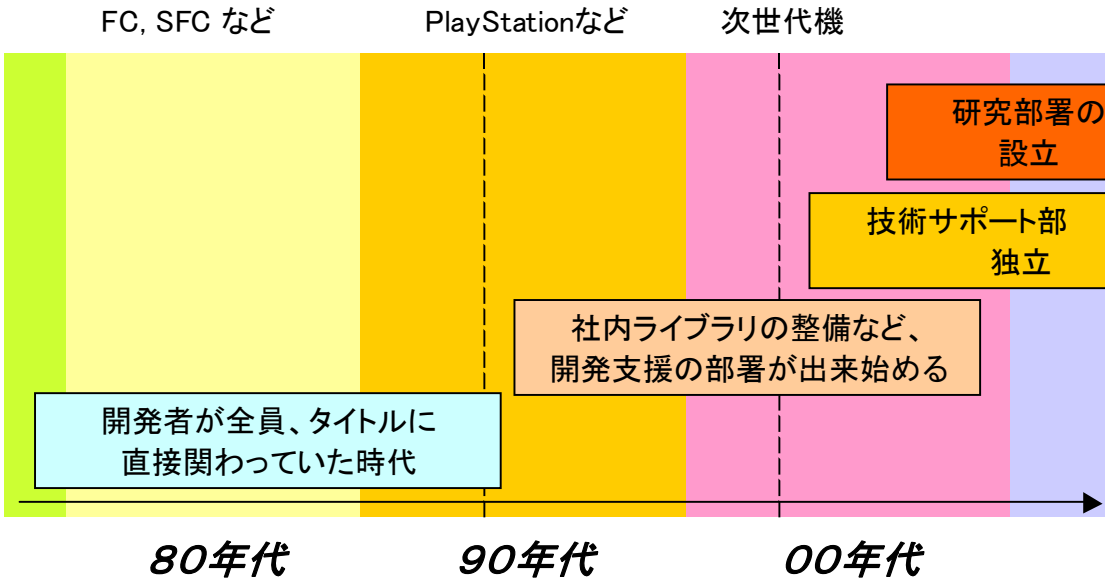


図 2.10-15　ゲーム開発企業の技術開発における組織的変化

2.10.8　ゲームデザインと技術

(1)　技術とゲームデザインの関係

技術開発にどれだけの重点を置けるかということは、ある程度以上は、少なくとも各タイトルにおいては、ユーザーの嗜好性にもかかっている。欧米のように技術導入に対して敏感なユーザーに対してタイトルをリリースする場合と、日本のように特に技術指向が目立たない国における、エクステンドな技術開発への力の入れ具合は異なるのである。

少なくとも、日本の開発現場では、日本のユーザーの非技術指向は、見えざる抑圧として、開発の現場が技術偏重になることを抑制している。つまり、欧米においては技術によってゲームデザインの土台を築こうとする一方で、日本では、ゲームデザインの領域を技術側から神聖不可侵のように捉え、ゲームデザインの領域に技術から介入することを恐れるのである。また、技術からゲームデザインを始めようとする傾向は殆どない。逆に、技術とは制約であるという考えが強い。これは欧米と日本で極めて対照的な点である。とすれば、ゲーム開発技術の「のびしろ」が少なかったこれまでの世代から、次世代機に入って技術的のびしろが大きくなった現在、技術の上にゲームデザインを展開する欧米のゲームが急速に発展し、相対的に日本のゲームの力が弱まるのは当然のことである。

また、こういった傾向が次世代機に入っても続いたために、日本のゲームデザインと技術は一つのデッドロック状態に陥っている。即ち「現行の技術レベルでゲームデザインを行う､そのゲームデザインでは技術が伸びない､技術が伸びないので技術レベルそのまま、現行の技術レベルでゲームデザインを行う…」と言った悪循環である。これは、ハードウェアと共に強い推進力を持って進化する CG 分野などでは比較的見られないが、純粋なソフトウェア分野である AI などでは特に著しく見られる点であり、日本のゲーム AI 技術は欧米に 10 年の開きがあると言われている（**2008 年度報告書ゲーム開発技術ロードマップ「プログラミング AI」を参照**）。

(2)　技術の上のゲームコンテンツ

このように、技術の大規模化、硬直化は、単に技術の問題のみならず、ゲームデザインの硬直化にも直結し､ゲームタイトルの力にも深刻な影響を及ぼしている。一方､現在は、欧米が高い技術力の上に新しいゲームデザインを実現しつつある時代であり、多くの日本の開発者は深い危機感を抱いている。また、欧米のゲームカルチャーと日本のゲームカルチャーの分離が進行し、日本のゲームが海外で受け入れられ難くなり、逆に海外のゲームが日本で売れないという事態を継続させ、相対的に大きな市場を持つ欧米のゲーム圏が資本力を持ってゲーム産業をリードするという現象を産み出している。そういった成長を背景に、巨大な資本を一つのゲームタイトルに集中させ、最低ラインが百万本、最高が 1 千万本というスケールの開発が標準となり、同時に技術開発に資本投入が為されている

世界市場を背景とした強力な技術開発が各社だけで難しい日本のゲーム産業においては、ゲーム開発技術の発展は、ゲーム産業を挙げて課題として行かなければ太刀打ちできないであろう。

もともと日本の得意分野であったゲームデザインも、既に広く世界のゲーム企業で学習され、日本のアドバンテージは相対的に現象している。「洋ゲー＝粗いゲームデザイン」は、既に遠い過去のものである。そして、次世代機では、自由なゲームデザインのためには確かな技術力、技術ポテンシャルが必要とされており、欧米では長期開発の中で新規の独自技術開発とその上に展開する新しいゲームデザイン開発を同時に実現するケースが増加している。また、世界販売を見越したローカライズの作業も、ゲーム完成後ではなく、ゲーム開発中から協調して始めなければならない段階に来ている（**本章「ゲームローカライズの歴史とこれから」参照**）。

(3)　ゲームデザインまでの距離

例えば、キャラクター3体とプレイヤー1体のアクション戦闘を組みたいという場合に、かつて数百～数千行と簡単なスプライトで済んでいた時代から、描画からアニメーションまで、多人数の連携した仕事が必要な現代においては、アイデアの着想から実現テストまでのアプローチが非常に長くなってしまっている。こういった現状で、開発後期までゲーム性のテスト・イテレーションも出来ないという課題がある。大規模開発かつ、ラピッドなイテレーションを実現するために、テスト環境を準備したり、開発工程を工夫したり、各社、さまざまな工夫が為されている現状であり、大規模かつ質が高いコンテンツを産み出すことが、世界市場における勝負の分かれ目になっている。

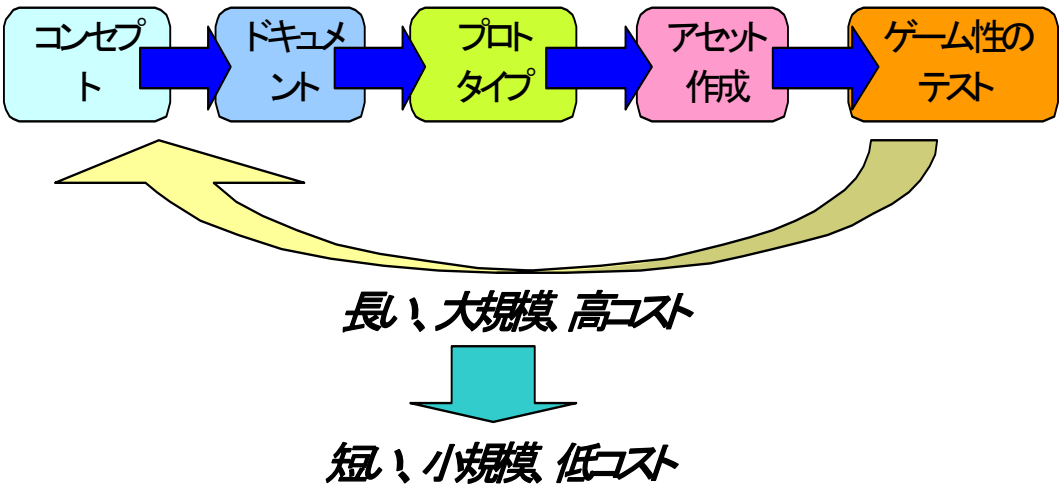


図 2.10-16　ゲームアイデアから実現までの遠い道のり

アイデアを思いついてから、ゲーム性を実現してテストするまでの距離（時間、コスト）が以前よりずっと長くなっている。こういった体制では、ゲームの開発終盤までゲーム性のチェックを行うことが出来ないという問題がある。現代では、たとえ大規模ゲームであっても、できるだけ迅速なゲームデザインのテスト＆エラーが軽い形で行える体制が求められる。

大規模開発のベクトルは多くの開発者を効率的な開発に目を向けさせたが、それは、同時に大規模化するデータとその連結によって、アイデアを自由にテストしにくい、身動きの取れない開発体制に陥られることになった。つまり、「コンテンツ・パイプライン」は、パイプラインというだけあって、その工程に簡単に干渉したり、変更したりすることは、それが大規模になれなるほど難しく､融通の効かないものになってしまっているのである。

簡単なデータ、とりあえずのデータ、少し特殊な処理を入れるテストなど、一旦出来上がったパイプライン上で実行するには、一人一人のデザイナーに個人的に依頼をしに回らなければならない。分業の弊害はそういった点にも現れる。そこで、そういった反省や先見の明のある企業は、たとえ大規模開発であっても、低コストでアイデアを試行錯誤できる環境の構築を目指した。プロシージャルによるデータの軽量化、事前プロトタイプの長期化、QA の強化など、大規模開発にこなれて来た企業は、今度は、ゲームデザインのプリプロセスとポストプロセスで徹底的にブラッシュアップをかける方向に移行しつつある。

(4)　企業のサイズと技術力の相関について

企業のサイズと技術力の相関について厳密に調査された結果は殆どない。また、何をもって技術力というのかも定かではない。現代はプログラミングだけで技術が語り得る時代ではなく、これまで記述したように技術の領野は 90 年代後半より大きく広がっている。つまり、そもそも、ある程度の規模の企業でなければ、自社内であらゆる分野の最先端にキャッチアップすることは難しい。そこで、アウトソーシングやミドルウェアの導入が検討される。また､(前世代までは大型タイトルに従事していたとしても) そもそも次世代機開発を行わなくなるという企業もある。

一方で、大企業だからと言って、必ずしも技術的レベルが自然に高くなれるというわけではない。技術をカバーし、各分野でトップを走ろうという方向を持つ仕事がなければ、また、そういった技術を社内で浸透しようとしなければ、結局は、開発ラインが複数集まった集合体にしか過ぎなくなってしまう。

社内の技術開発を積極的に進めるためには、

① 技術開発にふさわしい人間を集め
② 集中的に技術を研究・開発し
③ 開発ラインに還元する体制

が必要である。しかし、この 3 つを成功させることは、決して簡単なことではない。

①に関して言えば、タイトル開発ラインで活き活きとする人間と純粋に技術開発に打ち込みたい技術者を見分ける必要がある。開発と研究の二つの資質は違うものであることを認識しておかなければならない。

②は、解説して来た通り、広範囲、多様な技術、それぞれに適応した技術者を集めなければならない。また、そういった技術者のキャリアパスも新しく開発しなければならない。タイトル開発ラインのように、メインプログラマーのような明確な目標ポジションが見えないことが問題になる。

③は、一見簡単に思えるが、困難を極めるところである。もしも、開発ラインの中で開発した技術ならば、そのラインに吸収されることは、それほど難しくない。しかし、開発ラインの外の技術支援部署から、開発ラインへ向けて新しい技術を提案すると、そこには部署間の壁や、技術者以外の、プロデューサー、ディレクターなどを介して、技術の移転コストまでを含めて提案する必要が出て来るからである。これは企業が大きくなればなるほど、大きな問題になる。また、時間が経てばたつほど、技術支援部署と開発ラインの文化や方向性が違って来るものである。技術支援部署を独立させることは、他の部署との関係性を構築できなければ、完全にその企業の中で浮いてしまう危険さえある。独立させると同時に、その企業全体の中でのポジションと役割を明確に打ち出す必要があるのである。そのためには、逆に、部署という単位でなく、サウンド、物理、AI など、分野ごとの横のつながりを継続的に作っておくことが必要である。例えば、物理や AI 分野など、分野ごとに、週に一回ミーティングを行う、などである。そこで顔を合わせ、アイデアを出し合い情報を交換するなど、開発ライン、技術支援部署を超えた分野ごとに「盛り上がる」ことが出来る場を用意することが必要である。

2.10.9　これからのゲーム開発技術の発展のために

ゲーム開発技術ロードマップでは、各分野のゲーム開発技術の歴史を俯瞰して来た。また、本節では、そういった歴史的な運動を構成する流れを抽出し解説して来た。各分野のレポートでは、これからの展望を最後に記述して頂いている。この節では、これまで開設して来た 7 つの傾向を要約すると同時に、これからの日本のゲーム開発技術の発展に必要な方針を挙げて行きたい。

(1) ゲーム開発技術の発展の歴史（各分野の拡大と独立（専門化）、統合など）

ゲーム開発の大規模化と技術の高度化は、各分野の発展と分化、専門化、そして職業化を促した。80 年代から 90 年代初頭まで、ゲーム開発の必要な技術は 1～数人で保持することが出来たものが、現在では、数人～100 人規模のチームによって保持され、運用されている。専門家を職業化する欧米と違い、日本は複数の分野を経験させオールラウンダーを養成し、メインプログラマーを育てるというケースが多いが、それは開発ライン全体を調和して動かすスキルを向上させる利点と、各技術分野を専門化して持つ人材の養成を阻害するという欠点を持ち合わせている。つまり、日本の開発の現場射は、専門的技術を継続して保持し､発展させて行く体制に欠けているので､技術が深く定着しにくいのである。

専門化した職種はその技術を保持するのみならず、応用を通して各分野を発展させる役割を持っており、この機能が欠落している日本のゲーム産業の技術力が、専門化の進む欧米のゲーム産業に対して相対的に低くなったのは当然の帰結であり、日本が欧米の開発技術のキャッチアップに従事するのは必然である。

一つは、専門的人材を育て、そういった人材の開発の現場におけるキャリアパスを構築すること（開発者の育成による対応）が必要である。また、もう一つは、社内の技術部署の役割のひとつに発展的・パイオニア的研究を目標とし、長期的に高度なゲーム開発技術を開発して行く技術プランを構築すること（チームによる技術への対応）が日本のゲーム産業には必要である。

日本の開発の現場はプログラマ文化が主であり、エンジニアを受け入れ育む体制が出来てない。そのため、エンジニアの訓練を受けて来た場合も、数年でパイオニア的な仕事をする方向性が削がれてしまう。全てとは言わないが、部分的にせよ、エンジニア文化を開発の現場で育くんで行くことが必要である。

(2) データ、プログラム、スクリプトの形成

データ、プログラム、スクリプトという層の区別は、長い時間をかけて形成されて来たものであり、こういった区別は同時に、職業的な役割分担を可能にするものである。こういった分業体制の中でも、あいまいな協調体制の中でコンテンツの質をアップして行くことが出来るのが、日本の開発の美点であり問題点である。欧米は逆に、こういった分業によって、ゲームデザイナーの作家性を発揮できる環境を構築することを目標にしている。欧米における技術的な最初の到達点とは、ゲームデザイナーが自由にゲームを構築できる環境を構築するところにある。

どちらの方向性にせよ、もう一度大規模開発の上に、ゲームデザイナーの尖った個性を再現できるような開発体制の構築が必要である。

(3) 技術導入・流入経路

プラットフォームの変化、周辺ハードウェア機器の導入、新規基幹技術の導入によってキャッチアップ的に引き起こされる技術の底上げに加えて、現代では、学術的、或いは自社独自の技術開発によって他の企業から差別化するという傾向が見られる。ゲーム開発企業の競争力とは、タイトルにおける技術とゲームデザインという要素に分解すれば、特に欧米では、技術レベルによって他企業に打ち克つという意識が高い。一方日本では、ゲームデザインにおける意識が高く、技術はその次の評価項目という意識が高い。こういった意識の違いが、欧米では、博士号・修士号を持つ学生を引き入れ、さらに産学連携によって開発技術の向上に継続的に努め、技術力を対外的にアピールするという行動の原因となっている。現在、多くの技術が欧米経由で取得されており、日本のゲーム産業同士、或いは、日本から世界へ発信される開発技術は少ない。必然的に、日本のゲーム開発会社の技術的存在感は薄い（それは、毎年、GDC(ゲーム開発者の国際会議)で実感できる)。

こういった現状を打破するには、日本のゲーム産業における情報環境の改善を行う必要がある。ゲーム開発技術を蓄積し、部分的にせよ産業全体で共有し、業界全体として技術の底上げを行う。そして、公開した情報をもとに、学に対してゲーム開発技術の現状を知らせ、研究対象としてのフィールドを提供することで、産学によるゲーム開発技術の推進体制を築く必要がある。また、国内のソフトウェア業界から技術的に孤立した状態を改善し、技術交流を進める必要がある。

(4) 仮想化の流れ

アセンブラ言語でハードウェアを制御するという感覚でゲームを開発していた時代から、30 年を経て、ハードウェアとプログラマが触る言語の間には、数層に渡る仮想化・抽象化の構造が導入されて来た。そしてゲーム・ソフトウェアは大規模かつ高度に構造化されて来た。また、その恩恵として、開発効率と高度なハードウェア制御、大容量・高速処理が実現されて来た。

さまざまな仮想化・抽象化にはさまざまな技術が投入され、ゲームエンジンはその上に、高度に抽象化、かつハードウェアに特化した性能を最大限引き出すという二面性を持った構造物として発展して来た。ゲーム開発はもはや、魔術的なコードの集合ではなく、れっきとした科学的構造物である。また、科学的構造物であれねばならない。ゲーム開発環境、ゲームエンジンはその意味で、コンピューターサイエンスの研究対象として研究され得る対象であり、学術・産業をあげて研究対象とすることが必要である。

(5) 固定コンテンツの展開とプロシージャル

80 年代、まだゲームのデータリソースが少なかった頃、ゲームは小さな画面を計算リソースで動かし続けることで豊かに見せていた。こういった時代では、ハードウェアの性能を最大限引き出すことを目標とされていた。そこでは、2D 画面上のリソースとプロシージャル的な操作がよいバランスで機能していた。

90 年代からの 3D への変化は、大型化したデータを如何に効率よく動かすか、またグラフィックを美麗に見せるか、という点を重心として進んで来た。00 年代からは、そういったデータ偏重に対して、明確にプロシージャルという言葉を用いて、再び、ゲームステージを自律的に躍動したものにしようとする方向の運動が復権した。

そして、現在は、大規模なデータの上で、プロシージャルな展開する段階に来ている。単にリソースをうまく制御するところから、ゲーム内の状況に合わせたプロシージャルな処理を最大限、データの上に実行することで、これまでのデータとプロシージャルという対立を超えた段階へ上がろうとしている。

この段階は、技術とコンテンツが一体となった技術研究が必要な段階であり、プログラマとアーティストが協調して新しい機能を追及する時代である。技術者、アーティストの対立を超えた柔軟なコンテンツ技術開発を実現する必要がある。

(6) 技術からマネージメントへ

かつてゲーム開発技術は一人から数人でもカバーできる時代があった。しかし、現代のハイエンドな開発では数人から、技術サポート部を含めれば数十人で、ゲーム開発技術全体の中核をカバーする時代になっている。開発に特化する要員を含めれば数十人～数百人の規模になる。

そういった多人数が従事するゲーム開発においては、開発工程の構築技術、開発工程の管理技術が重要になっている。こういった技術は、もともと、先に大規模化していた IT 産業において育まれた技術であり、その技術が米の IT 産業から PC ゲーム業界、そしてコンシューマーゲーム業界へと伝播して来たのである。現在でも、ゲーム開発者が参考にする開発工程技術の参考書は、殆どが一般の IT 産業の著者が書いた書籍である。

開発工程構築の技術は、先に述べた「ツールチェイン」「コンテンツ・パイプライン」、「リソース管理技術」「プログラム統合環境・ビルド環境」として、開発初期のラインの確立において応用される。

開発工程管理技術は、継続的に効率のよい開発体制を維持するための技術である。一度確立した体制はタイトル開発が終わるまで、小規模な改革を繰り返しながら運用される。欧米では 2004 年～2008 年は、そういった開発工程技術の導入期であり、大規模化するゲーム開発をうまくマネージメントしようとする意気に満ちていた。日本のゲーム業界でも、少し時期が遅れながら、開発工程構築の技術、開発工程管理技術の重要性が認識され、さ

まざまな試行錯誤が行われている。しかし、まだ十分ではない。

しかし、人間関係が非常に密な日本において、開発の現場において工程管理の原理を導入することには、さまざまな抵抗がある。また、IT 産業で育まれた開発工程管理技術をそのままの形で適用することはできない。ゲーム開発には、さまざまなリソースを管理するという側面があり、またアーティストとエンジニアのコラボレーションであり、ゲーム開発工程、特有の管理ノウハウを積み重ねる必要がある。

(7) ゲームデザインと技術

ゲーム開発技術は純粋な技術であると言っても、特定のゲーム分野と共進化をするという傾向がある。例えば、欧米では FPS 分野を通じてゲーム AI 分野が進化し、また、リアリティーのある画面を求めて、シェーダ技術の導入・進化が促された。一方、日本では、どちらかと言えばリアルよりは幻想的な、また、体験型よりはストーリーテリングに重きを置きながらゲームが進化して来た。偶然とは言え、欧米では前者のリアル指向の技術が深く大学・研究機関の研究や技術と結びつき飛躍的な進化を遂げた。日本のゲームタイトル中心主義は、そこまで学術と強い関係を結ぶことが出来なった。

現在、ゲームデザインの可能性を引き出すことは、技術的基盤なくしては不可能である。大きな技術的基盤が各社で確立した現在だからこそ、それに飲まれない新しいゲームデザインへ調整して行くことが必要である。ゲームデザインと技術の対立と緊張関係が、よいコンテンツを生み出すのである。

CG にせよ、AI にせよ、リアリティーを重視する欧米のゲーム開発技術に対して、日本はその後を追いながらも、日本のコンテンツ自身が求める技術を探し当て、育むことが重要である。日本が、独自の技術の潮流を産み出すとき、初めて、欧米の技術と対等に対峙することが出来る。例えば、日本のゲーム AI 技術は、欧米のように FPS 分野を前提に進化するのではなく、日本特有のジャンルを苗床として共進化させることができる可能性を持っている。そして、そういった東と西と、起源を異にする別系統の技術の融合こそ、世界のゲーム産業技術を次の段階へ登らせる原動力となるはずである。

2.10.10　ゲーム開発技術ロードマップ調査の終わりに

ゲーム開発技術がますます広大、かつ多様に発展して行く時代において、ゲーム開発者が、ゲーム開発技術の歴史の中で一体、今、自分が何処に立っているのか、これまでの発展がどのような流れになって自分の足元に流れ込んでいるかを知ることは重要なことである。ゲーム開発技術の潮流は、その時代の開発者を知らずのうちに教育し、鍛え、次の時代へ推し進める役割を持って来たのである。

歴史は与えられるものではない。獲得するものである。狭い歴史しか知らないものは、狭い発展しか想像できず行動を束縛される。大きな歴史を知るものは、より大きなスケールで現代と未来を見て､大きく自由な行動を取ることが出来る｡ゲーム開発技術の歴史は、ゲーム開発技術者によって、再構築されなければならない。

本ゲーム開発技術ロードマップ調査は、開発者にゲーム開発技術の歴史のこれまでの広がり、そして、これからの広がりを知る機会と情報を与え、より大きく自由な開発活動を行う足場を提供することを目的として行われたものである。

残念ながら、現在、日本のゲーム開発技術の力は、欧米の大規模資本、大規模開発の前に、存在感をますます薄くしつつある。我々が強く大きなゲーム開発技術の潮流を作って行くためには、何をすればよいか考えたとき、導くべきその潮流の現在までの流れを知っておくことが、まず必要だと考えたのである。行き詰まった流れ、停滞した流れ、偶然うまく行っている流れ、十分強いがより質を高くしたい流れ、さまざまな流れの原因を明らかにし、改善し合流させて行くことで、日本は日本なりの開発技術の潮流を再び強く導くことが出来るはずだと考えるのである。

全部で 21 分野の先達に執筆の依頼をし、インタビューを重ねることで、少しずつ、この 30 年間に渡る日本のゲーム開発技術史の全体像が浮かび上がって来た｡そこでは､我々、日本の開発者が得意な分野で世界をリードした時代の姿や、逆に欧米の開発者が得意な分野で台頭して来た経緯が明らかになって行った。ゲーム開発技術史の複雑な流れの一本一本の糸が見え始め、その糸がどのように現在に繋がっているかも辿ることが出来た。そして、これから我々が為すべきことも、そして、それがどう未来に繋がって行くかも、今や高い視点から見下ろし俯瞰することが出来る。

最終的に､本調査は､ゲーム開発者に､歴史の最先端に立つ意識を与えたい｡我々に今、必要なのは、自分たちでゲーム開発の歴史を築いて行くという意識そのものである。ゲーム開発技術は、何もかも欧米から降って来るものを待ったり、教えられたり、キャッチア

ップするだけのものではない。それは、我々自身が創造し発信して行かなければならないものでもある。

デジタルゲームは今や世界中で行われている。ゲーム開発技術は、場所や文化を超えた極めて普遍性の高いものであると同時に、タイトルやその国の文化やゲームデザインを支える、極めてローカルなものでもある。大切なのは、それぞれの国、それぞれの企業の開発技術を発表し、対峙させ、新しいものを積み上げて行こうとする姿勢である。いかなる開発者も、歴史の前に立って、歴史を作って行く一員である。

我々は何を産み出すことが出来るだろうか？

歴史は我々の手のうちにあり、それは我々の技術開発への意識によって、いかようにも形成して行けるものである。そんなワクワク感こそ、ゲーム技術者の醍醐味の一つであり、埋もれてしまった好奇心や冒険心を今一度取り戻し、日本のゲーム開発技術の潮流を、力強く再出発させる原動力となることを、本調査は望むものである。

3. 様々な切り口から見たゲーム開発

3.1 技術革新と家庭用ゲーム産業

小山　友介（芝浦工業大学）

多くの産業（特に製造業）において、新技術の採用は製造コストの低下と、それに伴う製品価格の低下をもたらす。しかし、ソフトウェア産業においては、技術が発達してハードウェアが高度化するほど、開発規模・コストともに肥大化してゆくという問題を抱えており、家庭用ゲーム産業も例外ではない。加えて、家庭用ゲームではゲーム機の本体価格（ハードウェア価格）・ソフトウェアの価格ともにも上昇傾向にある。結果として、技術革新を重ねるほど、消費者、ハードウェア生産者、ゲームソフト開発者の負担が上昇してゆき、勝利者がどこにも居ない、というジレンマを抱えることになる。

本節では、ハードウェア価格とソフトウェアの開発費から、家庭用ゲーム産業の置かれている現状を据置機中心に議論する。

3.1.1 家庭用ゲーム機の地位の変化と、技術的な収斂

1990 年代半ばまでのコンピューターは、非常に高価であっただけでなく、性能も非常に限られていた。そのため、画像処理が中心のゲーム用コンピューターと、事務処理が中心となるオフィス用コンピューターや PC との間には、アーキテクチャに大きな差異が存在していた。1990 年代に始まったコンピューター性能の急速な向上によって、各プラットフォームの差が解消されていったことが、ゲーム産業内での家庭用ゲーム機の地位にも大きな影響を与えている。

第 1 世代／第 2 世代のゲーム機で採用された技術は、アーケード（ゲームセンター）で既に実績があるものを家庭用ゲーム機に移植したものであった。このことは、ハードウェアで遊べるゲームの「顔」として、ハードウェアと同時（もしくは数ヶ月以内）に発売されたゲームタイトル（ローンチタイトル）を見れば一目瞭然である。1983 年のファミリーコンピュータのハード同時発売タイトルは『ドンキーコング」『ドンキーコング Jr」『ポパイ』で、2 ヶ月後に 4 本目のタイトルとして『マリオブラザーズ』(いずれも任天堂)が発売された。セガ・マーク III では『ハングオン』(1985、セガ)、PC エンジンではアーケードゲームのキャラクターを当時大ヒットしていたビックリマンに入れ替えた『ビックリマンワールド」(1987、ハドソン)で、いずれもハード発売から少し前にアーケードでヒットしたタイトルである[1]。基本的に据置型の家庭用ゲーム機は、数年前からアーケードで使わ

[1] メガドライブのローンチタイトルはアーケードのヒット作の続編である「スペースハリアーII」で、これもアーケードゲームに準ずる。なお、任天堂はアーケードから撤退したため、スーパーファミコン以降のローンチタイトルは「マ

れて信頼性の面で確証されている技術を、大ロット生産によってコストを大幅に下げることで低価格を実現していた[2]。任天堂の開発者であった故横井軍平氏の言葉にあるように、「枯れた技術の水平思考」が実践されていたのである。なお、プレイステーションでは『リッジレーサー』(1994、バンダイナムコゲームス)、サターンでは『バーチャファイター」(1994、セガ)がローンチタイトルであり、アーケードゲームの移植でハードウェア性能を示す傾向は第 3 世代まで続いた。

第 3 世代である 1990 年代から、ビジネス・技術の両方で環境が激変する。ビジネス面では映像と音楽が融合した「マルチメディア」が将来莫大な市場を生み出すと考えられており、ゲームが現段階で既に市場が成立しているマルチメディアの期待の星、普及の尖兵として注目されるようになり、ゲーム機の莫大な普及台数が技術開発の償却を容易にするという発想の元で、家電業界からの参入が相次いだ。技術面では、1990 年代初頭から一部のアーケードゲームで採用されていた 3DCG が徐々に一般化しつつあった。加えて、コンピューター産業全体の技術革新速度が急上昇し始めた時でもあったことが、市場の競争に大きな影響を与えた[3]。

こういった中、フィリップスの CD-i と松下電器（当時）の 3DO は発売が早すぎたこともあって性能不足で早々に姿を消し、セガは「家庭用に 3DCG が入るのはもう一世代先」という判断を覆してサターンに 3D 機能を追加した[4]。任天堂は新ハードの開発が難航し、ハード競争に大きく出遅れた。最終的にこの世代で勝利したハードは、過去の遺産にとらわれず、完全に 3D に特化したマシンを開発したソニーのプレイステーションだった。

第 3 世代ゲーム機以降の技術トレンドの変化は、ゲーム産業内での家庭用ゲーム機の立場も変化させた。アーケードゲームを開発する会社にとって、3DCG 基板の開発は技術的ハードルが高く、自社で開発できていたのは実質上ナムコとセガの 2 社のみだった。そのため、第 3 世代の家庭用ゲーム機は、両社の最もハイエンドなアーケードゲームには劣るが、それ以外のゲーム会社の基板より高性能だった。ソニーがナムコと共同開発した互換基板（システム 11）の外販をきっかけとして、多数のアーケードゲーム用の互換基板を生み出された。互換基板はアーケードゲーム会社に普及し、アーケードゲーム会社のハード

リオ」シリーズとなる。

[2] 例えば、ファミリーコンピュータ用の CPU はリコー製のカスタムチップ（6502 ベース）を用いているが、300 万個を一気に発注したという話がある（Sheff 1993、山名 1994．このエピソードについては否定的な意見もある）。TV などの新製品の多くは月産数千台からスタートすることを考えると、同一の製品を大量生産するという点で、家庭用ゲーム機は他の電気製品とはまさに桁が違っていた。

[3] メガドライブに採用された CPU である MC68000 は、当時もまだアーケードでは現役だった。MC68000 をモトローラが最初にリリースしたのが 1979 年であり、当時のゲーム産業のアーキテクチャの発展が非常に緩やかであったことがわかる。

[4] 元々、サターンは当時のアーケードゲーム用基板 System32 を簡略化したものとして開発が進められていた（山名前掲書、126 ページ）。ソニーがフル 3D のゲーム機を開発しているという話を聞き、3D 性能を急遽追加した（麻倉 1998、202 ページ）。このことがサターンの設計を複雑化し、開発の難度が上昇したことでサードパーティの参入が難しくなっただけでなく、コストダウンを難しくしプレイステーションとの値下げ競争の中で赤字を拡大させた。そのため、3D 機能の追加がプラスに影響したかの判断は難しいが、3D 機能が無いままに発売された NEC の PC-FX のセールスが全くふるわず、NEC がゲーム機市場からの撤退したことを考えれば、3D 追加の判断によって次世代機（ドリームキャスト）につながったと言える。

ウェア開発能力は低下していった[5]。互換基板の普及は世代が進むにつれて強まり、現在のアーケードゲーム用基板は家庭用ゲーム機の互換基板と（PC の高性能化に伴って普及した）組み込み Windows が搭載された PC 互換基板がほとんどとなっている。

第 3 世代ゲーム機の時代は、Windows95 の登場によって PC の普及が爆発的に進み、それに伴って PC 用のグラフィックボードで激しい競争が繰り広げられた時期と重なる。PC でも 3DCG の普及は急激に進んだが、そのグラフィクス性能を誇示するベンチマークとしてゲームが用いられた。ゲーム産業の技術革新は、それまでのアーケード主体から、家庭用ゲーム機と PC 中心に移行した。最も高性能なゲームマシンの地位は、発売直後は家庭用ゲーム機が占め、しばらくすると次々と新しい CPU とグラフィックボードを生み出して直線的に性能向上する PC が抜き、それを前世代機から爆発的に性能を向上させた家庭用ゲーム機が抜き返す、というサイクルを繰り返すようになった。

1980 年代から現在までの、アーケードゲーム、家庭用ゲーム機、PC の技術的な特性の変化を図 3.1-01 に示す。第 2 世代までの家庭用ゲーム機は技術的にはアーケードゲームの下位にあったが、画面の描画性能では PC を凌駕していた。それが第 3 世代でほぼ同等となり、第 4 世代～現世代では逆にアーケードゲームが家庭用・PC より技術的に下位となっている。

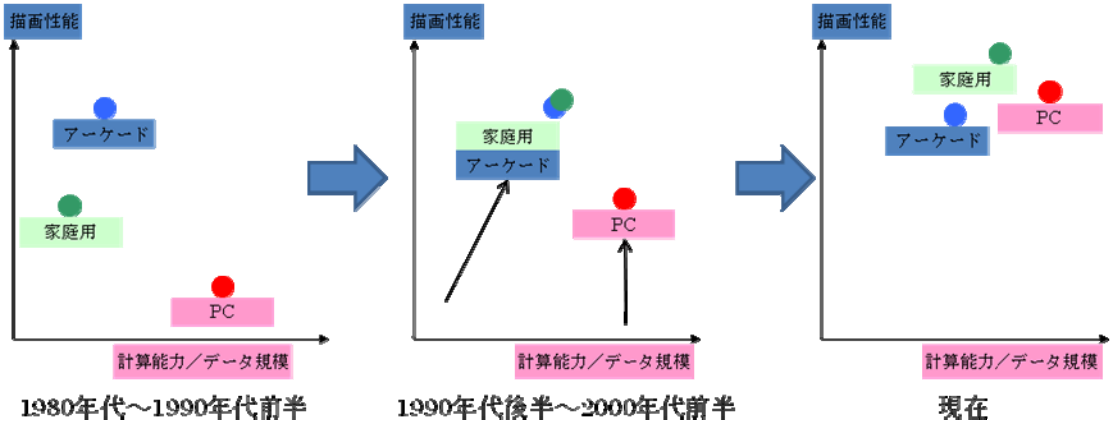


図 3.1-01　1980 年代から現在までの、アーケードゲーム、家庭用ゲーム機、PC の技術的な特性の変化

現在では、 CPU のアーキテクチャが x86 系（PC）と PowerPC 系（家庭用ゲーム機）であることを除くと、両者にほとんど差がないと言っていい。第 2 世代までの家庭用ゲーム機はハードウェア性能を極限まで使い切るため、汎用 OS すら存在しなかったが、現在では全てのマシンに OS が搭載されている。ネットワークに接続されているのも同じである。PC と家庭用の双方で発売されるゲームタイトルも多く、技術的には同一アーキテクチャに収斂する寸前である。

5 この傾向を決定づけたのは、1998 年にセガが発売したドリームキャスト互換基板である NAOMI 基板である。

3.1.2　据置ハード価格の上昇と端末普及重視型ビジネスモデルの危機

AV 機器や PC などの技術革新が激しい耐久消費財では、市場の成長期に性能の向上と製品価格の低下が同時に進展することが多い。発売当初は割高だった製品も、長期的には手頃な価格となり、最終的には一人一台レベルにまで低下する。下図は消費者物価指数からとった、携帯型オーディオ、薄型 TV、PC 用プリンタの価格推移である（http://www.stat.go.jp）。製品価格が急激に低下しているのがわかるだろう。

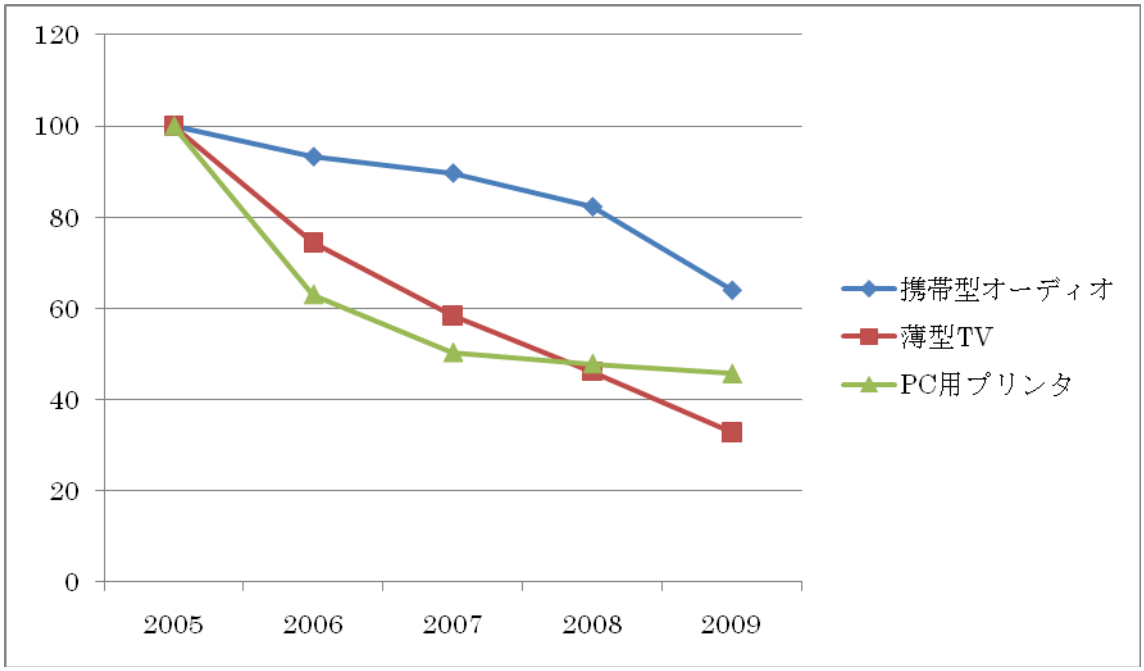


図 3.1-02　消費者物価指数の推移（2005 年=100）

この「常識」から考えると、家庭用ゲーム産業はかなり異様である。表 3.1-01 は、各世代の据置型ゲーム機が発売されたときの本体価格の一覧である[6]。各世代の下の（）に入る数字は、各世代の本体価格の平均を示している。

[6] ファミリーコンピュータ以前に発売されたさまざまな機種、3DO や PC-FX などの市場を形成できなかった機種は除く。

表 3.1-01　据置き型ゲーム機の発売時の本体価格

世代	発売年	企業	ハード名	新発売時 本体価格
第 1 世代 （14933）	1983	任天堂	ファミリーコンピュータ	14800
	1983	セガ	SG-1000	15000
	1985	セガ	セガ・マークⅢ	15000
第 2 世代 (23600)	1987	NEC-HE	PC Engine	24800
	1988	セガ	メガドライブ	21000
	1990	任天堂	スーパーファミコン	25000
第 3 世代 (36533)	1994	セガ	サターン	44800
	1994	SCE	プレイステーション	39800
	1996	任天堂	ニンテンドー64	25000
第 4 世代 (32375)	1998	セガ	ドリームキャスト	29900
	2000	SCE	プレイステーション 2	39800
	2001	マイクロソフト	Xbox	34800
	2001	任天堂	ゲームキューブ	25000
第 5 世代 (38258)	2005	マイクロソフト	Xbox 360	39795
	2006	SCE	プレイステーション 3	59980(60G) 49980(20G)
	2006	任天堂	Wii	25000

据置型ゲーム機を販売する企業の中で、任天堂だけはハードウェア価格上げないようにコストを絞り、過度の性能上昇を避ける戦略を採用している。任天堂だけは黒字を継続しているが、第 5 世代で据置機を発売している残りの 2 社のゲーム部門はかなり苦戦しており、ゲーム以外の収益部門で支えている状況である。本節では、苦戦しているソニーとマイクロソフトを中心に議論し、任天堂については最終節で取り上げる。

ハードウェア価格は第 4 世代で一度低下するものの、全体的に本体価格が上昇傾向であることがわかる（25000 円をずっと守っている任天堂を除いた場合、本体価格の平均はさらに跳ね上がる）。価格性能比で見ると発売当初のゲーム機は非常に高いコストパフォーマンスであるのは間違いないが[7]、この間に CPU を使う機器である PC が価格を飛躍的に低下させていることを考えると、ゲーム機本体の価格推移の傾向は電子機器の中ではかなり特殊である。

ハードウェア価格がじわじわと上昇していったのは、第 2 世代では別売の周辺機器だっ

[7] PC とデジタルカメラに関しては性能向上が著しく商品の更新サイクルが短いため、平成 17 年以降は物価指数の作成にヘドニック法が用いられている。市場の実勢価格と物価指数の間には乖離が存在するため、比較資料としては用いなかった。

たディスク系メディアの駆動装置が内蔵された（第 5 世代ではハードディスクも内蔵された）こともあるが、それ以上に、家庭用ゲーム機が「枯れた技術」を利用して安価なハードをつくるニッチ商品から、莫大な投資をして開発した技術を採用する最先端の製品へと変化し、ハードウェア価格に上乗せされている開発費の償却部分が大きくなったためである。第 5 世代で投資金額が跳ね上がったことで、ゲーム産業はその期待に応えることが難しくなっている[8]。

また、本体価格が上昇しているにも関わらず、発売当初のソニーとマイクロソフトのゲーム機本体の販売価格は原価を割っている[9]。そのため、ハードウェアが十分普及し、ゲームソフトの販売本数が増加して十分なロイヤリティ収入が得られるまでは赤字が増え続ける。ソニーのゲーム部門はプレイステーション 3 の発売以降は 2008 年 10-12 月期と 2009 年 10-12 月期を除くと赤字であり、プレイステーション 3 の開発と普及にかかったコストを回収できる目処はたっていない。マイクロソフトでゲーム部門が含まれているエンターテイメント&デバイス部門は 2009 年が 1 億 6900 万ドル、2008 年は 4 億 9700 万ドルの黒字だが、 Xbox 時代からの累積赤字は 50 億ドルを超えており[10]、こちらも回収の目処はたっていない。他分野へのシナジー効果を狙って据置型ゲーム専用機の開発に巨額の R&D 投資を行うビジネスモデルは、破綻の危機にある。

3.1.3　ソフトウェア開発費の爆発

一方、ソフトウェア（ゲームタイトル）は、ゲーム機本体以上にコストが上昇している。表 3.1-02 は、 一般的なアクションゲームの開発コスト・期間の一例である[11]。世代交代とともに、急速に開発コストが上昇していることがわかる。

---

[8] ソニーがプレイステーション 2 の CPU である Emiotion Engine の開発にかけた費用は 100 億円（「ソニー「プレステ 2」開発へ　画質、映画並み」、日本経済新聞 1999 年 3 月 1 日号）、半導体のためにかけた投資額は 1200 億円である（「プレステ 2 用　LSI　1200 億円投資」、日本経済新聞 1999 年 3 月 5 日号）。これがプレイステーション 3 では Cell などのゲーム用半導体に投資した金額は約 2000 億円に跳ね上がる（「ソニー、半導体投資縮小　PS3 用　次世代の量産見送り」　日本経済新聞 2007 年 2 月 13 日夕刊）。同記事によると、Cell に全て自前で投資した場合にはさらに数千億円かかり、回収の可能性が不透明なためにゲーム用の大がかりな新規投資を今後は控えるとある。最終的に、ソニーは Cell を共同開発した東芝に Cell 生産設備を売却している（「ソニー「セル」生産設備買収、東芝、半導体事業の中核に。」　日本経済新聞 2007 年 10 月 19 日号）。

[9] 発売当初のプレイステーション 2 の製造コストは 440 ドル超（「PS2 の製造コストを試算　440 米ドルと製品価格を上回る」、日経エレクトロニクス 2001 年 1 月 29 日号、175-186 ページ）、Xbox は米国での販売価格 299 ドルのところを 300 ドル超（「Xbox を開けてみた」、日経エレクトロニクス 2001 年 12 月 17 日号）、プレイステーション 3 の部品・製造コストは 800 ドルを超えるという試算がある。「全貌を現した PS3 と Wii その開発思想を読み解く」、日経エレクトロニクス 2006 年 12 月 4 日号、79-88 ページ）。また、Wii は発売当初から部品コストは 15000 円を下回っていたようである（日経エレクトロニクスのプレイステーション 3 と同一記事）。

[10] 「Xbox360 のゲームオーバー？」、日経 ITpro 記事（2007 年 5 月 24 日）
http://itpro.nikkeibp.co.jp/article/COLUMN/20070522/271875/?ST=win

[11] 経済産業省、『ゲーム産業政策～ゲーム産業の発展と未来像～』、2006 年、30 ページの表を一部改変・追記。

表 3.1-02　一般的なアクションゲームの開発コスト・期間の例

世代	発売年	予算	工数(人月)	実期間	メディア	容量
PS	1994	2 億	350	6 ヶ月	CD	0.6G
PS2	2000	7~8 億	1000	2 年	DVD	4.7G
PS3	2005	14~16 億	2000	3 年	BD	50G

開発費が 5 倍以上になったにもかかわらず、ゲーム 1 本の価格は、6090 円（PS）→7140 円（PS2）→7329 円（PS3）[12]と、2 割強しか上昇していない。仮にゲーム会社がゲーム 1 本販売するにあたって受け取る金額を 4000 円とすると[13]、開発費 16 億円を回収するには 40 万本の売上が必要となる。日本国内販売のみでこの数字をクリアできるタイトルは非常に少ない[14]。図 3.1-03 から解るように、21 世紀に入ってから北米と欧州の市場が急拡大しているため、海外市場での売上も併せて資金回収するスキームが構築できれば開発費の回収が見込めるが、海外市場で苦戦している日本のゲーム会社は非常に厳しい状況にある。

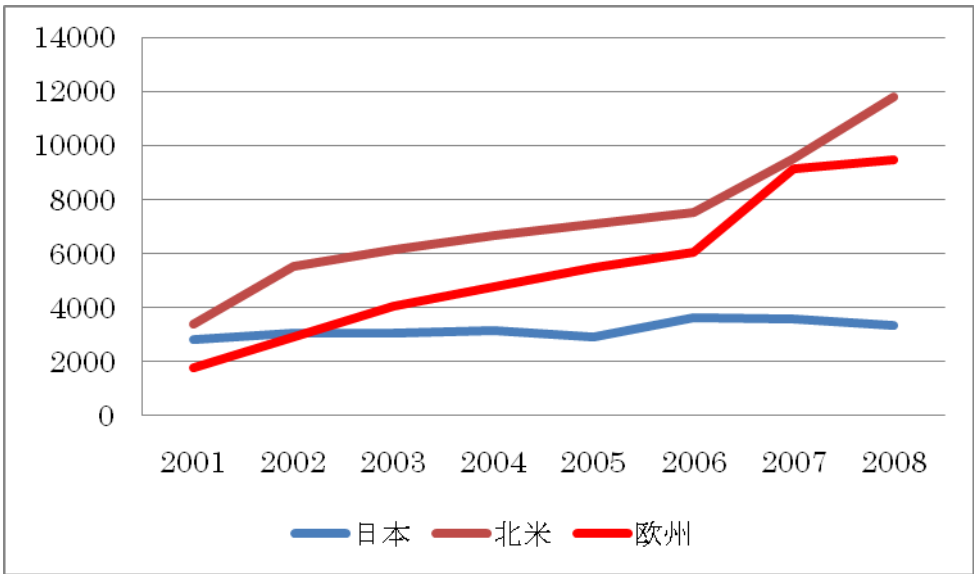


図 3.1-03　世界のコンシューマーゲームソフトウェアの市場規模推移（単位：億円）（2001～2008 年）

[12] それぞれのハードでシリーズ新作をハード同時発売している、バンダイナムコゲームスのリッジレーサーシリーズの価格を用いた。PS が『リッジレーサー』、PS2 が『リッジレーサーV』(2000)、PS3 が『リッジレーサー7』(2006)。
[13] ゲームが 1 本の販売価格のうち、流通とハードウェア会社（ライセンス料＋メディアの委託生産料）の取り分を除いた金額がゲーム会社の収入となる。正確な数字が明示されることはないが、本稿では一つの目安としてソフトウェア価格（7000 円）の 60%弱にあたる 4000 円をゲーム開発会社の取り分として計算した。
[14] 『ファミ通ゲーム白書 2009』のランキングによると、2008 年に国内販売本数が 40 万本を超えたのは 25 タイトルしかない。また、2008 年 4 月～2009 年 3 月にプレイステーション 3 で発売されたタイトル数は 152、販売タイトル数が 6,029,705 本であり、1 タイトルあたりの販売本数は 4 万本を少し下回る。

表 3.1-03　北米トップ 100 タイトル中の日本製タイトル数

	任天堂	任天堂以外	合計
2005	14	16	30
2006	20	15	35
2007	25	9	34
2008	22	13	35
2009	25	10	35

出典：http://vgchartz.com

3.1.4　家庭用ゲーム機の存続の危機？

現在までの家庭用据置き型ゲーム機の産業内での地位の変遷をまとめると、「ニッチ→トップランナー　→　PC との併走」となる。この次にどうなるのか、というと「存続の危機」ではないだろうか。技術革新が進むにつれ、アーケード、家庭用、PC に間に大きな性能差が存在していた頃にあった棲み分けが崩れ 3 者の垣根がほとんど無くなった。アーケードと同じゲームが家庭用や PC で遊べるようになり、PC の価格が家庭用ゲーム機と同水準かそれ以下に低下した現在では、家庭用とアーケードの存在意義が揺らいでいる。ゲーム市場は新興国へと拡大していくことは間違いないが、既に PC を保有している新興国の人に別途ゲーム専用のコンピューターを買わせるのは難しい。据置型ゲーム機は PC、携帯型ゲーム機も携帯電話（特に iPhone）で代用される可能性が高い。

こういった事態を示唆しているのが、日本国内でのゲームプラットフォームの拡散である。図 3.1-04 は家庭用・PC ネットゲーム・携帯電話ゲームの市場規模の推移を示したものである。家庭用ゲームはニンテンドーDS によって市場環境が 2006 年に好転したが、その後はまた伸び悩んでいる。一方、携帯電話ゲームと PC ネットゲームは着実に市場規模を拡大している。PC ネットゲームは最も成長力が高い分野にも関わらず、日本のゲーム企業の国際競争力が最も低い分野である。携帯電話は日本だけが独自アーキテクチャで「ガラパゴス化」している領域なため海外企業の参入可能性は非常に低いが、PC ネットゲームと同様に国内で「コップの中の競争」に明け暮れている間に、別アーキテクチャで進展している海外市場への進出が不可能となる可能性がある。

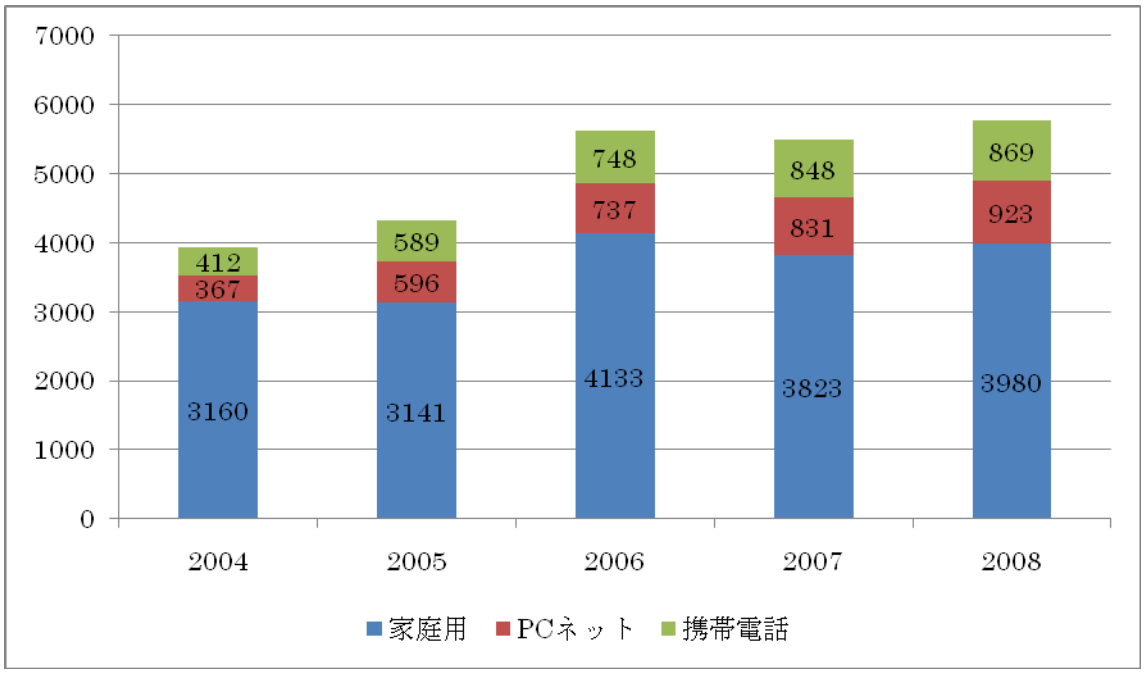


図 3.1-04　家庭用・PC ネットゲーム・携帯電話ゲームの市場規模（単位：億円）

出典：デジタルコンテンツ白書 2009

3.1.5　ゲームのためのイノベーション？

ここまでで議論した家庭用ゲーム産業の状況をまとめると、「閉塞感」の 3 文字に集約できるだろう。パッケージゲームを販売して利益を上げるビジネスモデルは、日本のゲーム会社が家庭用の据置機で行う場合、損をする可能性が高い投資でしかない。日本の家庭用ゲーム市場で据置機から携帯機への重心シフトが発生したのは、企業のリスク選好を考えると非常に自然である[15]。

日本のゲーム産業だけでなく、全世界規模で家庭用ゲーム市場を考えても、据置型ゲーム機は難しい状況にあるといわざるを得ない。ゲーム専用のコンピューターというハードウェアを採算ラインに載せるのが非常に困難になっているためである。据置機のゲーム開発期間は長期化が進んでいるため、第 5 世代が市場の中心を占める期間はこれまでより長くなることが予想されているが、それでもプレイステーション 3 と Xbox360 の開発・製造コストをソフトウェア売上のロイヤリティ収入で回収できる見込みがたたない。現状では、両社が第 6 世代を発売する合理的理由が見いだせない。

これまでゲーム産業の技術進化の方向は、描画性能の向上とゲームメディアの容量拡大

[15] 日本の重心シフトに関しては、少子高齢化も影響している。少子高齢化でユーザーの平均年齢が上昇して TV の前に座ってプレイできる時間が減少した影響で、ハード、ソフトともに携帯型ゲーム機（DS、PSP）の売上が据置機（Wii、プレイステーション 3、Xbox360）の売上を上回る状態となっている（小山 09）。海外ではまだ据置機の売上が多いが、少子高齢化は世界的な傾向であるため、将来的には日本と同じ傾向となる可能性がある。

に向いており、ハード・ソフトともに開発コストは上昇する方向にあった。ソニーとマイクロソフトはこの方向に沿った技術革新投資を行った結果、閉塞的状況に陥った。ソフトウェア開発費の高騰は日本国内向けだけでゲーム開発を行うことを困難にし、ハードウェア開発費の極端なまでの上昇は世界全体での市場規模拡大にも関わらずハードウェア事業の継続を困難にしている。

ゲーム産業でソニーとマイクロソフトが行った技術革新は、ハードウェアのアーキテクチャでリスクを取り、斬新ではあったが、消費者に与える経験としては既存のゲームの枠から出たものではなかった。むしろ、両社の事業領域への波及効果が強く意識され、ゲーム産業に過剰な負荷を与えるイノベーションだった。

ソニーがプレイステーション 3 の開発で取ったリスクの最大のものは、 Cell の開発と Blu-ray ディスクの採用だった。いずれも、プレイステーション 3 がビジネスとして成功すれば、本業の AV 事業に貢献する仕組みだった。Cell はプレイステーション 3 の量産で一気に償却を進めてデジタル家電に採用することを目指したものだったし、Blu-ray ディスクはDVDの次の映像ディスクでHD-DVDとの争いに勝利することを目指したものだった。第 1 節で 90 年代のマルチメディアブームとゲーム産業の関連について触れたが、発想はほぼ同じだったと言っていい[16]。マイクロソフトがゲーム産業に進出したときに取ったリスクは、Xbox のときは PC アーキテクチャ（Xbox の CPU はインテルだった）のゲーム機への進出であり、Xbox360 では PC とのハードウェアの互換性が低い状況で（Xbox360 は PowerPC アーキテクチャを採用している）共通で使える新しい開発プラットフォーム（XNA）を生み出すことだった。Windows アーキテクチャを携帯電話や情報家電にまで拡げるという意味で家電業界がマルチメディアブーム時に考えたことを逆向きに行っているものである。

一方、任天堂がイノベーションで取ったリスクは全く別のものであった。DS では 2 画面とタッチペン、Wii ではモーションセンサーを内蔵したリモコン型のコントローラーを採用した。インターフェースを大胆に変更することで、これまでのゲームにない面白さを生み出す可能性を目指したのである[17]。半導体の技術開発は主にゲーム機の低消費電力化とそれに伴う小型化に用いられた。他の 2 ハードが採用した画面解像度の HD 化は、HD 表示ができる TV の普及がほとんど進んでいないこともあって採用しなかった。これらの技術的挑戦は、ハードウェア性能の向上に真っ向から挑むイノベーションと比べると小手先で技術的にも劣って見えるのは確かである。だが、これまでにない新しい楽しさを生み出し、両者のハード以上に消費者から支持されたのは間違いない。

図 3.1-05 は今井（2008）にある、クリステンセンの「イノベーションのジレンマ」の

[16] マルチメディア時代の家電メーカーと当時のソニーの違いは、「リビングルームのコンピュータを独占することで莫大な量産効果を発揮し、ウィンテル連合軍に対抗する」という意図を明確に持っていたことである（西田 2008）。

[17] ゲーム＆ウォッチ向けに十字キーを開発した頃から、任天堂はインターフェイスにさまざまな挑戦をする企業である。スーパーファミコンではコントローラーの縁に L ボタンと R ボタンを配した 6 ボタンのコントローラー、ニンテンドウ 64 ではアナログスティックと振動パックが新しく開発されている。さらに言えば、ビジネス的には大失敗だったバーチャルボーイも、インターフェイスで斬新な挑戦をした結果の産物と言っていいだろう。

解説図である。縦軸が提供されている商品の性能、横軸が年単位の時間経過をあらわし、消費者の要求水準と企業が提供できる製品の性能水準がそれぞれ曲線で示されている。この図をゲーム機に当てはめてみることで、任天堂のイノベーションが成功した理由が明確となる。まず、ゲーム機の性能水準は初期の非常に低かった状態からある時期（第 3 世代）から急上昇する。一方、消費者がゲーム機に要求している性能水準は、当初はゲーム機の性能と比べると相対的に高いところにあるが、時間経過による要求水準の上昇スピードは緩やかである。任天堂がファミリーコンピュータを発売した頃は企業が提供できる性能水準は不十分であり、性能が高いものがそのまま評価された。一方、世代を経た現在のゲーム機では、全てのゲーム機が消費者の要求水準を超えており、競争の優劣は製品性能とは別の局面で決定される。

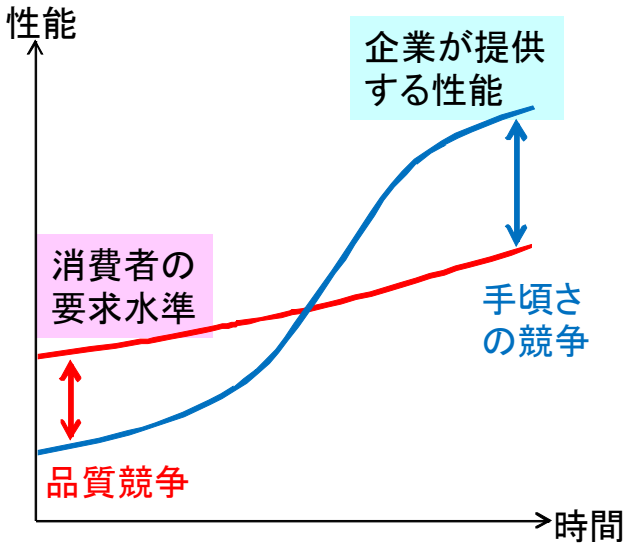


図 3.1-05　イノベーションのジレンマ

第 5 世代のゲーム機の競争では、競争の舞台となったのは価格、手軽さと目新しさにあった。そのことは、DS と Wii のユーザーに大人や高齢者が比較的多いことが示している。前節で触れた、携帯電話ゲームと PC ネットゲームについても同様の議論が可能である。携帯電話のゲームは、家庭用の携帯型ゲーム機からみてもシンプルで技術的には下位にある。PC ネットゲーム（特に MMORPG）は、ゲームプレイの「密度」としては既存のゲームとは比べものにならないぐらい希薄である。そういった、既存企業が目を向けなかったジャンルのゲームが、美麗なグラフィックで重厚長大なゲーム[18]が林立する家庭用ゲームを避けていた消費者にとって手頃（reasonable）であったことは想像に難くない。

[18] 家庭用ゲームは、「パッケージとして販売する」という大前提があるため、企業は軽いミニゲームと受け取られるゲームを販売することに躊躇してしまう。結果として膨大なボリュームなゲームが増え、マニアには支持されるがそれ以外からは敬遠されてしまう悪循環に陥ってしまいがちである。

3.1.6　まとめに代えて：選択と集中

ここまでの議論では、「家庭用ゲーム機ビジネスは袋小路に陥っており、閉塞感が漂っている」ことを指摘したに過ぎない。打開策を提唱するのは筆者の能力を超えるが、ゲーム産業の出来事を引きながら 1 点だけ指摘しておきたい。

ゲーム産業が技術的には劣った領域から挑戦を受けたのは、今回が初めてではない。そもそも、スペースインベーダー（タイトー、1978 年）の大ブームで市場として確立したアーケードに対して、ファミリーコンピュータが家庭用ゲーム機として参入したときの状況は、まさにこの状況である。アーケードゲームは（1985 年に施行された新風俗営業法の影響も重なって）市場規模を大きく落とした時期があったが、女性やファミリーが足を運びやすいような雰囲気の改善や大型筐体の体感ゲームやクレーンのプライズ機の成功もあり、最終的にファミリーコンピュータの発売前よりも市場規模は拡大した。既存企業は家庭用ゲーム機という新市場を得たことでさらなる飛躍を遂げることができただけでなく、多くの企業が新規参入することで活気づくことになった。

また、任天堂が 1989 年に発売した携帯型ゲーム機であるゲームボーイも、家庭用据置ゲーム機が第 2 世代に移行しようとしている時期にモノクロ 4 階調のグラフィクス性能で市場参入し、ヒットした。このときも、既存ゲーム企業は新市場にうまく適応することができた。

現在の袋小路的状況は、既存市場の行き詰まりと新規市場への適応が上手くいっていないことの双方から生まれたものである。既存企業にとって理想的な状況は現状に上手く適応することだが、それが上手くいかない場合にも他に方法がある（あった）はずである。その一つは、米国の IT 系企業によくある事業の買収／売却である。IT 系ベンチャーの場合、ある程度の成功を収めたあとの選択肢としては、上場して資金を集めてより激烈な市場競争に参入してゆくことと、事業価値が高いうちに Google やマイクロソフトなどの業界の巨人に事業を売却してしまうことの 2 つがある。欧米市場で出遅れたこともあり、日本企業のゲーム市場内でのプレゼンスは低下しているが、それでも成長する新興国の PC ネットゲームのベンチャー企業を買収する体力はあるはずである。もうひとつの方法は、家庭用ゲーム機に進出したあとでアーケード事業から撤退した任天堂のように、事業ドメインの集中を決断することである。Wii や DS 向きのアイデアあふれるゲームが生み出せず、他の据置機で開発する資金力もないなら、PS2 までと同様の従来型ゲームへの根強い需要がある PSP 向けに特化するのも方法の一つだろう。

参考文献：

[1] 小山友介「家庭用ゲーム産業の『ハリウッド化』」（出口弘・田中秀幸・小山友介（編），『コンテンツ産業論』,東京大学出版会,2009 年）

[2] 小山友介「日本ゲーム産業の共進化構造―イノベーションリーダーの交代―」,『ゲーム学会誌』,第 1 巻第 1 号,63-68,2006 年

[3] 今井賢一『創造的破壊とは何か　日本産業の再挑戦』,東洋経済新報社,2008 年

3.2　ゲームと価格のイノベーション

新　清士（IGDA 日本）

3.2.1　1983 年にファミリーコンピュータが作ったモデル

今の家庭用ゲーム産業が直面している最大の課題は、これまで前提として考えられてきた収益の基礎モデルが崩れようとしている点にある。

1983 年に発売された任天堂の「ファミリーコンピュータ」は、現在に至るまでの家庭用ゲーム機ビジネスの基本的なひな形を作った。ハードウェアとソフトウェアの分離というモデルである。ファミコンというゲーム機の中身を触ることはできず、また単体では何の機能も果たすことができない。そこにソフトウェアを収納した ROM カートリッジを差し込むことで、実際に機能を追加することができるという仕組みだ。そして、ハードウェア企業は、ハードウェアは、できるだけ原価に近い価格か赤字で販売をして、ソフトウェアの収入によって回収する。

このシンプルさによって、対抗機種であるマイコンとしてプログラムなどユーザーがデータを改変できる余地を提供していた他のゲーム機と比較して、高い性能を確保しながら、価格を抑えることができた。それが「ファミリーコンピュータ」の爆発的なブームを引き起こす一因になっていった。その後、ゲームは音楽 CD や書籍などと同様にパッケージとして小売店に流通させるというのが基本であり、収益を生み出す基本構造が確立された。

そして、1994 年のソニー・コンピュータエンタテインメント「プレイステーション」で、メディアが CD-ROM に、2000 年の「プレイステーション 2」で DVD、2006 年の「プレイステーション 3」で Blu-ray Disc に変わっても、この構造には基本的な変化は起こっていない。

ビジネス的には、これによりハードウェアを持つプラットホームホルダーと、ゲームを開発し発売する、ゲームパブリッシャー、ゲームディベロッパーという役割が明確に区分けされ、ゲームソフトウェアを開発する企業は、ハードウェアの技術成長に併せて、自社の投資をスケーラブルに行っていけるという好循環のサイクルが形成された。もちろん、開発費はだんだんと引きあがっていったが、この成長サイクルの中で、3D グラフィックスが登場し、その技術が高度化していく現象が起きたということはできる。その高度化する技術にこそ、ゲームの付加価値性が存在していたとも言える。

3.2.2　ファミコン時代から 1250 万倍にまで進んだ技術革新

一方で、この 27 年に渡るハードウェアの技術的成長を物語るには、様々な指標から指摘できるが、ソフトウェアを提供しているメディアのデータ量の増加を通じて、劇的な変

化を見てみよう。85 年の『スーパーマリオブラザーズ』（任天堂、ファミリーコンピュータ）を開発するために必要だったデータサイズは 40KB ほどしかなかった。しかし、DVD では、8.54GB を収録することができ、現在使用できる Blu-ray Disc では、最大 50GB ものデータを収録することができる。1GB は 10 億バイトに相当するため、DVD ではスーパーマリオの 21 万 3500 倍に相当するデータを格納でき、Blu-ray Disc では、さらに多く 1250 万倍ものデータを格納することができる。

ところが、メディアのイノベーションの加速化は、必ずしも、収益構造と一致することがなくなってきている。PS2 と PS3 と比較しても、コンテンツの容量は増大し、表現能力は格段に上昇しているが、当然、販売本数が 5 倍以上に跳ね上がるということは起きていない。また、どのゲームも、元々の「スーパーマリオ」が 40KB で作られていながら、現在遊んでも十分におもしろさを感じられるように、常に 50GB もの膨大なデータ領域を必要とするわけではない。

つまり、家庭用ゲーム機市場のハードとソフトウェアを分離し、ソフトウェアをパッケージ化して高度な技術を使ったゲームを展開することで、付加価値を形成するという基本モデルは、オーバースペック化したことで、バランスが悪くなってきたのである。

3.2.3　過剰なイノベーションが準備したソーシャルゲームブーム

また、過剰なイノベーションは、データを大量に必要としない低い技術で実現できる領域のゲームによって、市場形成を行える土壌も生み出した。昨年当たりから日本でも大きな存在感を持ち始めた「ソーシャルゲーム」がその典型例だ。

その背景にあるのは、テレビを必要としないコンピューティング環境のユビキタス化と高性能化、また、ブロードバンド回線の一般化だ。さらに、コンピューティング環境の発達は、クラウド型のインターネットに接続することを前提としたコンテンツプロバイダーの登場も生み出すようになった。

どの技術も、広義の意味で「ムーアの法則」の影響を受けている。ムーアの法則は、集積回路におけるトランジスタの集積密度は、18～24 か月ごとに倍になるという経験則のことを指すが、類似の法則性として、通信に関係する「ギルダーの法則」通信網の帯域幅は 6 ヶ月で 2 倍になるといったものなど、ユーザーの所有しているコンピューティング技術の環境が、指数関数的に増大しているという法則は複数存在している。

この状況は、実際にインターネットに接続をしていることを前提とした、軽いゲームアプリケーションのヒットタイトルが登場するようになった。ソーシャルネットワークサービス（SNS）の米 Facebook は、2004 年にスタートして以来、5 年あまりで全世界 3 億アカウントのユーザー数を集めることに成功している。

その Facebook 上でアプリケーションを展開したい企業に対して、API の公開を 2007 年 5 月に開始した。それにより、「ソーシャルゲーム」と呼ばれるジャンルが登場し急激

な人気を獲得する企業が登場するようになった。例えば、トップ企業の Zynga は提供している 41 アプリに登録している合計ユーザー数は、2 億 4000 万人に達している（3 月 1 日現在)。昨年 11 月には、第 2 位だった 5000 万人以上のユーザーを抱えている英 Playfish を米エレクトロニック・アーツが 3 億ドルで買収し、世界のゲーム業界に衝撃を与えた。

これらのゲームは、Flash や Java といった技術を利用することによって、専用のソフトウェアをダウンロードすることなく、ウェブブラウザ上だけでゲームを展開する仕組みになっている。また、SNS に展開されている性質上、自分の友人を同じアプリに巻き込むことができると有利になるというゲームシステムになっているために、口コミ効果が起こりやすい。

また、基本料金が無料で遊べるというところにも特徴があり、ユーザーは有利にゲーム展開を進めるために、有料で販売されているバーチャルなアイテムやグッズ、ポイントを購入する形で収益を上げるようになっている。実際にユーザーがゲームに課金する率は、10%以下であり、大量の無料ユーザーを、一部の有料のユーザーがお金を支払って支えるという収益構造になっている。

3.2.4　アイテム課金モデルのルーツは韓国から

これらのノウハウは、まず韓国から登場した。韓国では 1990 年代の最後の時期に、ネットワークカフェが流行したことによって､オンラインゲーム産業が急速に立ち上がった。初期に参入した企業は、ゲーム単位の月額課金を行うことで高収益を獲得することができた。パッケージゲームと違い、一度お金を支払ったら終わりではなく、そのゲームのユーザーになった場合には継続的にユーザーがお金を支払ってくれるという点で、家庭用ゲーム機と収益構造が大きく違っている。また、ゲーム自体も､「作品」ではなく､「サービス」になったと言われるようになった。1 タイトル作ったら終わりなのではなく、サービスを開始してから、継続的にユーザーの満足度を得られるようにサポートを行い続けなければならない。そのため、そもそもの開発コストの構造が違っている。

さらに、新しいビジネスモデルが出てきたのが、同じく韓国から出てきた「アイテム課金型」のモデルで､2000 年代前半に登場している。これは様々なオンラインゲームが登場し、過当競争になる中で、後発に参入する企業が、急激にユーザーを獲得するための手段として登場した。無料でゲーム自体は提供し、継続的に遊び続けるユーザーには課金を行うように促す仕組みになっていた。ソーシャルゲームと同じく実際にお金を払うユーザーの数は 10%以下と考えられている。実際のところ、ユーザーが購入するものは、ゲーム会社のサーバー内にある無視できる金額で無限に再生産できるデータに過ぎないために、お金を支払うユーザーはいないのではないかと考えられた。しかし、そうは考えることはなく、ユーザー側にはお金を通じてでしか手に入れる手段がないという非対称性から、価値を見いだすユーザーが現れるようになりアイテムに対して、お金を支払うユーザーが登場

するようになった。そして、ユーザーは無料で遊べるということから、多数のユーザーを獲得し、課金にも成功するゲームが次々に登場するようになった。

3.2.5　家庭用ゲーム機のコストでは開発できないソーシャルゲーム

そして、このモデルが、欧米に広がる形で、SNS と結びつく形で、ソーシャルゲームの誕生を促した。

韓国型のオンラインゲームではクライアントソフトをダウンロードする形のスタイルを採っており、コンピュータスペックの上昇に併せてグラフィックを豪華にする競争が繰り広げられたために、開発コストは上昇するようになっていった。現在では、一本当たり、数億円はかかるという日本のコンシューマゲーム機と変わらないような高コスト体質に悩むケースも増加している。

一方で、ソーシャルゲームの一本当たりの開発費の相場は、現在でも数百万から 1 千万円程度、開発期間も 3 ヶ月から半年と、数億から数十億円の開発費に、2～3 年の開発期間を求める家庭用ゲーム機とは大きく異なっている。現在のソーシャルゲームの収益構造に、既存の家庭用ゲーム機向けゲームのコスト構造を乗せることは不可能といってよい。家庭用ゲーム機のゲーム開発コストが高すぎるからだ。

現段階では、ゲーム自体も、数 MB 程度というデータサイズの内容でもある。そのため、できるだけ早期にサービスの骨格を固めてしまい、サービスに入ってしまう。そこから、段々とコンテンツを充実させて、ユーザーを巻き込みながら収益を継続的に上げ続けるという構造になっている。

3.2.6　日本でも登場してきたソーシャルゲーム

日本にも、昨年から今年にかけて登場してきている。特にグリーが成功している「釣り★スタ」などに代表される日本の携帯電話向けの SNS サービスに向けたソーシャルゲームだ。昨年 8 月にスタートした mixi の「mixi アプリ」や DeNA の「モバゲータウン」なども、次々にソーシャルゲームの展開を進めている。

これらのものは、Flash の技術を使った簡単なシステムで、常にユーザーの携帯電話は、サーバーにアクセスしてデータを取得しながら遊ぶ。ゲームのテンポもゆっくりしており、音楽といったものもなく、高度に完成した家庭用ゲーム機用ゲームに比べると、あまりにも大したことのない素朴なゲームのように見える。しかし、ゲーム機がテレビの前に座らなければならないのに対して、携帯電話の普及率は高く、一般的なユーザーは無料で遊べるソーシャルゲームに大挙して流れるという現象が起きている。例えば、mixi アプリで最も人気を集めているサンシャイン牧場はユーザー数は 460 万人を超えている。

やはり、国外のソーシャルゲームと同じく、実際に課金をユーザー全体でならした場合

には 4.4 円程度（2009 年 7-9 月期、mixi、DeNA 決算）という低い金額になる。しかし、10%程度であれ、一度課金を行い始めたユーザーの中には、高いコストを支払うユーザーも登場しており、結果的には一度ユーザーを抱えると高収益構造を生み出すという好循環を作り出している。

ポイントは、インターネット回線を通じて、ユーザーに直接データを受け渡すことができ、決済などの仕組みも簡便であるために、パッケージ流通と比較して、中間の流通コストが無視できるほどに小さくすることができる点だ。また、サーバー上のデータ自体の再生産コストも、ほとんど無視できる。また、ユーザーの側にとっては、同じように課金してゲームを遊んでいるとしても、パッケージソフトを 5〜6 千円かけて、リスクを負って 1 本購入するよりもかかったコストが月数千円程度と相対的に安い。そして、SNS の特徴を活かした、ソーシャル性がゲームの付加価値を上昇させており、十分に楽しめる内容になっている。

3.2.7　前払いモデルから後払いモデルへとユーザーが移行

家庭用ゲーム機は、1983 年に登場してきた構造から、ずっと一直線にハードとソフトを分離したパッケージモデルとして技術的な上昇を続けてきた。しかし、そこにはるかに安価な価格構造を持ち、ユーザーに近い存在として登場したソーシャルゲームにより、市場を現実的に奪われようとしている。ソーシャルゲームは、ゲームとユーザーの間の「価格と価値」の関係に、大きなイノベーションを引き起こそうとしている。

パッケージゲームは、購入することで、ゲームを楽しむ権利を前払いしていたと考えることができるが、ソーシャルゲームはまずゲームは無料で試して、価値があると判断されるとお金を払い出す、後払いの構造を持っている。

また、ソーシャルゲームはブラウザが動作すれば遊べるものも多いために、機種を選ばないゲームも少なくない。パソコンや携帯電話、スマートフォンでなくとも、ブラウザを搭載した家庭用ゲーム機や携帯ゲーム機でも動作する場合が少なくない。つまり、今後、ソーシャルゲームの価格構造を持ったゲームは増加すると考えるべきであろう。そのため、ユーザーの消費行動の変化を考えなければならない。ユーザーがゲームの中に感じている価値自体に大きな変化が起きつつあるのだ。

3.2.8　現在の変化の予兆は 5 年あまり前から存在していた

ただし、これらのことは、突然起きたことのように思えるが、実際にはそういうわけではない。すでに、前述したが、アイテム課金に基づく価格構造は、韓国のオンラインゲームが先行して、世界中の市場で大きく成長している。その間には 5 年あまりのタイムラグがある。

日本の家庭用ゲーム機にとって市場は、日米欧と意識されるが、オンラインゲームの場合はパソコンとインターネット回線があれば市場が形成できるために、日本企業がほとんど進出できていない、中国、東南アジア、東欧・ロシア、南米といった地域にも進出している。ネクソンの「メイプルストーリー」はその成功の代表格であり、北米でも一定の市場形成をしている。

また、日本国内においてもゲーム機離れの兆しは存在していた。2004 年の Benesse 教育研究開発センター『第 1 回子ども生活実態基本調査報告書』では、中学生女子の 55%、高校生女子の 80.6%が「ゲームをほとんどしない」と答えており、明確に家庭用ゲーム機で遊ぶ習慣がなくなっていることを伺うことができる。この頃は、携帯電話の爆発的な普及が進み始めていた時期であり、パッケージを購入するという習慣が消滅し始めていた。女性ユーザーが購入しなければ日本での大ヒットは難しいと言われるが、すでにそうなる兆しが出ていた。

それらの条件は、ソーシャルゲームが爆発的な人気を得られるための基礎的な条件を作り出していたと考えることができる。ユーザーは、ソーシャルゲームの登場によって、家庭用ゲーム機とは違うコスト構造を持ったゲームにより用意に乗り換えられるようになった。

3.2.9　ゲームは無形の財としての性質が強くなる

今から、どのような将来が起きるのかということを予見することは容易ではない。しかし、価格と価値の構造が変化したということは、様々な方法を提案できる可能性も広がったという側面も指摘できる。

ゲームがパッケージから解放されるということは、ゲームが「保険」のような金融商品的な無形の財としての性質を強く持つようになったと言い換えることができる。保険は、基本的な構造は似ているが、様々なオプションを組み合わせることで、どのような価格と価値の構造を持たせるのかは、自由な幅がある。つまり、現在出ているソーシャルゲームが提案している価格モデルがすべてではない。まだ、潜在的に隠されているビジネスモデルは存在すると考える方が自然だ。

むしろ、今後のゲーム開発では、ゲームの価格の構造の中に、価格構造を組み入れた上で技術を選択し、戦略を作ることが当然という市場環境に変わったと言える。ユーザーにとっての「価格と価値」の意味は大きく変わったのである。これは、これまで以上に、ゲームの企画段階から、ゲームはサービスとしての性質を持つようになり、その性質は今後のゲーム市場でより明確に姿を現すに違いない。

3.3　ゲーム開発者教育

三上　浩司（東京工科大学）

3.3.1　概要

ゲーム開発者教育は 1990 年代に専門学校や各種学校を中心に広まった。それまでの情報処理やコンピューター教育を土台に、ゲーム開発カリキュラムを充実させていった。現在では、ゲーム開発に関連する部門を多くの専門学校が設置しており、映像制作やゲーム開発に特化した専門学校も多数存在する。

一方、大学教育では、学部レベルでゲーム教育を実施する大学は存在しない。現在は情報系やメディア系、芸術系の学部組織にゲーム関連の学科の設置事例が増加している。大学教育では、様々な学部の中で教育される内容の中で、ひとつの分野としてゲーム開発をとらえる形で教育が進んできた。理工系学部や情報系の学部を中心に発展した技術系の教育や、芸術系学部におけるインタラクティブアートやゲーム制作、人文科学系学部でのエンタテインメント産業のマーケティングなどが例にあげられる。

近年では学生の理工系離れやプログラミング離れの対策として、ゲームを掲げる事例も多くなっている。従来の理工系学部や情報系学部を改組し、ゲーム関連の教育カリキュラムを整備しようとする事例が増えている。

本項では、学問分野の形成や教育手法の構築の観点から、主に大学での高度人材育成という視点からゲーム開発について述べる。

3.3.2　大学におけるゲーム開発者教育の課題

ゲーム開発にはプログラミング教育だけではなく、シナリオ制作、各種デザイン（ゲームデザイン、キャラクターデザイン、レベルデザインなど)、グラフィック制作、音楽・音響制作など、さまざまな要素技術が必要である。さらに、チームでの開発が基本となるため、開発マネジメントも重要な教育要素である。これらを総合した教育がゲーム開発者教育には必要であり、これらは大学におけるゲーム開発教育の実施を困難にしている。次にいくつかの点について、大学でのゲーム開発者教育についての課題を述べる。

(1)　教育スタッフの課題

(a)　人材の確保

ゲーム制作に必要な要素技術を専門的に教育する人材を集めることは必須である。そのためには制作現場の経験のある人材は適任である。一方、大学の教育スタッフとして学部や学科の運営にあたる教員の採用に際しては、研究実績や学位なども評価される。この傾向は新たに学部を設置するケースや学部を設置するケースでより顕著である。しかし、日

本ではゲーム開発はこれまで産業での実践が学術分野よりも先導している。さらに、近年増加傾向にあるものの、企業での研究開発事例や制作事例を国際会議等で発表する事例も欧米と比較して少ない。そのため、ゲーム開発を専門としながら学術分野でも実績が評価される人材が少なく、雇用したくても雇用できないケースもある。

(b) 教育体制の整備

一方で、仮に上記のような人材を確保することができた場合でも、十分な教育体制を整えるには課題が多い。大学であっても健全な経営が求められており、無尽蔵に人的リソースを教育に注ぎ込むことはできない。例えば、私立大学では教員一人当たりが担当する学生数は 1 学年当たり 10 名から 15 名程度になっている例が多い。これは、逆の見方からすれば、100 名から 150 名程度の学部を想定した場合、専任教員は 10 名程度しか採用できないということを意味する。これらの教員で、語学教育や基礎教育、人文科学など専門分野以外の教育をカバーしなければならない。受け入れる学生の人数を増やせば教員は多く確保できるが、教育環境への投資が増大する。また、最終的な進路を考慮すると、学問分野にはそれに合った適正な定員が想定されるため、単に学生数を増やすことは根本的な解決にならない。

非常勤の教育スタッフを雇用することはひとつの解決策である。しかし、継続的な指導や体系的なカリキュラム構築などは困難である。また、制作現場のスタッフは抱えているプロジェクトの状況により出講が流動的に可能性もある。そのため、制作ラインや制作チーム以外に開発スタッフを抱えることができるような企業から確実に出講できる教育スタッフを確保しなければならない。

(2) カリキュラムの課題

日本の大学のカリキュラムは 4 年間完成するカリキュラムとなっている。学部によって多少異なるものの、多くの大学では、1、2 年次には基礎的な教育、3 年次以降は専門的な教育となる。4 年次は卒業研究や卒業制作などが中心となるため、通常の講義や演習はほとんどない。各学年は前期と後期に分かれており、それぞれ 15 週の授業実施が文部科学省で定められている。それぞれの学期ごとにカリキュラムが分かれており、学期間には約 2 か月の休暇がある。

ゲーム開発には様々な要素教育が必要である。半期ごとにさまざまな要素技術を学び、最終的にこれを統合する実践的な教育を 4 年という限られた期間で行うのは困難である。実際はプログラムやグラフィックス、サウンドなど、多様な分野に分かれ、1 年時から 3 年次までにスキルを高め、それを 4 年次に統合する方法論が最も現実的である。しかし、実際は 4 年次には卒業研究や就職活動などがあり、それほど講義や演習などのカリキュラ

ムを用意するのは困難である。そのため、多くの場合では各要素技術の習得がバラバラになり、ゲームを統合する部分に関しての教育の機会を設けるのは難しい。ゲーム開発者教育のためには、多くの試行錯誤やコミュニケーションが必須であり、そのためには多くの試行錯誤やプロジェクトの経験は重要である。当初は少ない技術でプロトタイプを作り、年次を重ねるごとに技術レベルを上げるような複数年にわたるスパイラルな実習環境を構築することが課題である。

(3) 学生募集の課題

近年では、理工学系の学部の志願者が減少傾向にあることから、情報系の学部にゲーム開発の要素を加える事例が増加している。ゲーム開発というキーワードはは高校生に対して訴求力があり、実際にいくつかの大学で受験者数が増加している。しかし、こうして志願してくる学生の多くは、ゲームをプレイすることに興味はあっても、ゲームの開発経験やプログラム開発経験をもとに志願しているわけではない。現在は、高等学校教育でコンピューター教育は増加しているものの、ゲームなどのコンテンツ制作を実施している例は極端に少ない。そのため、学生はユーザーとしての興味が主体でゲーム開発を死亡して大学を選ぶ傾向がある。そのため、きわめて初歩的な部分からクリエイティブ教育からスタートする必要がある。さらに、専門的なスキルを高めるまでに、当初の期待と現実の差を理解して挫折するケースもある。ゲームに限らずコンテンツ制作にかかわる高度な人材育成のためには、大学や専門学校のみではなく、中学校や高校などと連携した人材育成を考慮する必要がある。

(4) 就職の課題

欧米の大学などでは、卒業後に就職活動をするケースもある一方で、日本の大学では、4 年間の在学中に就職先を確定し、卒業後すぐに就職することが一般的である。そのために、3 年次に就職活動を開始し、4 年次の前期には大半の学生が内定を得る。そのため、3 年次の後半から 4 年次の前半は、教育カリキュラムの構築が困難になる。近年の就職難では、4 年次後期まで就職活動が伸びるケースがあり、特に問題になっている。

ゲーム産業を目指す若者が大勢いる一方で、他の学問分野がターゲットとする産業と比較するとゲーム産業そのものに現時点では大きな雇用がない。また、特に企画部門（プランナ、ゲームデザイナなど）や管理部門（マネージャ、アシスタントなど）では大学でのゲーム教育と企業の募集する人材に、現時点では関係性が少ない。そのため、事実上すべての大学の学生がターゲットとなるため、より競争が激化する傾向にある。

3.3.3　教育の実例

本節では、実際の教育の事例として東京工科大学[1]の事例を紹介する。東京工科大学はメディア学部において、産学連携のカリキュラムを提案し、平成 16 年度文部科学省大学改革推進事業「現代的ニーズ取り組み支援プログラム」に「インタラクティブゲーム制作の実践教育」として採択された[2]。

(1)　実施概要

東京工科大学では、メディア系の学部の草分けでもある「メディア学部」(1999 年設置)のカリキュラムを基盤として、ゲーム産業で求められる国際的かつ総合的なゲーム開発力の教育を目的としたカリキュラムを構築した。

従来、ゲーム制作に必要な技術は多様な分野に散在しており、同一の教育カリキュラムを持つ学部の中でこれらが共存している例は少ない。東京工科大学にはリアルタイム CG 技術やアニメーション技術、音響・音楽の専門家が在籍しており、多様な分野の教育の実施が可能であった。

一方、産業界で必要とされる人材も多岐に渡っており、外部から多くの実務経験者の協力を得なければ、一貫した教育の実施は不可能である。これらの産業界の人材をゲーム開発会社の経営者として、国際的なゲーム開発経験を持つ山路氏がコーディネートした。これにより産学の連携によるゲーム制作の基盤技術を統合的に教育するカリキュラムを構築した。

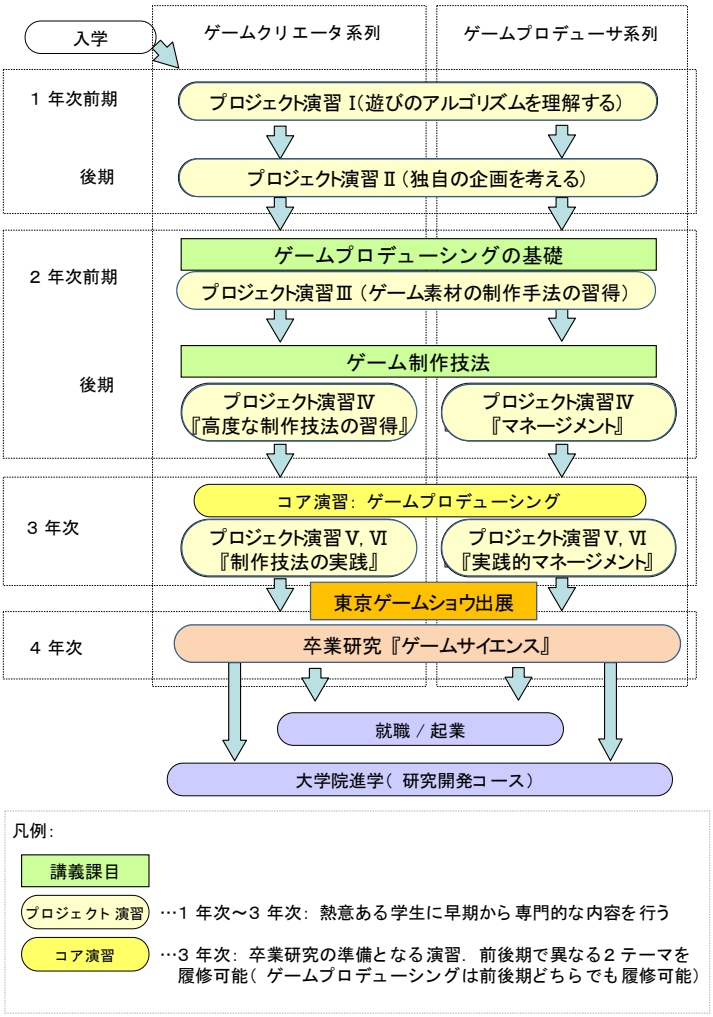


図 3.3-01　東京工科大学メディア学部のゲームカリキュラム体系

(2)　人材育成の工夫

ゲーム開発人材を育成にあたってのカリキュラム内での工夫について、3.3.2(2)のカリキュラム概要の中から「プロジェクト演習」と「卒業研究」について紹介する。

(a)　プロジェクト演習

プロジェクト演習は、先に述べた 3.3.2(2)ならびに(3)を軽減するために考案した仕組みである。次にいくつか特徴的な点について述べる。

① 早期教育・一貫教育

1 年次からゲーム開発を体験するプロジェクトを経験させることで、早期に遊ぶ側から作る側への意識の変革を促す。また、3 年間の継続の演習であるため、各学期ごとにゴー

ルは設けるものの、3 年間を通じての目標を明確に提示することができる。特に 3 年次では、毎年 9 月から 10 月にかけて開催される、「東京ゲームショウ」を成果発表の場として位置づけた。1 年次から企画を練り上げ、必要なスキルを身につけながら、2 年先のゴールを目指す仕組みをとっている。開発に必要な要素技術を核学生が学びながら、同時に統合することを学ぶ仕組みである。長期にわたるプロジェクトであるため、モチベーションの維持のため、学生には大きな目標として東京ゲームショウでの出展を掲げた。

さらに、2010 年からは、国際的なゲーム開発イベントである「Global Game Jam」への参加もゴールとして設定した。Global Game Jam では世界中のゲーム開発者が、48 時間以内にテーマに沿ったゲームを開発する。短期間にクオリティを維持して題目に沿った開発を行うことは制作経験者にとってもハードルは高い。学生の習得したスキルが産業界でどの程度通用するのか、自らが理解することができるため、今後の課題なども明確になる。

② チームの企画に合わせた指導と既存の授業との連動

本演習では基本的に学生が独自の企画のもとゲームを開発する演習である。それぞれのチームが企画する内容によって、習得すべきスキルは異なる。そこで演習では企画に合わせて必要なプログラムやグラフィックス、サウンドなどの技術について、コンサルティングをした。最終的に学生が必要な技術を定め、当該技術を身につくことができる学部の講義や演習によって習得する仕組みをとった。大学の講義や演習として用意されていない技術は、原則として学生が独自で学ぶものとし、教員はそのサポートをした。そのため、各チームが独自に様々なスキルを身につけていった。

③ 上級生による指導を活用

本演習では 1 年生から 3 年生まで同時に企画の進捗状況のプレゼンテーションと講評を毎週実施している。プロジェクトの進捗状況が異なるが、下級生にとっては上級生の経験を自分たちの企画に生かすことができる。当初から、上級生が下級生の師範となる仕組みを想定したため、この方式をとった。1 年生から 3 年生のほかに、すでに卒業研究に着手している 4 年生や大学院の学生もスチューデントアシスタントとして参加している。経験に基づいた、企画やチーム運営に対するアドバイスなどを与えている。その結果、200 名の受講者に対して、2 名の専任教員と 3 名の非常勤教員で担当ができている。

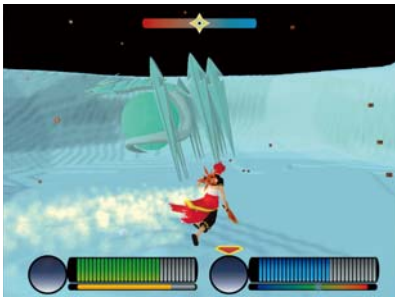


図 3.3-02　学生のゲーム作品の例

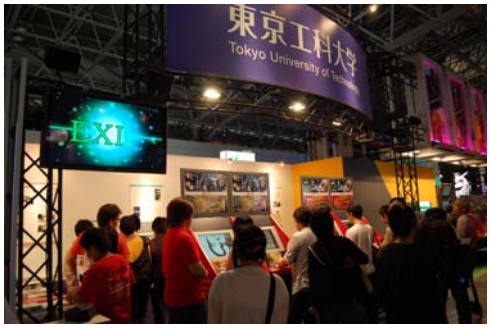


図 3.3-03　東京ゲームショウでの出展の様子

図 3.3-04　Global Game Jam の様子

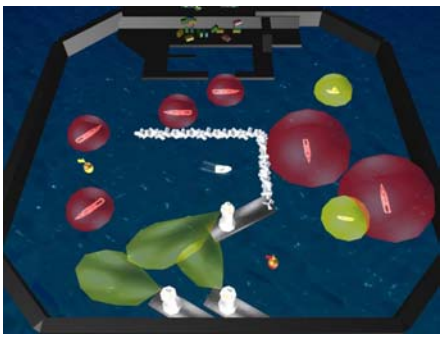


図 3.3-05　Global Game Jam で学生が制作したゲーム

(b)　卒業研究

4 年次の卒業研究「ゲームサイエンスプロジェクト」の定員は 20 名である。例年定員を大きく上回り、選抜面談を実施している。選抜の結果、配属されている学生は大きく分けて 2 タイプ存在する。ひとつは、上記の「プロジェクト演習」に参加し、ゲーム開発の体験を持っている学生である。ゲーム開発に関連した問題発見や解決をするための基盤が整っている学生が多い。もう一方は、技術系の素養の強い学生である。リアルタイム 3DCG のプログラミングスキルが高く、技術的な難易度の高い研究を志向する[6]。

①　研究テーマの設定

「ゲームサイエンスプロジェクト」では、学生が自ら調査し研究テーマを提案する。そのテーマが成立するかどうかを教員が判断し、卒業研究のテーマとして認定する。CG の基礎研究やリアルタイムグラフィックにおける表現手法の提案、ユーザビリティ、立体音響、シナリオ分析などがある。これまでに執筆された卒業論文は Web で外部にも公開されている。(http://www.teu.ac.jp/aqua/GS/)

②　対外的活動

卒業研究の研究成果はプロジェクト演習で開発したゲームとともに、東京ゲームショウに出展している。東京ゲームショウでは、専門学校の出展が多く、学術研究の出展物は少ない。しかし、研究やゲーム制作のゴールを産業界の世界的なイベントにすることは、制作期間が長い研究やゲーム制作のモチベーションを維持可能にし、また研究内容に対する現場からのフィードバックを得ることができる。

図 3.3-06　東京ゲームショウでの研究展示の様子

(3)　東京工科大学の教育事例のまとめ

ゲーム分野における教育はまだ黎明期である。実際に、学生の要望も高く、学部の定員の約 20%に当たる 100 名が積極的に参加している。これらの学生は、早くから自主的に技術の習得を行っており、人材育成という面では効果のある連携が実現している。しかし、

実践的な技術開発という面ではまだまだ課題も多く、今後の継続した発展が望まれる。

3.3.4 展望

(1)　産学連携の重要性

今後、更なる効果を求めるには、大学と産業界のさらなる連携は必須である。大学の研究成果を業界に広める仕組みや、業界のニーズを研究者に例示する機会は必須である。大学での開発者教育の特徴に、困難な問題に時間をかけて取り組めるという点がある。特に大学院の修士課程や博士課程を含めると、産業界の制作プロジェクトの中では取り組むことが困難な課題に取り組むことも可能である。急速に発展している欧米の技術力や開発力に対抗し、国際的な競争力を維持するために、産学連携は避けることができないと考えられる。

(2)　アカデミックな研究と現場的な研究

学部での研究の期間は就職活動など学生活動を考慮すると、実質は 1 年以下である。この期間で研究能力を高め、意義ある研究成果を出すことは容易ではない。また、実践的な開発現場で必要な技術のなかには、学術的な意義が少なく学位論文として成立しにくいものもある。大学のアカデミックな研究と制作現場の実践的な研究との差異を理解する必要がある。それを考慮したうえで、産学連携に適した研究テーマを設定することは、今後の産学連携の拡大に向けて重要であるといえる。

(3)　ミドルウェアの活用とその課題

グループワークで役割を分担したとしても、学生がゲーム開発に必要な技術を習得してゲーム制作を実践するには時間がかかる。そのため、コンテンツ開発の実際に体験し、試行錯誤によって成長する機会を得るのが難しい。そのため、開発を支援し、かつ自由度を妨げない開発ツールやミドルウェアが必要である。現在は多くのゲームエンジンがフリーになっているため、これらを有効に活用することが可能になっている。

一方で既存のミドルウェアを扱うことに主眼を置いてゲームを開発することが、大学における教育として産業界の発展に寄与するとは考えにくい。すでに多くの専門学校は主要なプラットフォームの開発環境を導入し、開発現場のツール教育を充実させていることから、大学が目指すゲーム教育は明確に専門学校との違いを考える必要がある。

参考文献：

[1] 東京工科大学， http://www.teu.ac.jp/index.html

[2] 東京工科大学，現代 GP，http://www.teu.ac.jp/gendai/006965.html

[3] クリエイティブ・ラボ，http://www.teu.ac.jp/clab/

[4] 三上浩司，渡辺大地，山路和紀「産学連携による高等教育機関におけるゲーム人材の育成と先端研究」CEDEC2007，レギュラーセッション，2007

[5] 金子満，三上浩司，山路和紀「専門教育における産学連携」，教育改革 IT フォーラム，分科会 B-2，2007

[6] ゲームサイエンスプロジェクト，http://www.teu.ac.jp/aqua/GS/

3.4　集合的なユーザーの振舞いを解釈するための手法について（―ユーザーの偏り（文化差）を考える―）

井上　明人（国際大学 GLOCOM）

「製品」のデザインについての議論において、アフォーダンス理論をデザインに応用することを唱えたドナルド・A・ノーマン（『誰のためのデザイン』1990,新曜社）の「ユーザーを中心にしたデザインを」という議論は、デザインを論じる文脈の中では非常に有名だ。

だが、クラウス・クリッペンドルフは、『意味論的転回』(2009,エスアイビーアクセス)の中でノーマンのデザイン論を批判している。ノーマンの言う「ユーザーを中心にしたデザイン」という理念は素晴らしいものだが、「ユーザー」とは何か、ということだ。「ユーザー」とは果たして単一のものだろうか。我々が「ユーザー」として捉えている存在はいかなるものだろうか。

人類のほとんどにあてはまるような文化差がそれほど関係のないような認知的構造はある。たとえば、錯視のメカニズムなどは、全人類がほぼ共通に備えているものの一つだ。しかし、実際には、製品の「ユーザー」という存在は、決して単一なものではない。様々な偏りがあり、地域差、社会階層、メディアとの接触経路などの多様な要因によって「ユーザー」には多種多様な文化差が生じる。

本稿では、この「ユーザー」の多様性がなぜ生まれるのか。そして、コンテンツ・デベロッパーや、マーケターといった人々は、この多様性をどのように解釈し、対処してゆくべきか。そして、これに対処することの意味とは何なのか、ということについて考えてみたい。

3.4.1　ユーザー間ネットワークの自律性

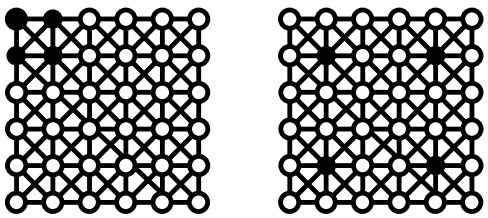


図 3.4-01

まず、ユーザー間のクラスターが維持される構造を簡単にモデル化してみよう。

いま、二つのユーザー間のネットワークを考えてみたい。一つは、影響力のあるユーザ

ーが相互に近しい位置にあるネットワーク A。もう一つは、影響力のあるユーザーが、ネットワーク全体に分散した形で存在しているネットワーク B である。簡単に図にすると、図 3.4-01 のような形になる。影響力のあるヘビーユーザーは黒丸（●）で、影響力を持たず特にゲームに興味のないライトユーザーは白丸（○）で示してある。

さて、あるゲームタイトル X がリリースされたとき、次のような 3 つの仮定を置いて、二つのネットワークの振る舞いの違いを、考えてみよう。

仮定 1 : 影響力のあるユーザー自身は新しいゲームを買う。
仮定 2 : 影響力のあるユーザーは隣接するユーザーにゲームの素晴らしさを伝え、購入の意志決定をさせることができる（いわゆる「布教」ができる）。
仮定 3 : 影響力のあるユーザーが隣接するユーザーに影響力を行使した後には、影響力を発揮するモチベーションは、通常は維持されない。しかし、影響力のあるユーザーが、隣接するユーザーも自分が素晴らしいと思ったゲームを素晴らしい、と思っていることを確認できたとき、モチベーションは維持され続け、引き続き影響力のあるユーザーであることができる。（つまり、「あれ、やっぱいいよね」と友達同士で確認できると、モチベーションが維持される）

さて、第一ステップとして、仮定 1 を適用してみると、ネットワーク A においても、ネットワーク B においても、影響力を持つユーザー4/36（11.1%）の購入は期待できる。ここでは、ネットワーク構造の差は特に現れない。

そして、第二ステップ。仮定 2「影響力のあるユーザーは隣接するユーザーにゲームの素晴らしさを伝え、購入の意志決定をさせることができる」を適用すると、ネットワーク A とネットワーク B の差は露骨にあらわれる。ネットワーク A では、影響力を持つユーザー同士が相互に隣接しあってしまっているため、影響力の行使できる対象となるライトユーザーは未購入者全体のうちの 15.6%(5/32)しか、期待ができない。一方で、ネットワーク B では、影響力のあるユーザー層がキレイに分散しているため、未購入者の全員に影響力を行使することができる。（図 3.4-02）

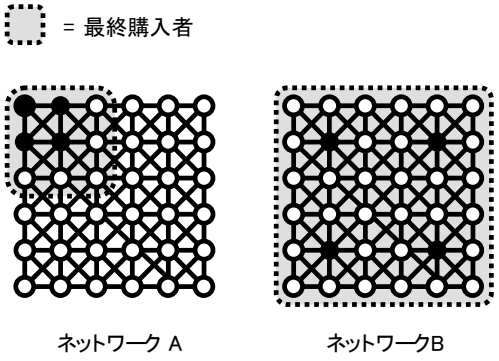


図 3.4-02

ネットワークAのような、ヘビーユーザー層が極度に凝り固まった人間関係しかもたないような場合、社会的／地理的／趣味のネットワークが重なるような特定クラスターだけに固まっているような場合というのは、ありていに言えば「オタク同士でしか会話をしない」ようなネットワークだと言える。一方で、ネットワークBはヘビーユーザーが分散しており、初期のファミコンブームや、2000年代後半の任天堂がターゲットとしたユーザーに近いだろう。「オタク層」という形で凝り固まったユーザー層を狙うのではなく、「お母さん」や「小学生」をターゲットとすることで、影響力を効果的に行使することができたのかもしれない。

また、仮に影響力の行使できる範囲の強さを隣接する2マスにまで増やしてもネットワークAでは影響力が行使できるのは12/32。3マスでも21/32となり、影響力の行使効率は低いが、ネットワークBでは影響力の行使できる範囲は1マスあれば十分である。特定のクラスターのオタクが、どんなにがんばって影響力を行使するよりも、影響力をもったユーザーが分散している方が影響力の範囲は大きな形であらわれやすい、ということだ。

さらにこのネットワークの違いが現れるのは第三ステップである。第二ステップまででは、一つのゲームタイトルがリリースされ、購入の意志決定がなされるまでを考慮したが、仮定3では一度のゲームタイトルがリリースされた後も、影響力のあるユーザーのテンションがどの高く維持されたままかどうか、ということを考える。仮定3は、つまり隣接するユーザー、すなわち「友達」などが「あのゲームは最高だよね！」と頷いてくれたときだけ、「あのゲームは神！」という形で信仰が成立し、影響力のあるユーザーであり続けることができる、というものだ。もし、自分がある作品の布教者だったとしても、周囲の人のテンションがそこまで高くないと、影響力のある「信者」ユーザーであり続けることはできない。もっとも、「信者」とまで言うと極端かもしれないが、小学生や中学生男子が、家でも学校でも友達とゲームのことを話しているような環境下でゲームをプレイし続けることは、新しいゲームの情報を得るためのネットワークとしても優れているし、攻略情報を手に入れたり、友人や家族と対戦することによって一つのゲームをより深く、長く楽しむこともできる環境でもある。その一方で、周囲に誰も話のできる相手がいないまま、ひっそりとした「孤独な隠れゲーム好き」になるのは、小中学生男子が「みんなでわいわいとゲームをやる」ような環境よりも、新しいゲームの情報も、対戦相手も全て自分で苦労して手に入れなければいけないので、かなりコストがかかり難しいことだと言える。

さて、このような場合には、仮定2では、マイナスの条件としか思われなかった影響力のあるユーザー間の隣接が、むしろプラスの効果として働く。一方で、仮定2ではプラスの効果だと思われた影響力のあるユーザーの分散がマイナスに働いてしまう。その結果、ネットワークAでは、ネットワーク構造が維持され続け影響力のあるユーザー・クラスターの位置は固定されたまま存在し続ける。一方で、ネットワークBのような極端な分散状態のネットワークでは、影響力のあるユーザーは隣接する位置にあるユーザーからの同意

が得られないため、ネットワーク上から「影響力のあるユーザー」は全て消滅してしまう。（図 3.4-03）

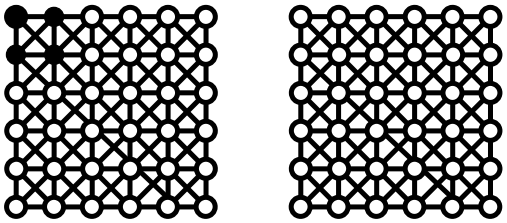


図 3.4-03

つまり、こういうことだ。ヘビーユーザーが極端にクラスター化された、ネットワーク A は、爆発的な波及効果が薄いという点ではダメな構造だが、持続的なクラスター維持のためのネットワークとしては優れている。一方で、ネットワーク B は、爆発的な波及効果を考えた場合には極めてすぐれた構造だが、持続的に影響力が残り続けるための構造としては、役に立たない。2000 年代後半の DS/Wii のブームによって、数多くのライトユーザーを獲得したものの、ライトユーザーを「ヘビーユーザー」に転化することができなければ、形成されたネットワーク構造は維持することができなくなる。

以上、この仮定は話をわかりやすくするために、やや極端な仮定を用いているが、実際には、もう少し仮定 2 と仮定 3 の条件の働き方はケースバイケースで強さが変わるだろうし、ネットワーク A とネットワーク B の二つは少し単純化しすぎている。

また、このモデル理解はネットワーク的な概念による理解でもあるが、よく知られているコンウェイの「ライフ・ゲーム」をモデルとして応用したものでもある。二次元のライフ・ゲームでは、スタート地点では単純な模様だったものが、複雑に絡み合いながら状態を維持したり、分散を繰り返したりするといった振る舞いが発生する（なお、コンウェイのモデルでは、隣接する 3 マス以上が●であれば新しく●が生まれ、周囲 2 マスか 3 マスが●であれば●は維持される）。興味を持たれた方は、「ライフゲーム」で検索すれば、ネット上で簡単にライフゲームを試してみることができるだろう。また、このライフゲームを応用したものでもっとも普及したゲームの一つが『シムシティ』(1989、マクシス)でもある。シムシティの、工場／商業地／住宅地などの関係性は、ライフゲームの応用版となっており、時間経過と供に自律的に相互関係を変化させていく。また、『シムシティ』では、工場や住宅地のように自律的に関係性を変化させていくものもあるが、その一方で「警察署」「消防署」「道路」などのように税金を投入して維持していくことによって、都市全体にポジティヴな効果を維持し続けることのできる施設もある。

このような資金を投じることで影響をコントロール可能な施設を投入することは、いわば TVCM のようなマス広告に予算をかけていくようなものだとも言える。任天堂のよう

な「広く影響力を狙う」戦略をとった場合、自律的な口コミネットワークを機能させるためには、仮定 3 のような変化を考えなければ、せっかく作り上げた影響力も、すぐに消えてしまう。そのため、任天堂は新しい影響力ネットワークが自律的に機能するようになるまで､長期にかけてマス広告を打つ戦略を採らなければならないということになるだろう。あるいは、マス広告以外にも､『ポケットモンスター』や､『ドラゴンクエスト』で採られたようなメディアミックス的な手法により、ユーザーのリアリティを囲い込む手法も有効だろう。

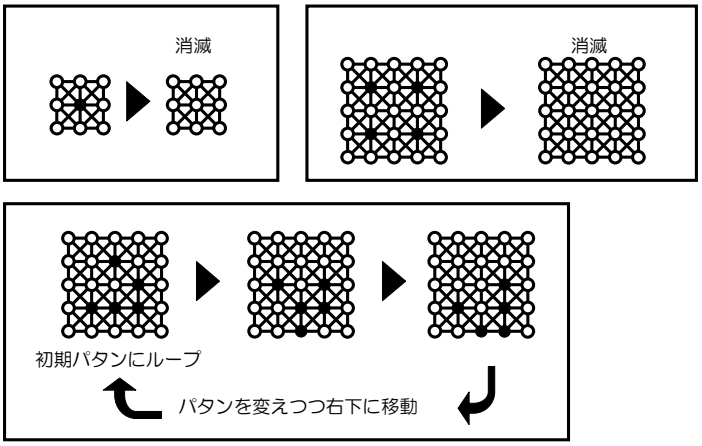
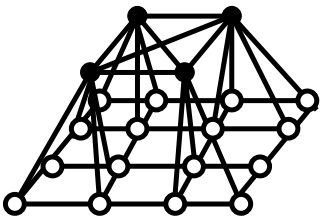


図 3.4-04

これもコンウェイのライフゲームを例にとってに説明しておくと、比較的近接する場所にいるユーザーの一定数を「●」に変化させる程度に強力な PR 活動が行われる必要がある、ということである。3×3 のマス目の中で、3 人のユーザーが強力に感化されている状態では、ユーザー間のクラスターは維持されないが、3×3 の間のユーザーの 5 人がちょうど良い位置取りで相互に影響力を確認できるような状態になれば、影響力のあるユーザーはゆっくりと移動を繰り返しながら、ユーザークラスターが維持されてゆく、ということが起こる（図 3.4-04)。

または、長期にわたってマス広告を打たずとも、ユーザー間のクラスターを長期にわたって維持する戦略としては、ヘビーユーザー同士のネットワーク上の「遠さ」がすぐに解消され、それでいてなお影響力を失わないようなネットワークを構想してみることもできるだろう。たとえば、二次元では距離が遠かったネットワーク B の影響力のあるユーザー同士の関係が、さらに第三の次元で見た場合には、実は近接してしまえるようなネットワーク C を考えてみる（図 3.4-05）ことが可能ならば、この問題は解消する。



ネットワークC

図 3.4-05

このようなネットワーク C ができあがるためには、つまり (1)一見、特に接点のなさそうなユーザー同士が、接点を見つけた途端に相互に繋がることができ、(2)かつ、似たもの同士だけでコミュニケーションを閉じすぎない、というようなネットワークである。このようなネットワークは概念的に言えば、インターネット普及以後は形成が可能になっているとも言えるのだが、インターネットはクラスター横断的なネットワークとして機能する一方で、ある側面でもクラスター間の断絶をブーストしている側面ももっている扱いの難しいネットワークである。Google などのインターネットのアーキテクチャの発展次第ではネットワークの伝播／クラスターの維持構造は変化してゆくものと考えられが、この点については本稿の扱う範囲を超えているため、言及するのみにとどめておきたい。

さて、以上のモデルは、あくまでユーザー間の自律的な行動を説明するためのごくシンプルなモデルである。実際の市場では、ユーザーごとの認知構造や、価格や製品の質などの様々な変数が大きな効果を示すため、このモデルだけで全てのユーザーの振る舞いを説明することはもとより不可能である。また、仮定 2 と比べて仮定 3 はシミュレーションのモデルとしての信頼性はそれほど高くない。様々な留保はつくものの、とりあえずこのモデルから最低限、示唆されるであろうことを簡単にまとめておきたい

① 「影響力のあるユーザー」同士が近しい関係でコミュニケーションをとりすぎており、ユーザーネットワーク全体にあまり接続されていない（中心性が低い）場合、影響力のあるユーザーの影響力はたいしたことがない。
② 「影響力のあるユーザー」が極めて社交的だったり、ネットワーク上の位置が分散されている場合、影響力のあるユーザーの行動はネットワーク全体にとって効果的に働く
③ 「影響力のあるユーザー」クラスターが長期にわたって維持されるためには、ユーザー同士が分散しているよりも、緊密なクラスターを形成していたほうが有利である（というシミュレーションを考えてみることができる）。
④ 企業側が中長期的に集中的なマス広告・メディアミックスなどを用いることで、ユーザーのクラスターが自律的に維持され続けるように仕向けることも不可能ではない（というシミュレーションを考えてみることができる）。

3.4.2　ユーザー／開発／マーケティングの相互作用

さて、次にユーザーの振る舞いと、ゲームの開発／マーケティングの三者の相互作用について考えるためのモデルを作ってみたい。

このモデルを考えるための導入として、ゲーム・マーケットではないが、近い分野でのコンテンツ産業の事例として、講談社が実施しているいわゆる「ジャンプシステム」のことを考えてみたい。「ジャンプシステム」とは、20 本程度の週刊連載漫画に対して、読者はがきを用いて人気アンケートを行い、週刊少年ジャンプのなかで連載させ続ける漫画を決定したり、不人気の連載に対しててこ入れを行うためのシステムである。ごくシンプルに、このジャンプシステムを図として書くのであれば、図 3.4-06 のような形で描くことができる。

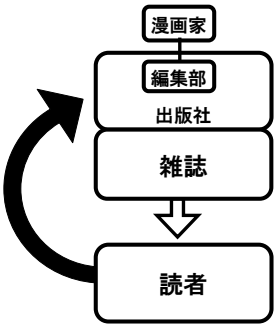


図 3.4-06

だが、この図 3.4-06 のイメージは「A.読者」と「B.漫画家―編集部」が直接コミュニケーションを取るという極めて素朴なものだが、先述したように、「読者（ユーザー)」のあり方は、メディアミックス戦略やマーケッティングなどによって、生成・変化をし続け、読者クラスターの内実も変化していく。すなわち、「ジャンプシステム」は、「A.読者（ユーザー)」、「B.漫画家―編集部」、「C.マーケティング」の三者間関係によって動的に再編成されるものだ、と捉えることができる（図 3.4-07)。

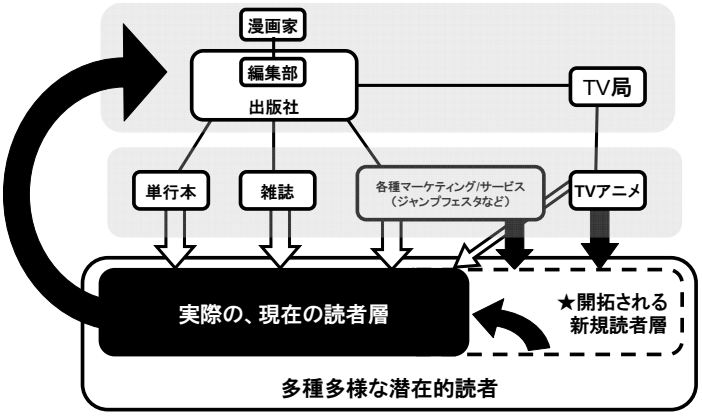


図 3.4-07

たとえば、若手漫画家の成長を描く『BAKUMAN』の中では、「ジャンプシステム」自体をネタにしつつ、編集者が「女の子からのファンレターを、作品のつくりかたにフィードバックしてはいけない」ということを漫画家に諭すシーンがある。これは、ジャンプシステム自体のサイクルの中でターゲットとされている読者層に偏りがある、ということを露骨に示した話である。週刊少年ジャンプで漫画家として描くということは、「ジャンプ読者＝少年とっての良い漫画」を描くということに他ならない。これは、実際の現在の読者層にとっての良い漫画を描くか、集英社のマーケティングのターゲット範囲となりうる読者にとって良い漫画を描く、ということだ。これは、すなわち、

① 「良い作品のあり方は、読者層の偏りに依存している」

ということだ。たとえば、ジャンプでギャグマンガとして面白かった連載が、突如「バトルもの」になることが良くあるが、これはジャンプ編集部や連載漫画家たちの意志決定である、という以上に、「バトルもの」にしたら、ジャンプでの人気投票順位が上昇してしまうようになっているからだ。現状のジャンプの読者層の偏りを考えれば、漫画家にはバトル要素を作中に挿入する強いインセンティブが与えられている。

さらに、このジャンプシステムは、

② 「良い作品のあり方を左右している読者層の偏りは、マーケッティングに依存している」

ということも意味している。たとえば、「少年週刊ジャンプ」が小学生女子をマーケティングのターゲット層として捉え、小学生女子の接触率の高いメディアへ広報を出したりしない限り、読者層の偏りは現状のマーケティングのされ方に依存することになる。（もちろん、厳密に言えばマーケティングだけでなく、該当マーケットの様々な社会的変数によ

って影響を被っているわけだが、ビジネスのプレイヤーにとってコントロール可能な部分として取り出すのならば、マーケティングだ、ということになる。）さらに言えば、

③　「マーケティング戦略は、マーケティング資源である「作品」内容に左右される」

ことも言える。いくら、小学生女子に向けたマーケティングをしたい、と考える営業マンや広報マンがいたとしても、週刊少年ジャンプに、小学生女子が喜ぶような作品が掲載されていなければ、マーケティングは成立しない。すなわち、「A.読者（ユーザー)」、「B.漫画家―編集部」、「C.マーケティング」の三者間が緊密に結びついた関係、とは、三者が相互に依存するような循環的な関係だと捉えることができる。

3.4.3　想定ターゲットユーザーの反応を、どのように推定するか

さて、上記のようにマーケティングの側が、どのようにターゲットユーザーを定めるかということの意味は大きい。これは、ユーザーの性質が決め→ユーザーの作品へのフィードバックの内容を決め、→作品のあり方を決める、という循環構造の中で理解することが重要であること示唆される。

さらに、作品を「評価」するための指標として用いられている指標をケーススタディ的にいくつか、取り上げ、ユーザーのフィードバックを理解するための方法を具体的に考えてみたい。

(1)　metascore

近年、作品評価のための手がかりとして、“meta score”と呼ばれる、ウェブサイト”metacritic”（metascorehttp://www.metacritic.com/）に掲載されたスコアが用いられることが多くなってきたようだ。

これは、欧米の主要なゲームレビュー記事や、日本語圏でのファミ通クロスレビューなどのメディアでの評価を 10 段階評価に変換して平均値を出したスコアである[19]（metacritic のサイトには、この metascore の他にも、ネットユーザーのスコアも掲載されている)。これは、ファミ通クロスレビューなどで掲載されている編集者 4 人の評価をソースとして評価情報などと比べた場合、評価者のサンプル数が数十倍以上になるため、評価としての信頼性が高い、とされている。また、単にサンプル数が多いということだけでなく、ネットのユーザーレビューにみられるような一部のユーザークラスターが熱心にネガティブな評価を書きまくる「炎上」と言われる現象とも無縁であるため、「相対的にマ

[19] <http://www.metacritic.com/about/scoring.shtml> によれば、”a METASCORE is a weighted average of reviews from national critics and publications for a given movie, book, TV show, videogame, or album. For those of you who enjoy math, read further.”

シな評価基準」として、参考にされていることが多いと言われている。

確かに、ファミ通クロスレビューなどと比べた場合、統計上のノイズの影響（サンプル数が少ないために少数のサンプルのブレに影響を大きく受けてしまうこと）は少ない。しかし、あたりまえだが、統計的なノイズが小さいということは、推計したい母集団と比較したときのバイアス（偏り）が小さい、ということとは根本的に意味が異なる。マーケティングのための事業部がないような企業がこの指標を用いることはありうるだろうが、基本的には、この metasore から推定される母集団とは「欧米圏を中心としたゲーム・ライターたち」でしかない。この「欧米圏を中心としたゲーム・ライター」という母集団を作為的にサンプルし、作り上げられた評価が metascore だ、ということになる。欧米のヘビーユーザー向けマーケットを対象としたフィードバックを得たい場合には、metascore はある程度まで参考になる数値だと言えるだろうが、異なるマーケット、異なるユーザーセグメントをターゲットにしている場合には、metascore はこれといって大きな意味のない数値である。

また、ゲーム・メディアの売上げは、主に媒体自体の売上収入と、広告収入の二本柱でできあがっている。そのため、ゲーム・ライターは、「想定読者が納得するようなレビュー」と、「広告主が怒らないようなレビュー」を書くためのインセンティブが与えられることになる。もちろん、インセンティブが与えられるだけで、必ずインセンティブに従うわけではなく、反骨精神あふれるライターは一定数存在するだろうが、ゲームライタートータルの平均値として算出した場合は、こうしたインセンティブ構造が metascore に一定程度反映されている確率が高いだろう。たとえば、ファンの多いシリーズ作品には、ある程度までポジティヴな評価を付けるようなインセンティブが存在するだろうし、広告主が大量に宣伝をしているゲームタイトルに対しては八百長の点数を付けることはないかもしれないが、広告の出てこない弱小メーカーのタイトルよりも「きちんとプレイした上でのレビュー」を心がけるようになるぐらいのことはありうる。すなわち、ファンが少なく、広告出稿量が大きくないオリジナルタイトルへの評価はそれほど信頼性が高くなくなる可能性がある。たとえば、ファミ通クロスレビューで『ポケットモンスター』がはじめて出てきたときの評価は 8,7,7,7 だった。

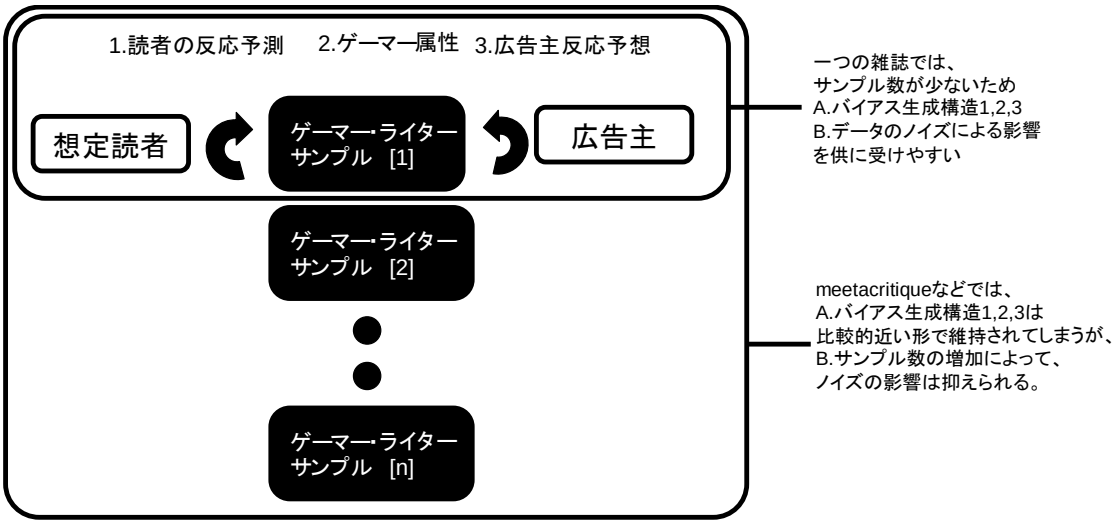


図 3.4-08

(2)　ネット上のユーザーレビュー

次に、ネット上のユーザーレビューを指標として用いる場合のことを考えてみたい。ネット上のユーザーレビューは、2009 年に『ドラゴンクエスト IX』が一般的には高い売上げを上げる一方で、ネット上での評価が「炎上」し、ネガティブな評価であふれたことによって、今現在著しく評判が悪くなっており、「ネット上のレビューはネガティブにふれやすい」ということが、業界の一部で「常識」として語られつつあるような節も見られる。

しかし、「ネット上のユーザーレビュー一般が使いものにならない」という認識がなされるのであるとすれば、それは誤解である。適当にサンプリングされたユーザー集団の平均値を見るような仕方だけで「ネット上のユーザーレビュー」は理解されるべきではない。

「ネット上のユーザーレビュー」が書かれるためには、いくつかのプロセスを前提として踏まえて考えておく必要がある。ネット上のユーザーレビューを書く 3 つのプロセスを得てサンプルとして絞られている。

まず、第一に企業のマーケティングによって、ターゲットユーザー層とされるサンプルが絞り込まれる。第二に、企業がマーケティング資源（広告費や作品のブランド・イメージなど）を動員して、実際に、購入させることに成功したユーザー層に絞り込まれる。第三に、「ネットでレビューを書く」という行動を自発的に選択するユーザー層として絞り込まれる。（図 3.4-09）

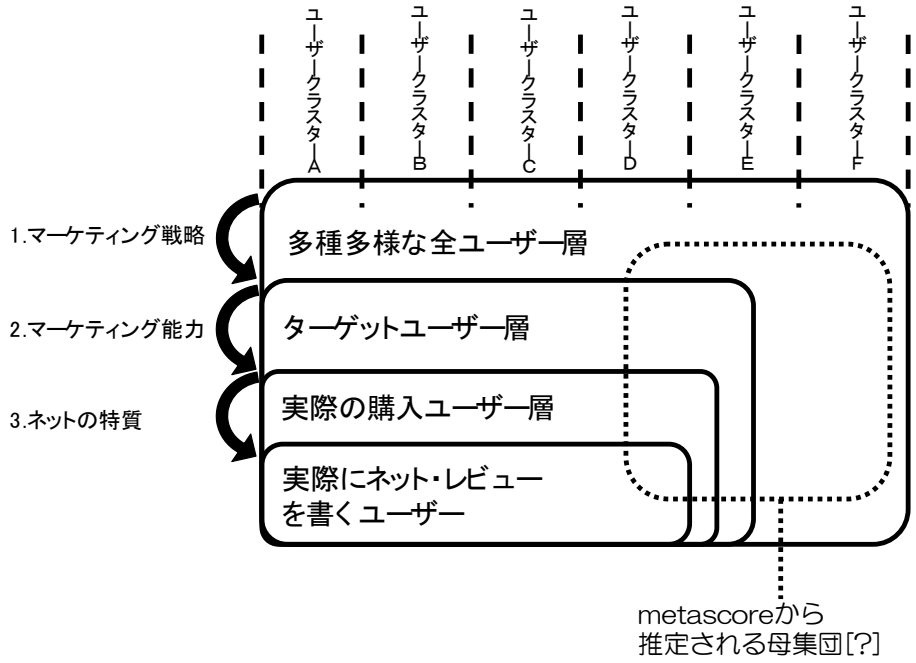


図 3.4-09

企業にとって実際に重要な母集団は、全世界の全人類ではなく、企業が設定するターゲットユーザー層全体という母集団になるだろう。このターゲットユーザー層全体と、実際の購入ユーザー層、そして実際にフィードバックを得られるサンプルユーザー層の 3 つの母集団の差が少なくなればなるほど、標本としての価値は高くなる。metascore から推定される母集団は､多くの場合､ターゲットユーザー層全体の母集団とは直接の関係はない。

サイバーカスケードなどの研究によって知られているように、ネット上の意見は特定のユーザー・クラスターが強力に意見を表明してしまうケースが少なからず存在する。そのため、Amazon などのレビューの平均値をみてしまうと、明らかにサイバーカスケードが生じていると思われる事例が出てくる。ターゲットユーザー層全体の反応を推定しようという目的にたつとき、こうした事例はあまりにもバイアスがひどいため、そのままでは参考にならないし、素直に参考にすべきではない。

しかし、より重要なことは、こうしたネット上のユーザーレビューは、metascore などよりも多種多様なユーザークラスターの評価が反映されている可能性が極めて高い、ということだ。metascore では、レビュアーの集団が、20 代～30 代を中心としたゲームに詳しい（そして、その多くが FPS 好きであろう）欧米の男性ゲームライターである可能性が高い。しかし、ネット上のレビューでは、RPG が好きな 20 代の女性や、アクションゲームが好きな 10 代のライトユーザーの男性など多種多様な層のユーザーが書き込んでいる。ネット上のユーザーレビューは、きちんとクラスタリングの処理ができれば、こうした多様なユーザー層のフィードバックを、知ることが可能になる。

ユーザーのクラスタリングをする手法は、TV であれば、F1 層や、M2 層といった年齢

と性別によるクラスタリングなどがよく知られているが、これはゲームごと、分析ごとにクラスタリングの工夫はケースバイケースで考えておいたほうが良いだろう。たとえば、『ウィニングイレブン』(コナミデジタルエンタテインメント)のようなサッカーゲームであれば、最低でも、サッカーファンの評価、ゲームファンの評価を切り分けておくことは必須だろう。サッカーファンは実際のサッカーの観戦体験やプレイ体験との比較によって評価を下す可能性が高いだろうし、ゲームファンはゲームとしての面白さ、という別の観点からの評価を下す可能性が高い。『三国無双』のようなゲームの評価の場合は、三国志ファンを切り分けて置くのはもちろんだが、ゲームファンの中でも、バーチャファイターやストリートファイターII シリーズなどの格闘ゲームのゲームプレイヤー層と、非格闘ゲームプレイヤー層とでは、大きく評価が違っている。現状では、こうしたユーザークラスタリングをするための優れたユーザー参加型レビューサイトはきちんと存在していないが、映画のレビューサイトで言えば、Cinema Scape などは高度なデータマイニングを行っておりユーザー間のクラスタリングをある程度まで行えている事例は存在する。日本のゲームのユーザーレビューサイトで言えば、mk2 がそれに比較的近い地位にあるが、mk2 のレビュー情報をクラスタリングするためには、数値を取得するだけではなく、テキストマイニングなどの様々な応用的手法を用いてクラスタリング処理を行っていくことが望ましいだろう。

(3) モデルユーザーによるテストと、購入者アンケート

最後に、企業によるモデルユーザーによるテストプレイと、購入者によるアンケートを採るという手法について述べる。これは、任天堂などのような体力のあるメーカーでは実際に両方の調査を行っている。

ネット・レビューなどでは、マーケティング戦略が対象とする想定ターゲットユーザー層の母集団の性質を推定することも、実際の購入層の母集団の性質を推定することにも、それぞれに乖離が大きく存在する。そのため、もともと推定したいと思っている母集団の性質を明らかにすることは、いくら高度な処理をしたとしても困難な点が残る。しかし、あらかじめモデルとするユーザー層を想定してテストプレイをやらせれば､｢想定するターゲット層」の反応を知ることは可能だろう。そして、直接に購入者からのアンケートを取ることができれば「実際の購入者層」の母集団の性質を推定することはかなり精度が高くなる。むろん、マスターバージョンに近い段階のゲームを十分な数のテストプレイヤーにプレイさせるためには、相当な予算がかかるし、アンケートに返送してくれるユーザーの回答率にも限界はかなり大きい。しかし、それでも「想定ユーザーの反応を知る」ための方法としてこれよりもましな方法はいまのところはないだろう。

また、言うまでもなく「想定ユーザーの反応を知り、フィードバックを活かす」体制をつくるということは、単にマーケッティング担当者の問題ではなく、開発担当者と非開発

担当部門が共同になって達成されなければならないことでもある。マーケッティング担当が日本の 10 代の中高生ライトユーザーに向けて売るための仕組みを構築している中で、開発担当者が欧米の 30 代へビーユーザー向けのゲームを開発していたとしたら、マーケットで良い評価を得るというのは困難だろう。広報・販路の拡大と一体になっている、ということはマス・マーケットに向けてのビジネスの中で製品開発をしていく上で極めて重要である。これには、ゲーム会社においてしばしば見られる、「開発部門が強すぎる」ことが一つの弊害となっていることも多いだろう。開発担当が想定しているユーザーと、広報の戦略がそりあわなければ、「想定ユーザー」などという話は、ただの中身のないそれっぽいだけの言葉になってしまう。ゲーム開発は、開発部門の抱えているテクニカルな複雑性が相当に高度なものであることは事実だが、「作ったものが評価されるプロセスとは何か」ということを考えれば、評価者＝ユーザーのあり方を左右する非開発部門との協同は、ゲーム業界における緊急の課題の一つだろう。

3.4.4 むすび

metascore のようなスコアを用いるということは、欧米のゲーマー層をターゲットにしたゲームを開発するのでもなければ、決して妥当な選択ではない。metascore の基準を数ある基準の一つの指標として参考にするのは悪いことではないだろうが、重要なことはさまざまなユーザー／母集団の「偏り」を理解しながら指標を使うことである。

metascore は「まっとうなスコア」なのではなく、偏りの表出の仕方が違うだけ、ととらえるのが妥当だろう。推定したい母集団や、サンプリングの手法によって偏りは常に発生する。たとえば、「偏りのない」調査としてよく知られているのは、選挙予測のための無作為調査だが、この調査が「偏りがない」というのは、母集団の性質を正しく推定することがすなわち、選挙結果を知ることそのものだからだ。「偏り」というのは、選挙結果と比べたときの偏りである。言うまでもないが、「選挙結果」とは公平無私な神様が選んだ結果ではなく、その選挙に投票した人々の恣意的な選好を反映したものである。「偏りのある母集団」と、サンプリングされた標本の間に差がほとんどない、ということが「偏りがない」ということだ。

マーケットに売出す、というのは販路を作りあげることであり、流通戦略・広報戦略を決めた時点で、＜公平な評価＞などということを考えることにはそもそも意味がない。マーケッティングにおいては、戦略的にターゲット層として捉えられるユーザー層の母集団の評価をいかにきちんと捉えるか、ということこそが第一に考えられるべきだろう。販促を行い、販路を構築したところで、すでに、対象とする母集団というのは一つの形をなす。作品を評価する母集団は、匿名的な無数の他者ではなく、企業自体が売る対象として選択した母集団である。

metascore を何よりも重視するような方法は、「誰に売りたいのか」「誰に評価されたい

のか」ということを、はじめから考慮することができなかったような場合に限られるだろう。企業の体力や業態によっては「誰に売りたいのか」ということを考えることは困難な場合も少なくないだろうが、十分な体力のある企業で metasore のようなもののバイアスを考慮せずに使うということは極めて危険である。それはすなわち、自らのビジネスが成功なのか、失敗なのかということを評価する指標を、自ら構築することできていない、ということを宣言することに等しい。

宮本茂は、GDC の講演で「奥さんメーター」と名付けて、自らの配偶者の反応を評価指標として用いている、ということを半ば冗談交じりに述べた。任天堂のユーザー定義・評価のサイクルは、週刊少年ジャンプのように高度にシステム化されているので、本当に奥さんメーターで全て評価しているということは、まずありえない。しかし、ユーザーを定義する、ということは自らの行為の意味を自ら構築するということとそのまま繋がっている。「奥さんメーター」は、冗談ではあっても、「自分は誰にむかって、何をしたいのか」ということを、自ら宣言するような行為でもある。ユーザーを捉え、定義するとは、開発・販売のための指標を自ら構築することであり、ある種のアイデンティティの宣言でもあるだろう。

3.5　現場から見たゲーム開発の変遷

山口　誠（株式会社グッド・フィール）

3.5.1　アセンブラの時代

アセンブラがゲーム開発の主言語だった頃の開発経験者が、現場から徐々に減っている現状をみて、非常に狭い視点だが一開発者として何か残そうと考えたのは、当時の技術資料が少ないことや、昨今の家庭用プラットフォームがどれもフル 3D の CG レンダリング可能で、C や C++が使え移植や技術転用が容易であることから、当初からゲーム開発ではそのような整備された環境が提供されていたとの誤解を防ぐためである。筆者は、学生時代に PC ゲームを数本開発した後に、就職して家庭用ゲーム開発を経験したが、現場の違いに愕然とした記憶があり、その違いについても書き残したいと考えた。

Wii と PS3、Xbox360 とではヒット商品が大きく異なる。ゲームプラットフォームというのは元来それぞれに特徴があり、注力している技術分野、採用する技術を選別する際の価値観も異なる。初期の家庭で遊ばれるゲームでは、PC、家庭用ゲーム機の間でハードウェアの違いが大きく、同じ PC(もしくは家庭用ゲーム機)であっても CPU や画像処理回路が異なり、それに応じたアセンブラでの記述が必要なため、そのプラットフォームでしか作れないタイトルというものが多く存在した。

3.5.2　プラットフォームの相違

(1)　PC ゲーム

国産 PC は、当初ビジネスや事務用途で商品化さたが、低価格化に伴いホビー向けの商品が登場した。当時、開発環境は未整備で情報も不足していたが、それでも個人で開発出来る仕組みが提供されていた点が、ゲーム専用のプラットフォームとは大きく異なっていた。PC の普及に伴って、ソフトウェアは、ユーザーが作成したプログラムコードが雑誌誌面に掲載されたものをタイピングしたり、電気店や PC 専門店などで流通していた磁気テープなどの商品で、入手ができた。開発者はマニアックなホビーユーザで、多忙の社会人より余暇が多く取れる学生などの若年層が目立った。初期のゲーム開発会社は、サロン化していた学校などのサークル、コンピューター販売店、出版社などを介して知り合ったメンバーが起業した。ソフトウェア開発は、地理的条件にあまり左右されないため、こうしたゲーム開発会社は日本各地で誕生した。

初期の PC のマニュアルには、メーカーが供給している BASIC の文法など入門者向けの解説しか無く、アセンブラでゲームを開発するための十分な情報は一般に公開されていなかった。また、別売の専門書についてもハードが販売されて随分期間を置いてから発売さ

れる場合が多かった。このため、必要な情報を得るには、基板回路を読み解き実験し調査したり、ROM 供給されていた BASIC のプログラムを逆アセンブラするなどして、I/O 関係の情報を収集したりしていた。

また、ネットなどの情報交換の場も今日ほど発展しておらず、情報誌の投稿欄などをチェックするか、各地にあったパソコンショップを中心としたコミュニティやコンピュータサークルなどから、情報を得ていた。

筆者自身は、学生時代に PC ゲームのプログラムをしていたが、こうした特別なサークルには所属していなかった。しかし、同じような活動をしていた周囲の友人の多くは、丸亀高校の数学同好会、九州大学の物理研究部、京大マイコンクラブなどの出身者、在籍者であった。

当時の PC のハードウェアは、業務用ゲーム機と比べ、画面処理能力が劣っていたため、それをプログラムテクニックでどう補うかが技術的課題であった。

ゲームセンターで遊んだゲームを家庭でも楽しみたいという要求から、シューティングゲームやアクションゲームなどリアルタイム性の高い物の移植や模倣作品が登場したが、そうしたタイトルの画像表現は家庭用ゲーム機に及ばなかった。このため、PC の特徴を活かし、データをカスタマイズできる様にした作品や CPU の高い計算力を活かしたシミュレーションゲーム、アドベンチャーゲーム、RPG といったジャンルが、PC ゲームの発展に貢献する事になった。

(2) 家庭用ゲーム

家庭用ゲーム機は、玩具として登場し、子供がカセットを差し込むだけで、簡単に遊べる ROM カセットとしてソフトが供給された。業務用メーカーが開発を牽引していため、ハードウェア仕様は業務用ゲームを移植し易くなっており、業務用ゲーム機に影響された構成となっていた。国内の業務用ゲーム開発の歴史は、家庭用ゲームより古く、遊技場や飲食店向けのピンボールやジュークボックスなどの輸入・サポート業を生業としていた企業が国産化を始めたのが起こりである。これら商品は、街角に置かれるため、簡単な説明文のみで遊び方が理解できるなど、プレーヤにコインを継続的に投入させる工夫がされていた。また開発体制は、ソフトウェアとハードウェアの双方を提供する必要があり、黎明期より、絵と音とプログラムの分業など組織的なチーム開発が実施されていた。こうしたチーム開発のノウハウが家庭用ゲームの開発にも引き継がれた、開発機器もホビーユーザが使う簡易的な構成の PC と異なり、初期の頃から業務用目的の高性能な ICE など強力なデバッグ環境がある汎用機やワークステーションが用いられていた。

しかし、ターゲットとなる家庭用ゲーム機の性能は、業務用ゲーム機と比べて安価なパーツで作られていて全体として性能が低く、計算処理も PC と比べて低い水準であり、そこをどう補うかが大きな技術課題であった。

市場に供給されたラインナップとしては、初期は業務向けタイトルの移植物が多く、その後、家庭向けに長時間遊べるアレンジやオリジナル作品が加わった。同様に、PC 系のソフト会社が市場に参入した際も移植作品などが登場したが、ユーザーがマニアックな PC 市場とは異なり、大衆向けのアレンジやオリジナル作品が登場する様になった。

3.5.3 アセンブラ時代のプログラマー雇用

ゲームの開発言語として、全プラットフォームで C 言語が標準となるのは、32 ビット CPU 以降で、8 ビット CPU では、メーカーが供給していた BASIC インタプリタ、アセンブラ、FORTH、GAME-PC や TL/1 などのゲーム開発専用言語が多用されていた。高級言語ほど記述は楽だが速度が遅いという欠点があった。16 ビット PC では MS-DOS の普及で C 言語の採用も進んだが、高速化部分はアセンブラとの併用が必要で、業務用や家庭用のゲーム機の多くは OS 未搭載で、高速化の為にアセンブラだけで開発されていた。

16 ビット以下のハードでは、多少のコードロスでも画像表示やレスポンスへの影響が大きく、コードの速度最適化が至上命題で、アセンブラが主役であった。アセンブラによるプログラミングの特徴は、命令コードをパズルの様に組み合わせてプログラムする点である。また、命令コードも CPU 毎に異なっていた。こうした事情から、大学や専門学校などの教育機関のカリキュラムだけでは、とても最適化したコードは書けず、長期の経験が必要だった。そうなると、新卒で生産力の高い者は、趣味やアルバイトなどで、自発的に長時間プログラムに従事した者に限られ、こうした人材は非常に稀少であった。

当時はプログラマーという職種の認知度が低く、PC と家庭用の何れにおいても優秀なプログラマーは不足していた。PC の場合、ユーザーにもそれなりにプログラムへの関心があったが、より一般向けの家庭用ゲームの場合、ユーザー層がプログラムに関心を持つことは少なかった。家庭用ゲームの開発は、ホビーユースの延長ではなく、組織的に行われており、開発文化の違いから、PC より深刻な状況にあった。

また、他業種のプログラム開発では、アセンブラは組み込み機器で使われる程度で利用頻度が少なく、スキルを有した他業種からの転職者を雇用することも難しかった。

筆者が新卒でプログラマーとしてゲーム会社に入社した当時は、プログラマーとして採用された殆どの者がプログラム未経験者だったため、入社 1〜2 年目で半数くらいが、職種を変えるか退職するかという状況が続いていた。

3.5.4 2D ゲームを作る技法

学生時代、PC ゲームを開発している頃、上司や先輩達に、Pascal などで発表されていたアルゴリズム、基礎となるブール代数やオートマトン、当時流行していた Lisp や Prolog による人工知能、C 言語の構造化の概念、Smalltalk やオブジェクト指向など、コンピュー

タ・サイエンスについて勉強しろ、と言われた。ゲームを開発する場合は、描画の最適化、思考ルーチン、マウスやウィンドウ周りのユーザーインターフェイスをどうするかなど、技術的な基礎知識を身に付ける様に求められた。

ところが、新卒で就職し家庭用ゲームの開発に携わる事になると、全く違った文化が待っていた。まず、美しくプログラムを書こうとすると叱られた。当時の家庭用ゲーム機では、リソースが不足していたため、美しさより、最適化しろと言うのである。今日、プログラムを学習された人なら、本当にエレガントなコードは、美しさと同時に最適化されたコードであると思われるかも知れない。しかし、それは、ビット単位でプログラムやデータを圧縮しなければいけない状況では、限定的にしか両立し得ない。

また、当時の PC ゲーム開発は規模が小さく、プログラムだけでなくタイトル全てを個人で請け負う場合も多かった。多人数で開発する場合でも、プログラム作業が部分ごとに完結しており、実質的には個人作業が多かった。しかし、家庭用ゲーム開発では、入社した会社が業務用ゲームの大手だったこともあり、多人数で分業することが求められた。業務用ゲームは、アクションゲームやシューティングゲームが多く、プレイヤ、敵、仕掛けなど、同じステージのプログラムを分業できた。このため、職人業的なゲーム開発のノウハウやコードの書き方などを、社内で共有できた。

新卒で入社したプログラマーは、オープニングやエンディング、ミニゲームなどの軽い作業から担当し、個々のザコ敵の実装から敵システム全体の実装へと 2〜3 年修行し、プレイヤや基本のゲームシステム（ゲームエンジン）が作れる様になるとチームリーダとして自立する。こうしたキャリアアップの中で職人的な伝統が伝えられ、各自が自分なりに 2D ゲームを作る技法を身に付けることができた。

2D ゲームを作る技法となると、それについての教科書も無く、実際に経験された方で無いと理解して頂くのは難しいのだが、自分なりに学んだ部分をいくつか書き留める。

(1)　理不尽さの解消

経験の少ない開発者がゲームを作ると、面白さは少ないのに、やたらと難しいバランス調整に陥りがちである。難しさという物は、ゲームである以上、ルールに則り、プレイ状況を理解した上で納得できる物でなければいけない。また難しさに応じた面白さも無ければ、やる気も起こらない。死んだり、ダメージを受けるなど、プレイヤに不利益な部分の表現については特に注意が必要で、理不尽な要素を注意して排除しながら、面白さを作りだして行かなければならない。

(2)　攻撃について

避けきれない程の近距離での弾を発射は、弾が確認出来ず原因不明のダメージにもつな

がる。また、避けきれ無い程に速い弾など回避策の無い攻撃は、攻撃自体にゲーム性が存在しないので配慮が必要である。

(3)　攻撃合図

敵や仕掛けなどの動作については、ある程度の予測が立てられる様にする。
銃を構えるなどのポーズをとらせた後に弾を発射させる、機械的に一定の時間間隔で弾を発射させる、一定の距離に近付くと攻撃するなど、プレイヤが経験から学習し対処できる様にする。

(4)　当たり判定領域の大きさ

当たっていないように見えるのに、ダメージを受けるのでは、プレイヤも反感を持つ。当たり判定領域は、見た目よりも小さめに取る方が無難である。

(5)　速度と加速度

緊迫感を演出する上で、敵や弾の動きの速度を上げると、同時に難易度を高める原因ともなる。そこで、はじめはゆっくり動くが徐々に加速したり、高速で向かって来ても近付くと減速するなど、加速度を利用する事によって難易度の微調整が可能となる。

(6)　目線の注意

画面中のプレイヤの目線についても気を配る必要がある。絵的な変化を持たせるなど単調さを廃する工夫を検討しなければならない。しかし、進行方向の反対から敵を出現させたりするのは、注目すべき場所から強制的に目線を外させることになり、プレイヤの状況把握を困難とさせるので注意が必要である。

(7)　挙動をみる

敵の動きなどで、機械的な単調さを廃したい場合、プレイヤの挙動に反応して、プレイヤに絡ませる事も検討できる。プレイヤの挙動と無関係な攻撃ばかりだとつまらなくなる。

(8)　レスポンス

操作に対するレスポンスの良し悪しは、爽快感や、継続してプレイするモチベーションを、プレイヤに与える。プレイヤが思い描く行動を、遅延なく実現できるかが重要である。

攻撃アニメーション中にジャンプボタンを押した場合のモーションのキャンセルなど、状態遷移に気を配り、プレイヤにストレスを与えない様な配慮を取る。

(9)　移動の引っかかり

キャラクターが移動中に、進行方向にある壁などの障害物に当たった場合でも、単純に止まるだけでなく、縁を迂回させるべきかなど、操作性を考えた検討が必要である。所持アイテム一覧における選択についても、アイテムの無い方向へカーソルを移動させた場合、右端から左端へとループさせるか、停止させるかなども十分に配慮をする。

(10)　敵の再セット

キャラクターの移動に合わせて画面がスクロールする場合、単純に発生位置だけで敵の再出現を行うと、敵を撃破した直後に少し戻っただけで再セットされて生き返るので、不自然さが伴なう。

3.5.5　C 言語の時代

筆者が知る限り、国内のゲーム開発において、大規模に C 言語が用いられたのはロードランナー（PC-9801 版・㈱システムソフト・1985 年)が最初で、PC では、その後、徐々に使われる様になった。家庭用ゲーム機向けの開発で、本格的に C 言語が普及し始めたのは 3DO、PlayStation、セガサターンが登場した 1994 年以降で、1996 年には NINTEND064 が発売され、それ以降のプラットフォームは C 言語対応となった。

それ以前にも、ミドルウェアメーカ製などの C 言語はあったが、特定のタイトルで僅かに使用されている程度であった。

また、この時代、業務用ゲーム機の 3D ゲームブームが起こり、その影響を受けて、一斉に、家庭用ゲーム機にも 3D 化の波が訪れた。

筆者の場合、C 言語と 3D 化の対応が同時期で、基本システムを開発する作業量が跳ね上がり、非常に苦労した。

2D 表示の基本システムでは、社内製のデザインツールで描画したデータを変換し、オブジェクト（スプライト）パターンとして表示させ、アニメーションさせ、背景を表示させるだけで良かった。しかし、3D 表示になり、デザインツールは Softimage などに変わり、モデリングしたデータを変換し実機で表示させボーンマトリックスによる 3D アニメーションを実現したり、PhotoShop などで作成したテクスチャーの変換ツールと、そのアニメーション・システムの構築が必要だった。その上、当たり判定についても、データ変換と実機の処理システムが必要となり、それまで無かったライトやカメラの要素も加わった。

当時、邦訳されていた 3D プログラミングの専門書は少なく、OpenGL プログラミングガイドとリファリンスマニュアルくらいしか役に立たなかった。この頃から、ゲーム開発に米国の専門書や論文が必要となって来た。

また、この頃までは PC の CPU の処理能力が家庭用ゲーム機のそれを凌駕していたが、以降、差が随分と縮まった。また、内蔵の RAM 容量もメガ単位になり、メディアとなる ROM カセットの容量も増え、CD-ROM を搭載する機種も現れた。全般として、メモリ容量が増えた。3D になり、データは巨大化して圧縮する必要は増したが、メモリ全体に対するプログラムコードサイズの割合は少なくなり、プログラムコードの圧縮は以前ほど重要な課題では無くなった。そうなると SIGGRAPH や他の学会などで発表された論文が利用出来る様になり、ゲーム開発に最新の技術が導入されるようになった。

90 年代中盤以降には、それまで UNIX ユーザーを中心とした一部の技術者しか利用していなかったインターネットが PC で普及し始めた。家庭用ゲームのネットワーク化も PC から始まり FPS、戦略 SLG、RPG など PC での人気ジャンルがスタンドアローンから次々にネットワーク化し、多人数で遊べる様になった。

また、この頃 PC では Windows が普及しはじめ、それに伴ない、国内メーカー独自規格の PC プラットフォームが姿を消して、Windows に一本化されていった。CG の表示能力も増して、家庭用ゲーム機に負けない表示能力を有する様になった。

2000 年代に入り PlayStation2 が発売され、3D 描画用のマクロコードがプログラミング可能となり、それに続いて Xbox や PC の DirectX8 がシェーダ機能を実装し、CG 表示がよりプログラマブルになった。しかし、ゲームの基本システムの開発は複雑で困難となり、3D ゲームの開発環境その物をゲームエンジンという形でミドルウェア化し販売する企業も出現した。

Windows の OS メーカーである MicroSoft が Xbox を発売し家庭用ゲーム機市場に参入した事で、PC と家庭用ゲーム機のプラットフォーム格差を吸収し、多少のハードウェア仕様の差はあっても、ゲーム開発の考え方としては、同じようなパイプラインで作れる様になった。

このプラットフォームの世代交代時期に前後して、C 言語での開発経験も増し、メモリ搭載量も増えたため、C++言語を採用したり、AI やフィジックスなどの複雑な処理も、徐々に取り入れられた。

家庭用ゲームのネットワーク対応は、インターネットを使っていない一般家庭もあり PC と比べると展開が遅れたが、徐々に対応はなされていった。特に Xbox では、Xbox LIVE と呼ばれる総合的なサービスの仕組みを提供し始めた。

3.5.6　C 言語普及によるプログラマーの変化

3D 化による開発量の増加と、脱アセンブラによる一般的な技術者雇用が可能となり、徐々に大学や専門学校、他業種からのプログラマー雇用の敷居が下がり、ゲーム開発に従事するプログラマーの数が飛躍的に増えた。また、仕事の細分化や専門化も進み、職人的な技術伝承の機会は消失した。

逆にタイトル毎の人件費が膨らみ利益率は下がり、論文からの最新技術導入が可能となり、市場が大きく、論文発表も多い英語圏、特に北米での技術先行が加速した。

大人数の連携作業が必要となり、バージョン管理ツールやアセットツールが重要になった。

仕事量の増加に対応するため、これまでプログラマーが職人的な技法で作成して来た様々な部分、例えばプレイヤーや敵の動き、画面の表示調整などを、独立した業務として分離し、その為の専用ツールを作成しノウハウを教育する事で、プログラマーで無い人でも担当出来る様な対応が求められるようになった。

3.5.7　3D 表示に伴なう追加要素

ゲームの CG 表現が 3D 化しても、2D ゲームの技法は継承できる。しかし、表示要素にはライトとカメラという概念、また、動きの要素には AI やフィジックスなど、新たに加わる要素もある。

(1)　ライト

様々な角度からライトが当る事によって 3D モデルは初めて立体感を表現する事が出来る。映画やドラマなどを観てもわかる様に、ライトには、屋外の太陽光や室内の電灯など、その空間にあらかじめ配置されている環境光と、カメラが撮影しやすい様に、カメラマンについて歩く照明さんが、被写体に対して当てるライトの 2 種類がある。この 2 種類を調整する事で、見やすい画面作りをする事が出来る。初期の 3D ゲームでは、この点を理解せず、照明さんのいない見にくい物が多く存在した。

(2)　カメラ

また、カメラにおいては、車載カメラがカメラマンの撮影する映像より機械的でガクガクした物であるのを思い浮かべるとわかる様に、カメラという物体として扱うよりも、敵やプレイヤーと同様にステージ中にカメラマンという存在を設けてシミュレーションさせ撮影した方が、より良い画作りが出来る。

要するに、プレイヤーが大きな崖から落ちたとして、重力落下する被写体と一定距離を

保って追いかけるより、クレーン車を使って等速で近づいた方が画面酔いのしない見易い映像が撮れるという事である。また、プレイヤーが走る時は、進行方向に視点を向け視野を広げたり、物陰に隠れて移動する場合は、四角くモニターに映し出されている投影面の四隅から被写体までの遮蔽判定を取り障害物の無い方向へ回り込んでやると良い。天井が低い場合はカメラマンが中腰でカメラを持つだろうし、部屋の様な狭い空間に入ると広角レンズで視野を広げるだろう。屋外で遠方にいる場合は望遠レンズを使ってズームするだろう。カメラマンが当たり前の様に行っている事をシミュレーションすれば良いと考える。

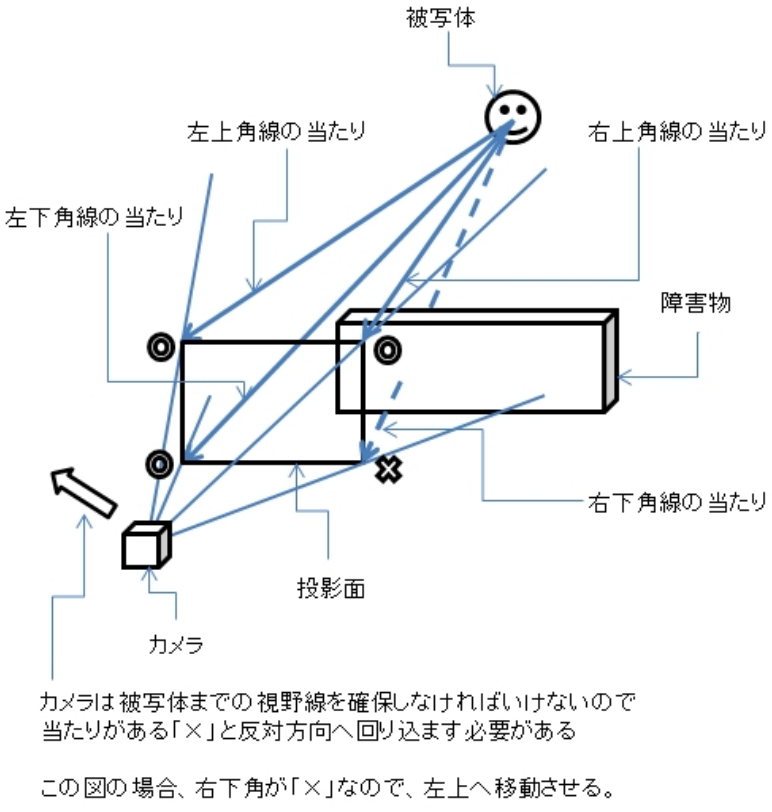


図 3.5-01　障害物の無い方向へ回り込み

(3)　コントローラー

3D の場合、カメラ位置が変わると画面の向きも変わるので、コントローラーで右に移動中のプレイヤがいるとして、カメラがプレイヤの周囲を 90 度以上回り込むと画面上の左右が入れ替わってしまう。移動前の方向のまま進むと右の指示を出しているのに、左に動くことになる。逆に移動後の指示に従うと、コントローラーの操作は変えていないのに、カメラの都合でプレイヤの進行方向が異なってしまう。カメラを動かした場合、コントローラーの反応にも、こうした点で破綻の無い対処が求められる。例えば、コントローラーの操作が途切れるまで、カメラの回り込みを抑制するなどの対処が可能である。

(4)　動作の切り替え

キャラクタ・モーションについても、2D の頃はパラパラ漫画の様に予め描いた絵を変えるだけの仕組みなので､固定視点で手描きのアニメーション的な表現だけでも許された。しかし、3D では、ボーン情報を入れてポーズ形状を計算したり、カメラを動かして、あらゆる視点から撮影できたり、実写の様なリアルな表現が可能となった。このため、例えば攻撃中にジャンプボタンが入った場合など、2D であれば、途中でアニメーションをキャンセルしてジャンプのアニメーションに繋いでも、それ程違和感が生じ無かったが、リアルな 3D 表現の中では許されない事が多く存在する様になった。

この場合、攻撃中のポーズからジャンプに遷移するためのモーションデータを用意したり、計算によって中割のポーズを作り繋いだり、攻撃が終了するまで、若干の動作開始を遅らせたりする処理が必要に成った。

(5)　AI

敵の動きにおいて、複数のザコ敵の動きをボスがコントロールするといった団体行動の AI は 2D 時代から存在したが、地形形状が 3D で複雑化したため、地形把握や移動経路の検索などの処理が複雑化した。

また、敵の動きも、個体ごとに顔の向きによる視野範囲や耳の位置による聴覚範囲などを計算し認知表現が取られる様になった。

(6)　フィジックス

CPU の処理能力が上がり物理挙動を正確に計算出来る様になった。

具体的には、積み重ねたブロックが崩れる様な剛体シミュレーションや、人体が衝撃により倒れ落ちる様なラグドール的な表現、破壊で破片が飛散る様なエフェクトシーン、負荷の高い流体の表現などの利用が可能となった。また、旗やスカートを揺らす様なクロスシミュレーションも広義的にはフィジックスの計算手法の一つと考えられる。

この技術を使う上で注意すべき点は、予め調整したアニメーションデータや適当にプログラムで動かしていた旧来の手法と比べると、動きはリアルにはなるが予めイメージした物に近いとは限らず、コントロールが難しく、処理も高負荷になるといった点がある。また、ゲームの処理の都合上、コリジョン形状と表示上の CG 形状が必ずしも一致しないのと同様に、フィジックスで使用するオブジェクトの形状も表示形状と必ずしも一致させない方が良い場合がある。ヘッドフォンの様な複雑な形状を扱う場合、キューブ形状で近似して高速化したり、表示上は複数のオブジェクトをひとつにまとめたオブジェクトとして扱うなどの工夫も必要になるかも知れない。

3.5.8　現在のプラットフォーム

現世代の家庭用ゲーム機、PS3 や Xbox360 などの HD 対応機種では、複数のプロセッサ・コアが搭載され並列分散処理が標準的に利用される様になった。以前ならマイクロコードやマクロコードで記述しなければならなかった、シェーダや Cell の用な専用チップコードまでもが C 言語に近い形で記述出来る様になった。表示回路であった GPU を、GPGPU と呼ばれる汎用的な並列プログラムで利用可能にもなっている。フィジックスや AI も、ジャンルによっては標準的に使用されるのが当然の状態となり、表示系では、テクスチャーが画像イメージだけでなく、法線マップの様な形状データ、オクルージョンマップの様なライト情報まで扱う様になった。ラジオシティ法など事前計算したライト情報をモデルに埋め込んだり、環境光のライトマップとしてデータ化して利用することも可能となった。モデルの形状変形もシェーダで行える様になった。一方で任天堂 Wii では、Wii リモコンという個性的なモーションデバイスでゲームの可能性を広げている。

また、ネットワーク利用も、対戦や多人数プレーだけで無く、アイテム課金やダウンロード販売も一般的になった。

プログラム言語としては、開発の効率化の為、Lua や Squirrel の様なゲーム開発用の汎用インタプリタや、Flash の ActionScript を画像表現に用いるなど、導入され注目されている。

ゲームに使えそうな先端技術を追いかけようとしても、細分化が進みすぎて、もはや個人では限界を感じる様になった。また、ゲームプラットフォームのユーザー数を見ると、主役は家庭用ゲーム機ではなく、DS や PSP などの携帯ゲーム機の方が多く、PC も同様にディスクトップでは無いノート PC やネットブックに中心が移行してしまった。更には、携帯電話も考えるとゲームユーザとは、どこを主流と考えて良いのかもわからない時代を迎えている。

3.5.9　ゲームとは何か？

筆者が PC でゲーム開発を始めた頃のゲームジャンルは、アクションゲームとシューティングゲームが主流で、その後、アドベンチャーや RPG などが登場しブームとなった。家庭用ゲーム機での開発に移った頃は、縦シューや横シューと呼ばれる 2D のシューティングゲームがブームだったが、数年で下火となり、変わって格闘ゲームが現れ一大ブームが訪れ、それも過ぎ去った。続いて 3D のアクションゲームがブームとなったが、それもコア層のマニアックなジャンルへと変貌しつつある。DS ではカジュアルなゲームも流行しているが、携帯電話や SNS では、それ以上にライトなゲームが流行っている。

昨今の DS のカジュアルゲームなどを見て、「あれがゲームでしょうか？」という声をよく耳にするが、過去のブームを振り返ってみてもヒットジャンルに一貫性は無く、同じジャンルのブームが再来する事もあるが、前回のブームとは何かしらの相違点もあり、長期

間同じ物が好まれるケースはむしろ少ない。ゲームはヒットビジネスで、ヒットビジネスは留まること無く変化していくのが常で、これからゲーム開発の仕事に従事したい方には、その認識を持って働かれる事をお勧めする。

3.5.10 開発者として

昔と比べ家庭用ゲーム開発は大規模になり、専門的、極小的な業務が増えたが、開発者は研究者と異なり、個人的な作業より、組織的に開発したゲーム作品（商品)で業績を計られる事が多く、優秀な開発者は、単なる技術者としての技能だけでなく、作品全体をコントロール出来る組織人としての能力が求められる事になる。即ち、その人がメンバーとして加わったから、作品全体が良くなった、という評価のされかたである。

自分自身も、日々精進して、少しでも面白い作品作りに従事する所存で、この次はスタッフロールでお目に止まれれば幸いである。

参考書籍：

書　名	著　者	出版社
Lode　Runner で学ぶ実践 C 言語	ビー・エヌ・エヌ企画部	（株）ビー・エヌ・エヌ新社

参考タイトル：

紹介した章	技術内容	ハード	利用タイトル
3.5.5	C 言語のゲーム開発利用	PC-8801	ロードランナー（システムソフト 1985 年)

4. ゲーム開発者に対するヒアリング調査

4.1 近藤敏信氏ヒアリング

略歴
1989 年、システムサコム入社。同僚と独立し、『メタルサイト』(1989、X68000)の開発に参加。メタルサイト開発チームを母体とする(株)フィルインカフェにて、『ルイン 神の遺産』(1993、PCE-SCD)の開発に参加。同社を離れ、『海腹川背』(1994、SFC)、『海腹川背・旬』(1997、PS)の開発に参加した後、(有)スタジオ最前線を設立。以降、同社にて、『どきどきポヤッチオ』(1998、PS)、『新世紀エヴァンゲリオン 麻雀補完計画』(2000、GBC)、以降主に DS、PSP などの RPG 案件の開発に参加。アニメーション作品への協力や、カードゲームや小説のイラストなど、多方面で活躍中。

(1) 近藤氏のキャリア

--近藤さんが、お仕事を始められたのは 20 歳ぐらいの頃ですよね。

近藤:そうですね。自分は高校卒業してそのまま東京に上京してきてシステムサコム(以下、サコム)という会社に入ったので。正確に言うと平成元年から働いています。

--最初に携わったタイトルは『メタルサイト』ですか？

近藤：メタルサイトはサコムを半年くらい働いて辞めた後に作っているんですよ。同期のメンバーと一緒にサコムを辞めてメタルサイトを作ったという流れですね。

--サコムでは、ドットワークを中心にやられていた感じですか？

近藤：割とビジュアルシーンで一枚絵みたいなものがゲームで使われてくるようになっていた時期で、サコムにいるグラフィッカーが、みなさんドッターさんだったので、でっかい一枚絵みたいなものを描く人がいないこともあって、自分が入社できたというところはありますね。
自分は漫画家を目指して上京してきていたので、その当時サコムが抱えていた RPG 案件のオープニングの絵やビジュアルシーンの絵を描いていましたね。ドット絵は当時の主流なのでドット絵も作業していましたが、特に一枚絵が多かったですね。

--『海腹川背』(以下、海腹)の時はいかがでしたか？　あれはまだスーパーファミコンの時代ですから、最終的にはドット絵に落とし込む形になっていたと思いますが。あれはキャラデザ以外に近藤さんはどのくらいまで担当されていましたか？

近藤：初代の海腹はキャラデザだけですね。海腹の開発チームの人がドット絵を描いていた感じですね。

--：当時のドット絵は、キャラクターデザインを持っていって、実際にああいうキャラクターになるわけですよね。

近藤：あれはちょっとイレギュラーで、企画を立てた人のボンヤリとしたデザイン画がありまして、それがドット絵になって、ボンヤリとしたデザイン画とドット絵を見て、イラストとして起こしているという感じなので、通常と逆の流れですね。
例えばドット絵の見栄えの良い TIPS みたいなものによってドット絵って見栄えがしていたわけです。ドット栄えするデザインってやっぱりあって。普通の絵のうまい人が描いているキャラクターが、ドット絵としていい絵かって言われると全然違っていて。
だから初期の頃って、プロのイラストレーターさんとかプロの漫画家さんとかがデザインするようなゲームがありましたが、大抵失敗している。「いや、そんなところにそんな細かいデザインしても見えないから」というものがあったりしていました。絵としてはよくても、ドット絵だと映えないと、見分けがつかないとか、そういう理由で上手くいかなかった感じがありました。
ですから、どっちかというとドット絵の人が「この辺 3 ドット同じ色が続くんだったら、ここにスリット入れたほうがカッコいい。」とか、ドット絵都合でデザインがなされていた時代でしたね。
そしてドット絵都合でデザインをやっていたのが、割と画面が細かくなってきて、色もたくさん出るようになってという風になってきて、ドット絵都合ではなくて、絵をちゃんと表現できるようになってきたのが 90 年代かなと。
そうなった時点で同時にドットの技術も色数で打ち消されるようになってきちゃうというか、より、生のデザインとして良い絵が、ドット絵としてもよいもの生み出す風に変わっていった、そういう過渡期でしたね。だから海腹の時はは、まだドット絵を主体として、それを逆にイラストに起こした感じでしたね。

--『海腹川背・旬』(以下、旬)の時には多分逆になっていますよね

近藤：そうですね、旬の時は自分が全部絵で描いているんですけど、それを取り込んだものを切り抜いて動かしています。ドット絵とかで起こさなくても PS の色数と解像度だっ

たら絵をそのまま取り込めば動く、というので「じゃあやってみようか」と。

--ペーパーワークでがっちり描いたやつがあって、ということですか？

近藤：そうですね、それを何枚も何枚も取り込んで、微調整、位置あわせくらいですね。

--近藤さんの色の使い方は、ちょっとパステルで輪郭ぼかすような感じですから、多分そういうのがやりやすかったんじゃないかなと思うんですが、あれはもしかしたら意図的なんでしょうか。

近藤：特にはないですが、絵としては意図していたことはあります。ちょっと話が飛んじゃうんですけれども、現状だと例えば、CG でプリレンダリングしているものをドット絵として使うときはわざとドット絵っぽく見えるように特殊な処理をしているんですけど。旬の時は逆にドット絵っぽくしたくなかったというか……ドット絵の方に近づけてしまうと、せっかく一枚絵でいっぱい描いているのに、ドット絵だと思われなくなかったんですね。絵として認識して欲しいという思いがあって。
ペーパーが切り抜かれて動いているような見栄えになると良いな、というのを狙っていました。

--「プレイステーション」(以下、PS)、「セガサターン」が発売になって、凄い需要が出たからだと思いますが、97 年に『OPTPix』という減色ツールが出ますよね。OPTPix の前、94 年に『pag1teto』というまた別のツールがありました。ちょうど 90 年代の真ん中から後半にかけてのところが、それまでのドットから、表現媒体は基本的にはドットですが、例えばキャラクターの 8×8 とか 16×16 の縛りがなくなったりとか、色数が 256 色とかフルに使えるようになったりとか……。

近藤：そうですね、あそこでガラっと変わりましたね。実際に絵を描く人はドット数とか色数から開放されたというか。普通に絵として描けば、ゲーム上にドットで出せるという風になったんで、大きく変わりましたね。

--90 年代後半になってそういう感じで来て、97 年に旬が出て、翌年が『どきどきポヤッチオ』(以下、ポヤッチオ)が出ます。ポヤッチオと旬って 1 年しか間が空いてないですけど、タイトルの間に違うことを試したとか、そういったことはありますか？

近藤：違うことを試したというよりは、旬の場合は当時最先端の PS などでゲーム機のスペックがあがって来て、先ほどいっていましたけれど、OPTPix みたいなもので色数から

も開放されたという所をすごく前向きに捉えて、生の絵をそのまま動かそうというのが旬で、それとは逆にポヤッチオの方は純粋にドット絵として勝負して、よりいいものを目指そうという思いで、ツールなど諸々の環境を整えてドット絵として一番いい状態というものを目指しました。旬で生の絵を動かそうとしていてきついなと思ったのは、まだ PS では VRAM とかメモリの容量がやりたいことに対してはちょっと足らないというか…厳しいところがあって。
そうするとセル組みしたりチップ組みしたりした方が、クオリティが出せる部分もあるわけですよね。ポヤッチオのドット絵も凄く細かく頭とか手とかに分かれて組み上げているんですね。PS の頃が本当に過渡期だったんだなと思いますね。なので、先端的なことと、今まで培ってきたものとを別個でそれぞれやってみたという感じです。

--近藤さんのこの流れというのは基本的に 90 年代どんどん変わっていく中で、制約の強かったハードが PS やサターンが出てきて絵が自由に描けるようになって、クオリティを上げていく形でやってきています。
それとちょっとかぶる形で、ゲームボーイカラー(以下、GBC）の案件もやるようになりますよね。実際にハードウェアのアーキテクチャとしてはちょっと前の時代のスーファミとかの頃に戻る感じになっています。
それで、例えば一回 PS でやったことは当然そのままでは使えなかったりしたと思うんですけども。そういう制約がまた付く所に戻ってきてやったことはありますか？

近藤：意外なことに、以前はは制約だったものが、環境が変わったことによって、制約ではなくなっていたということが沢山あって驚きました。例えば、ドット絵に関しても、当時はドット絵で描いていたんですけど、もうその辺の GBC の案件だと、ビジュアルもっと大きな絵をフルカラーで描いてそれを縮小・減色して使用していますし、ドット絵のチビキャラも 3D で起こしたものをプリレンダして、それにちょっと人間が手を加えていたりであるとか、あるいは本当にそのまま減色して使ったりとか。以前と作業のアプローチが全く変わったので、同じ『作る』と言っても、スーファミ時代の『作る』と、PS とかで減色ツールなどが発達した後で、『作る』では全然違っていましたね。

(2)　携帯アプリの存在によって、再び脚光を浴びたドット絵

--確かに 10 年くらい間が空いているので、その間にがらっと変わっちゃっているということですね。世間的にも、GBC に関しては 2000 年はハードウェアの時代として最後のほうだったかと思いますが、ちょうどこの頃 2001 年ぐらいから『i アプリ』といった携帯のアプリが「ドコモ」も「au」も始まって、僕の周りで見ていると、急に思い出したようにドット絵の需要が出てきて「ドット絵を描ける人もういないんです」みたいなことをどこの

スタジオも言っていたような気がしますが、近藤さんの所に話が来たりしましたか。

近藤：ああ、そうですね。やっぱり駆け込みでありましたね。うちにはドットに強い人間がいましたから……。先ほど言っていた GBC とかゲームボーイアドバンスなどと携帯との違いって、GBC などは ROM カセットじゃないですか。何が変わったかというと、ROM の要領が格段に大きくなったんですよ。なので、細かい 3D アニメーションの 1 コマ 1 コマの画像を減色して突っ込んでも、ROM に山ほど入ったんです。だからそういったフルカラーで描いたものを縮小して減色するという方法論が GBC には使えた。ですが、携帯のアプリは使用できるメモリサイズが小さいので、でかいものを縮小して大量に流し込むみたいなことをしちゃうと、瞬く間にメモリの使用可能サイズを超えちゃう。
結局携帯はまた昔の方法論でチマチマやるしかないということになったんですよね。そういう意味でドットの需要が急増したんですよね。ところがドッターさんは激減していたので、ドッターさんいないか？　みたいな話が業界を駆け巡ったっということなんでしょうね。

--その頃の携帯の時から単価は変わりましたか？

近藤：携帯は安かったイメージしかないですね。やっぱり携帯ってどれだけ人気があったといっても、やっぱりダウンロード 1 本 300 円程度ですから。昔、スーファミとかファミコンとかってカセット 1 本何千円でしたので、そういう意味で同じことやっていても昔のほうが良かったというイメージですね。

--最近はニンテンドー DS(以下、DS)案件を主にやられていますが、2D を表現するメディアとして、どんな感じに捉えていますか。

近藤：自分的には若干厳しい感じですね。厳しいというか液晶の解像度がもうすこしあると、絵かきとしてはうれしいですかね（笑)。そこを携帯機だからということで、割り切った目で見るのであれば、2D 的にも 3D 的にも色々おもしろいというか、何でもできちゃうんですけれども、最終的な出力があの解像度なので、1 ドットが 1 ドットが大切になるので、厳しくみる必要がありますね。

--ドット数が少ないから、絵を描くのがしんどいみたいなこともたくさん出てくるわけですね。理想の解像度というのは……最低 640×480 は欲しい？

近藤：そうですね。それくらいあるといいですね。そのくらいの解像度のあるマシンになれば、ドット絵と意識しないで縮小と減色で相当な部分やれちゃうんじゃないでしょうか。

--旬の時はあまり感じませんでしたが、ポヤッチオの時って、画面の面積に対してキャラクターがちょっと大きめですよね。あれはやっぱり、キャラが大きくなって使えるドット数が増えることによって、豊かに細かく描けるようになる、そういうものを意識されていたのでしょうか。

近藤：そうですね。イベントの内容をキャラクターで表現しようと思っていたので、ある程度の大きさはないとそれが表現ができない。当時としてあのくらいが限界かなと思ったんです。限界というのは、あれよりも大きくなってきちゃうと、本当にドット絵じゃなくて、一枚絵をとして、絵としてキャラクターを描かなければいけなくなるという部分で、原画が必要になってきますからね。

--アニメを作るみたいな話になってくる。

近藤：そうですね。ドット絵として描ける中では、なるべく大きいサイズにした感じです。

(3)　作業用ソフトウェアの変遷

--減色ツールは、パッケージのソフトウェアれていたのか、あるいは、ご自身である程度カスタマイズされていたんでしょうか。例えば、近藤さんはこういう使い方をするとか、他の方はこういう風にちょっと変えているとか。

近藤：ありましたね。タイトルごとに変えていました。

--90 年代後半はやっぱり OPTPix をってましたか。

近藤：そうですね、当時は OPTPix の減色が圧倒的に綺麗なので使用していた開発現場は多いと思います。-あの頃はあれ以外に選択の余地があまりなかったですしね。ウチの場合だと GBC とかゲームボーイアドバンスの場合などはただ減色するだけじゃなくて、その内の 3 色は BG に割り当てて、この色はスプライトの方に割り当ててと、特殊な減色をしなくてはいけなかったので、自社でそういうツールを作ったり、いくつかのソフトを併用していた感じですね。
そうこうしているうちに OPTPix がツールメーカから販売される形に変わって、ゲームボーイアドバンス用の BG とオブジェを別々に減色できたり、キャラマップ減色できたりとかそういった機能を追加したものがそれぞれのプラットフォーム用に発売されるようになったので、自社では作らなくなったという感じです。

初期の頃は、安価で販売されていて、普通に絵を減色するしか機能が無かったので、そういった各プラットフォームに合わせた部分というのはそれぞれが作っている感じでしたね。

-- OPTPix は多分、元々はウェブページとかの gif の減色を意識していましたよね。

近藤：ゲーム業界で一杯使われるのに気が付いて、PS2 用、ゲームボーイアドバンス用とかっていう感じでそれぞれの機能に特化したものが出るようになったんでしょうね。その分自社で作らなくても良くなったといえばよくなったんですけど、結構 1 本の単価がはねあがったのでびっくりしました。それでも、今、家庭用の開発で使ってないところは無いんじゃないでしょうか。

--減色ツールのちょっと前の話でいうと、ドット絵から実際のペーパーのワークを乗せるという風に意識を変えていく途中のところの辺り。
95 年ぐらいでは、線画をペーパーで描いてスキャンして、プロポーションはそれ使って、それに全部ドットを乗せていく、というやり方で描いていた。そういう使い方はいかがでしたか？

近藤：そうですね、確かに取り込みを多用するようになりましたね。スキャナが物凄い高価な代物だったのが、普通に買えるような値段にどんどん安くなってきて、そういった周りの物理的環境もどんどんハードル下がってきたところも大きいと思います。
環境の話ですけど、最初にサコムに入ったときは、まだカーソルでドットを置いていましたから。世の中的にマウスってものがそもそも標準じゃなかったんですよね。
それが X68000 が発売されて、最初からマウスが付いているとか、マウスというものが身近になった。それでスキャナで取り込んでマウスで修正してみたいなものが値段的にも世の中の状況的にも普通になっていったというのがあるとおもいます。

(4)　作業環境の変遷

--ちなみにマウスやスキャナといった作業環境の話で何か大きく変わったというトピックは年代別にありますか？

近藤：自分は X68000 ユーザーだったので、X68000 贔屓なところがあるんですけれど、X68000 を開発ツールに使っていたメーカーは多かったと思います。当時は、各メーカーがドットを置く専用の描画ツールを自前で作っていましたから、色数の多い X68000 は最適だったんですよね。
PC-8801 や PC-9801」では家庭用のグラフィックを扱うには痒い所に手が届かなかっくな

ってきた辺りが、X68000 だと届いたというのもあったと思います。
その後、ウィンドウズ時代では、Photoshop で描いて減色すればいいみたいな時代になりますよね。その時代と PC-9801、PC-8801 とのミッシングリングを実は業界の中では X68000 が埋めていたイメージがありますね。
Windows98 のときにはもう大体ウィンドウズに移行していましたが、95 の時にはまだ現場では X68000 使っているところはあったと思います。

--タブレットを使うようになったのはいつ頃からでしょうか？

近藤：タブレットを使い出したのは自分はウィンドウズになってからです。X68000 の時はさすがに無かったですから。

--作業環境がダイナミックに変わったのは 90 年代前半から後半にかけて？

近藤：そうですね、あそこはずいぶん変わりましたね。
2000 年以降で変わったのは 3D の方ですかね。そっちは圧倒的に変わりましたね。
90 年代の後半には 2D 関係の作業環境としてはある程度確立したという感じです。

--モニターが低価格になって、解像度が高くなって、絵描きさんの作業のやり方が変わったりというところはいかがでしょうか。

近藤：自分としては、便利になったとは思いますが、物凄い何か次元が一個変わったとうい感じはないですね。まあ徐々に変化が来ているからかもしれないですけど。
どちらかというと、モニターもそうですけれども、まずそもそもメモリが足りなくてちょっと大きな解像度のものを描こうとするとガリガリとハードディスクにずっとアクセスしていた頃から、最近では普通に何も考えずに描けるようになったとか、そういう方が大きかったかもしれないですね。
それもゲーム製作的には次元が違うというレベルまではないというか、ゲームの製作作業としては、フルカラーで描いたものを減色してそのまま出せる時代になった、という方が圧倒的に違いますね。

--例えば CRT から液晶に変わったときに発色や色の調整は何かされましたか・

近藤：開発環境が液晶に本格的に変わったのはやっぱり PSP とか DS とか実際のターゲット自体が液晶になった頃ですね。
PS2 時代だとまだ CRT ですから。そうすると CRT でどう見えるかという方が重要なので、

割と CRT で作業をしていました。自分のモニターでどう見えようが、もう PSP なら PSP の液晶に表示されるんだとしたら、それに合わせた方が良いわけですから……そうなってきたときに、同じ液晶のほうが CRT よりも調整しやくなったカンジでしょうか。
今では多くの人が、ゲーム機を液晶テレビに繋いでいますし、実際に映されるシーンというものが世の中で圧倒的に液晶の方が多くなってきましたよね。それにあわせるように自分の開発現場も液晶になってましたね。
会社の中にいるとおもしろくて、徐々に液晶に入れ替わっていくんですよ。うちで一番多かったのは CRT と液晶をツインモニタにして両方置くというのが、グラフィッカーさんのスタイルだったんです。液晶系の仕事するときはこっちみるし、CRT 見るときはこっちみたいな。
最初の頃は「やっぱ CRT だよね」っていっていた人たちがある時期から「液晶の方が綺麗だな、色が出てるな」と入れ替わっていった時期があるんです。

--そういうときの、ベンチマーク的な液晶モニターは決まっていましたか？

近藤：うちは普通の民生品を買っています。仮に PC の仕事をするにしても多分普通にスタンダードに普及しているもので見る人のほうが圧倒的に多いわけですから、そこで見栄えがいいのが一番たくさんの人たちが見栄えいいんだろうと。

--今、よく使っているソフトはどういうものがありますか？

近藤：現場は Adobe が多いですよね。そのほかに『SAI』(以下、サイ)を使い始めという話しもよく聞きます。特に絵描きさんの間では割と流行ってきているというか。値段が安いというのと、性能もそれなりにいいので。
あとは 3D 系では、昔は 2D のときは減色ツールでフルカラーのものをそれぞれのターゲットマシンに合わせて解像度とか色数を落としていましたよね。3D もある時期から、ハイポリで作っておけば、それをターゲットマシンに合わせたポリゴン数にしたり、ハイポリの状態からレンダ画像みたいなものをベイクしてテクスチャーとして持っていったりする技術が確立したので、2D で起こっていたことがここ数年で 3D でも起こっているという感じです。
先ほどの DS の話にもなりますが、作っているポリゴンも別にポリゴン数も気にしないし、色数も気にしないし。2D の方の絵も当然色数も大きさも気にしないで作ったものを DS に合わせて適正な解像度と適正なポリゴン数にツールが変換してくれるようになったので、そこから格段に楽になりましたね。

--今はいきなりパソコンに向かって絵を描かれるんでしょうか？

近藤：自分はまだ手で描いてますね。でもいきなり描く人は増えてきましたね。手描きをしないでいきなりで PC、というところに自分のポリシーがある人も中にはいますね。

--それはやっぱりある程度若い人ですか？

近藤：自分の知っている人ですと、自分より年上の方もいますので、必ずしも若いとは限らないとは思わないんですけれど、ただ古い人ほど自分のスタイルがありますからね。
そういう意味では若い人の方が多いんだと思いますけど、自分からしたら「いきなりモニターで描くんだ!?いう感じですね。

--近藤さんは、作業中はタブレットで調整されるんですか？

近藤：作業中はタブレットで、そうですね。マウスとタブレット併用です。

--機能面やアプリケーションがどれくらい充実をたら近藤さんは紙を捨てるんでしょうか。

近藤：今でも最高スペックで PC を組めば大丈夫なんだとは思いますが、自分はどちらかというとペンを走らせるスピードが早いんですよ。そうすると、ペンを動かしたときにマシンのスペックが足りないとカカカって明らかにサンプリングしたポイントが分かるような感じになっちゃうんです。そうするとゆっくり描かないといけないので、何かイラッとしちゃったりするので、紙の方に戻りました。

--やっと、ご自分のスピードにマシンが追いついてくれた。

近藤：そうですね。やっぱりちょっとしたラグとかサンプリングのポイント数が足らなかったりというのが嫌だった人は多いと思います。あと以前だと、Photoshop ではキャンバスを回せないからダメだった人も多いと思います。サイはそのあたりいろいろ改善されているので、サイで全部デジタルになった人も多いんじゃないですかね。今では Photoshop でもキャンバスは回せますが……。
現在は、下書きペン入れと簡単な色塗りまではサイでやってフィニッシュワークだけ Photoshop みたいな人も結構いるみたいなのですね。

--最近タブレットに液晶パネルが付いているモデルがありますが。

近藤：あれはちょっとだけ描画がペンに遅れてくるので自分はきついですね。あのラグは

多分スペックの問題というより、かなりどうしようも無いラグかなと思うんですけれど。そこが例えばタブレットだったら直接画面に描いているんじゃないけど、脳内補完がかかって作業できるようになるじゃないですか。だから脳内補完できるようになれば、多少のラグも含め大丈夫だと思うんですけど、液晶ペンタブの場合、目の前でラグが視認できてしまうのできついですね。
液晶ペンタブでなくても困ってないので使っていませんという幹事ですかね。紙禁止とか言われたら買うかもしれないですね。液晶自体回して描けるので、キャンパスを回転できないソフトでも問題ないですしね。

--タブレットは今値段が落ちてきたこともあって、普及しているのでそこらへんで実際使われている方はどうなのかなと。

近藤：タブレットは本当に普通なものになりましたね。おかげでデジタルで絵を描く人が増えましたね。実際、環境的にはデジタルだけでいけるようになっているだと思います。

(5)　海外と日本の事情

--海外のゲームで、ちょっと前の韓国のオンラインゲームが 2D でやっていました。あの技術はこのソフトを使っているんじゃないか、ああいうアプリ使っているらしいといった情報はご存知ですか？

近藤：そこは良く分かってないですね。海外との交流はあまりないものですから。

--日本はかなり独自進化を遂げてきたということでしょうか？

近藤：自分の周りではそうですかね。若干こう、ガラパゴス的なところがあると思うんですよね。日本は家庭用が強かったので。
例えば韓国系のオンラインゲームは PC ですよね。北米とか海外だと凄いグラフィックの FPS とかそういったものもたくさんあります。日本だとそういったものを作る PC メーカーがほとんど無くて、大体は 18 禁 ADV 系だったりしますから。
そのため PC 系の凄いグラフィックのゲームを開発する経験などが乏しかったですからね。、PS3 とか 360 の開発では日本の現場は大分苦労した人が一杯いたんだろうなと思います。

結局、海外の開発を見ていると、何百万ポリゴンとか使って元の 3D モデルを作ってターゲットに合わせてローポリ化してテクスチャーにハイポリの情報をベイクして……みたい

な、2D でいえばフルカラーで描いた画像を縮小して減色するという手法に対応するようなことを結構古くからやっていんです。その方法だと絵と一緒で自由にポリゴンモデルを作れますよね。
ですが、日本って割りと職人芸的に UV ちゃんと開いて本当にテクスチャーを一枚一枚丁寧に描くといった作業方法でやっている現場が多かったので……海外は Photoshop でバリバリフルカラーで絵を描いているんだけど、日本人は一所懸命ドット絵で広大な画像をちまちま全部手作業で埋めていく的な……乱暴にいっちゃうとそういう差を感じました。
最近は日本のほうもハイポリの方からローポリに持っていくといったような様々な技術が、3D のほうも定着してきたかなという感じです。

--海外に遅れること何年くらい、でしょうか。

近藤 : そうですね。一番差が開いているときで 3、4 年は遅れているなと感じてましたが、今は追いついているメーカーさんはちゃんと追いついていると思いますし、昔ながらの方法論で作業されているメーカーさんもあると思うので、まちまちかもしれませんね。

(6)　ゲーム以外の業務について

--近藤さんはゲームの仕事以外にもアニメのお手伝いをしたり、カードゲームや小説のイラストを手がけていますよね。アニメやイラストとゲームの仕事の違いを感じることはありますか？

近藤：そうですね、やはり違いますね。

--例えば、キャラクターの設定を 1 枚ペーパーで落とすにしても描くときに気をつけるところが……

近藤：ええ、全然違います。ただ最近になって一緒になってきちゃって困ってます。先程もちょっと話したと思うんですけど、昔はドット絵には制約があった。
自分はドット絵も理解しているので、ドット栄えするデザインはこうだとか、ゲームの中でアクションゲームのキャラなんだったら、こういう風になってたほうが、ゲーム的に映えるというのがわかっていたので、単純に絵の上手い下手とは別でゲームに適している、適していないという観点からもゲームのキャラクターというものをデザインできたんです。
そこのアドバンテージは自分の中にあるなと思っていたんですけれども、今はどんどん解像度もあがるしポリゴン数も凄い一杯出るし、こういう絵じゃないと表現できませんというものの制約が無くなってきたんですよね。

なので、ゲーム用のデザインと普通のイラストと差がなくなってきますよね。昔に比べると純粋に「絵で勝負」という方に近づいてきています。
だからゲームのキャラクターのデザインをするときに、昔はその他のイラストレーターさんに比べて自分はアドバンテージを感じてた部分があるんですけれど、今はもう全然そういうアドバンテージが無くて、本当にどれだけいいデザインか、いい絵かとか、どれだけいいキャラクターかというところになってきました。そういう意味でイラストの仕事とゲームの仕事の差がなくなって来たなと思うときが多々ありますね。

--カードゲームを作っているスタッフの方と打ち合わせをされるときに、アナログのゲームとコンピューターのゲームの違い、というのはありますか？

近藤：そうですね、自分がゲームデザインして一から開発をしたカードゲームがあったんですけど、それを開発しているときに凄いショックキングだったのが、カードゲームって人気が続くと第四弾、第五弾と出続けるんです。
それまで自分はゲームってエンディングがあるものだと思っていたので……要はレベル40でラスボス倒して終わりとか。そういうゲームをずっと作っていたので、終わりがないゲームをどう作ればいいんだろうというのに戸惑ったというか、分からなかったんですよね。
でも実際に自分でカードを開発してみて、ようやく終わりが無いゲームが理解できたました。そういう観点でMMOを見たりオンラインゲームを見たら。ああ、色々作れそう自分もやれそうだなと思いましたね。それ以前だと終わりの無いゲームってどう考えればいいの？　といった状態でしたから、カードゲーム開発は自分にとって良い勉強になった、いい糧になったと思いますね。

--海腹の頃のグラフィックチームって何人くらいの規模だったんでしょうか。

近藤：海腹ってグラフィックはほぼ一人。どちらもほぼ一人なんですよ。海腹川背のほうは一人の神ドッターさんが一人でだーっと置いた感じで、旬の方は自分がひたすら一枚一枚全部描いていたみたいな感じです。
ただ、それは当時としてもかなり例外的というか、それなりの人数でグラフィックチームを組んでやる時代になっていましたから。

--ポヤッチオも近藤さんがかなりたくさん一所懸命描いた感じですか。

近藤:ポヤッチオの時のドット絵は、キャラクターのドット絵担当の人間が一人いまして、

一人で黙々とこなしてもらいました。背景は結構人間いましたね。
当時だと普通は、4、5 人でやるべきドット絵を一人でやってもらった感じですね。。ポヤッチオも海腹も過渡期にいたのかなという感じはします。ポヤッチオや海腹以降って、どうやっても一人じゃやれるようなものじゃなくなってきましたが、当時はギリギリやれる人間が一人でがんばれば何かできちゃった。

--それが最新の案件だとどんな感じになりますか？

近藤：DS の RPG を例にだすと、関わった人数だけで何十人ってなってるんで、もう規模が海腹とは全然違いますね。しかもキャラクターなどのドット絵に関しては 1 ドットも置いてないかもしれない、というくらいの勢いなんです。全部プリレンダしていたり、高解像度で描いてるものを減色しています。
ピュアに UI くらいじゃないでしょうかね、ドットで置いているのは。ユーザーインターフェイス系以外はほとんどドットをおかずに減色縮小したりプリレンダしたりというもので構成しましたね。

--そのドットを使わないというジャッジというのは、最初のタイミングで決められていたのか、制作にあたってこの量になると減色したほうがいいとか……。

近藤：それはもう最初からですね。ウチでドットを置かなくなったのが『宇宙のステルヴィア』ってタイトルからになるんですが、あそこからプリレンダしたものが、何もしなくもてドット絵に見えるようになったんですよ。一発レンダで。

--修正しなくてもそのクオリティになっている。

近藤：そうですね、それ以前って何か本当にどうやっても人間が修正しないとまずかったんですけど、宇宙のステルヴィアからドット絵修正しなくてもドット絵に見えるねこれっていうものがプリレンダできるようになったので。
ちょっと技術的なところでお話しますと、たとえば、ドット絵でラインを取るときに、普通のトゥーン系のシェーディングって境界線の上にラインを引くんですが、それをアルファで抽出すると、シェーダが境界線を描いているので、ラインの太さが半分になっちゃう。それがあるので、ドット絵として使いづらいんですよね。
体の中の線は 1 ドットなんだけれど、主線 0.5 ドットだったりとかっていう問題があったりして。ですが、シェーダも進歩してラインがアルファと一致するようになったんですね。アルファで抜いたときにラインが 1 ドットのままで入るという風にちゃんと取れるようになったのが宇宙のステルヴィアの頃からですね。

物凄い高解像度ですとアルファの淵で削れちゃうドットは関係ないんですけれど、64×64 ドットか 32×32 ドットだと 1 ドットの比重が凄い大きいので、アルファに一致したラインが描けるかどうかというのが結構ネックで。そこが描ける様になったので大分楽になりましたね。

--本来はジャギを消すためにアルファをかけたいという需要の方が多かったので、ツールはどうしてもアルファ込みで処理しようとするのがデフォルトになっている。それがやっとジャギとかそういうのは自分で考えてやるからっていう風に普通のツールを上手く組み合わせてやれるようになったのがこれくらいの時期からという感じですか？

近藤：そうですね。レンダリングした結果がそのままドット絵として使えるようになったので、そこから大量生産ができるようになりました。

--3D のモデルを作ってからプリレンダで落として 2D を作るときにアニメーションパターンのことを考えると、モデルが一個あってそれ動かす方が当然楽なんですよね。一個一個全部バラに描くんじゃなくて。

近藤：そうですね。アニメーションパターンでいうと、枚数がもの凄い数になってきちゃいますからね。
それから、クオリティの統一みたいなところでいうと。ドットで置くっていうと技術をもった人がいないとだめで、人数がたくさんいる。しかし、レンダリングだと誰がやっても変わらない。元のモデルだけ監修すればクオリティを統一しやすいというメリットもあります。

--モデルに OK を出したら後はアニメーションパターンをつけて、プリレンダでコマを落としていくだけだからそこはおかしくなりようがない。

近藤：そうですね。ドット絵でたくさんの枚数を描いたときに、下手なスタッフがまざると、下手なドット絵が大量に混ざってしまいます。そうすると、それの直しを入れるだけで大変になるわけです。
ですが、3D モデルの場合だと本当に最初の 1 枚だけなんですよね、見なきゃいけないのは。そこでちゃんとしたモデルにしておけば、あとはそこからレンダリングされるもののクオリティは統一されるので。そう意味では監修しやすいといいうかクオリティは統一しやすいですよね。

--考えると、2D ベースのゲームだとしても昔より規模の大きいものが全然作りやすくなっ

ているということで。小さいキャラが一杯出てくる RPG が全盛だったスーファミ時代からすれば、もう少しチームを小さくしてやれそうみたいな感じですか。

近藤:そうですね。「同じをもの作れ」って言われるんだったら大分減らせると思いますね。ポヤッチオの時、ドッターさんが泣いていたのが、キャラクターの向きが 8 方向あることだったんです。同じボカっと殴るモーション作っても、8 方向なんで、残りの方向も全部描かなきゃいけない。
「修行だ」って言ってたんですが、3D の場合だと 1 方向やっとけば後の方向は勝手にレンダリングなんでできちゃいます。それだけでも労力 8 分の 1 みたいな。同じパターンの角度違いをひたすら描くのは苦行でしたね。そういうのが無いだけでも効率はあがっているのかなと思います。

(7) これからのドット、2D、3D

--これから先、ドットを打つ時代に戻ることはありそうですか？

近藤：もうあんまりないと思いますね。携帯で一回戻ったという感じはありますけど、携帯自体の解像度も上がってきたし容量も多くなってきたし、本当に「減色すればいい」くらいになっていくと思うんです。

--よっぽどコンセプチュアルにドットを使いたいという話でなければ……。

近藤：そうですね。ドットに知識はどんどん必要なくなっていますね。だだ心配なのは、今ゲーム業界で上の方でディレクションとかチーフやっているような人間というのは昔ドット絵を置いていた経験をしている人間がいるので、パレットがなんだとかドットがなんだとか VRAM とはなんだとか、すごい基本的なこともしっかり分かっている上で今のことがやれるという感じなんですけど、それを全く分からない人だけになっちゃった時にはいくら減色できるからドットの知識は不要だといっても色々困るんじゃないかな、と思うんです。その辺の技術の伝承じゃないですけれど、今いらなくなっているんだけれど、最終的に突き詰めていくとそういったことは知っていた方がいいってことは一杯あると思うんですけどね……。

--世代的に近藤さんは今年 40 になられたということですが、ドッター世代の一番下になるんでしょうか。

近藤：どうなんですかね。自分がちょうど業界に入ったときってスーファミが発売された

年くらいだったので。まだそれ以降入って来た人たちもドット絵を置いたりしていましたから一番下ということはないと思います。30 中頃ぐらいの方がまだおられると思いますが、消えていった人も多いと思いますね。

スーファミ時代から PS に移行する間の、アーケードシーン、あの辺の大きなドット絵前後で職を失った人たちが結構います。ドット絵がどんどん大きくなってきたときに、あくまで置いているのはドット絵なんですけれど、ドット絵の技術よりも、「そもそも表示サイズが大きいからデッサンが取れていないとまずい」とか「絵的に最低限これくらいの画力が無いとまずい」というのがあって。
結構昔は「俺、絵描けないんだよね」っていうドッターさんが一杯いたんですよ。本当に絵は描けないんだけれど、ドット絵は描けるって人がいました。
でも、いくらドットって言っても絵が描けないとやれないみたいな時期に入って結構職を失った人たちがいる感じはありますね。

--その後に例えば 3D できないとね、というのが……。

近藤 : そうですね。3D の方でも最初の方ってポリゴン数が少ないんで本当に 3D としていいものが作れるかというよりかは、ポリゴンの 1 ポリゴン 1 ポリゴンをどう扱うか的な、まさにドット絵職人的な感じだったんですけれど、そこから 1 ポリゴンの比重が下がって、いいモデルが作れるかどうかの方に移ってきましたよね。
本当に今は物勝負になってきて、ドット絵もポリゴンもそうですけど、初期の頃はそのツールが使えるかとか特別なノウハウを知っているか、どちらかというとオペレーター寄りの人というか、技術中心な人が活躍される時代から、技術的な問題が取っ払われるようになってきて物勝負というか、いかにいい絵、いいモデルみたいな方にどんどんシフトしてきましたね。
今、3D の方だと、実機に表示しているローポリモデルと元となるハイポリモデルと分かれてているので、この辺の方法論がきっちりと確立してしまったときに、モデラーの現場も大きく変わるんじゃないかなと思いますね。1 ドットから絵描きさんが解放された感じで、1 ポリゴンというものから完全に解放される、直前にいるのかなという気がしています。

5. ゲーム開発者の就労意識とキャリア形成の課題

藤原　正仁（東京大学大学院情報学環）

5.1 はじめに―問題意識と目的

ネットワークやプラットフォームの進化にみられる近年の劇的な技術革新を背景として、とりわけゲームをはじめとするインタラクティブ・コンテンツにおける先端技術の応用について緊要な課題となっている。

そこで、本調査委員会（デジタルコンテンツ制作の先端技術応用に関する調査研究委員会）では、3 年間にわたって国内外の関連情報を調査し、ゲーム開発技術の編纂と歴史的考察､展望へ向けてのゲーム開発技術ロードマップの作成､ゲーム研究者のネットワーク、ゲームにおける産学連携、IGDA カリキュラムフレームワーク、開発者の人材育成の現状と課題などを明らかにしてきた。

しかし、デジタルコンテンツ制作の先端技術応用を担う開発者の実態については、いまだ明らかになっていない。劇的な技術革新、国際競争の激化、顧客ニーズの多様化など、ゲーム産業を取り巻く環境が急激に変化していくなかで、開発者の将来キャリアは不透明な状況に置かれている。他方で、ゲーム開発に関する職業は、初等・中等教育期の若者にとって羨望の的となっている。開発者の生涯キャリア発達、次代を担う開発者の育成に資するためのキャリアに関する情報提供が求められていると言えよう。

先行研究を渉猟してみると、例えば、International Game Developers Association（2004）（以下、「米国調査（2004）」と称する）は、米国の開発者に対して質問紙調査（N=994）を行っており、満足感、仕事の安定性、タスク負担、会社組織、仕事の質、キャリア展望など、開発者の生活の質（Quality of Life: QOL）の実態について明らかにしている[1]。この調査は、2009 年にも実施され、経年的にデータが収集されており、2010 年第 2 四半期に調査結果が公表される予定である。

また、United Business Media（2009）は、プログラマー（Programmers）、グラフィッカー（Artists and Animators）、ゲームデザイナー（Game Designers）、プロデューサー（Producers）、サウンド（Audio Developers）、品質管理（QA Testers）の給与に関する調査（N=3880）を行い、とくに 3 年未満のエントリーレベルの平均年収について示している[2]。

一方、日本においては、藤原（2007）は、家庭用ゲーム産業の人事担当者に対して、質問紙調査とインタビュー調査を実施し、開発者の人材マネジメントに関する実態について解明している[3]。

社団法人コンピュータエンターテインメント協会（2008）は、家庭用ゲーム産業の人事

担当者に対して、質問紙調査とインタビュー調査を実施し、ゲーム産業における新卒開発者の採用を含めた人材育成の状況と新卒者に求められるスキルを明らかにしている[4]。

藤原（2009）は、家庭用ゲーム産業の人事担当者に対する質問紙調査・インタビュー調査、プロデューサーに対するインタビュー調査に基づき、個（開発者）と組織（ゲーム会社）の観点から、開発者の人材マネジメントやキャリアディベロップメントの実態を解明している [5]。

さらに、近年、アニメ産業においても、日本アニメーター・演出協会（2009）が日本のアニメーターに対して質問紙調査（N=728）を行っており、アニメーターの労働や生活実態を明らかにしている[6]。

このように、日本では、開発者の人材マネジメントやキャリアディベロップメントに関する調査研究が蓄積されてきている。しかし、開発者の実態に関する調査はこれまでに行われておらず、雑誌や書籍、ウェブサイトなどで断片的に紹介されるに留まり、客観的なデータに基づく十分な検討がなされてこなかった。ゲーム産業において、開発者の人材不足が指摘されるなか、よい点も大変な点も含めて開発者のキャリアに関する情報を蓄積し、提供していくこと（Realistic Job Preview: RJP）で、人材の確保のみならず、人材の定着にもつながるのではないだろうか。

そこで、本章は、これまでの調査研究を踏まえて、国際比較・業界比較・労使比較の観点から、ゲーム開発者の実態を明らかにすることを目的とする。加えて、ゲーム開発者が先端技術を応用したデジタルコンテンツ制作を行っていくために必要な環境を具体化することが本章の課題である。

5.2 調査の方法

5.2.1 調査対象

家庭用ゲーム産業に関わる開発者（プロデューサー、ディレクター、プログラマー、グラフィッカー、プランナー、サウンド、デバッガー、ネットワーク・エンジニア）を対象とした。これらの開発者を対象としたのは、ゲーム開発の中枢を担う職種のためである。組織的観点からすれば、非開発職も重要な役割を担う（石島,2009）[7]が、開発職と非開発職の分業と協働、非開発職のキャリア形成に関しては別の機会に譲ることとする。

5.2.2 調査形式

本調査は、独立行政法人メディア教育開発センターが開発した、リアルタイム評価支援システム（Real-time Evaluation Assistance System: REAS）を利用し、インタ

ーネット調査を実施した。財団法人デジタルコンテンツ協会のウェブサイト等で事前に告知したパスワードを入力することによって、本調査に回答できる仕組みを構築した。いわゆるオープン型のインターネット調査である。調査データの回収にあたっては、日本ベリサイン株式会社のサーバー証明書を取得した SSL に対応し、暗号化通信に配慮した。

また、全国のゲーム会社 605 社の人事担当者に対して調査協力依頼状を送付し、本調査への回答に協力いただいた。さらに、ゲーム開発者のネットワークを活用し、調査協力の告知に支援いただいた。とりわけ、国際ゲーム開発者協会日本（IGDA 日本）、株式会社デジタルスケープの支援のもと、本調査への協力依頼がメールニュースならびにウェブサイトに掲載された。

5.2.3 調査実施期間

本調査期間は、2009 年 9 月 24 日～10 月 31 日である。

5.2.4 有効回答数

上記の期間に回収された調査データ（有効回答数）は、477 票に達した。

5.3 回答者・回答者の所属企業の概要

5.3.1 回答者の属性

はじめに、回答者の属性について、質問紙調査結果をもとに概観する。(1)性別、(2)平均年齢、(3)最終学歴、(4)学問系統、(5)配偶関係、(6)子どもの有無、(7)主たる生計の維持者、(8)現在の役職、(9)現在の職種、(10)就労形態の順にみてみよう。

(1) 性別

性別は、「男性」が 87.2%、「女性」が 12.8%の割合となっており、男性が約 9 割に達している。International Game Developers Association（2005:12）[8]では、北米・英国・豪州（米国 66%、カナダ 18%、英国 12%、オーストラリア 4%）の開発者（N=6437）に対して、労働力多様性調査を実施しているが、日本と同様に、開発者の構成は「男性」が 88.5%、「女性」が 11.5%という結果を示している。

ゲーム産業は、世界的に女性の社会参画が遅れている状況にある。老若男女にエンターテインメントを提供していく上で、とりわけ人材の多様性（diversity）が求められているのではないだろうか。

(2) 平均年齢

「平均年齢」は 33.79 歳（SD=6.52）である。階層別では 30 代が最も多く、52.6%となっている（図 5.3-01）。20 代は 28.1%、40 代は 17.6%、50 代は 1.7%と少なくなっている。

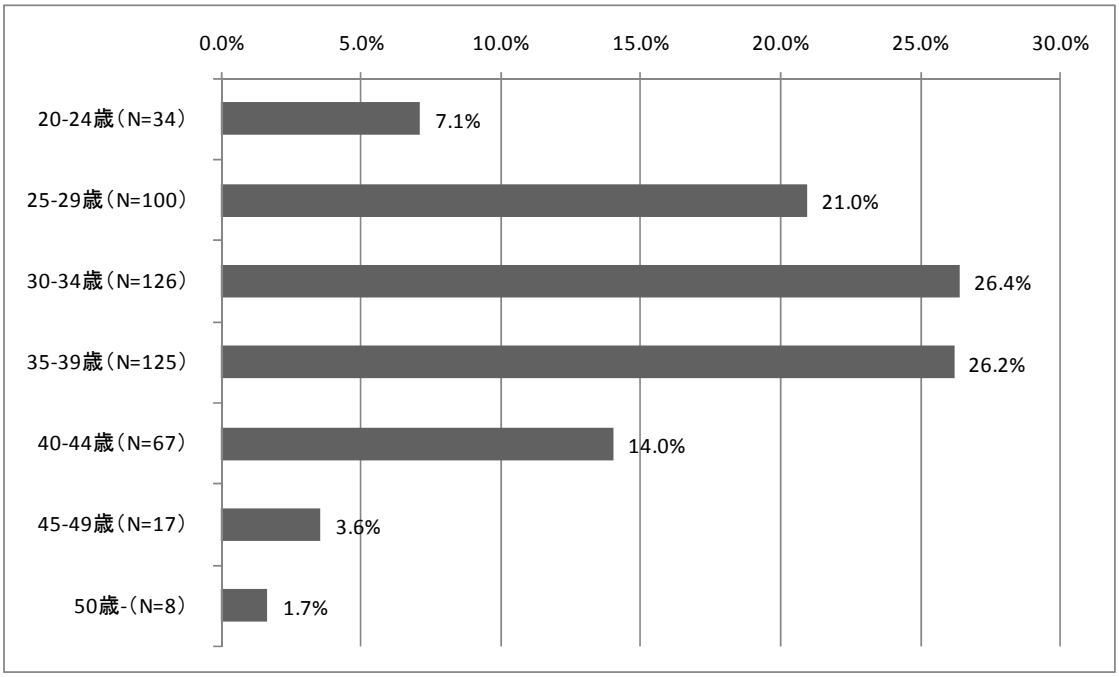


図 5.3-01　回答者の年齢層

(3) 最終学歴

最終学歴は、「大学卒業」が最も多く 44.0%、続いて「専修学校卒業」が 30.2%となっており、両者を合わせると 74.2%を占める（図 5.3-02）。日本では、次代を担う開発者の育成において、大学と専修学校が果たしている役割は非常に大きいことが分かる。

また、「高等学校卒業」は 10.7%、「大学院修士課程修了」は 9.2%となっており、それぞれ約 1 割程度の割合となっている。「高等専門学校・短期大学卒業」は 5.0%である。「大学院博士課程修了」はわずか 0.4%と少ないが、高度な専門知識を有する人材がゲーム産業に参入することにより、産業の活性化にもつながるであろう。

International Game Developers Association（2005:20）[8]によれば、北米・英国・豪州における開発者の最終学歴は、「大学卒業」（64.3%）、「高等学校卒業」（18.3%）、「大学院修士課程修了」（14.4%）、「大学院博士課程修了」（1.7%）、「中学校卒業」（1.4%）という結果を示している。すなわち、これらの地域の開発者で大学卒業以上の者は 80.4%に達しているのである。それに対して、日本の開発者で大学卒業以上の者は 53.6%に留まっており、26.8 ポイントの格差が生じている。近年、日本においても大学においてゲームに関

する教育が行われてきているが、欧米に比べて高等教育機関における日本のゲーム開発教育と次代を担う人材の輩出は遅れをとっている。

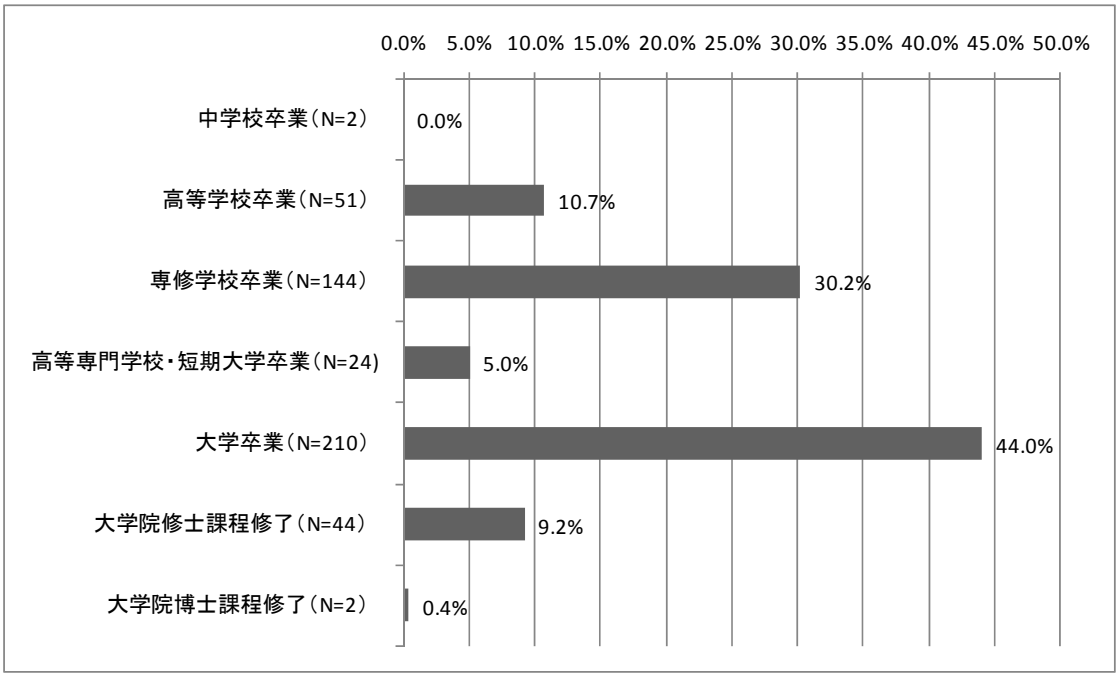


図 5.3-02　回答者の最終学歴

(4)　学問系統

学問系統は、「理・工系」（32.9%）が最も多く、続いて「芸術・表現系」が 24.5%となっている（図 5.3-03）。

これを最終学歴別にみると、専修学校卒業は「表現・芸術系」（36.8%）、高等専門学校・短期大学卒業、大学卒業ならびに大学院修士課程修了は「理・工系」が最も多くなっている（それぞれ 50.0%、35.7%、65.9%）。

さらに、職種別にみると、プロデューサー、プログラマー、プランナーは「理・工系」が最も多い（それぞれ 32.1%、58.8%、23.6%）。それに対して、ディレクター、グラフィッカーは「芸術・表現系」が最も多くなっている（それぞれ 27.9%、67.0%）。

このように、理・工系および芸術・表現系の職業的レリバンス（学校教育の職業的意義）は、ゲーム開発者にとって高い結果を示している。

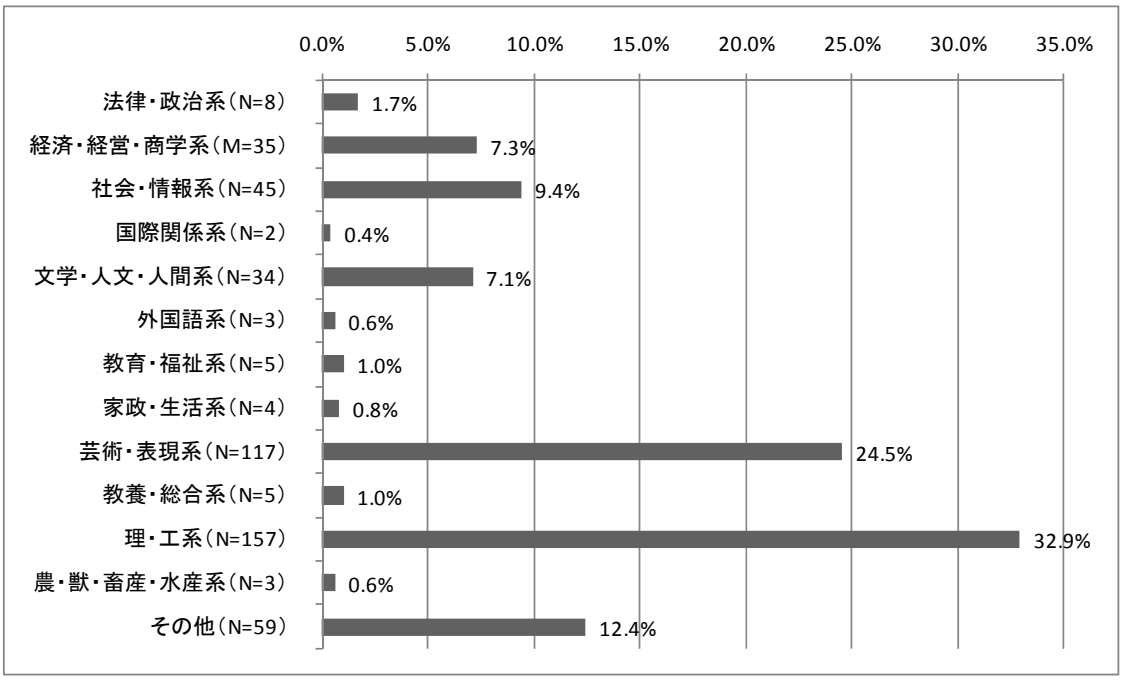


図 5.3-03　回答者の学問系統

(5)　配偶関係

配偶関係は、「未婚」が 58.7%、「既婚」が 40.7%、「離死別」が 0.6%となっている。

これを性別にみると、既婚率は男性が 42.1%、女性が 31.1%となっており、女性のほうが 11.0 ポイント低い。

さらに、年齢別にみると、既婚率は 20 代（9.0%）、30 代（47.8%）、40 代（67.9%）、50 代（62.5%）となっており、40 代の既婚率が最も高い。

(6)　子どもの有無

子どもの有無は、「あり」が 20.3%、「なし」が 79.7%となっている。

これを性別にみると、子どもがいる男性は 21.6%、子どもがいる女性は 11.5%となっており、とりわけ女性が子どもを育てながら働くことが困難な状況が窺える。

子どもがいる年齢層をみると、20 代は 2.2%、30 代は 20.7%、40 代では 46.4%、50 代では 37.5%であり、上記の配偶関係と比例している結果となっている。

(7)　主たる生計の維持者

主たる生計の維持者は、「本人」81.6%、「本人以外」9.4%、「その他」9.0%である。

主たる生計の維持者（本人）を性別にみると、男性は 85.1%、女性は 57.4%となっている。

(8)　現在の役職

現在の役職は、「一般」が 62.1%、「主任・係長クラス」が 16.6%、「課長クラス」が 7.5%、「部長クラス以上」が 13.8%となっている。

現在の役職を性別にみると、男性、女性とも「一般」が最も多い（それぞれ 58.7%、85.2%）が、女性のほうが 26.5 ポイント高い結果を示している。「主任・係長クラス」は、男性が 18.0%、女性が 6.6%、「課長クラス」は、男性が 8.7%、女性が 0%、「部長クラス以上」は男性が 14.7%、女性が 8.2%となっている。主任・係長クラス以上では、女性は男性よりも低い数値を示しており、いわゆるガラスの天井（性別などにより組織の中で昇進が制限された状態）が示唆される。

(9)　現在の職種

現在の職種は、「プロデューサー」が 11.1%、「ディレクター」が 9.0%、「プログラマー」が 27.5%、「グラフィッカー」が 21.6%、「プランナー」が 11.5%、「サウンド」が 3.4%、「デバッガー」が 1.3%、「ネットワーク・エンジニア」が 1.7%、「その他」が 13.0%となっている。「その他」には、アニメーター、シナリオライター、スクリプター、メカエンジニア、システムエンジニア、制作進行、マネージャー、経営者・役員、ローカライズコーディネーター、広報などが含まれている。

現在の職種を性別にみると、男性は「プログラマー」が 30.5%、女性は「グラフィッカー」（44.3%）が最も多い。このように、ゲーム開発職においては、男性＝プログラマー、女性＝グラフィッカーという緩やかな性別役割分業がみられる。

(10)　就労形態

就労形態は、「正社員」が最も多く 71.9%となっている。「契約社員」（7.8%）、「常用型派遣社員」（2.3%）、「登録型派遣社員」（4.2%）、「アルバイト・パート」（2.1%）のいわゆる非正規社員は少なくなっている。なお、「フリーランス・独立開発者」は 4.0%、「経営者」は 7.1%という結果になっている。

5.3.2　回答者の所属企業の概要

次に、回答者の所属企業の概要について、質問紙調査結果をもとに概観する。(1)資本金、(2)従業員数、(3)所在地域、(4)事業、(5)事業領域、(6)業績の推移、(7)従業員数の推移の順にみてみよう。

(1) 資本金

所属企業の資本金は、「50 億円以上」が最も多く 34.2%となっている。「1 億円未満」は 43.4%、「1 億円以上」は 56.6%である（図 5.3-04）。

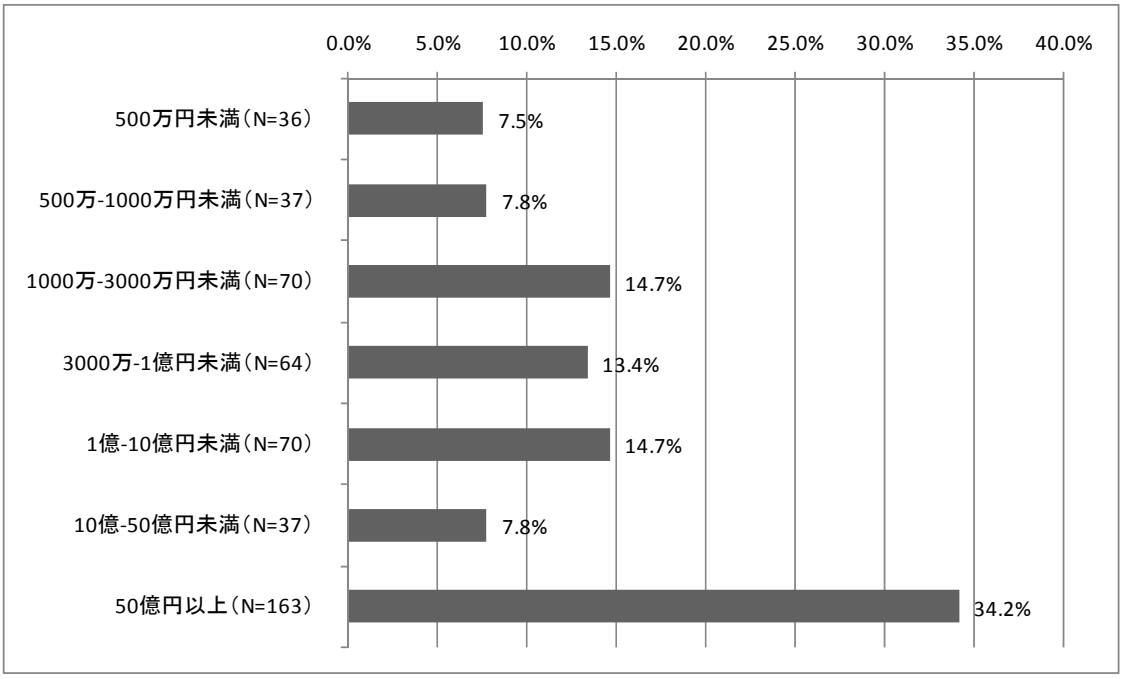


図 5.3-04　回答者の所属企業の資本金

(2) 従業員数

所属企業の従業員数は、「1001-3000 人」の層が最も多く 20.7%となっている。中小企業関連立法に基づき、従業員数 300 人以下を中小企業、300 人超を大企業とすると、「中小企業」は 54.1%、「大企業」は 45.9%の構成比となっている（図 5.3-05）。後述する企業規模別の統計は、以上の従業員数による整理に基づいて考察を行う。

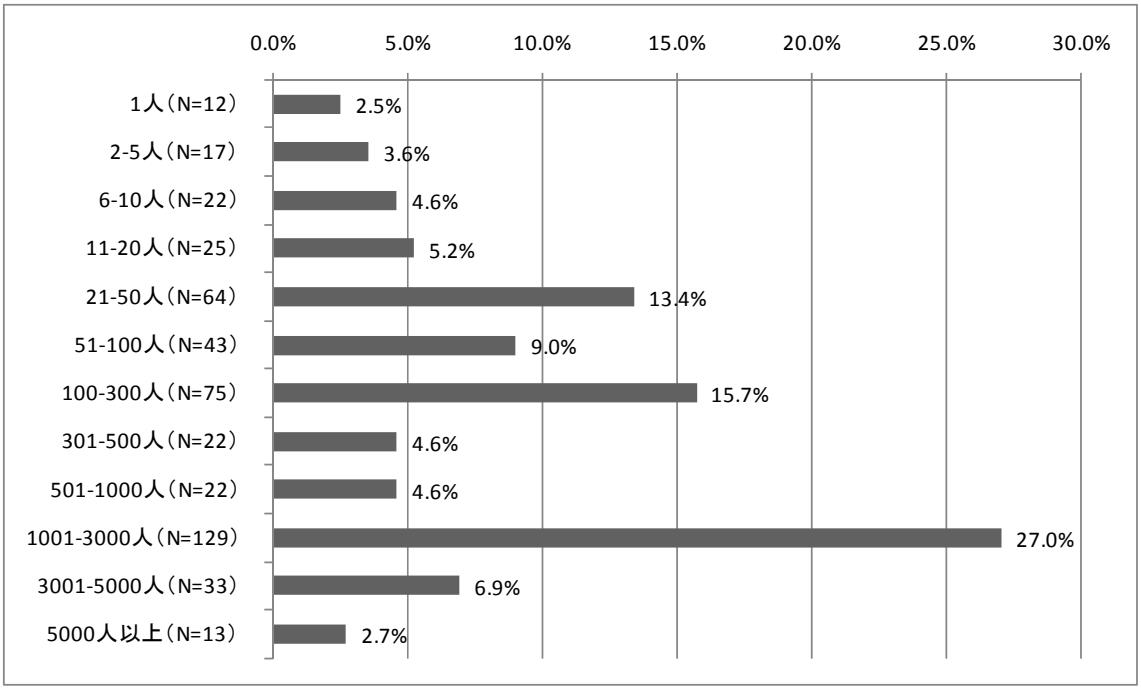


図 5.3-05　回答者の所属企業の従業員数

(3)　所属地域

所属企業の所在地域をみてみると、「関東地方」が最も多く 87.6%となっている。その他の地域は、「近畿地方」（7.8%）、「中部地方」（2.3%）、「九州地方」（1.0%）、「北海道地方」（0.8%）、「東北地方」ならびに「沖縄地方」は各 0.2%である。

(4)　事業

所属企業の事業をみてみると、「デベロッパー・パブリッシャー」が最も多く 52.2%、続いて「デベロッパー」（37.9%）、「パブリッシャー」（7.1%）、「その他」（2.7%）となっている。「その他」は、企画、シナリオ作成、音楽制作などが挙げられている。

(5)　事業領域

所属企業の事業領域をみてみると、「据置型ゲーム事業」が最も多く 61.2%、続いて「携帯型ゲーム事業」（47.4%）、「モバイル・コンテンツ事業」（24.5%）、「オンラインゲーム事業」（19.5%）となっている（図 5.3-06）。

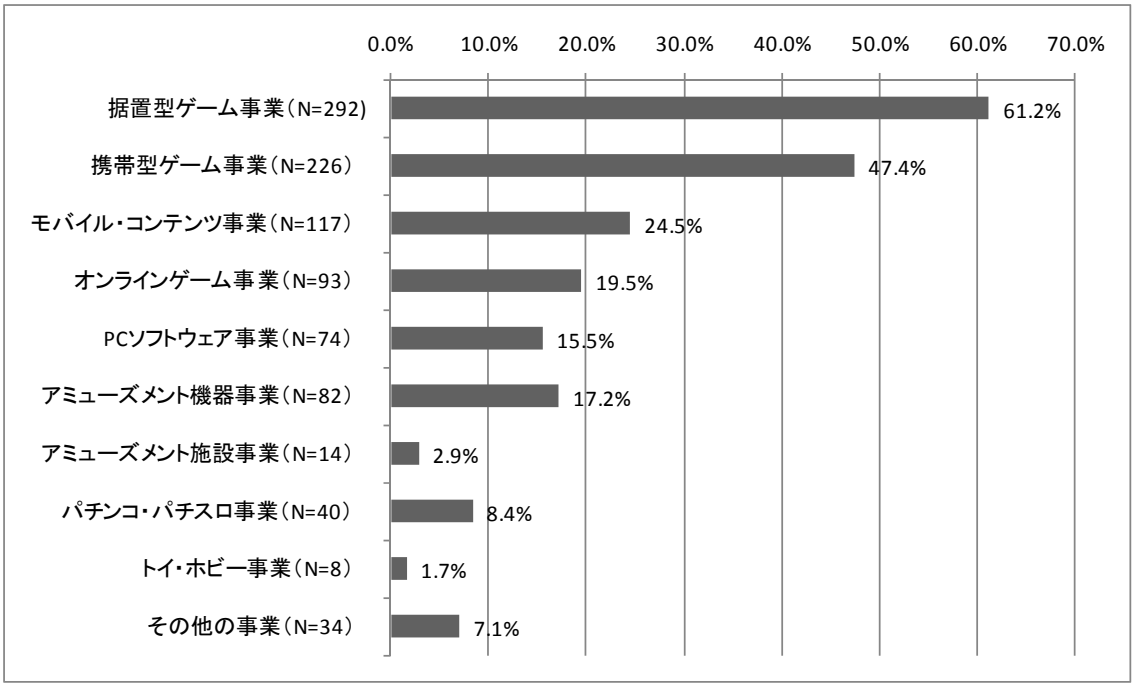


図 5.3-06　回答者の所属企業の事業領域

(6)　業績の推移

所属企業の業績の推移（対前年比）をみてみると、「減益」（55.6%）が最も多くなっている。「横ばい」は 28.1%、「増益」は 16.4%である。

企業規模別にみると、大企業の減益は 66.7%、中小企業の減益は 46.1%となっており、中小企業よりも大企業のほうが業績の低迷が見受けられる。

プラットフォーム別にみると、大企業では据置型ゲーム事業の低迷（20.5%）が顕著であるのに対して、中小企業では据置型ゲーム事業（15.1%）と携帯型ゲーム事業（14.7%）で減益が目立っている。

(7)　従業委員数の推移

所属企業の従業員数の推移（対前年比）をみてみると、「横ばい」が 37.3%と最も多く、続いて「減少」（33.5%）、「増加」（29.1%）となっている。

しかし、企業規模別で従業員数の推移をみてみると、大企業では「減少」が最も多く 45.2%、続いて「横ばい」が 37.4%、「増加」が 17.4%であるのに対して、中小企業では「増加」が最も多く 39.1%、続いて「横ばい」が 37.2%、「減少」が 23.6%となっている。したがって、大企業では企業業績に応じて人員削減を進めているのに対して、中小企業では企業業績に関わらず採用活動が活発であり、大企業に比べて開発者の採用コストが高いことが窺える。

5.4　開発者の勤続・給与

開発者の勤続年数はどのくらいの長さなのだろうか。また、開発者の給与はどの程度なのだろうか。開発者の勤続年数と給与について考察してみよう。

5.4.1　開発者の勤続年数

本調査では、現在のゲーム会社の勤続年数と、ゲーム産業の経験年数を尋ねている。それぞれ順にみてみよう。

(1)　現在のゲーム会社の勤続年数

まず、現在のゲーム会社の平均勤続年数は、6.59 年（SD=5.70）である。これを職種別にみると、サウンドが最も長く 10.3 年（SD=7.5）、続いてプロデューサーが 9.2 年（SD=7.0）、ネットワーク・エンジニアが 8.4 年（SD=8.4）、ディレクターが 6.7 年（SD=5.4）、プログラマーが 6.5 年（SD=5.0）、グラフィッカーが 4.6 年（SD=4.0）、プランナーが 4.4 年（SD=3.7）、デバッガーが 2.0 年（SD=0.9）となっている（表 5.4-01）。サウンドとプロデューサーの勤続年数が突出して長くなっている理由は、ゲーム会社での勤続年数が長い部長クラス以上の者が前者は 13.0%、後者は 10.2%と多く存在しているためである。

表 5.4-01　現在の職種×勤続年数

		ゲーム会社の勤続年数		
		度数	平均値	標準偏差
現在の職種	プロデューサー	53	9.2	7.0
	ディレクター	43	6.7	5.4
	プログラマー	131	6.5	5.0
	グラフィッカー	103	4.6	4.0
	プランナー	55	4.4	3.7
	サウンド	16	10.3	7.5
	デバッガー	6	2.0	.9
	ネットワーク・エンジニア	8	8.4	8.4
	その他	62	8.9	6.9

(2)　ゲーム産業の経験年数

次に、ゲーム産業の平均経験年数は、10.25 年（SD=7.72）である。これを職種別にみると、プロデューサーが最も長く 14.0 年（SD=6.4）、続いてサウンドが 13.6 年（SD=6.9）、

ディレクターが 11.5 年（SD=5.3）、ネットワーク・エンジニアが 6.7 年（SD=5.4）、グラフィッカーが 9.6 年（SD=11.6）、プログラマーが 8.7 年（SD=5.2）、プランナーが 8.0 年（SD=5.3）、デバッガーが 4.3 年（SD=3.7）となっている（表 5.4-02）。（社内）プロデューサーとディレクターは、協働開発者や他部門との連携に基づいてプロジェクトを束ねる役割を有しており、組織における信頼性（関係特殊的スキル）がとくに求められる職種である。したがって、（社内）プロデューサーとディレクターのゲーム産業の経験年数が他の職種と比べて長くなるのは、日本においては必然の結果である。

表 5.4-02　現在の職種×ゲーム産業経験年数

		ゲーム産業の経験年数		
		度数	平均値	標準偏差
現在の職種	プロデューサー	53	14.0	6.4
	ディレクター	43	11.5	5.3
	プログラマー	131	8.7	5.2
	グラフィッカー	103	9.6	11.6
	プランナー	55	8.0	5.3
	サウンド	16	13.6	6.9
	デバッガー	6	4.3	3.7
	ネットワーク・エンジニア	8	9.8	7.9
	その他	62	12.2	6.8

5.4.2　開発者の給与

(1)　平均年収

「平均年収」は 5,184,995 円（SD=2,817,970）である。年収の最小値は 1,000,000 円、最大値は 30,000,000 円、中央値は 4,800,000 円となっている。

国税庁長官官房企画課（2009）『平成 20 年分民間給与実態統計調査－調査結果報告－』[9]によると、平均年収は 4,296,000 円（平均年齢 44.4 歳、平均勤続年数 11.5 年）となっており、一般的な水準と比べると、ゲーム産業の平均年収（5,184,995 円）は約 89 万円高く、平均年齢（33.79 歳）は約 11 年若く、平均勤続年数（6.59 年）は約 5 年短い。

階層別にみると、「300 万円以上 400 万円未満」の層が最も多く 21.0%、続いて「400 万円以上 500 万円未満」18.6%、「500 万円以上 600 万円未満」15.9%という結果となっている（図 5.4-01）。

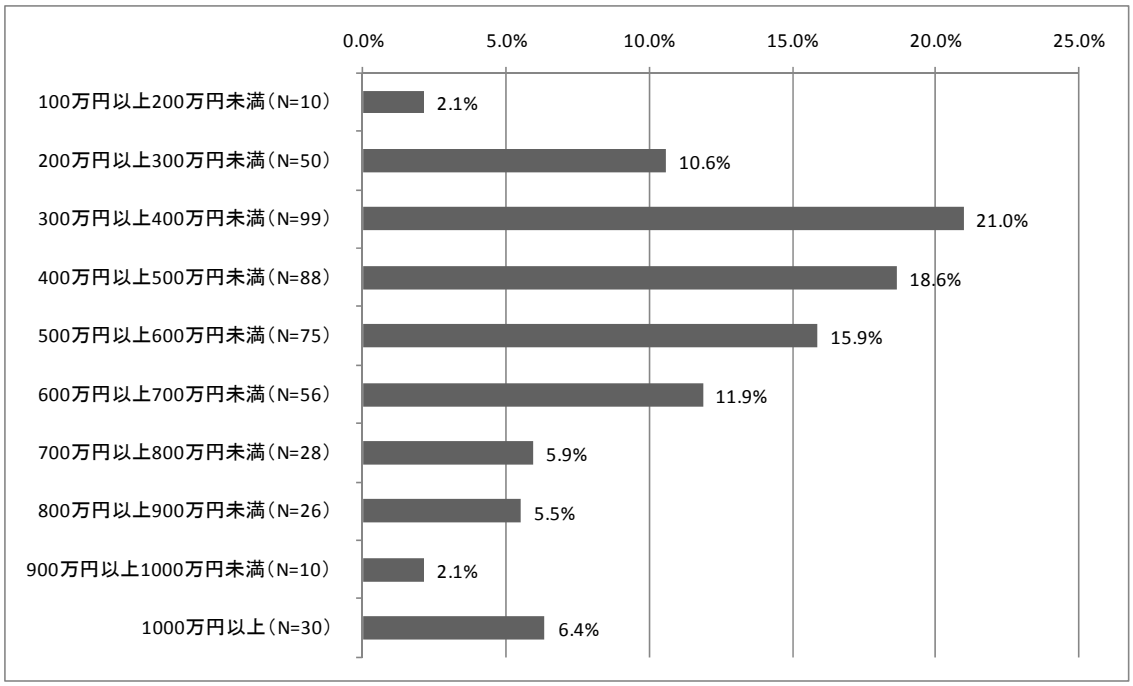


図 5.4-01　回答者の年収

(2)　最終学歴別年収

最終学歴別にみると、「中学校卒業」は 2,300,000 円（SD=1,838,478）、「高等学校卒業」は 4,751,765 円（SD=2,212,764）、「専修学校卒業」は 4,298,634 円（SD=1,895,361）、「高等専門学校・短期大学卒業」は 5,219,335 円（SD=2,359,125）、「大学卒業」は 5,770,033 円（SD=3,435,937）、「大学院修士課程修了」は 5,878,837 円（SD=2,030,387）、「大学院博士課程修了」は 5,600,000 円（SD=1,979,899）となっている。

しかし、最終学歴と年収との関係を考察する上で、年齢とゲーム会社の勤続年数を考慮する必要がある。平均年齢と平均勤続年数をみると、「中学校卒業」は 32.0 歳、3.0 年、「高等学校卒業」は 35.1 歳、5.7 年、「専修学校卒業」は 31.3 歳、4.5 年、「高等専門学校・短期大学卒業」は 34.4 歳、8.1 年、「大学卒業」は 34.8 歳、7.7 年、「大学院修士課程修了」は 35.3 歳、8.7 年、「大学院博士課程修了」は 38.0 歳、4.5 年である。年齢と年収（r=.587、p＜.01）、勤続年数と年収（r=.535、p＜.01）の間には有意な正の相関がみられる。

(3)　年齢別年収

年齢別にみると、「20 代」は 3,350,575 円（SD=1,046,575）、「30 代」は 5,152,578 円（SD=1,864,308）、「40 代」は 7,786,145 円（SD=4,178,993）、「50 代」は 9,475,000 円（SD=4,620,374）となっており、年齢が上がるとともに平均年収が上昇する、年功賃金体系が示唆される。20 代と 30 代との平均年収の差は約 180 万円、40 代と 50 代との平均年

収の差は約169万円であるが、30代と40代との平均年収の差は約263万円となっており、賃金カーブの勾配が極めて高い。

(4) 勤続年数別年収

勤続年数別にみると、「5 年以下」は 4,326,455 円（SD=2,310,105）、「6～10 年」は 5,686,775 円（SD=2,495,036）、「11～15 年」は 6,843,654 円（SD=3,720,327）、「16～20 年」は 7,903,745 円（SD=4,620,374）となっており、やはり勤続年数が長くなるほど平均年収が上昇している。しかし、年齢別の平均年収と比べると賃金カーブの勾配はそれほど高くない。

(5) 職種別年収

職種別にみると、「プロデューサー」が最も高く 6,925,000 円（SD=3,558,799）、続いて「ディレクター」が 5,636,279 円（SD=1,977,377）、「サウンド」が 5,590,625 円（SD=2,257,005）、「ネットワーク・エンジニア」が 5,225,000 円（SD=2,034,523）、「プログラマー」が 4,641,390 円（SD=1,796,069）、「グラフィッカー」が 4,238,588 円（SD=1,699,936）、「プランナー」が 4,096,340 円（SD=1,341,286）、「デバッガー」が 2,583,333 円（SD=664,580）となっている。ただし、ネットワーク・エンジニアとデバッガーは、有効サンプル数（度数）が少ない点に留意が必要である。「その他」が高い数値（7,186,230 円、SD=4,758,974）を示しているのは、役員や経営者がここに多く含まれているからである。

米国では、United Business Media 社がゲーム開発者の給与調査を 2004 年秋から毎年行っている。『Game Career Guide 2009 Fall』（pp.31-38）[2]によれば、勤続年数が 3 年以下のゲーム開発者の平均年収は、プログラマー（Programmers）が最も高く 64,500U.S.$、続いてサウンド（Audio Developers）が 53,269U.S.$、グラフィッカー（Artists and Animators）が 47,692U.S.$、ゲームデザイナー（Game Designers）が 45,896U.S.$、プロデューサー（Producers）が 45,279U.S.$、品質管理（QA Testers）が 27,894U.S.$となっている。

日本のゲーム産業は、後述するように、年俸制による成果主義制度を導入しつつも、能力主義と職務主義の折衷・混合型の人事管理制度となっている。例えば、部長、課長、主任、一般などの職位と、プロデューサー、ディレクター、プログラマー、グラフィッカー、プランナーなどの職種、マネジメント（管理職）とスペシャリスト（専門職）などの資格などが混合しているため、米国のような職務等級制度での運用が難しくなっている。そのため、単純に米国調査と比較することはできないが、米国ではプログラマーが、日本では

プロデューサーの平均年収が高くなっている点は興味深い。日本のプロデューサーは勤続年数が長いことが影響していると考えられる。

表 5.4-03　平均年収（クロス表）

		現在の年収		
		度数	平均値	標準偏差
最終学歴	中学校卒業	2	2,300,000	1,838,478
	高等学校卒業	51	4,751,765	2,212,764
	専修学校卒業	144	4,298,634	1,895,361
	高専・短期大学卒業	24	5,219,335	2,359,125
	大学卒業	210	5,770,033	3,435,937
	大学院修士課程修了	44	5,878,837	2,030,387
	大学院博士課程修了	2	5,600,000	1,979,899
年齢	20代	134	3,350,575	1,046,575
	30代	251	5,152,578	1,864,308
	40代	84	7,786,145	4,178,993
	50代	8	9,475,000	4,620,374
勤続年数	5年以下	282	4,326,455	2,310,105
	6-10年	107	5,686,775	2,495,036
	11-15年	52	6,843,654	3,720,327
	16-20年	36	7,903,745	2,732,180
現在の職種	プロデューサー	53	6,925,000	3,558,799
	ディレクター	43	5,636,279	1,977,377
	プログラマー	131	4,641,390	1,796,069
	グラフィッカー	103	4,238,588	1,699,936
	プランナー	55	4,096,340	1,341,286
	サウンド	16	5,590,625	2,257,005
	デバッガー	6	2,583,333	664,580
	ネットワーク・エンジニア	8	5,225,000	2,034,523
	その他	62	7,186,230	4,758,974

5.5　開発者の求職・転職行動

開発者はどのようにして仕事を得ているのであろうか。求職行動・転職行動から考察してみよう。

5.5.1　開発者の求職行動

(1)　ゲーム産業への参入理由

ゲーム産業への参入理由について尋ねた結果をまとめたのが図 5.5-01 である。これをみると、「ゲームが好きだから」という理由が最も多く（74.6%）、続いて「自分の能力・個性を活かせるから」（57.7%）、「仕事が楽しそうだから」（52.0%）となっている。また、続いて「コンピューターが好きだから」（35.4%）という結果となっている。

職種別にみると、特徴的な点がいくつか浮かび上がってくる（表 5.5-01）。プロデューサーは、「将来性を感じたから」という理由が他の職種と比べて最も多くなっている（43.4%）。プログラマーは、「ゲームが好きだから」（84.7%）、「コンピューターが好きだから」（65.6%）が他の職種と比べて突出している。グラフィッカーは、「自分の能力・個性を活かせるから」（70.9%）と「仕事が楽しそうだから」（59.2%）が他の職種と比べて最も多い。

ゲーム産業では、さまざまなメディアにおいて、「ゲームが好きだから」という理由だけでは参入が難しいと言われている。しかし、ゲームが好きということを根底としつつも、それに追加して「コンピューターが好き」、「自分の能力・個性を活かす」、「将来展望」などの動機が求められているように思われる。

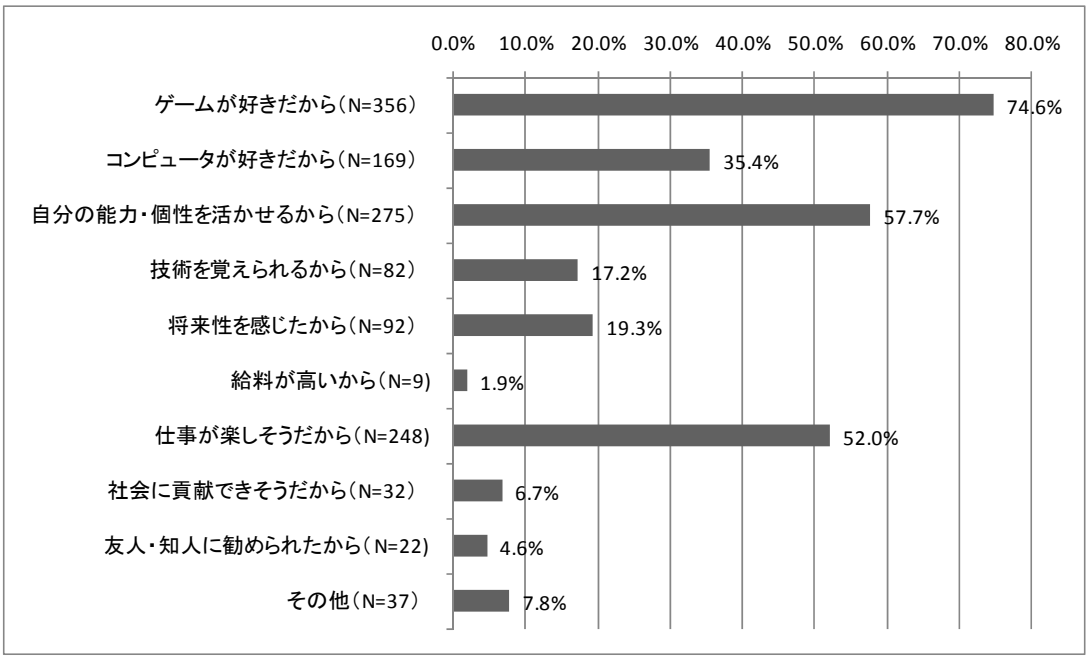


図 5.5-01　ゲーム業界への参入理由

表 5.5-01　ゲーム産業への参入理由×現在の職種

	プロデューサー		ディレクター		プログラマー		グラフィッカー		プランナー	
	度数	%	度数	%	度数	%	度数	%	度数	%
ゲームが好きだから	39	73.6%	31	72.1%	111	84.7%	71	68.9%	41	74.5%
コンピューターが好きだから	10	18.9%	11	25.6%	86	65.6%	19	18.4%	10	18.2%
自分の能力・個性を活かせるから	24	45.3%	26	60.5%	67	51.1%	73	70.9%	31	56.4%
技術を覚えられるから	2	3.8%	5	11.6%	36	27.5%	22	21.4%	2	3.6%
将来性を感じたから	23	43.4%	12	27.9%	14	10.7%	8	7.8%	12	21.8%
給料が高いから	0	0.0%	1	2.3%	1	0.8%	3	2.9%	1	1.8%
仕事が楽しそうだから	21	39.6%	21	48.8%	71	54.2%	61	59.2%	30	54.5%
社会に貢献できそうだから	5	9.4%	2	4.7%	9	6.9%	4	3.9%	6	10.9%
友人・知人に勧められたから	1	1.9%	2	4.7%	4	3.1%	6	5.8%	3	5.5%
その他	4	7.5%	3	7.0%	10	7.6%	6	5.8%	4	7.3%

(2)　現在のゲーム会社への就職理由

現在のゲーム会社への就職理由について尋ねた結果をまとめたのが図 5.5-02 である。これをみると、「自分の能力・個性を活かせるから」（54.1%）が最も多く、「仕事が面白いから」（34.2%）、「技術を覚えられるから」（26.4%）となっている。

現在のゲーム会社への就職理由を職種別にみてみよう（表 5.5-02）。プロデューサーは、「経営者に魅力を感じたから」（15.1%）が他の職種と比べて最も多い。ディレクターは、「自分の能力・個性を活かせるから」（65.1%）と「会社の将来性を考えて」（30.2%）が多い結果となっている。プログラマーは、「技術を覚えられるから」が 35.9%、「一流会社だから」が 23.7%、「地理的条件が良いから」が 22.1%となっており、他の職種と比べて特徴的な点として指摘できる。プランナーは、「仕事が面白いから」が 38.2%、「実力主義の会社だから」が 14.5%と顕著である一方で、「どこも行くところがなく、やむなく」（21.8%）という理由も多くなっている。

現在のゲーム会社への就職理由を企業規模別にみてみると、5 ポイント以上の格差がある項目は、「一流会社だから」（大企業 38.8%、中小企業 3.1%）、「経営者に魅力を感じたから」（同 5.0%、13.2%）、「どこも行くところがなく、やむなく」（同 7.8%、14.3%）、「技術を覚えられるから」（同 29.7%、23.6%）、「福利厚生が充実しているから」（同 8.7%、2.7%）、「給料が高いから」（同 1.4%、6.6%）という結果となっている。

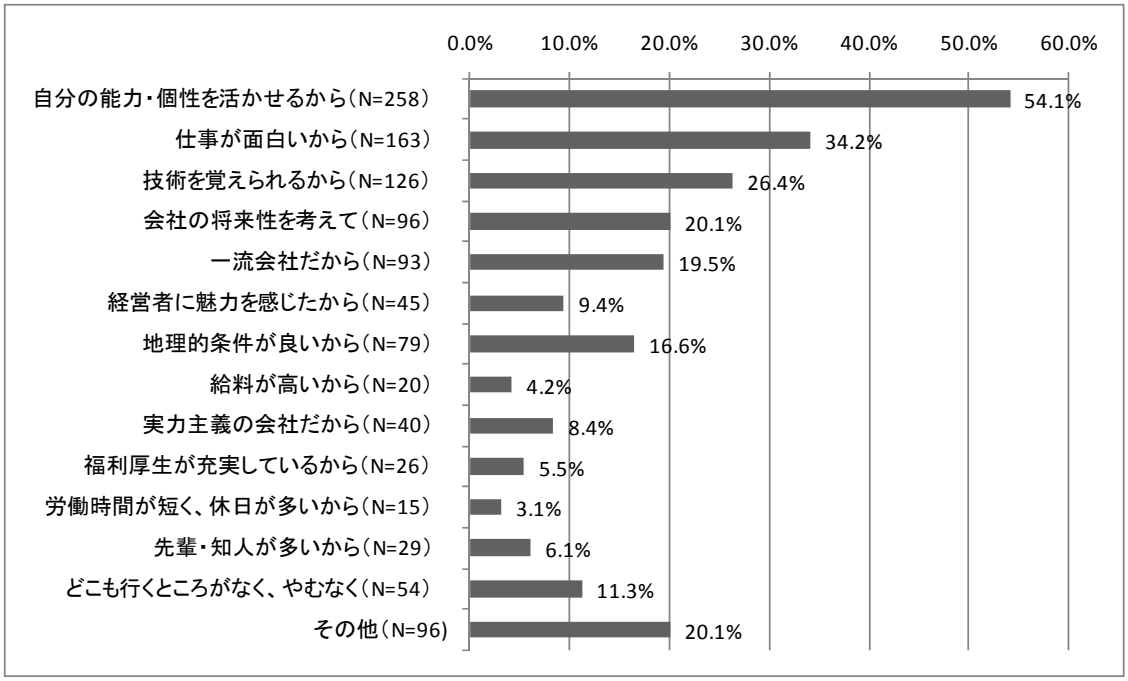


図 5.5-02　現在のゲーム会社への就職理由

表 5.5-02　現在のゲーム会社への就職理由×現在の職種

	プロデューサー		ディレクター		プログラマー		グラフィッカー		プランナー	
	度数	%	度数	%	度数	%	度数	%	度数	%
自分の能力・個性を活かせるから	28	52.8%	28	65.1%	58	44.3%	57	55.3%	30	54.5%
仕事が面白いから	18	34.0%	11	25.6%	49	37.4%	28	27.2%	21	38.2%
技術を覚えられるから	4	7.5%	7	16.3%	47	35.9%	36	35.0%	10	18.2%
会社の将来性を考えて	15	28.3%	13	30.2%	25	19.1%	17	16.5%	8	14.5%
一流会社だから	9	17.0%	8	18.6%	31	23.7%	24	23.3%	12	21.8%
経営者に魅力を感じたから	8	15.1%	5	11.6%	14	10.7%	4	3.9%	6	10.9%
地理的条件が良いから	7	13.2%	4	9.3%	29	22.1%	18	17.5%	7	12.7%
給料が高いから	0	0.0%	4	9.3%	5	3.8%	6	5.8%	4	7.3%
実力主義の会社だから	6	11.3%	4	9.3%	10	7.6%	4	3.9%	8	14.5%
福利厚生が充実しているから	2	3.8%	0	0.0%	8	6.1%	10	9.7%	3	5.5%
労働時間が短く、休日が多いから	1	1.9%	0	0.0%	3	2.3%	6	5.8%	1	1.8%
先輩・知人が多いから	1	1.9%	3	7.0%	9	6.9%	8	7.8%	2	3.6%
どこも行くところがなく、やむなく	1	1.9%	2	4.7%	15	11.5%	16	15.5%	12	21.8%
その他	12	22.6%	10	23.3%	28	21.4%	14	13.6%	9	16.4%

(3)　現在のゲーム会社への就職経路

現在のゲーム会社への就職経路を示した結果が図 5.5-03 である。これをみると、「企業の求人情報」が 31.7%と最も多く、続いて「友人」(19.1%)、「民間の求人ウェブサイト」(13.8%)となっている。「その他」が 14.9%と多くなっているが、学校経由の就職（求人情報、教員や就職担当者からの紹介、会社説明会、制作発表会)、社長からの誘い、起業などが挙げられている。

現在のゲーム会社への就職経路を年齢別にみてみよう（表 5.5-03)。20 代は、「企業の求人情報」(38.1%)が最も多く、「民間の求人ウェブサイト」(25.4%)と続く。また、学校経由の就職も多いのが特徴である。30 代は、「企業の求人情報」が 31.1%と最も多くなっている。加えて、30 代、40 代、50 代と年齢が高くなるほど、「友人」を通じた就職が多くなっている。すなわち、年齢が高くなるほど人的ネットワークが転職に有利に働いており、開発者個々人の関係性のマネジメントが重要となっている。しかし、いずれの年代においても「企業の求人情報」は重要な就職経路であり、企業による適切な情報提供が求められている。

企業規模別にみてみると、とりわけ大企業において「企業の求人情報」が重要な就職経路となっている（大企業 50.2%、中小企業 15.9%)。一方、中小企業においては「起業」（同 0%、14.7%)、「友人」(同 14.2%、23.3%）などにみられるように人的ネットワークが重要な就職経路となっている。

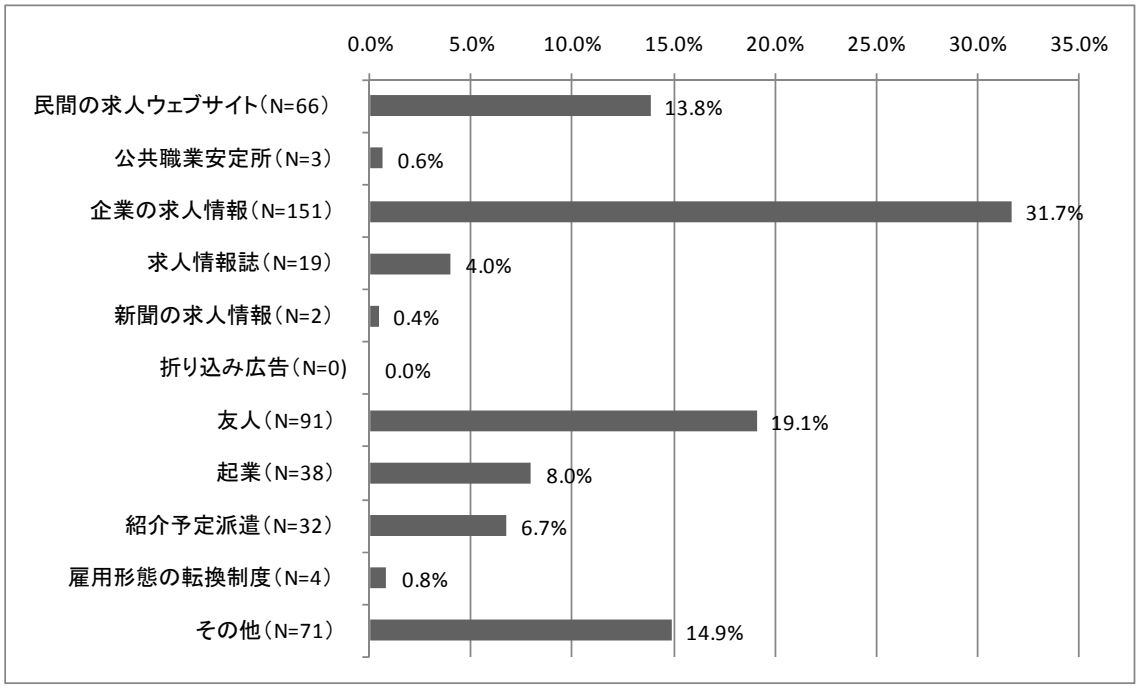


図 5.5-03　現在のゲーム会社への就職経路

表 5.5-03　現在のゲーム産業への就職経路×年齢

	年齢							
	20代		30代		40代		50代	
	度数	%	度数	%	度数	%	度数	%
民間の求人ウェブサイト	34	25.4%	24	9.6%	8	9.5%	0	0.0%
公共職業安定所	0	0.0%	3	1.2%	0	0.0%	0	0.0%
企業の求人情報	51	38.1%	78	31.1%	20	23.8%	2	25.0%
求人情報誌	0	0.0%	10	4.0%	7	8.3%	2	25.0%
新聞の求人情報	0	0.0%	2	0.8%	0	0.0%	0	0.0%
折り込み広告	0	0.0%	0	0.0%	0	0.0%	0	0.0%
友人	14	10.4%	57	22.7%	18	21.4%	2	25.0%
起業	2	1.5%	22	8.8%	13	15.5%	1	12.5%
紹介予定派遣	8	6.0%	23	9.2%	1	1.2%	0	0.0%
雇用形態の転換制度	1	0.7%	2	0.8%	1	1.2%	0	0.0%
その他	24	17.9%	30	12.0%	16	19.0%	1	12.5%

(4)　仕事の継続理由

現在の仕事を継続している理由を示した結果が図 5.5-04 である。最も多いのは「お金を得るため」（70.6%）、続いて「この仕事が楽しいから」と「現在の仕事が好きだから」がいずれも 54.1%、「自分の才能や能力を発揮するため」（53.0%）となっている。

日本アニメーター・演出協会（JAniCA）が 2008 年度にアニメーターを対象に行った調査[10]によると、「現在の仕事が好きだから」（65%）が最も多く、続いて「お金を得るため」（55%）、「この仕事が楽しいから」（53%）という結果になっている。

ゲーム開発者とアニメーターが仕事を継続している理由には、上記のとおり「仕事が好き・楽しい」、「お金を得るため」という点で共通性がみられるが、ゲーム開発者は自分の才能や能力を発揮したいという強い自己実現欲求を持っている点でアニメーターと異なる。

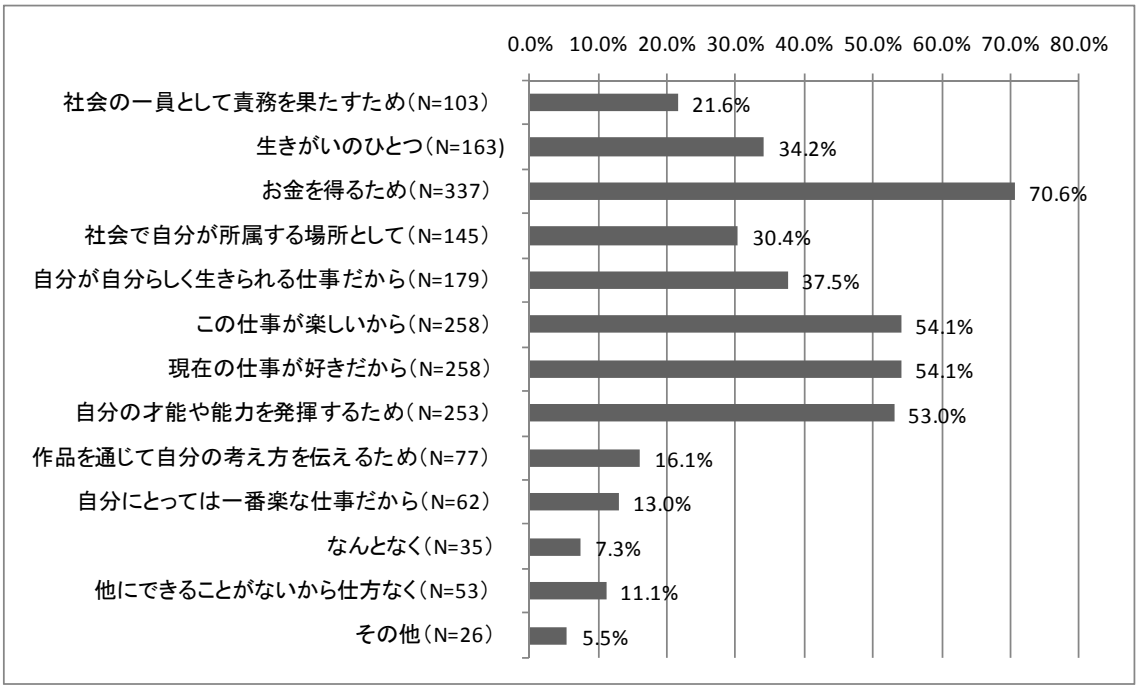


図 5.5-04　仕事の継続理由

5.5.2　開発者の転職行動

(1)　転職経験

転職経験が「ある」者は 59.1%に対して、転職経験が「ない」者は 40.9%となっている。約 6 割の開発者が転職経験を持っている。

それでは、どのような人が転職をしているのだろうか。転職経験をクロス集計でみてみよう（表 5.5-04）。

年齢別にみると、30 代の転職経験が最も多く 59.9%に達している。

企業規模別では、大企業に属して転職経験を有する者は 34.4%に対して、中小企業に属して転職経験を有する者は 65.6%になっている。

職種別にみると、グラフィッカーの転職経験者が多く 25.5%を示している。グラフィッカーには、派遣労働者などの非正規労働者が多く含まれているためである。

表 5.5-04　転職経験（クロス表）

		転職経験			
		ある		ない	
		度数	%	度数	%
年齢	20代	54	19.1%	80	41.0%
	30代	169	59.9%	82	42.1%
	40代	52	18.4%	32	16.4%
	50代	7	2.5%	1	0.5%
企業規模	中小企業	185	65.6%	73	37.4%
	大企業	97	34.4%	122	62.6%
現在の職種	プロデューサー	33	11.7%	20	10.3%
	ディレクター	26	9.2%	17	8.7%
	プログラマー	60	21.3%	71	36.4%
	グラフィッカー	72	25.5%	31	15.9%
	プランナー	32	11.3%	23	11.8%
	サウンド	7	2.5%	9	4.6%
	デバッガー	5	1.8%	1	0.5%
	ネットワーク・エンジニア	4	1.4%	4	2.1%
	その他	43	15.2%	19	9.7%

ところで、転職経験と年収の関係はどのようになっているのであろうか。「転職経験あり」の場合は、「転職経験なし」の場合と比べて、約23万円年収が下がっている。転職経験と年収との間には、有意な負の相関関係がみられる（r=－.106、p＜.01）。したがって、日本のゲーム産業においては､転職をしないほうが経済的報酬に利得があると言えよう(表5.5-05)。

表 5.5-05　転職経験×現在の年収

		現在の年収		
		度数	平均値	標準偏差
転職経験	ある	282	5,089,989	2,592,810
	ない	195	5,323,545	3,119,232

(2) 転職回数

転職経験のある者のうち、転職回数を調べたところ、「1 回」が 32.7%、「2 回」が 26.0%、「3 回」が 15.3%、「4 回」が 7.8%、「5 回以上」が 18.1%という結果となった。転職回数は、「1 回」から「4 回」までは逓減していくが、「5 回以上」になると増加している。

それでは、どのような人がどの程度転職をしているのであろうか。表 5.5-06 に沿って具体的にみてみよう。

年齢別でみると、20 代では転職回数が逓減していくのに対して、30 代では転職回数が逓増していることが分かる。

企業規模別でみると、中小企業に所属している人の転職回数が顕著に多くなっている。

職種別でみると、グラフィッカーの転職回数の多さが目立っており、勤続年数の短さ（4.6 年）と符合する結果となっている。

表 5.5-06　転職回数（クロス表）

		転職回数									
		1回		2回		3回		4回		5回以上	
		度数	%	度数	%	度数	%	度数	%	度数	%
年齢	20代	28	30.4%	15	20.5%	6	14.0%	3	13.6%	2	3.9%
	30代	47	51.1%	43	58.9%	29	67.4%	13	59.1%	37	72.5%
	40代	13	14.1%	13	17.8%	8	18.6%	6	27.3%	11	21.6%
	50代	4	4.3%	2	2.7%	0	0.0%	0	0.0%	1	2.0%
企業規模	中小企業	51	55.4%	47	64.4%	33	76.7%	15	68.2%	39	76.5%
	大企業	41	44.6%	26	35.6%	10	23.3%	7	31.8%	12	23.5%
現在の職種	プロデューサー	7	7.6%	12	16.4%	5	11.6%	3	13.6%	6	11.8%
	ディレクター	7	7.6%	6	8.2%	3	7.0%	4	18.2%	5	9.8%
	プログラマー	29	31.5%	12	16.4%	6	14.0%	4	18.2%	9	17.6%
	グラフィッカー	18	19.6%	21	28.8%	14	32.6%	6	27.3%	13	25.5%
	プランナー	8	8.7%	9	12.3%	4	9.3%	3	13.6%	8	15.7%
	サウンド	2	2.2%	3	4.1%	0	0.0%	0	0.0%	2	3.9%
	デバッガー	1	1.1%	1	1.4%	2	4.7%	0	0.0%	1	2.0%
	ﾈｯﾄﾜｰｸ･ｴﾝｼﾞﾆｱ	2	2.2%	0	0.0%	1	2.3%	0	0.0%	1	2.0%
	その他	18	19.6%	9	12.3%	8	18.6%	2	9.1%	6	11.8%

先と同様に、転職が必ずしも年収の増加に結びつくとは限らない（表 5.5-07）。開発者は、金銭的報酬よりもむしろ、自分の才能や能力を発揮できる職場を求めて転職をしていることが窺える。

表 5.5-07　転職回数×現在の年収

	転職回数				
	1回	2回	3回	4回	5回以上
	平均値	平均値	平均値	平均値	平均値
現在の年収	5,451,957	5,136,096	4,546,667	5,193,182	4,649,440

(3)　前職の業界

本調査では、転職経験がある者を対象にして前職（直近の仕事）についても尋ねている。これをみると、前職が「ゲーム産業」の者は 66.0%に達している。約 7 割がゲーム産業内で転職をしており、ゲーム産業全体で開発者の育成が行われていることが示唆される。

一方、前職が「ゲーム産業以外」の者は 33.0%となっている。具体的には、インターネット業界、アニメ業界、放送業界、音楽業界、出版業界、パチンコ業界、自動車業界、教育機関、官公庁など幅広い業界から、ゲーム産業への参入が行われている。

(4)　転職理由

それではなぜゲーム産業に転職をしているのであろうか。図-5.5-05 によってみてみよう。「会社に魅力を感じなかったから」（36.2%）、「会社の将来性を考えて」（34.4%）というように、会社そのものに不満や不安を感じて転職をしている者と、「自分の能力・個性を活かせなかったから」（33.3%）というように、仕事に対する不満を感じて転職している者とに大別される。

「その他」（35.8%）の内訳をみると、「ゲームをつくりたかったから」、「ゲームをつくるのが夢だったから」、「ゲーム業界に入りたかったから」、「スキルアップのため」、「起業のため」という前向きな意欲が多い一方で、「プロジェクトの終了のため」、「契約期間満了のため」、「会社の倒産のため」、「体調不良のため」というように転職を余儀なくされている者もいる。

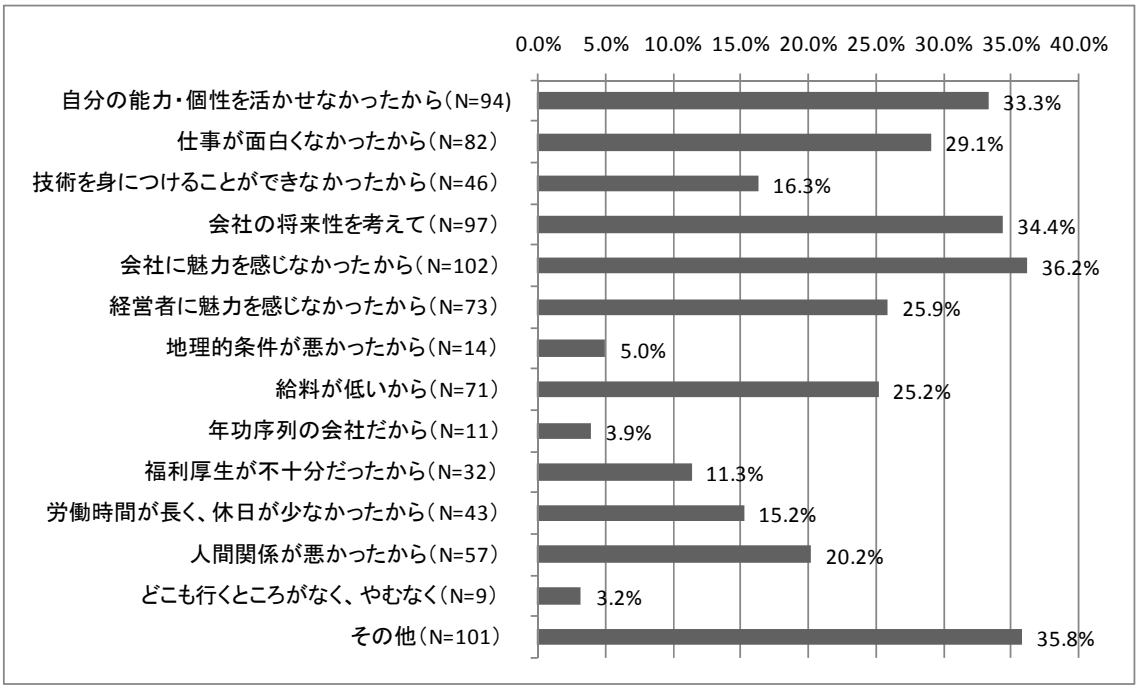


図 5.5-05 転職理由

5.6 開発者の働き方

開発者はどのような働き方をしているのであろうか。就労形態、勤務形態、労働時間、繁忙期から考察してみよう。

5.6.1 勤務形態

図 5.6-01 は、開発者の勤務形態について示したものである。開発者全体でみると、「裁量労働制」（42.1%）が最も多く、「始・就業時間が一定している通常業務」（30.0%）、「フレックスタイム勤務」（24.9%）と続く。

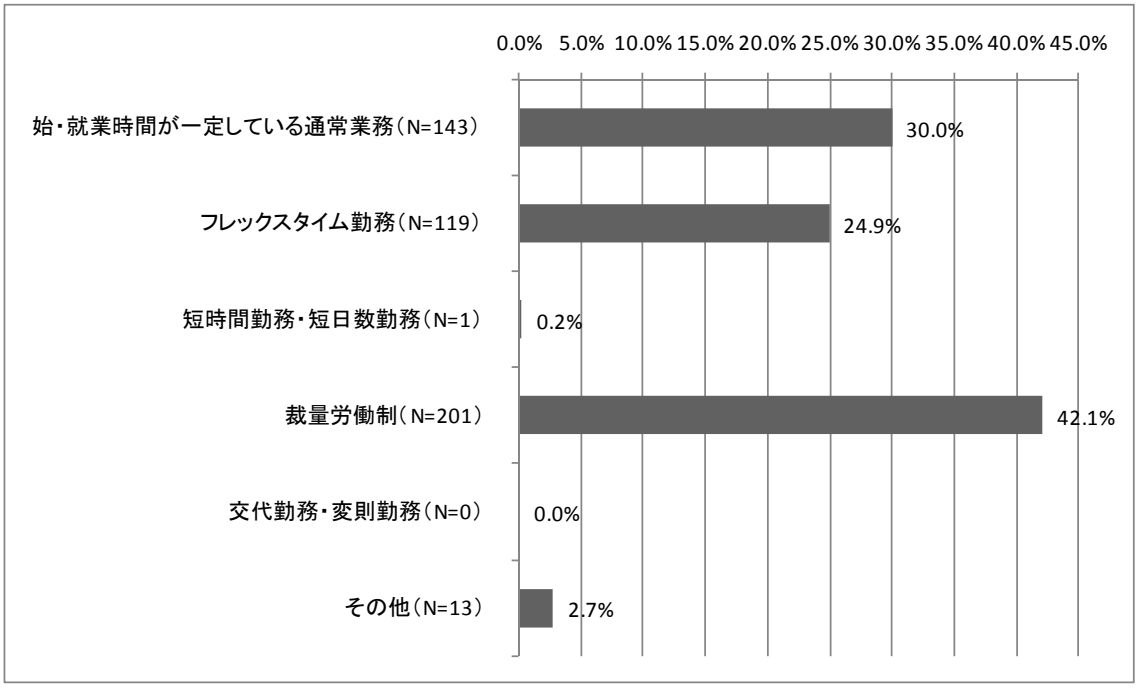


図 5.6-01　勤務形態

5.6.2　労働時間

開発者の労働時間はどのくらいの長さなのだろうか。本調査結果（日本調査（2009））と、米国調査（2004）とを比較してみてみよう（表 5.6-01）。日本では、「45-55 時間未満」（39.6%）が最も多く、次いで「35-45 時間未満」（31.7%）、「55 時間以上」（27.5%）となっている。米国でも日本と同様に「45-55 時間未満」（38.4%）が最も多くなっているが、「55 時間以上」の層が日本よりも 7.6 ポイント少ない結果となっている。しかし、プラットフォームの進化やマルチプラットフォーム化とともに開発期間の長期化、開発ラインの複線化などが生じているため、2004 年に行われた米国調査と単純比較はできない点に留意が必要である。

表 5.6-01　労働時間

	日本調査（2009）		米国調査（2004）	
	度数	%	度数	%
35 時間未満	6	1.3%	47	4.8%
35–45 時間未満	151	31.7%	353	35.8%
45–55 時間未満	189	39.6%	379	38.4%
55 時間以上	131	27.5%	196	19.9%

事業領域によって、労働時間には、変化がみられるのであろうか。労働時間を事業領域とのクロス表でみてみよう（表5.6-02）。

労働時間をみると、「55時間以上」働いている者のうち、「携帯型ゲーム事業」が最も多く（35.0%）、続いて「PCソフトウェア事業」（29.7%）、「モバイル・コンテンツ事業」（29.1%）、「据置型ゲーム事業」（26.4%）となっている。すなわち、開発期間の長期化が懸念となっている「据置型ゲーム事業」よりも「携帯型ゲーム事業」、「PCソフトウェア事業」、「モバイル・コンテンツ事業」のほうが労働時間は長くなっており、生産ライン数などが影響しているように思われる。

表 5.6-02　労働時間×事業領域

		据置型ゲーム事業		携帯型ゲーム事業		モバイル・コンテンツ事業		オンラインゲーム事業		PCソフトウェア事業	
		度数	%	度数	%	度数	%	度数	%	度数	%
労働時間	35時間未満	3	1.0%	2	0.9%	1	0.9%	0	0.0%	2	2.7%
	35-45時間未満	91	31.2%	54	23.9%	38	32.5%	27	29.0%	19	25.7%
	45-55時間未満	121	41.4%	91	40.3%	44	37.6%	44	47.3%	31	41.9%
	55時間以上	77	26.4%	79	35.0%	34	29.1%	22	23.7%	22	29.7%

表 5.6-03 によって、時間外の補償について尋ねた結果をみてみると、日本と米国のいずれも「時間外は補償されない」が最も多くなっている（日本 37.5%、米国 47.2%）。しかし、日本では「割増料金で時間外が支払われている」（25.4%）という相対立する項目が多くなっている。割増料金で時間外が支払われている勤務形態は、「就業時間が一定している通常の業務」（28.0%）と「フレックスタイム勤務」（37.0%）であり、「裁量労働制」のうち 46.8%が時間外は補償されないという回答を示している。一方、米国では「プロジェクトの終了時に休暇で補償されるが、時間外は計算されない」（19.6%）という項目が次いで多くなっている。

表 5.6-03　時間外の補償

	日本調査(2009)		米国調査(2004)	
	度数	%	度数	%
従業員の通常の 1 時間ごとの率で、時間外が支払われている	52	10.9%	23	2.3%
割増料金で時間外が支払われている	121	25.4%	28	2.8%
時間外が計算されて、時間外と同等の時間を休暇で補償されている	18	3.8%	24	2.4%
プロジェクトの終了時に休暇で補償されるが、時間外は計算されない	34	7.1%	193	19.6%
プロジェクト報奨金(一時金)が支払われている	5	1.0%	28	2.8%
時間外は補償されない	179	37.5%	465	47.2%
その他	68	14.3%	46	4.7%

日本では、週労働時間が「55 時間未満」までは給与が増加する傾向にあるが、「55 時間以上」になると給与は増加しない（表 5.6-04)。つまり、「55 時間未満」までは時間外手当が補償されるが、「55 時間以上」になると、時間外手当は支給されない傾向にあることが窺える。

表 5.6-04 現在の年収×労働時間

		現在の年収		
		度数	平均値	標準偏差
労働時間	35時間未満	6	4,316,667	1,929,162
	35-45時間未満	151	5,136,140	2,699,254
	45-55時間未満	189	5,325,158	3,035,726
	55時間以上	131	5,075,609	2,662,953

5.6.3　繁忙期

米国調査（2004）では、生活の質（Quality of Life: QOL）の観点から、繁忙期について丁寧に調査が行われている。表 5.6-05 によって、日本調査と比較してみよう。

まず、繁忙期の労働時間であるが、日本では「あらゆるマイルストーンの前」（53.2%）が最も多く、「最終的なβテストの時」（22.2%)、「毎月」（15.3%）と続く。この傾向は米国調査とあまり大きな変化はなく、日本でも同様の結果を示している。

次に、繁忙期の長さをみてみると、日本では「2 ヶ月以上」43.6%（「2 ヶ月～3 ヶ月未満」と「3 ヶ月以上」の合計）が最も多くなっているのに対して、米国では「1～2 週間未満」(29.1%) が最も多い結果となっている。米国調査は 2004 年に行われており、その後、技術革新に伴って、ゲーム開発規模が急激に拡大していったことを考慮に入れる必要があ

り、単純には比較ができないが、日本では繁忙期が長期化していることが窺える。

さらに、繁忙期の労働時間をみてみると、日本では「55～65 時間未満」（31.9%）が最も多いのに対して、米国では「65～80 時間未満」（35.5%）が最も多い結果を示しており、日本よりも米国のほうが繁忙期に長時間労働となっている。日本では繁忙期は慢性的になっているのに対して、米国では繁忙期は短期間ではあるがその間は長時間労働となっていることが指摘できる。

ジェイソン・デラロッカ（Jason Della Rocca）氏が CEDEC2007 や DiGRA2007 で、「生活の質の向上がゲーム産業の未来を決定する」と報告しているように、とりわけ適正な労働時間管理がゲーム開発の現場において求められているのではないだろうか。

表 5.6-05　繁忙期

		日本調査（2009）		米国調査（2004）	
		度数	%	度数	%
繁忙期の働き方	繁忙期はない	17	3.6%	24	2.4%
	最終的なβテストの時	106	22.2%	205	20.8%
	あらゆるマイルストーンの前	254	53.2%	569	57.7%
	毎月	73	15.3%	166	16.8%
	その他	27	5.7%	－	－
繁忙期の長さ	1 週間未満	39	8.2%	116	11.8%
	1-2 週間未満	44	9.2%	287	29.1%
	2 週-1 ヶ月未満	79	16.6%	228	23.1%
	1 ヶ月-2 ヶ月未満	107	22.4%	142	14.4%
	2 ヶ月-3 ヶ月未満	74	15.5%	185	18.8%
	3 ヶ月以上	134	28.1%		
繁忙期の労働時間	35 時間未満	2	0.4%	21	2.1%
	35-45 時間未満	23	4.8%	41	4.2%
	45-55 時間未満	73	15.3%	121	12.3%
	55-65 時間未満	152	31.9%	302	30.6%
	65-80 時間未満	134	28.1%	350	35.5%
	80 時間以上	93	19.5%	129	13.1%

表 5.6-06　繁忙期に対する認識

	日本調査(2009)		米国調査(2004)	
	度数	%	度数	%
何も問題がなくても誰もが繁忙期を経験する	222	46.5%	329	33.4%
すべての開発者は長時間働くが、管理はされない	151	31.7%	183	18.6%
管理者は、各自に与えられた仕事が終了しかどうかについて関与しない	66	13.8%	340	34.5%
個人や家族の理由のために繁忙期を拒否することは可能である	178	37.3%	312	31.6%
管理者は、ゲーム産業におけるビジネスでは繁忙期を一般的なものと考えている	312	65.4%	513	52.0%
管理者は、繁忙期を必要悪とみなして、従業員への影響を最小にしようとする	156	32.7%	387	39.2%
管理者は、週 40 時間労働のような繁忙期の廃止方針を適用する	24	5.0%	23	2.3%
その他	33	6.9%	43	4.4%

事業領域によって、繁忙期の労働時間には、変化がみられるのであろうか。表5.6-07によって、繁忙期の労働時間と事業領域のクロス表をみてみよう。いずれの事業領域においても最も多い結果を示しているのが「あらゆるマイルストーンの時」である。これを事業領域別にみると、「据置型ゲーム事業」が最も多く59.6%、次いで「オンラインゲーム事業」（51.6%）、「PCソフトウェア事業」（51.4%）、「携帯型ゲーム事業」（50.9%）、「モバイル・コンテンツ事業」（44.4%）となっている。

繁忙期の長さを事業領域別にみてみると、据置型ゲーム事業と携帯型ゲーム事業で「3ヶ月以上」が最も多い（それぞれ28.4%、29.2%）。しかし、モバイル・コンテンツ事業は「1ヶ月-2ヶ月未満」（25.6%）、オンラインゲーム事業は「1ヶ月-2ヶ月未満」と「2週-1ヶ月未満」が拮抗しており（21.5%）、PCソフトウェア事業は「2週-1ヶ月未満」（31.1%）が最も多くなっている。つまり、繁忙期の長さについては、据置型ゲーム事業と携帯型ゲーム事業は「1ヶ月-2ヶ月未満」、オンラインゲーム事業は「2週間-2ヶ月未満」、PCソフトウェア事業は「2週-1ヶ月未満」となっており、とりわけ据置型ゲーム事業と携帯型ゲーム事業で繁忙期が長くなっていることが把握できる。

繁忙期の労働時間を事業領域別にみてみると、携帯型ゲーム事業は「65-80 時間未満」（30.1%）が最も多い。それに対して、据置型ゲーム事業は「55-65 時間未満」（31.8%）、オンラインゲーム事業は 36.6%、PC ソフトウェア事業は 33.8%となっており、携帯型ゲーム事業と比べて繁忙期の労働時間は短い。

表 5.6-07　繁忙期の労働時間×事業領域

		据置型ゲーム事業		携帯型ゲーム事業		モバイル・コンテンツ事業		オンラインゲーム事業		PCソフトウェア事業	
		度数	%	度数	%	度数	%	度数	%	度数	%
繁忙期の働き方	繁忙期はない	10	3.4%	6	2.7%	6	5.1%	7	7.5%	4	5.4%
	最終的なβテストの時	57	19.5%	55	24.3%	37	31.6%	24	25.8%	21	28.4%
	あらゆるマイルストーンの前	174	59.6%	115	50.9%	52	44.4%	48	51.6%	38	51.4%
	毎月	33	11.3%	40	17.7%	16	13.7%	12	12.9%	10	13.5%
	その他	18	6.2%	10	4.4%	6	5.1%	2	2.2%	1	1.4%
繁忙期の長さ	1週間未満	21	7.2%	13	5.8%	12	10.3%	7	7.5%	4	5.4%
	1-2週間未満	20	6.8%	23	10.2%	17	14.5%	18	19.4%	13	17.6%
	2週-1ヶ月未満	54	18.5%	33	14.6%	24	20.5%	20	21.5%	23	31.1%
	1ヶ月-2ヶ月未満	67	22.9%	58	25.7%	30	25.6%	20	21.5%	15	20.3%
	2ヶ月-3ヶ月未満	47	16.1%	33	14.6%	10	8.5%	12	12.9%	8	10.8%
	3ヶ月以上	83	28.4%	66	29.2%	24	20.5%	16	17.2%	11	14.9%
繁忙期の労働時間	35時間未満	1	.3%	1	.4%	1	.9%	0	.0%	0	.0%
	35-45時間未満	11	3.8%	7	3.1%	6	5.1%	6	6.5%	4	5.4%
	45-55時間未満	41	14.0%	29	12.8%	21	17.9%	11	11.8%	8	10.8%
	55-65時間未満	93	31.8%	68	30.1%	34	29.1%	34	36.6%	25	33.8%
	65-80時間未満	87	29.8%	69	30.5%	31	26.5%	25	26.9%	23	31.1%
	80時間以上	59	20.2%	52	23.0%	24	20.5%	17	18.3%	14	18.9%

5.7　開発者の育成

開発者の育成はどの程度行われているのであろうか。人材育成、Off-JT、キャリア開発支援の実施状況から考察してみよう。

5.7.1　人材育成の実施状況

図 5.7-01 は、OJT（On the Job Training）、Off-JT（Off the Job Training）、キャリア開発支援の実施状況について示した図である。ゲーム開発者に対して行った本調査と、人事担当者に対して行った藤原（2007）[3]を比較している。

これをみると、OJT については、ゲーム開発者の約 50%が実施されていない（「あまり実施していない（28.7%）＋「全く実施していない」（21.2%））と感じているのに対して、人事担当者の 93.3%が実施している（「十分実施している」（33.3%）＋「ある程度実施し

ている」（60.0%））と回答している。

Off-JT についても、OJT と同様にゲーム開発者と人事担当者との間で大きな乖離がみられる。Off-JT は、ゲーム開発者の 63.3%が実施されていない（「あまり実施していない（37.3%）＋「全く実施していない」（26.0%））と感じているのに対して、人事担当者の86.6%が実施している（「十分実施している」（43.3%）＋「ある程度実施している」（43.3%））と回答している。

キャリア開発支援については、ゲーム開発者と人事担当者との間でそれほど乖離はみられない。ゲーム開発者は 74.8%、人事担当者は 63.3%が「実施していない」と回答している。

このような労使間における人材育成の認識乖離の他に、企業規模間の格差も存在している。OJT、Off-JT、キャリア開発支援のいずれにおいても、中小企業は大企業に比べて実施率が低い。「十分実施している」と「ある程度実施している」の合計をみると、OJT（大企業 65.7%、中小企業 36.9%）、Off-JT（同 55.2%、20.9%）、キャリア開発支援（同 39.3%、13.2%）という結果となっている。

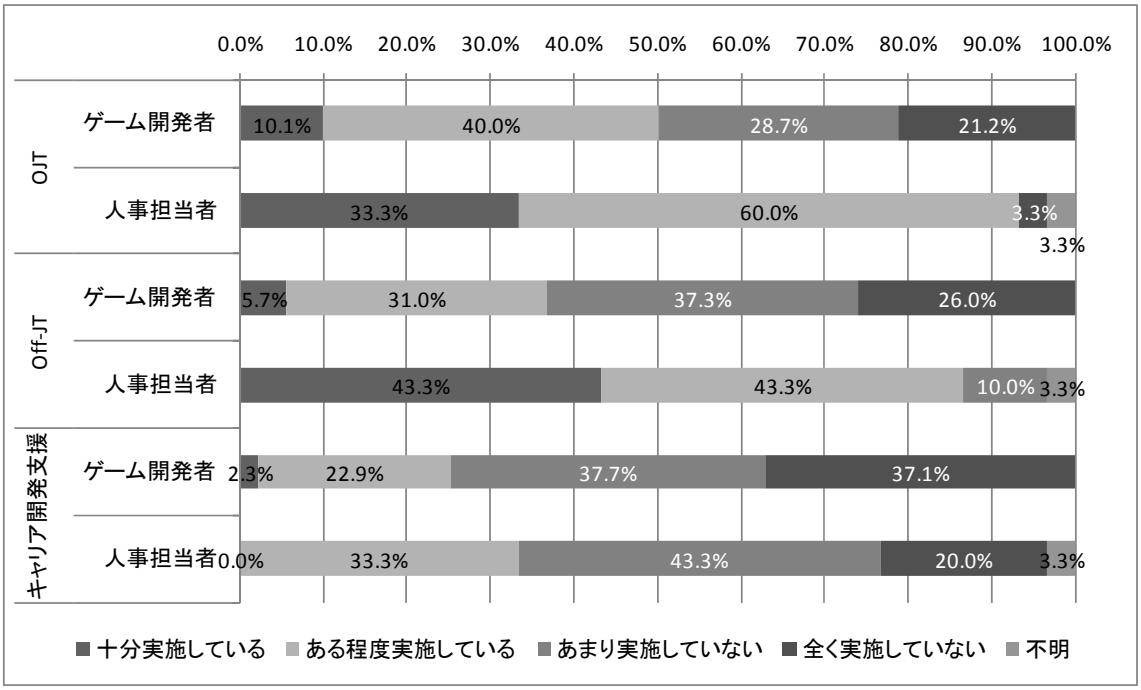


図 5.7-01　人材育成の実施状況

5.7.2　Off-JT の実施状況

Off-JT の具体的な取り組みについてみると、「新入社員研修」が最も多く（59.1%）、次いで「CEDEC（CESA Game Developers Conference）」（53.7%）となっている（図-5.7-02）。

CEDEC は、国内ゲーム産業を代表する人材交流や情報交換の場として、毎年、規模が拡大してきており、多くの期待が寄せられている。また、ゲーム開発者のうち 45.5%が職場で「週刊ファミ通（エンターブレイン社)」を閲覧し、ゲームやゲーム産業関連情報の収集を行っている。その他、「GDC（Game Developers Conference)」(31.4%) や「SIGGRAPH」（28.3%）にも約 3 割のゲーム開発者が参加している。

しかし、企業規模別にみると格差が顕著となっている。例えば、「とくに行っていない」をみると、大企業は 5.9%であるのに対して、中小企業は 33.3%となっている。また、「新入社員研修」は、大企業は 85.4%実施しているのに対して、中小企業は 36.8%の実施に留まっており、48.6 ポイントの格差が生じている。そして、「階層別研修」の実施状況は、大企業は 49.8%、中小企業は 3.9%という結果となっている。

海外の研修機関・研究会への参加でも企業規模間格差は著しい。「SIGGRAPH」の参加率は、大企業は 53.0%、中小企業は 7.4%、「GDC」の参加率は、大企業は 53.0%、中小企業は 13.2%となっている。

国内の研修機関・研究会への参加も同様の結果を示している。例えば、「CEDEC」の参加率は、大企業は 75.3%に対して、中小企業は 35.3%に留まっている。

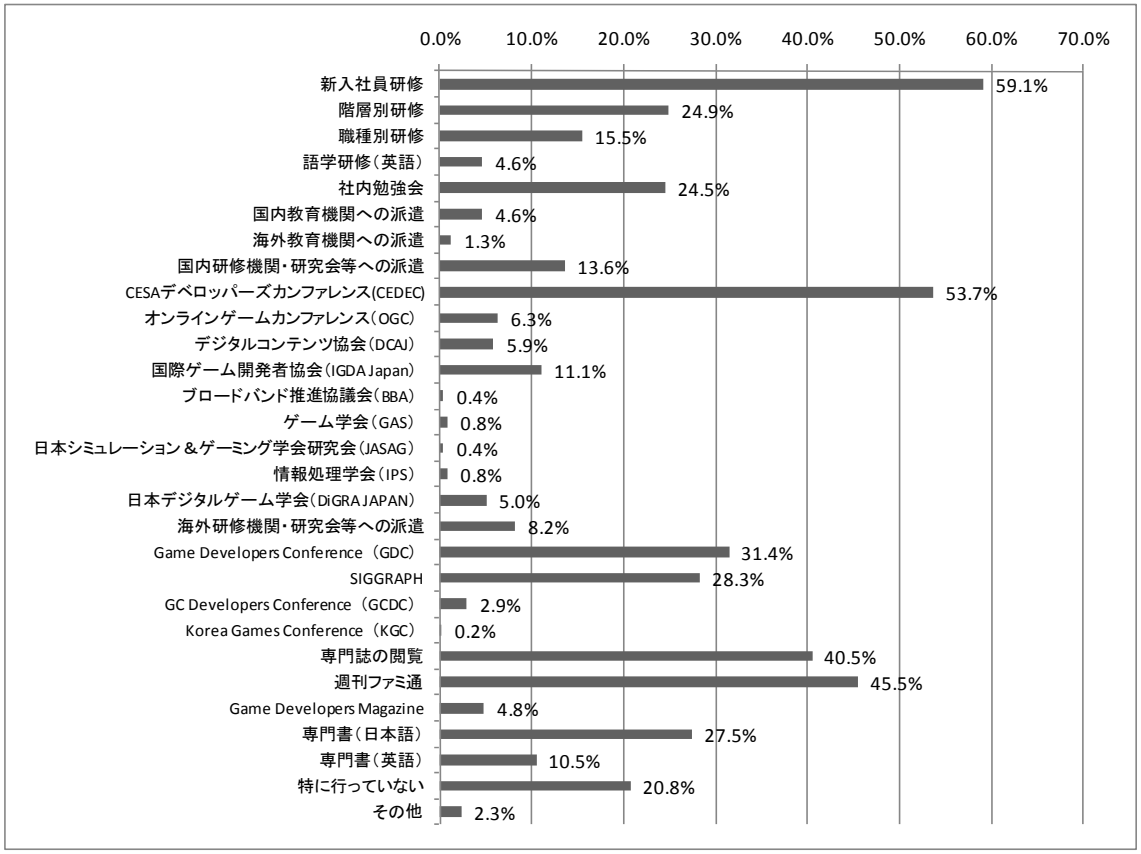


図 5.7-02　Off-JT の実施状況

5.7.3　キャリア開発支援の実施状況

図 5.7-03 は、ゲーム開発者のキャリア開発支援の実態について伺った結果をまとめた図である。これをみると、「特別な支援はしていない」（39.2%）が最も多くなっており、「自己負担で自己研鑽に努めている」が 28.3%に達している。しかし、「心身の健康管理に対する支援」は、34.8%実施されている状況となっている。専門業務型裁量労働制の導入において、健康・福祉を確保するための措置が義務づけられているためであろう。その他、「社内公募制度」（21.4%）が導入され、ゲーム開発者の自律的なキャリア形成が支援されている。

ところが、企業規模別にみると、キャリア開発支援も大きな格差が存在する。「特別な支援はしていない」は、大企業は 15.1%に留まるのに対して、中小企業は 59.7 にも達し、格差は 44.6 ポイントとなっている。また、「自己負担で自己研鑽に努めている」は、大企業は 21.5%であるのに対して、中小企業は 34.1%となっている。

このような課題を克服するために、とりわけ中小企業のゲーム会社は「中小企業雇用創出等能力開発助成金」などを利用して、開発者の教育訓練の充実を図っていく必要があるだろう。また、行政機関、業界団体などは、とりわけ中小企業に対する支援を積極的に行っていく必要があるだろう。

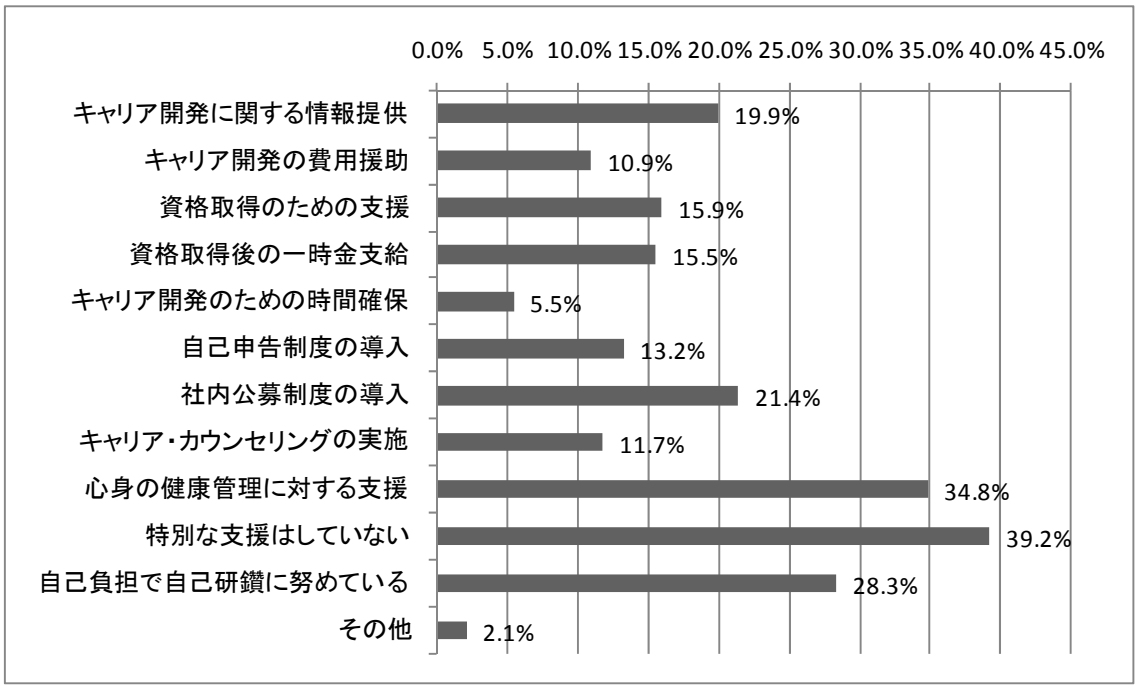


図 5.7-03　キャリア開発支援の実施状況

5.8　開発者に対する評価・処遇

開発者の評価・処遇はどのように行われているのであろうか。評価に対するゲーム開発者の認識、処遇制度から考察してみよう。

5.8.1　開発者に対する評価

図 5.8-01 は、評価に対するゲーム開発者の認識を示した図である。これをみると、「過去の成功や実績に基づいた意見や提案が評価される」68.1%（「大変そう思う」＋「そう思う」の合計）が最も多くなっている。これまでに蓄積された知識が組織において有利に働いていることが示唆される。

また、「給与制度は、成果や業績的な要素が大きい」が 59.3%を示しており、開発者の外発的動機づけを維持・向上させるための評価が行われている。しかし、「個人の業績に連動して、報奨金等の一時金で処遇されている」（28.8%）、「評価結果に基づく給与ダウンは頻繁に発生している」（22.0%）、「評価結果に基づく降格人事は頻繁に行われている」（19.3%）に留まっており、成果主義的な評価は一側面に過ぎない。

他方で、「個人の業績に連動して、昇進や昇格等のポストで処遇されている」者は 44.2%となっており、開発者の内発的動機づけを維持・向上させるための評価も行われている。

ところが、評価に対する公平性・公正性には改善の余地があるといえよう。具体的には「評価基準は開発者に公開されている」（30.2%）、「評価結果・評価理由は本人に十分開示されている」（38.1%）、「評価に関する苦情処理の仕組みがある」（20.5%）と低い結果を示していることから、評価プロセスや評価結果の透明性、納得性が担保される仕組みが必要である。つまり、手続的公正を担保することが、開発者のモチベーションとコミットメントにおいて重要である。手続的公正の原則を構築することにより、管理者と開発者の間の心理的距離は縮まるであろう。

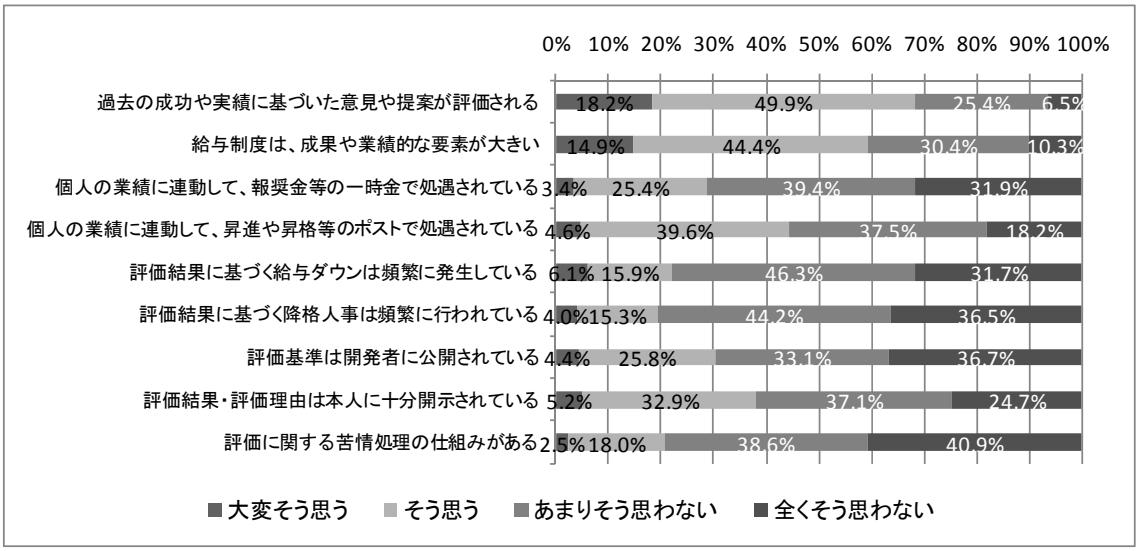


図 5.8-01　開発者に対する評価

5.8.2　開発者に対する処遇

開発者の基本給は、「年俸制」（50.5%）が最も多くなっている（表 5.8-01）。「職務給・役割給」（32.5%）が導入されつつも、「能力給」（32.3%）や「定期昇給」（26.6%）といった年功的な運用も並行して行われている。したがって、日本のゲーム産業は、年俸制による成果主義制度を導入しつつも、職能資格制度と職務等級制度の折衷・混合型の人事管理制度となっていることが指摘できる。

賞与・一時金は、「企業業績に連動した賞与・一時金」（25.6%）が最も多くなっており、いわゆるボーナスが支給されている。個人業績や開発チーム、開発タイトルに応じた賞与・一時金は、あまり支給されていない状況となっている。

諸手当は、「通勤手当」（71.1%）が顕著に高い数値を示している。また、日本のゲーム産業では「時間外手当」は、開発者の 36.5%は支給されていると回答している。勤務形態別にみると、「フレックスタイム制」の 58.0%、「始・就業時間が一定している通常業務」の 30.8%、「裁量労働制」の 28.4%において、時間外手当が支給されている結果となっている。

表 5.8-01　開発者に対する処遇

		度数	%
基本給	年俸制	241	50.5%
	定期昇給	127	26.6%
	年齢給	61	12.8%
	業績給・成果給	73	15.3%
	職務給・役割給	155	32.5%
	能力給	154	32.3%
	退職金の基本給組み入れ	25	5.2%
賞与・一時金	個人業績に連動した賞与・一時金	51	10.7%
	開発チーム別業績賞与・一時金	43	9.0%
	開発タイトル報奨金	63	13.2%
	企業業績に連動した賞与・一時金	122	25.6%
	ストックオプション	40	8.4%
諸手当	通勤手当	339	71.1%
	時間外手当	174	36.5%
	扶養手当	111	23.3%
	住宅手当	98	20.5%
	資格手当	36	7.5%
	役職手当	119	24.9%
	その他	9	1.9%

5.9　開発者の満足度

開発者は、どの程度満足しているのであろうか。仕事の満足度、ストレス要因、ゲーム産業でのキャリアチェンジ、ゲーム産業で働くプランから考察してみよう。

5.9.1　仕事の満足度

ゲーム開発者の仕事の満足度は、総じて高くなっている（図 5.9-01）。「大変満足」（8.0%）、「満足」（38.8%）の合計は、46.8%に達している。それに対して、「不満」（18.0%）、「大変不満」（7.1%）の合計は、25.1%と少なくなっている。

仕事の満足度を年収とのクロス表によってみてみると、仕事の満足度は年収に応じて高くなっている（表 5.9-01）。つまり、年収が高ければ満足度も高く、年収が低ければ満足度も低くなる。しかし、開発者は、金銭的処遇だけで動機づけられるのではない。先述し

とおり、自分の才能や能力を発揮するため、仕事が楽しいから、現在の仕事が好きだから、仕事を継続しているという事実から、内発的動機づけを高めるための施策が重要である。

また、仕事の満足度を仕事の継続理由とのクロス表によってみてみると、いずれの開発者も「お金を得るため」という目的で仕事を継続している結果となっている（表 5.9-02）。ところが、満足度が高い開発者ほど、「生きがいのひとつ」、「自分が自分らしく生きられる場所だから」、「この仕事が楽しいから」、「現在の仕事が好きだから」、「自分の才能や能力を発揮するため」という回答が多くなっており、働くことの意味を自己認識していることが把握できる。一方で、満足度が低い開発者ほど、「他にできることがないから仕方なく」という理由が多くなっており、働くことの意味を見出せない者が多く存在している結果となっている。このような動機づけの低い開発者に対しては、自己効力感を向上させるような施策を行っていく必要があるように思われる。

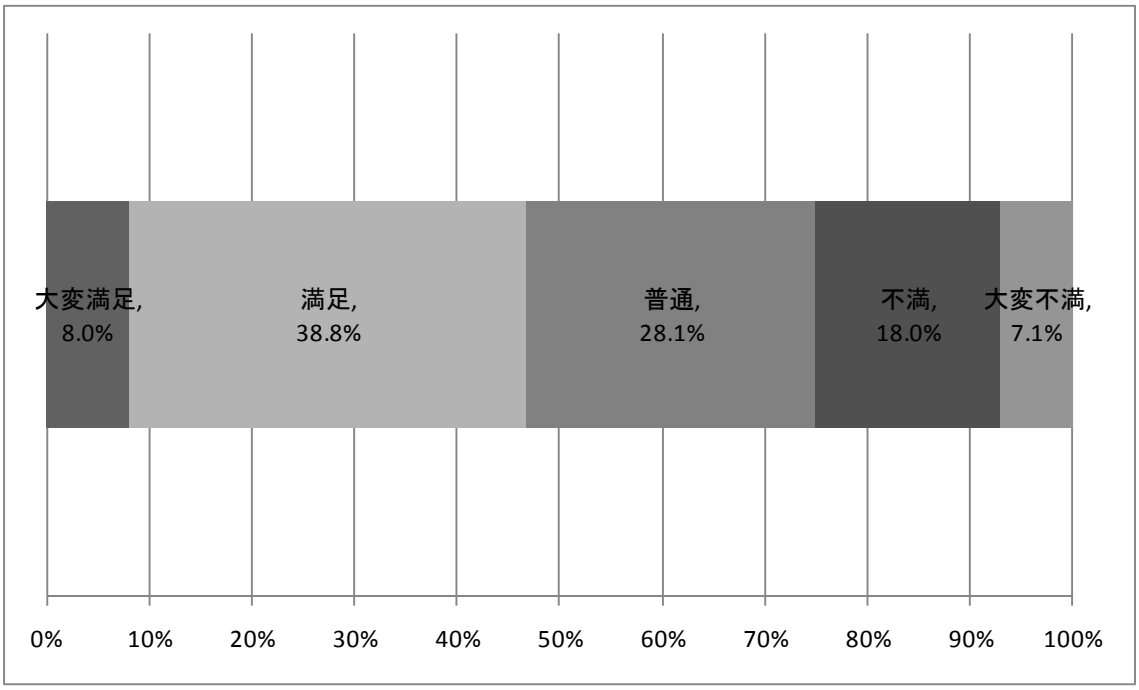


図 5.9-01　仕事の満足度

表 5.9-01　現在の年収×仕事の満足度

	仕事の満足度				
	大変満足	満足	普通	不満	大変不満
	平均値	平均値	平均値	平均値	平均値
現在の年収	6,262,632	5,391,617	5,068,796	4,906,657	3,983,515

表 5.9-02　仕事の継続理由×仕事の満足度

	仕事の満足度									
	大変満足		満足		普通		不満		大変不満	
	度数	%	度数	%	度数	%	度数	%	度数	%
社会の一員として責務を果たすため	9	23.7%	41	22.2%	31	23.1%	16	18.6%	6	17.6%
生きがいのひとつ	27	71.1%	80	43.2%	29	21.6%	21	24.4%	6	17.6%
お金を得るため	23	60.5%	127	68.6%	98	73.1%	65	75.6%	24	70.6%
社会で自分が所属する場所として	11	28.9%	59	31.9%	42	31.3%	29	33.7%	4	11.8%
自分が自分らしく生きられる仕事だから	21	55.3%	96	51.9%	39	29.1%	18	20.9%	5	14.7%
この仕事が楽しいから	29	76.3%	132	71.4%	61	45.5%	31	36.0%	5	14.7%
現在の仕事が好きだから	27	71.1%	142	76.8%	57	42.5%	27	31.4%	5	14.7%
自分の才能や能力を発揮するため	23	60.5%	117	63.2%	65	48.5%	37	43.0%	11	32.4%
作品を通じて自分の考え方を伝えるため	8	21.1%	25	13.5%	24	17.9%	19	22.1%	1	2.9%
自分にとっては一番楽な仕事だから	9	23.7%	22	11.9%	20	14.9%	9	10.5%	2	5.9%
なんとなく	0	0.0%	6	3.2%	13	9.7%	10	11.6%	6	17.6%
他にできることがないから仕方なく	1	2.6%	8	4.3%	15	11.2%	19	22.1%	10	29.4%
その他	1	2.6%	6	3.2%	10	7.5%	4	4.7%	5	14.7%

5.9.2　ストレス要因

もし、ゲーム開発者にストレスが発生しているとしたら、どのような要因があるのだろうか。これを尋ねた結果を米国調査（2004）と比較したのが表 5.9-03 である。日本では「その他」（22.6%）が最も多くなっているが、この内訳をみると、ゲーム業界や会社の将来性に対する不安、技術革新と自己が持つ知識や技術の陳腐化に対する不安、労働条件に対する不満、プロジェクト管理に対する不満、評価制度の曖昧さなどが挙げられている。それに続くのが「会社の財政難」である。会社の財政難は、パブリッシャー（5.9%）よりもデベロッパー（21.0%）のほうが顕著である。

米国に目を向けてみると、「官僚的な組織」（37.1%）が最も多い結果となっている。米

国の開発者はより自由な組織風土を好んでいることが窺える。

表 5.9-03　ストレス要因

	日本調査(2009)		米国調査(2004)	
	度数	%	度数	%
官僚的な組織	67	14.0%	366	37.1%
管理者と開発者の関係悪化	83	17.4%	157	15.9%
協働者との関係悪化	57	11.9%	31	3.1%
会社とパブリッシャーの関係悪化	15	3.1%	68	6.9%
会社の財政難	98	20.5%	82	8.3%
次のプロジェクトがどこから来るのかわからない	23	4.8%	114	11.6%
全てはうまくいっている	26	5.5%	35	3.5%
その他	108	22.6%	82	8.3%

5.9.3　ゲーム産業でのキャリアチェンジ

表 5.9-04 は、ゲーム産業でのキャリアを変えられるとしたらそれは何かについて尋ねた結果を、米国調査（2004）との比較で示した表である。これをみると、日本では「より面白いプロジェクトで働く」（45.9%）が最も多くなっている。それに対して米国では「より多くの収入を得る」（24.9%）が最も多くなっている。繰り返しとなるが、日本のゲーム開発者にとっては、金銭的処遇などの外発的動機づけを高めるための施策よりむしろ、人材の適材配置などの内発的動機づけを高めるための施策が重要であることが窺える。

表 5.9-04　ゲーム産業でのキャリアチェンジ

	日本調査(2009)		米国調査(2004)	
	度数	%	度数	%
より面白いプロジェクトで働く	219	45.9%	180	18.3%
より多くの収入を得る	83	17.4%	246	24.9%
より大きな責任を引き受ける	61	12.8%	85	8.6%
より短い時間で働く	39	8.2%	236	23.9%
仕事安定性を獲得する	64	13.4%	190	19.3%
何も変えられない	11	2.3%	37	3.8%

5.9.4　ゲーム産業で働くプラン

表 5.9-04 は、ゲーム産業で働くプランについて尋ねた結果を、米国調査（2004）との比較で示した表である。これをみると、「生涯、ゲーム産業で働く」プランの者は、日本も米国も最も高い数値を示している（日本 59.7%、米国 47.6%）。しかし、日本のほうが米国よりも 12.1 ポイント高い数値となっており、生涯、ゲーム産業で働きたいという意志を持つ開発者が多く存在していることが把握できる。

表 5.9-05　ゲーム産業で働くプラン

	日本調査（2009）		米国調査（2004）	
	度数	%	度数	%
2年以内に他の産業で仕事を探す	36	7.5%	144	14.6%
2-5年の間ゲーム産業で働き、他の産業で仕事を探す	83	17.4%	196	19.9%
5-10年の間ゲーム産業で働き、他の産業で仕事を探す	73	15.3%	168	17.0%
生涯、ゲーム産業で働く	285	59.7%	469	47.6%

また、表 5.9-06 によって、ゲーム産業で働くプランと仕事の満足度のクロス表をみると、当然の結果であるが、満足（47.4%）、大変満足（11.6%）の者ほど「生涯ゲーム産業で働く」プランを持っているのに対して、不満（33.3%）、大変不満（30.6%）の者ほど「2 年以内に他の産業で仕事を探す」という回答が多くなっている。

表 5.9-06　ゲーム産業で働くプラン×現在の満足度

		ゲーム産業で働くプラン							
		2年以内に他の産業で仕事を探す		2-5年の間ゲーム産業で働き、他の産業で仕事を探す		5-10年の間ゲーム産業で働き、他の産業で仕事を探す		生涯、ゲーム産業で働く	
		度数	%	度数	%	度数	%	度数	%
仕事の満足度	大変満足	1	2.8%	2	2.4%	2	2.7%	33	11.6%
	満足	4	11.1%	20	24.1%	26	35.6%	135	47.4%
	普通	8	22.2%	33	39.8%	24	32.9%	69	24.2%
	不満	12	33.3%	24	28.9%	15	20.5%	35	12.3%
	大変不満	11	30.6%	4	4.8%	6	8.2%	13	4.6%

5.10　おわりに―開発者の声･･･離脱・発言？

これまで、定量的データを用いて開発者の実態（勤続・給与、求職・転職行動、働き方、人材育成、評価・処遇、満足度など）を捉えてきた。しかし、数値では捉えきれない開発者の声もある。これに関連する理論に、Hirschman（1970）の離脱・発言モデルがある[11]。Hirschman は、組織における人間の行動と市場における人間の行動を、離脱（Exit）と発言（Voice）へ二分した。例えば、労働者が企業に対して不満を持ったときに離脱（Exit）することで不満を解消するか、あるいは発言（Voice）することで不満を解消するかということに注目したものである。

本調査では、178 名から自由記述に回答をいただいたが、紙幅の関係上、全てについて記述することは難しい。そこで、開発者のキャリアに注目して、開発者の声を紹介しよう。

―――産業としてのゲーム業界の成熟度が増している割には、キャリアプランに頭打ち感があるように感じる。以前からキャリアに対してフラットなスタイルで納得していたのは、成功報酬的なインセンティブによる評価ありきだったが、業績不振や、市場飽和等で、処遇が一般企業のようなスタイルに移りつつあるのであれば、キャリアポストを有る程度作っていくべきだと思う。（男性、30 代、プログラマー）

―――ゲーム業界は、キャリアやステップアップなどが不透明でかつまた労働環境に対して整備されていないという印象を持っています。これらは、実際の現場におけるプロジェクトの進捗や、プロジェクト間の連携などにも影響がでており、例えば開発者が OJT などを正しく受けられず放置されることによって、成長を独力にのみ頼るため、能力や適用範囲にも偏りが発生していると考えられます。他のプロジェクトで採用している技術や方法も正しく共有をされず、埋もれてしまう現状があります。これらが改善されることでよりよい開発者の増加や、開発環境の整備、より面白く品質の高いゲームの作成などにつながるのではないかと考えています。（男性、30 代、プログラマー）

―――キャリアのゴールが見えにくく、もがいている。ゲーム開発に関わる若年層がゲーム開発に対して、意識の低さが顕著にみられる。（男性、40 代、プロデューサー）

―――社内でキャリアパスを描くことが非常に難しいです。うまく世代交代が出来ずに昔の方法論で現開発に取り組んでいるため、問題意識の共有や開発環境のあり方に関する興味が低く、若手の芽を伸ばせない雰囲気があります。（男性、30 代、グラフィッカー）

―――会社によるが、使える人材を雇うばかりで、その後の人材開発のできていない稚拙な会社が多い。中間管理職が、クオリティ面での管理だけはできるが、各スタッフのキャリアパス/育成のできない人が多い。これが日本ゲーム業界が世界に後れを取っている原因ではないか。（男性、30 代、グラフィッカー）

―――ゲームバブル期のキャリア組が上司として居残り、下の世代を押さえつけ、才能を発揮できず、処遇や伸び悩んでいる不景気と開発費の増大を理由に、新しいプロジェクトにチャレンジさせてもらえない。（男性、30 代、ディレクター）

―――経営や経済などに疎い人材が管理職に就くので、マネジメントやビジネスでの観点から物事を見れないでいる感じがする。特に、ゲーム産業黎明期からこの業界に従事している人は、その傾向が強いと感じる。そのため評価制度やキャリア形成などにまで考えが至っていない。（男性、30 代、グラフィッカー）

―――クリエイティブな職種においては、それなりのキャリアパスが保障されてほしいが、組織が大企業化、官僚化することによって、一般企業の管理職みたいな仕事が増えつつある。大組織の場合、分業化、専門化のメリットはあるものの、今まではなかった斬新なアイデアを商品化するためには、社内のハードルが高すぎる。PS2 の一極集中化時代が終わり、プラットフォームは多様化しつつあるにも係らず、大組織では柔軟な対応が遅れている。だからといって、ベンチャーが活性化するためには、市場や流通の構造に大きな壁が立ちはだかっている。（男性、40 代、ディレクター）

―――個人的には、社内で違う仕事を求めるか、転職するか（ゲーム業界以外も含めて）、今後のキャリアを検討しているところです。（男性、30 代、ネットワーク・エンジニア）

―――ゲーム制作における日本特有の欠点をいくつも持ち（制作をなあなあで進行する所、閉鎖的な所など）同時にワンマン経営であり、割り切った会社との付き合い方が要求されてきた。生活は保障されるが、長い目で見て、あと 10 年生き残れる会社かどうかは疑問であり、それまでに、会社に依存しないキャリアを持つ必要性を感じている。（男性、30 代、グラフィッカー）

―――40 歳以降のキャリアの展望がまったく見えません。これといったヒット作の実績を持たないため転職もままならず、年齢だけ重ねてしまったため、自分のスキルを発揮できない仕事でも受けざるを得ず、結果、あまり高評価も与えられず給与カット、のようなスパイラルを止めることができません。もっとゲームデザインの仕事それ自体や、プロデューサーにならない場合のキャリアパスなどについて、同じ悩みを持つ人たちで突っ込んだ話ができるオープンな場があれば、とも思いますが、基本的に「業界」の未来は憂えても、「個人」の憂いの共有はなかなか難しいのかも知れません。その意味で、今回のアンケートには意義があると思います。（男性、30 代、プランナー）

―――現在の会社に入社してもうすぐ 20 年経とうとしています。諸外国から見れば驚く程長い期間、同じ会社に所属している訳ですが、居た分だけの経験、人脈、そして愛社精神はそれなりに育まれている事を考えると、別な会社へ移る、と言う事は年齢や家族等を持っている関係で考え難い状況です。キャリアをそれなりに積んできた現在、自分の中にはもっと外へ出て活躍したい、と言う欲求があり、ゲーム開発を通じたより海外との連携や、拠点を持つ事を立場を越えて考える様になっています。これも最終的には

活動をよりグローバルに、且つ、コスト意識を持ってパフォーマンスの高いものをもたらす、と言う部分が基本ですので社内ベンチャーの様なものを模索しています。ゲーム業界と言いますかクリエーターの評価方法は未だ曖昧な部分が多く、個々に左右されるものだけに、数値化が難しいと思います。業界に関しての私の個人的な見解は、確かにエンタテインメントは衣食住と言う原則的に人間に必要なものからは離れていますが、人間が存在している以上、遊び、の部分は決して廃れない、と確信しています。そこに今もたまたまデジタルと言うメディアを使ってゲームと言うものをアウトプットしてはいますが、他の方法論で「遊び」と言うジャンルを形作っていけるのもゲーム業界のフレキシブルな所ではないか、と思っています。勿論、空に浮いた様な職種、業界ではありますが、クリエーターが高齢化している中、しっかりと家庭を築ける業界であり続ける事も、自分に課せられた 1 テーマ、とも思う次第です。（男性、40 代、サウンド）

―――幼児期の子どもがいる母親ですが、出産後、復職して現在までがんばってきましたが、子育てをしながら続ける仕事としては、業界自体が若すぎて、働きにくい部分があります。子どもの急病などで早退・遅刻あるいは休暇もままならない状態です。そのため、多くの女性技術者はキャリアをあきらめざるを得ない状況にあると思います。そもそも、定年退職まで働ける仕事ではないことはわかっていても、この状況は働き盛りの 30 代には少し酷です。かといって、満足できるまで働けば、妊娠しにくい年齢に差し掛かってしまい、治療が大変になるなど、多くの問題を抱える女性技術者をたくさん見てまいりました。残業しなければならないのは仕方ないとしても、子供を犠牲にしてしかるべきという風潮はもうやめて、保障（たとえば保育料の一部負担など）をきちっとしてほしいと考えます。（女性、30 代、ネットワーク・エンジニア）

―――儲けようという意識の薄いままダラダラと受注仕事をこなしていましたが、折悪しく不況により受注が当然のごとく減少、それに伴い売り上げも落ち込み、今になって危機意識が芽生えたのか、自ら企画立案する提案型企業を標榜するようになりましたが、社内の危機意識は希薄に感じられ、仕事が降りてくるのを待つ人が多い状態です。おそらく大多数の社員が不満を抱えてると思われますが、転職を図るにもリスクがあるため今の会社に甘んじています。正直、根本的に改善される望みが薄く、自分の力を活かせる場所があればそこに行きたいのですが、これまでの仕事内容がキャリアにプラスにあまりならない物が多く、かといって日常業務に忙殺され自己研鑽の時間も満足に取れないという、ある種の閉塞状況に陥っています。（男性、30 代、グラフィッカー）

―――開発者のキャリアや、働きやすい環境が経営側に理解されにくいです。特に社員が望む働く環境が、営業側の社員と開発側とでは全然違うのに一緒くたにされていて、作り手としては日々ストレスを感じています。（女性、20 代、グラフィッカー）

このような開発者の声は枚挙にいとまがないが、転職者が多い（約 6 割）という事実から、多くのゲーム開発者は、発言オプションを選択せずに、組織を離脱してしまっている

のではないだろうか。満足度調査でみたとおり、もちろん、多くの開発者が仕事に満足を感じているが、勤続年数の短さや平均年齢の若さをみると、将来的なキャリアを描けないのが実状であろう。実際に、「キャリアプランに頭打ち感がある」、「キャリアやステップアップが不透明」、「キャリアのゴールが見えにくく、もがいている」、「40 歳以降のキャリアの展望がまったくみえない」というキャリア・ミスト（自己の将来キャリアにかんする不透明感）（加藤,2004:164）[12]があり、「社内で違う仕事を求めるか、転職するか」悩んでいる者もいる。また、「会社に依存しないキャリアを持つ必要性を感じている」と、ゲーム産業を取り巻く環境変化に伴って、自律的なキャリアを形成していくことが重要であると指摘する者もいる。その他に、「しっかりと家庭を築ける業界であり続けること」、「多くの女性技術者はキャリアをあきらめざるを得ない状況にある」というワーク・ライフ・バランスの問題を指摘している者もいる。総じて、開発者はこれまでのキャリアを回顧的に意味づけて、自己あるいは自己と組織の関係のなかから将来を展望しているのである。

如上の開発者の声は、組織の離脱（Exit）を意味するのか、それとも組織への発言（Voice）を意味するのか、改めて検討していく必要があるだろう。

5.10.1 要約と結論

本調査は、ゲーム産業の暗黙の了解事項を数値で確認しただけに過ぎないかもしれない。しかし、ゲーム開発者の実態を総合的に把握し、よりよい環境を整備していくために客観的なデータを得ることができたことに意義がある。それでは、本調査で得られた知見をどのように活かしていけばよいのであろうか。改めて課題を整理しながら、展望を見出そう。

第一に、ゲーム産業や仕事に対する動機づけに関する問題である。「ゲームが好きだから」、「自分の能力・個性を活かせるから」、「仕事が楽しそうだから」と高い期待を持ってゲーム産業に参入をしている者が多いことから、開発者のモチベーションを維持・向上させるための施策が必要である。とりわけ「仕事の満足度」との関連からは金銭的処遇がその基礎となるであろうが、そのような経済的なインセンティブだけでは不十分である。最も大切なのは、開発者の能力や才能が活かされ、それが他者から公平・公正に評価、承認される環境づくりを整備することである。例えば、開発者の希望するプロジェクトや部署への人材配置が挙げられる。具体的には、社内公募制や社内ベンチャー制などのように、開発者の自律的なキャリアを支援できるような環境が必要である。また、開発者に対する表彰制度も有効であると思われる。しかし、制度づくりだけをしてもあまり意味がない。開発者のモティベーション・マネジメントと関連づけて運用していくことにより、開発者の職務満足やリテンション（人材の確保・維持）[14]にもつながるであろう。

第二に、労働時間管理、とりわけ繁忙期の労働時間管理の問題である。日本のゲーム産業は、世界的にみても長時間労働であることが多くの論者から指摘されてきた。よい作品を生み出すために、「繁忙期は一般的なもの」という認識が強固なものとなっている。米国

では「時間外手当は補償されないが、プロジェクトの終了時に休暇で補償される」という考え方が中心であるが、日本では「時間外は補償されない」と「割増料金で時間外が支払われる」という相対立する考え方が中心となっている。日本では「裁量労働制」の導入が進んでいるため、労使関係の問題が今後拡大していくだろう。裁量労働制は、業務の遂行の手段および労働時間配分の決定を労働者の裁量に委ね、その場合の労働時間については、一定の時間労働したものとみなすもので、労働時間ではなく労働の質や成果に応じて報酬を決定する制度である。とりわけ、創造的労働のための裁量性を本質とするものであり、労働者の主体的な働き方を可能とし、その能力発揮を促進するとの趣旨にある以上、開発者の業務遂行の自律性が確保されなければならない。労働時間管理は、開発者の生活の質（Quality of Life: QOL）に関わる問題であり、過剰労働やその弊害を未然に防ぐ上でも重要である。

第三に、キャリア形成に関する問題である。中年期（ミドル）以降のキャリア展望が見出せない者が多く存在するが、一企業のみならず、業界全体で開発者のキャリア形成を支援していく必要があるだろう。なぜならば、人は生涯を通じて成長するものだからである。キャリア形成の問題を自己責任に収斂するのではなく、社会的連帯（social solidarity）のもとに捉える必要がある。一部の企業においては、「新入社員研修」や「CEDEC への派遣」、「心身の健康管理に対する支援」などが行われているが、「特別な支援はしていない」、「自己負担で自己研鑽に努めている」という回答も多く存在する。にもかかわらず、「生涯ゲーム産業で働きたい」という開発者は約 6 割も存在している。また、転職者の約 7 割が「ゲーム産業」からの転職であり、ゲーム産業全体で開発者の育成が行われている。開発者の立場に立てば、「より質の高いゲームをつくりたい」、「ゲーム産業で働きたい」という意志が強いのである。国際ゲーム開発者協会日本（IGDA 日本）などの開発者コミュニティや業界団体、学術団体などの社会関係資本（social capital）を活用することにより、開発者の生涯キャリア発達を支援していくことは、ゲーム産業の活性化にもつながるであろう。

第四に、上記と関連して、産学官の連携に関する問題である。ゲーム産業の事業領域は、「据置型ゲーム事業」を中心としつつも、「携帯型ゲーム事業」や「モバイル・コンテンツ事業」、「オンラインゲーム事業」などへ拡大しつつある。多様な顧客ニーズに対応し、より質の高い作品や新しいジャンルを開拓して、ゲーム産業がさらなる発展を目指していくためには、人材の育成と技術の応用が必要である。我が国では、ゲームの産学官連携組織として、全国に先駆けて 2006 年に福岡ゲーム産業振興機構が設立された。福岡ゲーム産業振興機構の設立のきっかけとなったのが、福岡のゲーム会社の連携組織 GFF（GAME FACTORY'S FRIENDSHIP、GATE FOR FUTURE）である。GFF は、九州大学や福岡市という学と官とも連携しつつ、「FUKUOKA ゲームインターンシップ」や「シリアスゲーム産業育成事業」などに取り組んでいる。前者は産学官連携による人材育成と人材獲得を、後者はシリアスゲーム開発のノウハウの蓄積とシリアスゲーム開発者の育成を目指している。このような先駆的な産学官連携を参考にして、企業間連携や産学官連携による効

果的かつ効率的な研究開発システムを構築していくことが必要である。

第五に、企業規模間格差に関する問題である。大企業と中小企業との間では、Off-JT や開発者のキャリア支援の実態に大きな乖離がみられた。例えば、「新入社員研修」、「階層別研修」などの Off-JT の実施率や、CEDEC、SIGGRAPH や GDC などの「国内外の研修機関・研究会」への参加率をみると、中小企業は大企業に比べて大幅に低い結果を示している。また、中小企業では、開発者のキャリア形成について「特別な支援はしていない」ため、「自己負担で自己研鑽に努めている」開発者が多く存在する。開発者の流動性が高いゲーム産業においては、内部育成よりもむしろ外部調達によって人材が確保されるため、人的資本への投資が積極的に行われない状況にあるのではないだろうか。とりわけ中小企業では、開発者のキャリア形成に対して積極的にはなれず、その結果、多くの離脱者（Exit）が存在しているのではないだろうか。中小企業のゲーム会社においては、「中小企業雇用創出等能力開発助成金」などを利用して、開発者の教育訓練の充実とリテンションを図っていく必要がある。また、行政機関や業界団体などによる中小企業への支援も重要である。健全な市場競争のもとでイノベーションが促進されることにより、中小企業が新たに創業され、ゲーム産業全体の活性化につながるだろう。

以上、開発者のモティベーション・マネジメント、労働時間管理、キャリアディベロップメント、産学官連携、企業規模間格差の 5 点について、それら課題と展望について指摘した。そして、開発者が先端技術を応用したデジタルコンテンツ制作を行っていくために必要な環境を具体化し、それに対する施策を提言した。

経済産業省が『ゲーム産業戦略～ゲーム産業の発展と未来像～』[14]を 2006 年に公表してから 3 年が経過した。ここに示された具体的な施策の実施に対する評価と、さらなる企業努力や開発者のキャリア形成、そして新たな政策提言に向けて、本章が役に立てば幸いである。また、劇的に変化する開発者を取り巻く環境のなか、ゲーム開発者が生涯キャリアを形成し、新たな価値を創造できるよう、このような調査が経年的に行われていくことが期待される。

5.10.2 残された課題

本章では、開発者の立場から、先端技術を応用したデジタルコンテンツ制作を行っていくために必要な環境が明らかにされたが、開発者個人と組織との共生関係のあり方については今後の課題として残された。開発者個人と組織の成長・発展に関して、さらなる研究の余地がある。組織は開発者個人を創造的人材（人的資源）としてマネジメントするが、「創造的な成果を目指す個人にとっては、社会構造のなかで自らが代替可能な選択肢としての資源として認知されることは本意ではない」（山下・山田,2010:59）[15]。すなわち、創造的な価値を共創する組織観ならびに開発者の人間観を改めて問い直す必要がある。

また、開発者のキャリアは、オーガニゼーショナル・キャリア（特定の組織内で展開す

るキャリア）からバウンダリーレス・キャリア（特定の組織内にとらわれない境界のないキャリア）[16]へと既に変容しているのかもしれない。もしそうであるならば、開発者個人のキャリア発達のみならず、開発者個々人の関係性のなかから形成されるキャリアにも焦点を当てていく必要があるだろう。

参考文献：

[1]International Game Developers Association, *"Quality of Life White Paper"*,（2004）
[2]United Business Media, *"Game Career Guide Fall 2009"*,（2009）
[3]藤原正仁, "我が国のゲーム開発会社の人材マネジメント"『ゲーム産業における開発者人材育成事業報告書』社団法人コンピュータエンターテインメント協会,（2007）pp.45-80
[4]社団法人コンピュータエンターテインメント協会,『ゲーム産業における新卒開発者人材育成事業報告書』,（2008）
[5]藤原正仁, "変容する日本のゲーム産業と人材マネジメント"『デジタルコンテンツ制作の先端技術応用に関する調査研究報告書』財団法人デジタルコンテンツ協会,（2009）pp.230-260
[6]日本アニメーター・演出協会,『アニメーター労働白書 2009』,（2009）
[7]石島照代,『ゲーム業界の歩き方』ダイヤモンド社,（2009）
[8] International Game Developers Association, *"Game Developer Demographics: An Exploration of Workforce Diversity"*,（2005）
[9] 国税庁長官官房企画課,『平成 20 年分民間給与実態統計調査－調査結果報告－』,（2009）
[10] 日本アニメーター・演出協会,『2008 年度実態調査概要報告』,（2009）
[11]Albert O. Hirschman, *"Exit, Voice, and Loyalty: Responses to Decline in Firms, Organizations, and States"*, Harvard University Press,（1970）＝矢野修一訳,『離脱・発言・忠誠：企業・組織・国家における衰退への反応』ミネルヴァ書房,（2005）
[12]加藤一郎,『語りとしてのキャリア：メタファーを通じたキャリアの構成』白桃書房,（2004）
[13]山本寛,『人材定着のマネジメント：経営組織のリテンション研究』中央経済社,（2009）
[14] 経済産業省商務情報政策局文化情報関連産業課『ゲーム産業戦略～ゲーム産業の発展と未来像～』
[15]山下勝・山田仁一郎,『プロデューサーのキャリア連帯：映画産業における創造的個人の組織化戦略』白桃書房,（2010）
[16]Arthur, M. B., and D. M. Rousseau, "*The Boudaryless Career: A New Employment Principle for a New Organizational Era*", Oxford University Press,（1996）

6. 海外におけるゲーム関連技術の現状調査

6.1 GDC2010 における海外産業動向 ～ソーシャルゲームについての側面を中心に～

新 清士（IGDA 日本）

6.1.1 はじめに

3 月 9 日から 13 日の日程で米サンフランシスコにて、世界最大のゲーム開発者向けカンファレンス「Game Developers Conference 2010（GDC、ゲーム開発者会議）」が開催された。

不況の影響で、参加者数は減少すると思われていたが、実際には、参加者数は 1 万 8250 人と、昨年より 1250 人増加した。増加を促した要因は、4 億人ものユーザーを集めたソーシャルネットワークサービス「Facebook」を中心に、急激に広がっている「ソーシャルゲーム」関連の講演の充実にあったことは間違いない。GDC では、全体で 450 あまりの講演が開催されるが、前半二日間で、新たに開始されたサミットの一つ「Social & Online Game Summit」だけでも 20 もの関連講演が行われ、これまではコンシューマゲーム機関連の講演が中心だった。

後半のメインカンファレンスでも、数多くのソーシャルゲームに関連する講演が行われおり、多くの注目を集めていた。

昨年までは、ここまでソーシャルゲームはトピックの中心になっていなかったものが、今年いきなり、ゲームの表舞台に出てきた格好だ。率直に驚かされたのは、日本で想像していたよりはるか速い速度で、ゲーム産業全体の再編が進んでいることを実感した。

このレポートでは、そのソーシャルゲームに関するセッションを中心に、なぜこうした急激な変化が引き起こされたのかを紹介する。

6.1.2 ハイペースで 2 億人のゲームユーザーを集めた Facebook

Facebook のガレス・デイビス氏（プラットホームマネージャー）の行った「How Friends Change Everything という基調講演で、サービスの全体像を紹介しながら、自らの強みを強烈にアピールした。強さを支えるのは「数」の論理である。桁違いのユーザーを集めることで、他を圧倒する。

Facebook の中核は「オンライン上の現実のアイデンティティ（Your real identity online）」だという。すべての友人とオンライン上で繋がる関係性を持てる点だと。現在、Facebook の登録ユーザー数は 4 億人で、そのうち 2 億人が毎日アクセスするアクティブ

ユーザーであり、一人当たりのユーザーの平均アクセス時間は 55 分にも及ぶ。ゲームを日常的に遊んでいるユーザーは、2 億人に登る。

Facebook 上に、提供されている API を使って、自由に他の企業がソフトウェアを提供することができるが「オープン化」を行いゲームのサービスが開始されて、2 年 4 ヶ月で、この数字である。昨年 3 月に 1 億台を超えた「ニンテンドーDS」シリーズでさえ、4 年 3 ヶ月かかっている。すさまじいペースといわざる得ない。

Facebook の優位性は、ユーザーのデータのすべてをインターネット上の自社サーバーに置く「クラウド型」のサービスである点にある。ユーザーにとって最も大切なものは、自分自身がサーバーに預けたデータ、つまり「アイデンティティ」である。自分自身のデータをサービスに預ければ預けるほど、そのサービスを別のサービスで代換えすることは難しくなり、そこでしか得られない価値は高まっていく。それにも関わらず、サービスは基本的に無料で提供されており、コンピューターとブラウザがあれば大半の機能をどこでも享受することができる。ユーザーがサービスに参加するまでの障壁は極端に低い。

ゲームでも、価格を付けられない価値は、自分自身のプレーした結果が反映されたデータである。そのデータを各企業が直接的に押さえているのである。その上、ユーザーがどのようにゲームを遊んでいるのかを 100%把握することができる。

しかも、ユーザーは、既存のコンシューマゲームのように料金の先払いの必要性がない。遊んで気に入ってくれれば、後払いでアイテム課金などの仕組みに乗ればいい。ユーザーがお金を支払うようになるのはプレイヤーの 10%以下であることは一般的によく知られているが、それでも数千万人というユーザー数を抱えることができれば、10%のユーザーが、一人 10 ドル月に支払ってくれるだけで、莫大な利益を生み出す。

強いのは当然と言うことができる。

6.1.3 デバイスを問わないサービスへと広がる Facebook

クラウド型サービスは、特定の物理メディアに依存していないために、好きなようにユーザーに供給するデータの内容を決めることができる。そのデータを有料化することも、無料で提供することも自由にできる。昨年までは、Facebook は無料で利用できる有効な口コミマーケティングツールだとゲーム産業からは考えられていた。今年は、違う段階に入った。「プラットフォーム」として、既存のゲーム機などのハードウェアを凌駕するほどの影響力を持ち始めたのである。

Facebook は、外部の企業が無料で勝手に利用できる API を「Facebook Connect」という形で提供している。この機能を利用すると、他のハードも部分的に Facebook のサービスと連携させることができる。

カジュアルゲームと呼ばれるパソコン向けのダウンロード販売型のゲームの結果を Facebook のデータと連動させることができる。iPhone 向けのゲームアプリでも、この機

能を使うことができる。

既存のゲーム機でも、昨年 7 月より「ニンテンドーDSi」のカメラを使って撮影した写真は、この API を利用して Facebook にアップロードできる。昨年 11 月より、「Xbox360」は、写真や、Facebook を見ることができる機能など、様々な面で連携を進めている。同様に、「プレイステーション 3」もゲームで獲得したトロフィーや購入したゲームの情報を投稿できる機能が追加された。

この状況をデイビス氏は、「マルチデバイス・プラットホーム」と呼んだ。ある人は、パソコンから、ある人は携帯電話デバイスから、ある人は Xbox からアクセスする。提供されている機能は、共通しているわけではないが、中核にユーザーのアイデンティティを統合する存在として Facebook がある。

もちろん、デイビス氏は、各ハードウェア会社よりも、自社の方が上といった発言をしたりはしない。しかし、言外に「ハードウェアが優位であった時代は終わる」という事実を突きつける。ネットに接続できるハードウェアは、超巨大化した Facebook に関わらずにいることは難しい状況が完成しつつある。

6.1.4 機能拡張によって競合関係化が進む

Facebook の進撃は容易には止まらない。米 Many-eyes.com による世界でトップを取っている SNS を集計したページでは、Facebook が全世界の大半を押さえている。

日本、中国、ロシアなど、一部の地域で独自 SNS がシェアトップを握っているが、世界レベルで規模に置いて、対抗できる可能性があるのは、08 年にウルムチで起きた動乱以後、Facebook へのアクセスを政府がアクセスブロックを行っている中国ぐらいだろう。

米調査会社の InseideNetwork のジャスティン・スミス氏の講演「The State of Social Gaming: Industry Overview and Update」では 3 月 1 日現在の集計の Facebook ユーザーの内訳は、北米のユーザーが 1 億 4000 万人で 36%、欧州が 1 億 3000 万人で 34%だが、アジアもインド、インドネシアなど 7000 万人と 18%も占めている。

さらに影響力は機能追加を通じて増加しようとしている。例えば、Facebook は、独自仮想通貨である「Facebook Credit」を順次拡張していくことを明らかにしている。Facebook が販売された金額の 3 割を取るというオンライン決済の仕組みだ。このシステムは、アイテム課金販売のための仕組みとして使いやすく、8300 万人のユーザーを集めている人気農場育成ゲーム「FarmVille」（Zynga）がまず対応した。

これは、アップルにとってはやっかいな仕組みだろう。Facebook に対応したアプリを無料で iPhone 向けにサービスを展開し、課金は Facebook Credit を採用するようなケースが出てくれば、直接的な競合関係になるからだ。早晩、他のハードも同じ問題に直面することになるだろう。誰が課金プラットフォームと、ビジネスプラットフォームを握るのかという争いは激しくなっていくと思われる。

6.1.5　2007 年に考えられたソーシャルゲームに必要な要素

歴史的なイノベーションのパターンの類似性がある点から、ソーシャルゲームの優位性を論じたのは、米大手パブリッシャーのエレクトロニックアーツ（EA）のゼネラルマネージャーのベン・コーシンズ氏だ。「Kings of Convenience - What Walmart Tells US About the Future of Gaming」で紹介された。アメリカ最大のスーパーチェーンウォルマートが成長する基盤は何があったのかを説明しながら、今のソーシャルゲームブームが短期で終わるのではなく、歴史的な決定的な変化と見ていた。

コーシンズ氏は、2007 年のノルウェーで行われたカンファレンスで、ゲーム産業の未来を予測した。その際、まったく、はずれてしまった予測は「Wii」が大失敗するということだったと、失笑気味に話した。しかし、それ以外のものは、すべて当たっていると述べた。それは、次のようなものである。

- 「ユーザー生成コンテンツ」が一般化する。
- ユーザーのローカルな環境ではなくインターネット上の「クラウド」サーバーにユーザーのデータを置くことが普通になる。
- シンプルなインターフェースが求められるようになる。
- これまでのゲーマー以外の大衆市場に広がりが生まれる。
- ソーシャルネットワークが重要になる。
- ハイパーリンクの手法が多様化する（つまり、ブラウザゲームが重要になる）。
- 求められるパソコンのスペックが低いもので動かすことが前提になる。
- ゲーム世界に持続性が生まれ、ゲームのレベルアップの深さも広がる。

これらの要素をすべて満たしている代表例が、ソーシャルゲーム大手の Zynga の 6000 万人以上のユーザーを集めている「FarmVille」だという。これらの予測はソーシャルゲームという言葉の定義内容そのものといっても過言ではない。他方、これらの機能は、家庭用ゲーム機向けゲームでは、全部は実現されていない。

6.1.6　スーパーマーケット登場の歴史とソーシャルゲームの発展は似ている

コーシンズ氏は、20 世紀の初頭に車が登場する以前のアメリカ社会では、移動は、徒歩や馬車であり、町の中心部には人通りも少なく、お店で小麦や砂糖など生活必需品を買う場合には、店員と話しながら一品一品、量り売りで購入するのが基本的だった。

それが、1930 年代になって、その様子は大きく変わり始める。自動車が登場し普及が始まり、それに合わせるようにスーパーマーケットの業態が登場したためだ。アメリカ全土で、1920 年代には 800 万台だった自動車は、1930 年代には 2300 万台にまで増加する。世界恐慌の影響も受けることなく、自動車は普及し続けた。

普及に合わせて道路も変わった。1910 年代には、馬車道の延長線上に過ぎず、ぬかるみだらけで、でこぼこだった道路は、1920 年代には平らな道路へと整備され、1940 年代になると舗装が一般化する。

1916 年に、大量に並べられた商品を棚から、自由に選んで、最後にレジカウンターで精算するという、現代まで続く「スーパーマーケット」がテネシー州メンフィスに登場する。このフォーマットは、自動車社会が進むにつれて一般化し、より多くの品揃えと巨大な駐車場を持つ、ウォルマートに代表される、現代のショッピングモール方式へと発展していく。

小規模小売店は、対面サービスの面では質が高い。しかし、コストが高く「不便」だった。また、店にいる常駐する専門家のアドバイスは、消費者が商品を手にするプロセスが簡単になるほど、重要ではなくなる。そして、一度、新しい技術やビジネスモデルが登場すると、その成長は指数関数的な急成長をみせることになり、旧来、存在していたものを凌駕する存在へと変わる。

結果的に、現在、アメリカでシェア 1 位のウォルマートには、途方もない数の商品が置かれている。一方で、対面販売を基本とした小規模の小売店の数は、毎年減少している。

この歴史的なイノベーションのパターンは、既存の家庭用ゲームと、台頭するソーシャルゲームに、そのまま当てはめることができると指摘する。どこにでもユーザーの周り存在するようになったコンピューター環境は自動車であり、道路は回線が速くなっていくインターネット網というわけだ。ディスクメディアを使って使用する、既存の家庭用ゲーム機は、道路に接続されていない 19 世紀のアメリカ社会というわけだ。

6.1.7　家庭用ゲーム機が抱えていた「不便」を解消するソーシャルゲーム

既存のゲームは、質が高いが、ゲーム機を購入したり、ゲームショップに行ったり、ゲーム自体の価格を前払いする必要がある点など、様々な「不便」を抱えている。しかし、それしか選択肢がなかった時代には問題ではなかった。

ソーシャルゲームという新しい技術が登場したことで不便になったのだ。ゲームの質は、明らかに既存のものより低いが、ジャンルとしては、既存のゲーム機が提供しているものと同じ分野のゲームを展開することができる。

さらに、オンライン流通は、コストを劇的に下げることができ、価格をゼロに近いところから始めることができる。

ユーザーは、価格が低いために、ハイエンドのゲームほどの高い水準を期待しないでゲームを気軽に始める。結果的に、既存のゲームに比べて、ユーザーにとって「より高い便利さ」を実現することができ、大量のユーザーを集めることができる。また、ゲームへの参入コストが低さは、ゲーム専門メディアや、レビューサイト情報を集めた「メタスコア」といった「専門家情報」の役割を相対的に低下させる効果が現れる。

ウォルマートの品揃えがものすごい量であるように、ソーシャルゲームは誕生してから、まだ、一般に広がり始めて 3 年も経過していないにもかかわらず、すでにゲームのジャンル数はコンシューマー機向けゲーム以上に広がる膨大な品揃えを整えつつある。

そして、次にやってくるのは、それらの技術が高度化して既存のゲームと質の面でも肩を並べるという段階であることは、容易に理解できる。

6.1.8　ソーシャルゲームでもフランチャイズタイトルが鍵に

昨年 11 月に EA に買収された Facebook 向けソーシャルゲーム会社大手の英 Playfish の共同設立者のクリスチャン・セーゲルストローレ氏による講演「The Relentless March Towards Free…and What it Means to the Game Industry」では、別の見方から似た議論を展開した。動画サイトの YouTube が誕生から 3 年で、2009 年にアメリカだけで 1 億 5000 万人がオンラインで動画を見るようになった。ソーシャルゲームは、2 年間でゼロから 2 億人のユーザーを抱えるようになっている。これを「Youtube 効果」と呼んでいた。

しかし、セーゲルストローレ氏は、この現象によって何が起きていないのかに、着目する。パッケージゲームでも「モダン・ウォーフェア 2」のような「大規模予算をかけた大ヒットタイトルは死んでいない」、「Yahoo のようなポータルサイトもなくなっていない」、無料ゲームが中心となったからと行って「(ゲーム産業全体の) 利益も消し飛んだわけではない」。

そのために次に起きてくることは、既存のフランチャイズタイトルが強くなると見ている。実際、先行する iPhone 向けゲーム市場では、その現象が起こり始めている。08 年にアップルのネットワーク販売仕組みである「App Store」をスタートした直後は、ビールを飲むアプリ「iBeer」のようなアイデア一発のようなアプリが大ヒットした。

ところが、現在、ランキングの上位にいるのは「ロックバンド」、「グランドセプトオート」、「シムズ 3」、「Madden NFL 10」、「アサシンクリードⅡ」など、上位 10 位のうち 8 タイトルが、既にブランドが確立されているタイトルだと指摘する。

この現象は、ソーシャルゲームにも広がるだろうと分析していた。今後、多くの企業が既存のフランチャイズタイトルと組み合わせることは、確実に一般化してくるだろう。ユーザーにとってもわかりやすい評価軸でもあるからだ。

一度、ビジネススキームやプラットフォームの方向性が定まると、一気に予想を超えてその分野の成長が始まることは、様々な技術分野で、過去繰り返されてきた現象である。今、ゲーム産業の新しい方向性が定まりつつあると、ソーシャルゲームで成功している人びとは考えつつある。ソーシャルゲーム分野は、一見多様な選択肢があるように見えるが、すでに最初の混沌の時期を確実に終える段階に入ろうとしているようにも見える。

6.1.9　ソーシャルゲームでは存在感の薄い日本

一方で、Facebook 関連の講演を聴いていて痛感したのは、日本企業の存在感のなさである。Facebook が、そもそも日本では人気がなく、大きく広がりそうな気配もないので当然ではある。国内プラットフォームに乗って成長するというサイクルを続けてきた日本企業にとっては、参入は容易ではない。世界的に競争も激化している。

Facebook の急成長によって、収益を上げている欧米企業は、日本のことを気にかけていない。なぜなら、日本以外の地域でユーザー数は増加し続けているため、自らがいる市場に注力していればいいからだ。

アイテム課金市場は、スミス氏によると北米だけでによると、2008 年が 5 億ドルで、2009 年が 10 億ドルだったのが、2010 年は 16 億ドルにまで急成長を続けると予測している。日本の家庭用ゲーム機のソフト市場は、約 3300 億円であるため、その半分のサイズにまで達する計算だ。

2 億 4000 万人ものユーザーを抱えている、トップパブリッシャーの Zynga は、新しいゲームを開始から 7 日間で、ゼロから 1000 万人まで集めるだけのパワーを持っている。実際、「FarmVille」は 11 人で 5 週間で開発してリリースしている。しかし、一度リリースして成功したら、人海戦術でアップデートを行い、コンテンツを充実させていく。

そして、ユーザー行動を分析して、的確に有料のアイテムを提供していく。それを繰り返して新しいゲームを順次リリースしていく。

Zynga はすでに巨大戦艦に喩えられるほど強力な企業になっており、他の企業にとって対抗していくことが容易でない。これも市場の成熟を加速化させる要因になるかもしれない。

6.1.10　終わりに

これらの新規プラットフォームと、新しい市場形成が、日本のゲーム産業と関係ないところで進行し、世界のゲーム産業の様相を劇的に変えようとしている。富が生まれる場所がシフトしつつあり、完全に日本は後追いになりつつあるという実情は、今年の GDC から明確に見えてきた実情である。

今、日本では、Facebook は人気がないためにこれらの大きな変化を実感することは容易ではない。しかし。自社単独で既存のゲーム機の受注仕事の収益を使って、小さくソーシャルゲーム開発していくというペースでは、とても世界の速度に追いつけないと危機感を募らせている、現地に参加した開発者の声は少なからずあった。

変わらなければ、ある日突然、押し寄せる津波に飲み込まれる。現地で、変化の津波を実感した日本の開発者の声は多かった。日本市場にこの大きな変化が押し寄せるまでの猶予の時間はそれほどないだろう。

6.2 GDC2010 における海外のゲーム関連技術についての調査

三宅　陽一郎（株式会社フロム・ソフトウェア）

6.2.1 概論

GDC2010 は、昨年よりやや規模の縮小した形で開催された。2010 年という年は、次世代機(ここでは 2005 年以降の新プラットフォーム、PlayStation3, Xbox360, Wii 世代機を指す)に入って、2 つ目の次世代機タイトルのリリースが始まる時期であり、1 つ目のタイトルで構築した開発基盤の上に、2 つ目のタイトル開発の技術を着実に積み上げているという印象を強く受けた。ここでは、特にゲーム AI 分野の講演に特化して、2 つのタイトル「Killzone 2」(Guerrilla Games, 2009) [1]と「The Sims 3」(EA, Maxis,2009) [2]（邦題：シムピープル）のゲーム AI 技術の詳細を解説し、対照的な AI の実装を要求する両タイトルにおけるゲーム AI 技術の比較を通して、ゲーム AI 分野全体の広がりを模索して行く[3]。最後に、この先鋭的な仕事を通して、これからの 10 年のゲーム AI 技術の展望を解説する。ここでは、AI と言った場合、キャラクター、及び、キャラクターの持つ知能そのものを指すものとする。本報告のベースとなる講演は以下の二講演である。

- Alex J. Champandard, "On the AI Strategy for KILLZONE 2's Multiplayer Bots"(AI Summit: Case Studies: AI in Recent Games)　[1] [4]
- Richard Evans. "Modeling Individual Personalities in The Sims 3" [2]

6.2.2 Killzone 2 の AI 技術

(1) Killzone 2 概観

Killzone 2 は FPS（一人称視点シューティングゲーム）であり、オフラインモードとオンラインモードが存在する。オフラインモードについては、GDC2009 の講演に基づき昨年の報告書で解説した[5]。Killzone 2 のパス検索システム、射線判定システムについては、そちらをご覧頂きたい。また、Killzone のパス検索システム、射線判定システムも、2008 年度の報告書で解説している[6]。Killzone 2 のオンラインモードは、2 チーム 16 人に分かれた 32 人の対戦であり、AI がチームを組んで行動することが出来る。今回の、GDC2010 の講演は、このオンラインモードにおける AI 技術の講演であり、以下、この講演の内容とその背景を解説する。

人工知能には、記号操作を主として知能を構築しようとする流れと、ニューラルネットのように数値的なシステムで知能を構築しようとする流れがある[7]。前者は明示的に知識

を表現する方法であり、後者は知識は非明示的に数値の配列の中に含まれている。

Killzone 2 は前者のようにシンボリックな思考システム（HTN プラナー）を持つと同時に、後者のように戦局全体を数値的ダイナミクスによって認識する知能を持った AI である。Killzone 2 のマップの特徴は、前作同様に、は障害物が多く段差が激しく、立体的構造物に満ちている。Killzone 2 の AI 技術が目指していたものは、こういった複雑なマップでも、長期間（数分）に渡って AI が正しい位置取りをしながら、戦術的に行動し、かつ AI チームとして大局的に戦略的に行動できる知能の構築である。

(2) エージェントとチームの関係

各エージェント（AI）が賢明である、という意味と、チームとして賢明であることは意味が異なる[8]。各エージェントは知覚できる周囲の局所的な情報に基づき行動するのに対して、チームは、大局的な戦局の趨勢に基づいて行動しなければならない。また、チームとそこに属するエージェントは無関係ではない。チームには組織としての構造と、情報の流通が整備されなければならない。Killzone 2 の AI では、最も上位に属する「Strategy AI」という司令官 AI の下に「Squad AI」（部隊 AI、チーム AI）が複数配置され、「Squad AI」には複数の「Individual AI」が属する。命令は上から下へ伝達されるが、各 AI が判断した局所的戦局が、上位へと伝達される。また、「Individual AI」（個体 AI、エージェント）は、命令がない、或いは無効化された場合には、上位へ向けて命令の再発行を求める。

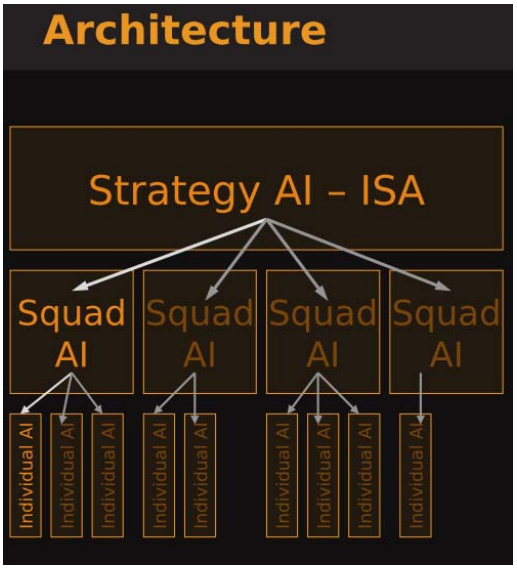


図 6.2-01　Killzone 2 における AI 組織図[1]　階層構造になっている。

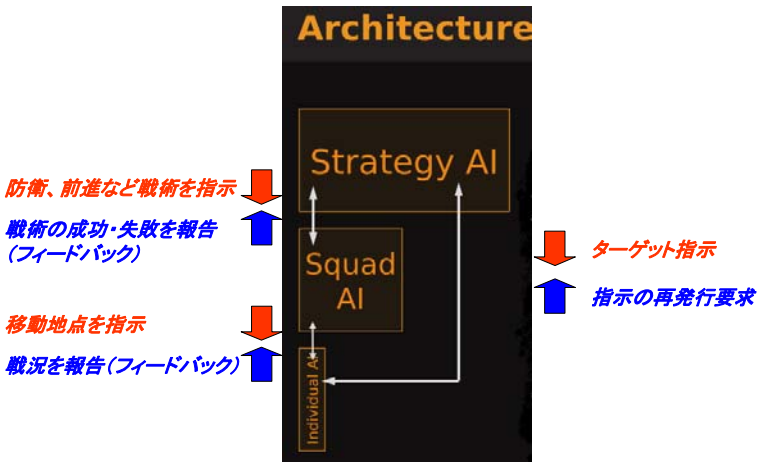


図 6.2-02　Killzone 2 における AI 組織の階層構造間の情報伝達フロー[1]

各階層の間には相互の情報の流れがある。上位から下位には「命令」が、下位から上位には「報告」「要求」が伝達される。

(3)　各 AI のアーキテクチャー

各エージェントのアーキテクチャーは、HTN プラナー(Hierarchical Task Network Planner)をコアとするエージェント・アーキテクチャを取っている。プラナー[7]とは行動プランを生成するシステムのことで、ここでは一連のタスクを階層的に生成するアルゴリズム(HTN Planner)である。周囲の脅威(threat)の情報、上からの命令(order)、AI 同士のメッセージ(message)などから、現在の周囲の環境世界（World State）を再構成し、そこからタスクプランを生成し、順番にそれを実行して行く(task execution)。

各AIのアーキテクチャ

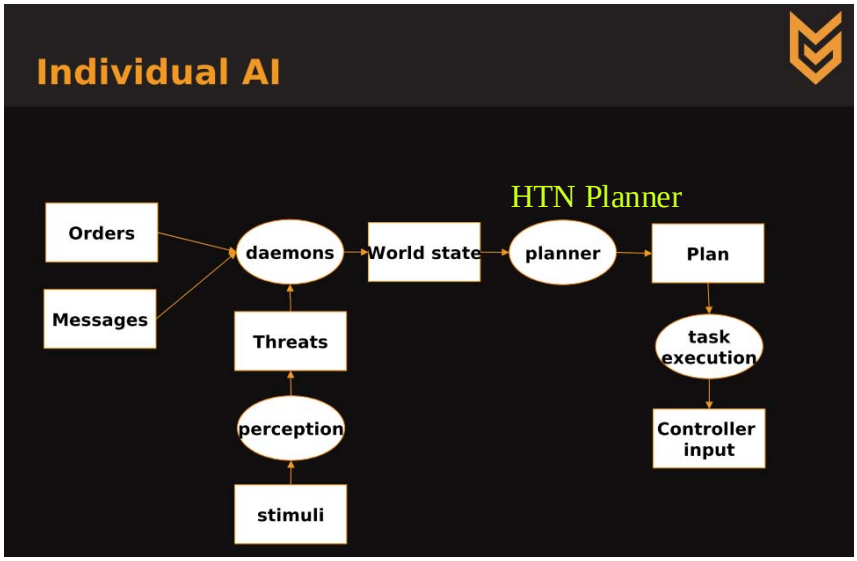


図 6.2-03　Killzone 2 の各エージェントの思考構造[1]

階層型プランニングとは、階層的な分解によって最終的なタスクリストを導くアルゴリズムである[9] [10] [7]。階層型タスクネットワーク(HTN)は、タスクに分解するところまでは同じであるが、分解によって（論理的に）導出したタスクを、実行可能な順序に従って向き付けられたネットワーク状に再構成する過程を含むものである。また、この過程で、情報が補完されることでタスクがカスタマイズされる過程を含む。最終アウトプットは、タスクを要素とする向き付けられたタスクネットワークである。

「Killzone 2」の場合、タスクネットワーク形成の部分は単に分解したタスクを単にスタック状に並べて実行するので、それほど凝った HTN にはなっていない（むしろ、単に階層型プランニングと言った方がよい。おそらく前提条件が設定されているのと、実行順序が規定されるので HTN と言っているのであろう）。公開された擬似コードの部分には、「前提条件」(precondition)とタスクがセットになったコードが階層的に記述されており、ゲーム状況によって満たされた「前提条件」に対応するタスクリストが実行タスクとしてスタックされて行くことになる。

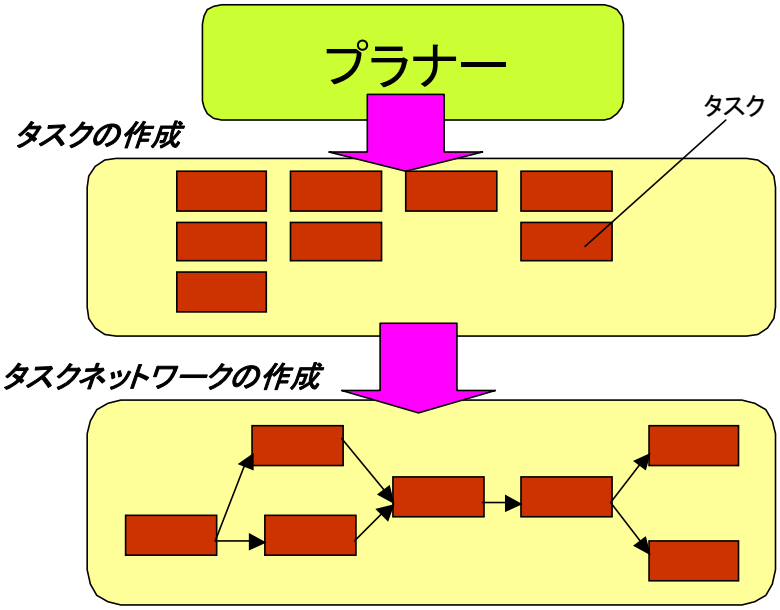


図 6.2-04　HTN プランニングのイメージ

導出したタスクの集合を規則に従ってタスクネットワークとして再順列する

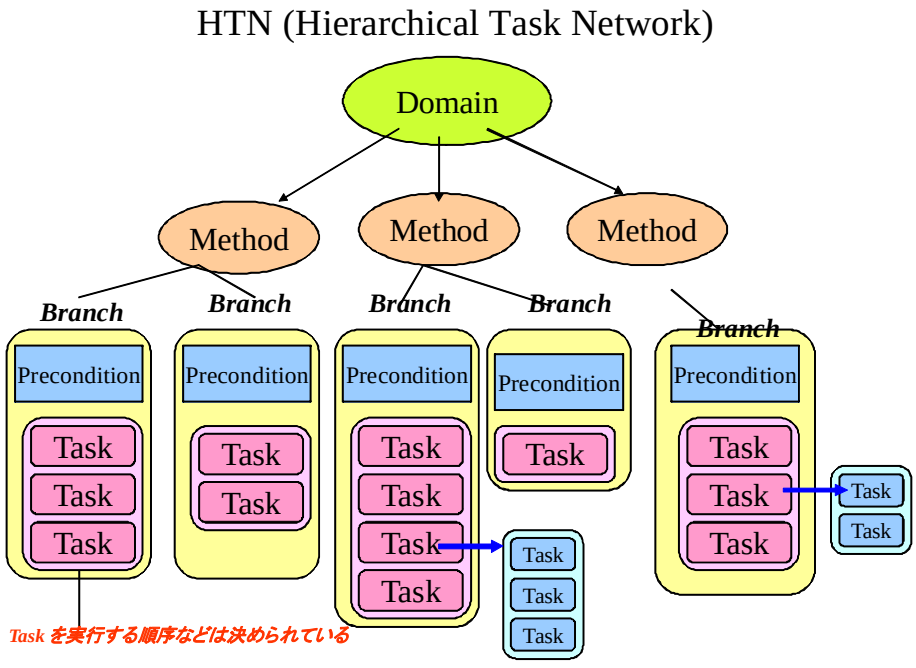


図 6.2-05　Killzone 2 における階層的タスク分解のイメージ

前提条件(precondition)をクリアしたブランチでタスクが階層的に分解される。同時に、各タスクは実行順序も決定される。

HTN (Hierarchical Task Network)

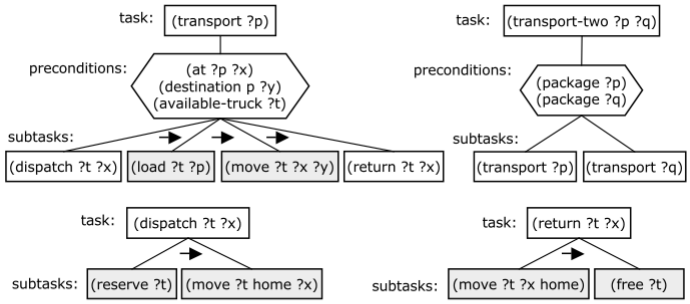


Figure 1: Methods for transporting a package ?p, transporting two packages ?p and ?q, dispatching a truck ?t, and returning the truck. Arrows are ordering constraints. The shaded subtasks are *primitive* tasks that are accomplished by the following planning operators: (load ?t ?p) loads ?p onto ?t; (move ?t ?x ?y) moves ?t from ?x to ?y; (reserve ?t) deletes (available-truck ?t) to signal that the truck is in use; (free ?t) adds (available-truck ?t) to signal that the truck is no longer in use.

Dana Nau et al., “SHOP2: An HTN Planning System”,
Journal of Artificial Intelligence Research 20 (2003) 379-404

図 6.2-06　一般的な HTN の解説(1) [11]

ここでは 2 つのタスクが前提条件を満たした場合の分解が解説されている。

HTN (Hierarchical Task Network)

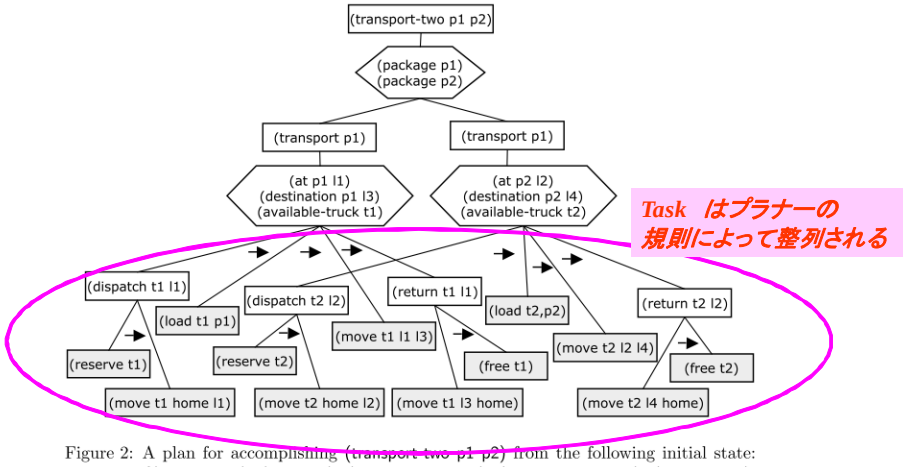


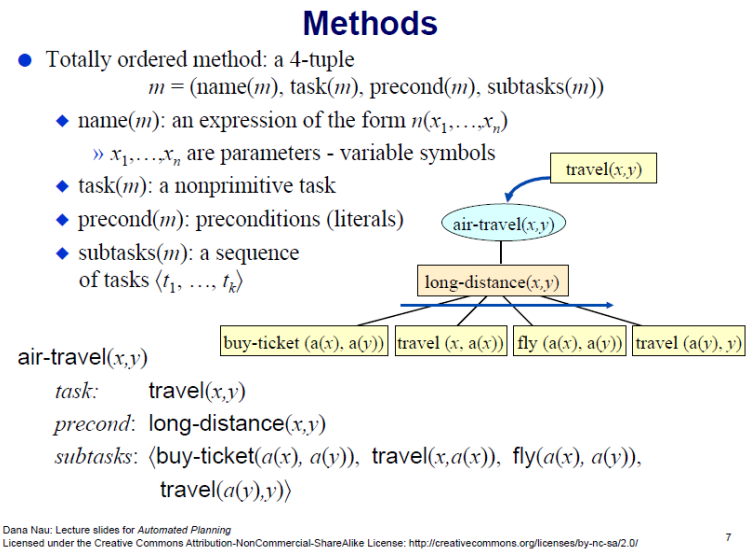
Figure 2: A plan for accomplishing (transport-two p1 p2) from the following initial state: {(package p1), (at p1 l1), (destination p1 l3), (available-truck t1), (at t1 home), (package p2), (at p2 l2), (destination p2 l4), (available-truck t2), (at t2 home)}.

Dana Nau et al., “SHOP2: An HTN Planning System”,
Journal of Artificial Intelligence Research 20 (2003) 379-404

図 6.2-07　一般的な HTN の解説(2) [11]

一つ前の図の二つのタスク生成が同時に起こった場合、生成されたタスクの実行順序が調整され、再順列されていることがわかる。

HTN (Hierarchical Task Network)

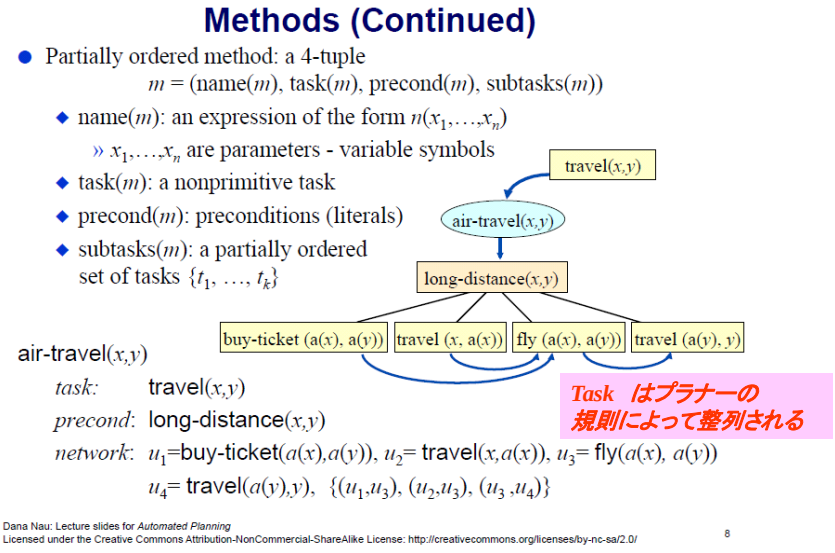


Dana S. Nau,"Hierarchical Task Network Planning", Lecture slides for Automated Planning: Theory and Practice http://www.cs.umd.edu/~nau/cmsc722/notes/chapter11.pdf

図 6.2-08　一般的な HTN の解説(3) [12]

タスク生成と全タスクの実行順序が subtasks(m)に指定されている。

HTN (Hierarchical Task Network)



Dana S. Nau,"Hierarchical Task Network Planning", Lecture slides for Automated Planning: Theory and Practice http://www.cs.umd.edu/~nau/cmsc722/notes/chapter11.pdf

図 6.2-09　一般的な HTN の解説(4) [12]

一つ前の図で生成されたタスクがネットワーク的に再順列されている。タスクネットワークの生成である

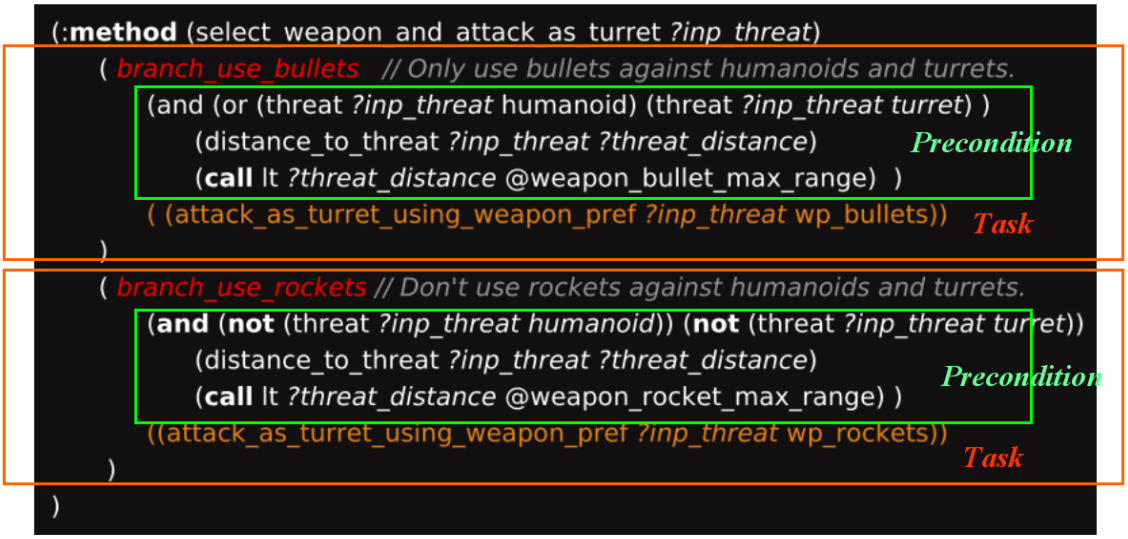


図 6.2-10　Killzone 2 の HTN プラナーの擬似コード[1]

前提条件とその下のタスクがセットになっていることがわかる

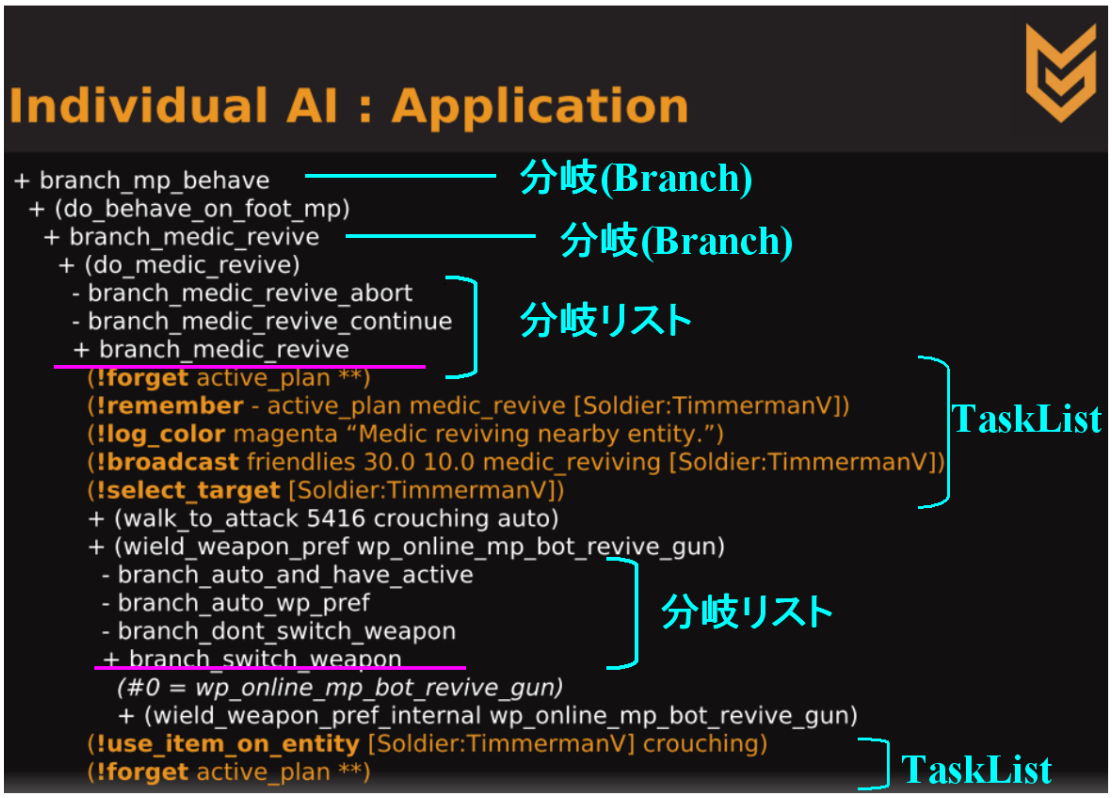


図 6.2-11　Killzone 2 の擬似コード[1]　タスクが実際に展開されて行く様子

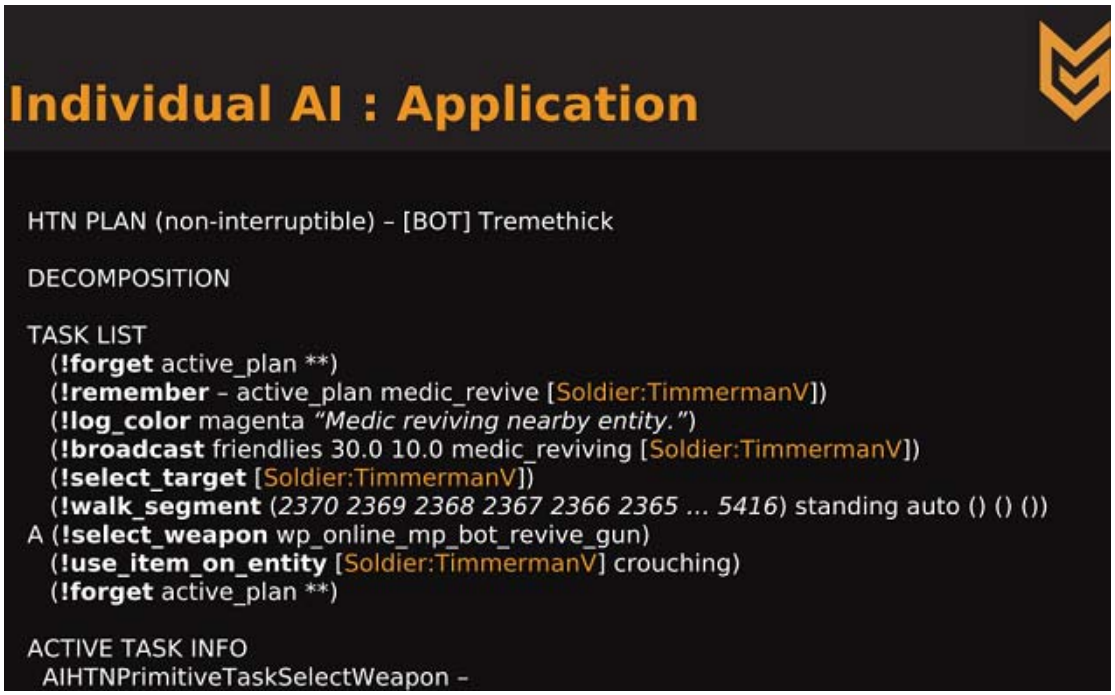


図 6.2-12　Killzone 2 の擬似コード[1

生成されたタスクリスト　A は現在、Active なタスクを表現している。分解された順番に実行される。これは兵士を介抱しに行くプランである。

(4)　Squad AI

Squad AI の思考構造も、殆ど、各 AI の思考構造と同じである。ただ、命令(order)は、上位の「Strategy AI」から降りて来るものであり、また、下位の各エージェントからメッセージが届く仕組みである。それらの情報から現在の周囲の環境情報を再構成し、HTN プラナーによって、兵士への命令リスト（タスクネットワーク）をアウトプットする仕組みである。

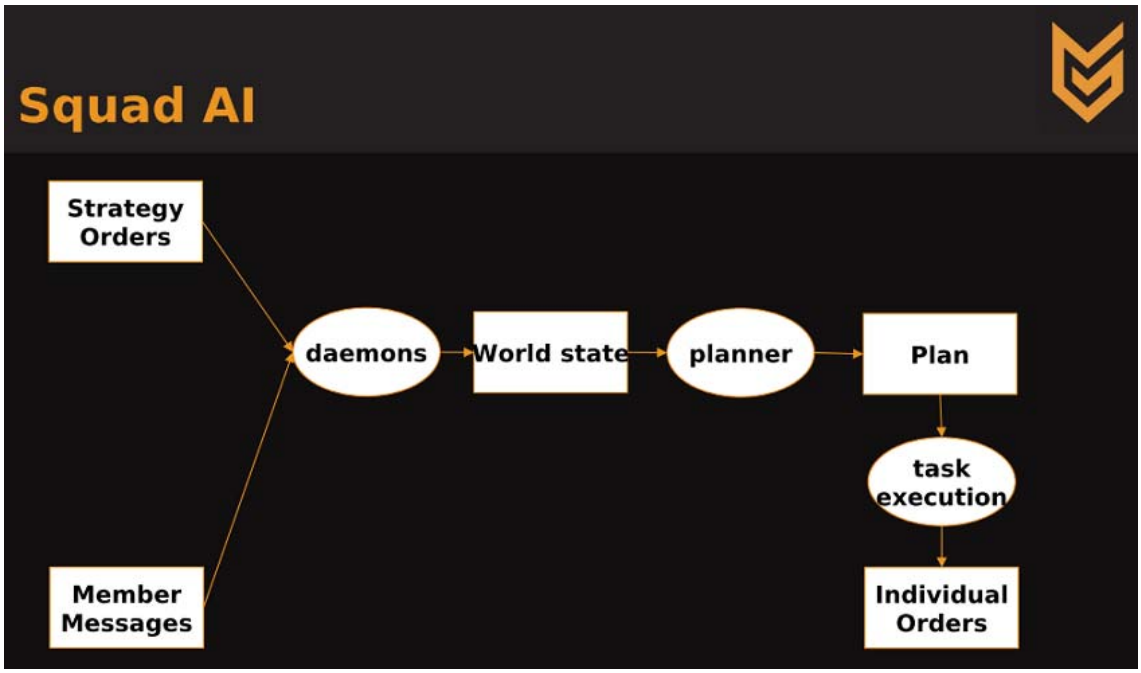


図 6.2-13　Killzone 2 の Squad AI の思考構成 [1]

(5)　動的なチーム生成

さて、ここで、GDC2008 で講演のあった Halo3 における Squad の生成について思い出してみよう(2008 年度の報告書で解説) [6,13]。Halo3 における Squad は、あくまでゲームデザイナーの手によってカスタマイズされたものであった。そして、そうして作った複数の Squad たちを、戦況が必要とする複数のチームタスクに割り当てたのであった。これは、比較的狭いエリアで戦場を作り出すという目標があった上で、Squad を形成する Halo3 のゲーム性の要求から来ている。一方、プレイヤー、AI が流動的にマップ全域を動き回る Killzone 2 においては、Squad は戦況に応じて動的に作られる。集められるメンバーは、目的に応じて、或いは、コアとなるメンバーからの近い距離にいる AI を Squad メンバーとして割り当てられる仕組みとなっている。

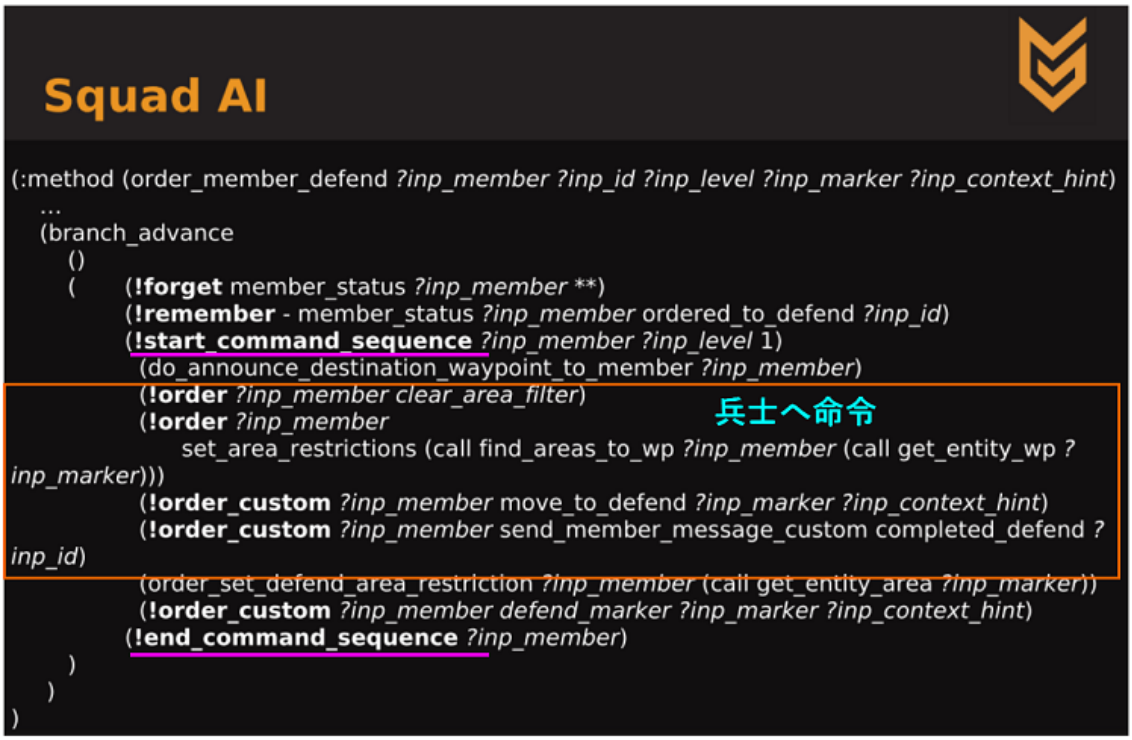


図 6.2-14　Killzone 2 Squad AI の HTN プラナーの擬似コード[1]

条件に合う兵士を探して命令を送っている

スカッドを目的に応じて動的に構成

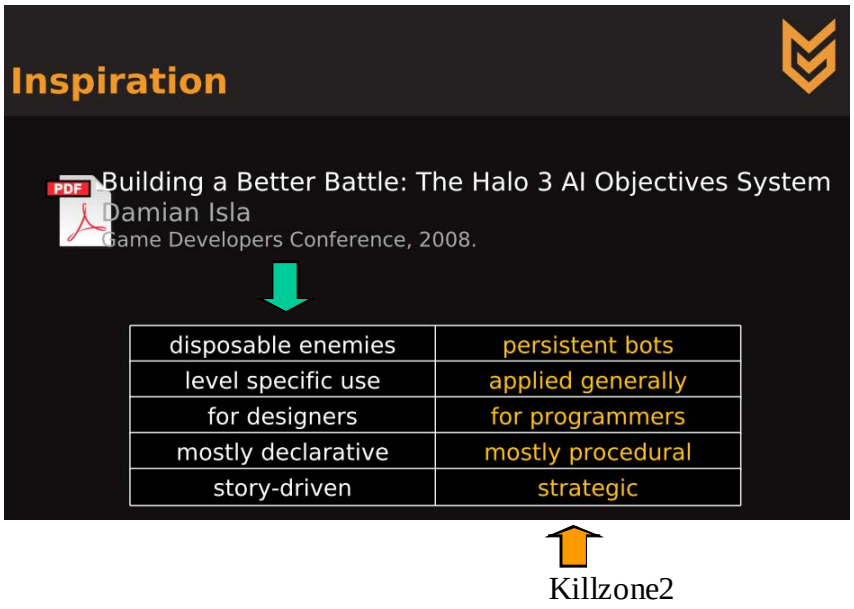


図 6.2-15　Halo3 と Killzone 2 の Squad 編成の比較[1]

Halo3 の Squad はデザイナーが事前に指定するが、Killzone 2 では動的にゲーム内で構成される

(6) 戦略思考

最上位の Strategy AI、Squad の戦略的思考について解説する。まず、戦略的思考の基礎となるのが、戦況全体の把握である。Strategy AI に各 AI から報告が上がって来る情報は、兵士がいる場所の情報の集合に過ぎない。戦況を把握するには敵兵士全体、味方兵士全体の勢力を常にモニターしておく必要がある（これはチートと呼ぶかどうかは、それほど簡単な議論ではない。一切、実際に知覚したデータ以外の情報を廃するならば、それは人間的にも正確ではない。人間は見えない情報も、とりあえずなんとか補完した上で行動している)。

まず、そういった戦局把握のために、マップ全体地形をウェイポイントで埋め尽くす（実は、こういった工程はそれほど自明ではない。ナビゲーションメッシュやウェイポイント自体を自動的に配列、かつ質が高くできればよいが、複雑な地形においては技術的な困難がある。Killzone 2　が自動的に配置しているのか、デザイナーが手動で置いているのかは不明)。勿論、このデータの第一の目的は各 AI が移動用に用いるデータである。各 AI のために、各ポイントに動的に情報を埋め込むことも可能だが（例えば、敵のいるウェイポイントに印を付ける)、Squad が用いる場所スケールとして小さ過ぎる。そこで、一定数のウェイポイントを 1 クラスター（塊）として、マップ全体をクラスター分割する。具体的には、一定数のウェイポイントがなるべく四角になるように自動処理して分割して行くのである。こうしてマップを一定の大きさに分割した後は、各クラスターの隣接情報を再構築する。隣のクラスターとのリンク情報を用いて、クラスターを要素とするネットワークグラフを構築する。こうして出来たグラフを戦略グラフ(Strategic Graph)と呼ぶ。そして、この戦略グラフ上に動的に戦局の情報を、影響マップ（Influence map）のテクニックを用いてモニターして行くことで戦局を把握するのである。

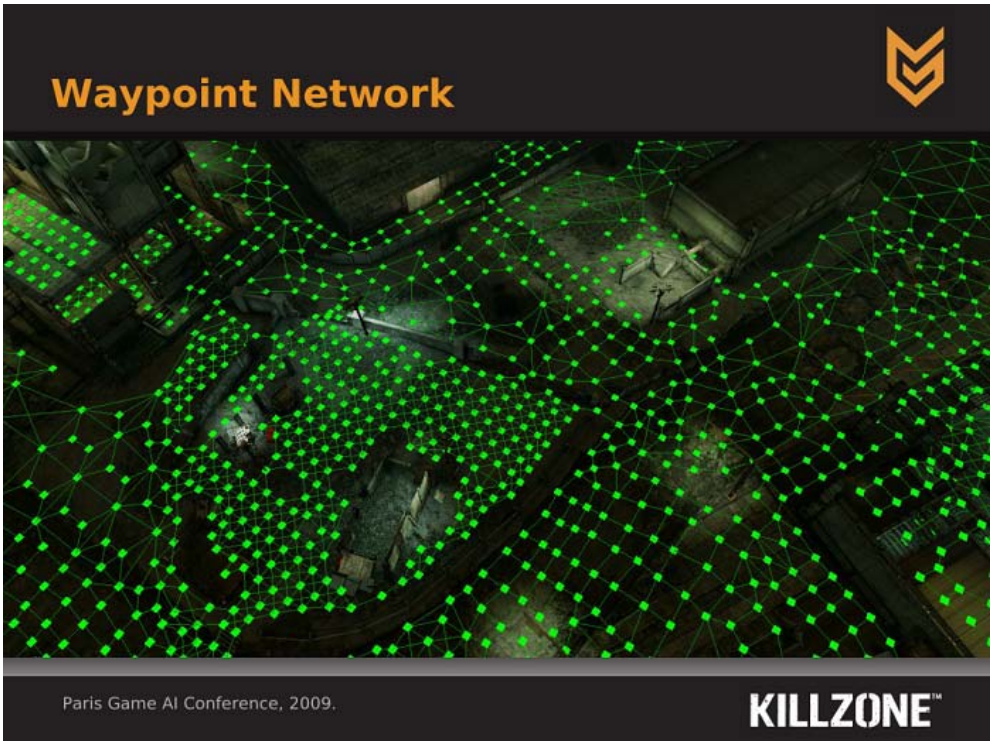


図 6.2-16　Killzone 2 のウェイポイントマップ[1]

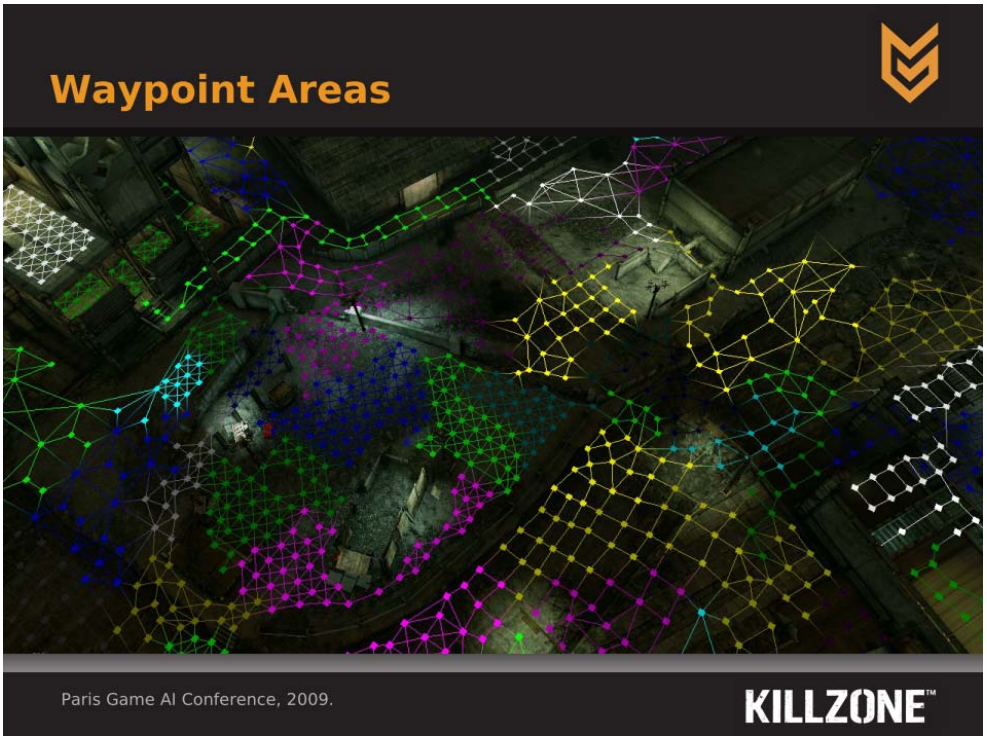


図 6.2-17　ウェイポイントのクラスター化[1]

ウェイポイントを一定数ごとに（なるべく）四角い形になるようにクラスター化（自動処理）

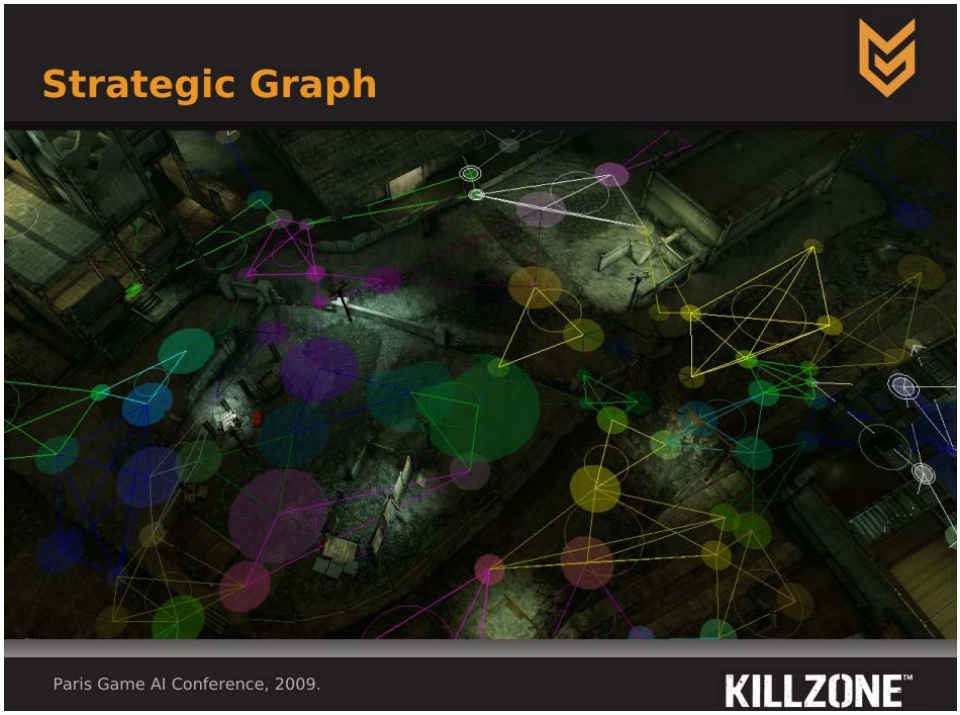


図 6.2-18　戦略グラフの作成[1]

クラスタ同士が接するポイントで各クラスターを結ぶ。クラスターを一つの要素として作ったグラフを戦略グラフと呼ぶ

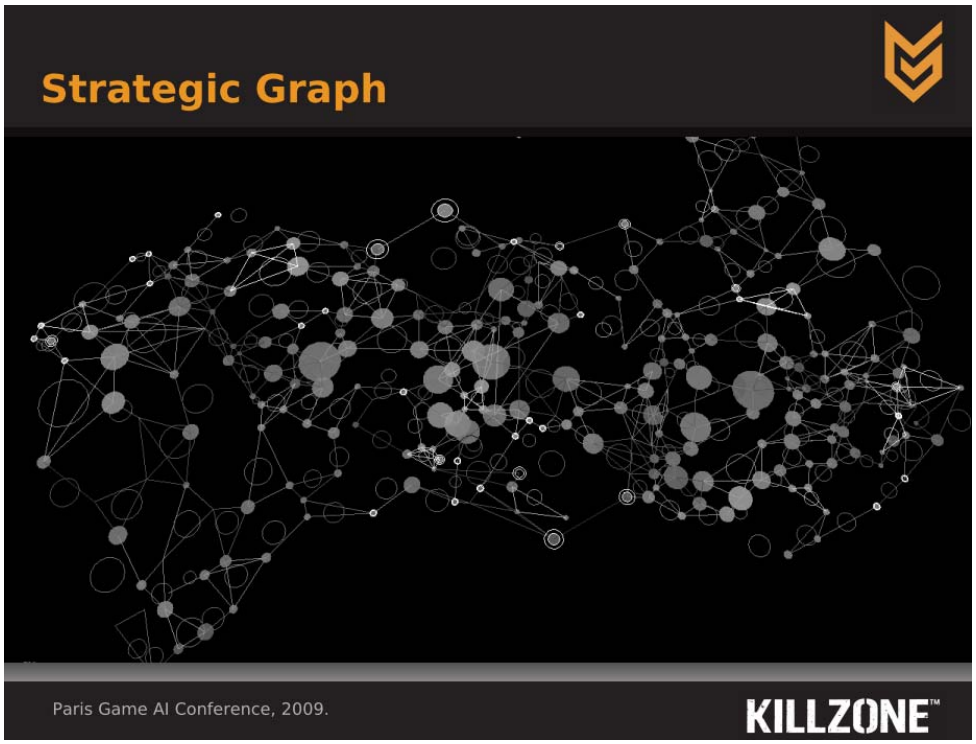


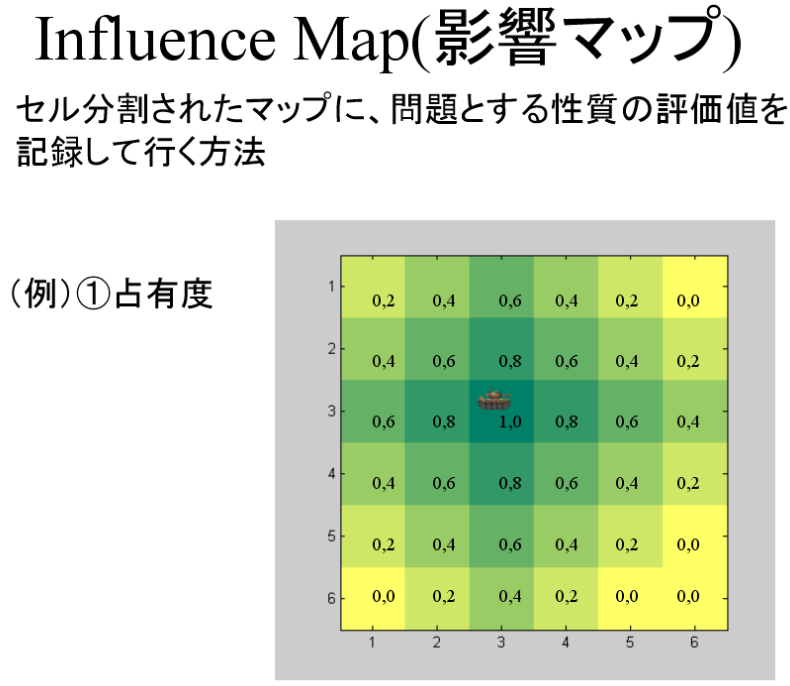
図 6.2-19　Killzone 2 戦略グラフ[1]

戦略思考の基礎となる世界表現である。

(7)　影響マップ

影響マップ(Influence Map)の方法は、戦略ゲームでよく用いられる方法で、マップ上の勢力情報などをモニターするために用いられる。影響マップ原理は単純で、影響を与える（例えば、脅威となる影響）キャラクターがいる位置から距離などに応じて、その影響度を減少させつつマッピングして行く。逆（マイナス）の影響を持つものがいる場合は、マイナスの値をマップ上にマッピングして行く。最終的に、複数のキャラクター、或いは複数のオブジェクトからの影響度を重ね合わせる（足し合わせる）ことで、マップ全体の「影響度マップ」が出来上がる。この影響度マップを基に、例えば、敵の勢力の強い地域や弱い地域が把握できるようになるという仕組みである。「Age of Empire」(Ensemble Studios) シリーズでは、影響マップを地形解析に用いており[14]、また、SimCity シリーズ(EA、Maxis) では、動的に街の発展アルゴリズムとして用いている[15]。

Killzone 2 における影響マップは、味方の死んだポイント、砲台、キャラクターの位置の情報を基に、敵、味方の脅威度を計算し戦略マップにアップデートされる。各 Squad を目的に応じたパス検索アルゴリズムを所持しており、エリアを限定（防衛 Squad なら味方陣地周辺など）したり、影響度マップの情報を利用して Sqaud のパスを決定する。例えば、敵と遭遇をすることが目的であれば、危険度の高いエリアを通るようなパスを決定すればよく、安全な道を通りたければ、敵の脅威度の少ないパスを辿ればよい。



http://www.fdaw.unimaas.nl/education/4.5GAI/slides/Influence%20Maps.ppt

図 6.2-20　影響マップの一般的解説(1) [16] 戦車の占有度の影響力のマッピング

この他の敵、味方の戦車の影響度などを加味しつつ全体の影響マップが出来上がる。

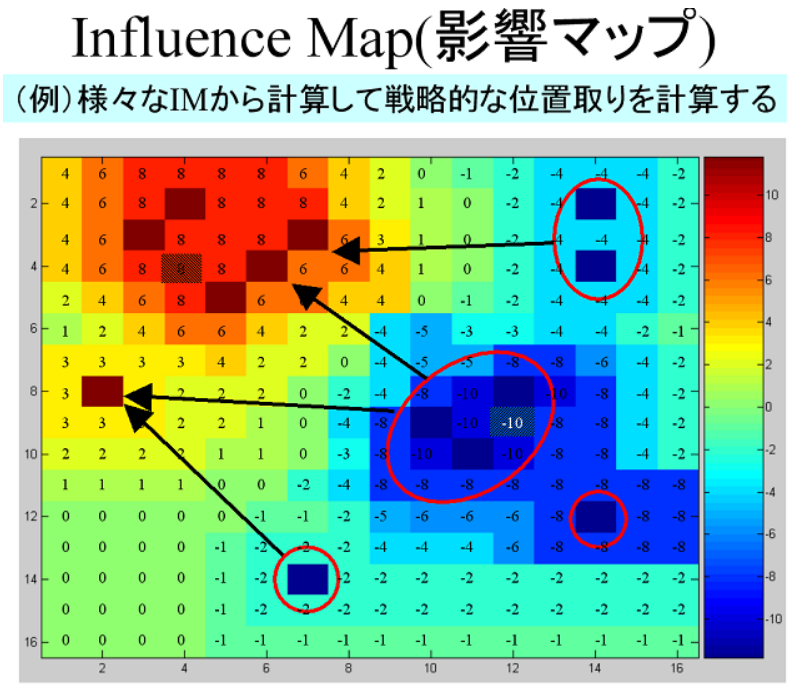


図 6.2-21　影響マップの一般的解説(2) [16]

敵の勢力が赤、味方の勢力が青として影響マップが形成されており、この情報を基に戦略的移動を行う

図 6.2-22　Killzone 2 の影響マップ[1]

先に作成した戦略用グラフ上にリアルタイムに敵、味方勢力の影響度をマッピングして随時アップデートする。

図 6.2-23　Killzone 2 の戦略的パス検索[1]

例えば、影響度をもとに敵と遭遇しにくい、或いは遭遇しやすいパス、或いは防衛エリア内でのパスなど、ミッションに特化したパス検索を行うことが出来る。

(8)　Killzone 2 の階層構造

以上のように「HTN プランニング」「戦略マップ上の影響マップ手法」が Killzone 2 の主要な技術であるが、この節では、Killzone 2 に見られる AI システム全体の性質について考察してみよう。

Killzone 2 AI では AI システムの随所に階層構造が取り入れられている。

- 「Strategy AI - Squad AI - Individual AI」階層構造、
- 「HTN プラナー」階層構造
- 「Strategy Map - Waypoint Map」階層構造

という 3 つの階層構造が用いられている。

階層構造を導入することは、AI にスケーリングを与えることである。スケーリングとは、ここでは複数の尺度で物事を捉える能力のことである。高度な知性であればあるほど、時間、空間における複数のスケーリングを持っている。例えば、時間においては、瞬時、3

分、1 時間、1 年、一生など、知性はさまざまな時間スケールの思考を行うことが可能であり、また、空間においても、身の周り、街、国、地球、など、様々な空間スケールの思考が可能である。こういったスケーリング能力を持つおかげで、人間は、世界の様々な諸相を捉え思考することが出来るのである。

こういった高度な AI を実現するには、逆に、階層構造を人の手で AI に導入すればよい。上記の階層構造はそれぞれ、組織としての階層構造は時間と空間のスケーリングを（上位になればなるほど長期大局、下位になるほど、短時間局所）、HTN は時間的階層構造（長期プランから短期プランに分解して行く）、Strategy Graph は、Waypoint という局所的スケールから、大局的な戦略的なスケールへと AI の思考を移す空間階層化の機能を実現しているのである。

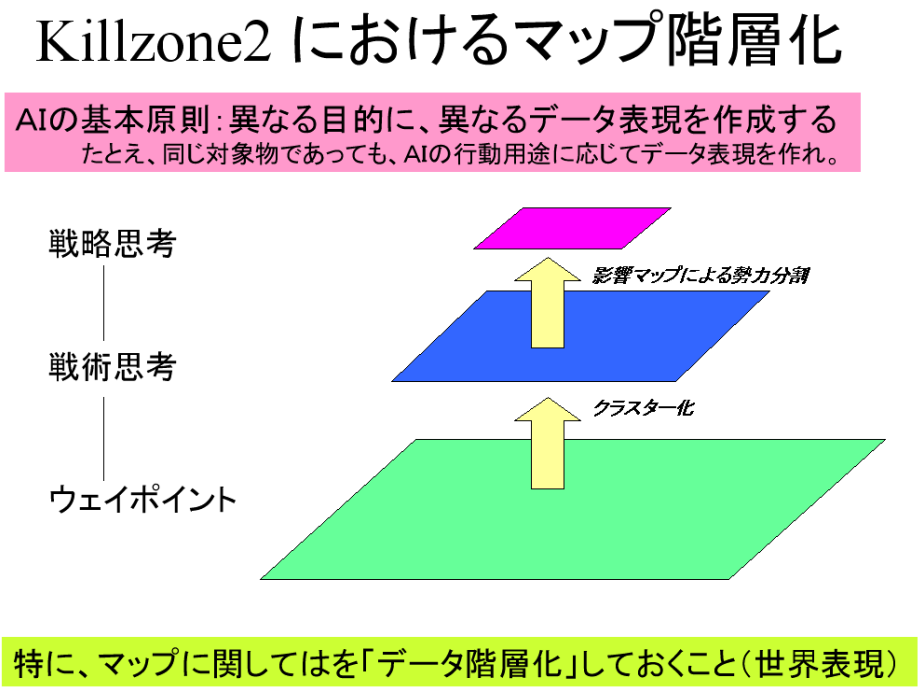


図 6.2-24　Killzone 2 における地形データ階層化

もとの地形からまずウェイポイントデータへ、さらにそれをクラスター化した戦略マップ、そして、その上にさらに影響マップを用いて勢力図に分割している。こうした階層化は、各階層スケールの思考を可能にする。例えば、ウェイポイントは詳細な地形に応じたパス移動を実現し、戦略マップはクラスターを粒度とする思考を可能にし、影響マップはマップ全体の勢力図を見越した戦略思考を可能にするのである。人工知能の標語に「異なる目的には、異なるデータ表現を作成せよ」という言葉がある。知能の機能は、事物の表現と思考がセットになって初めて実現する。思考だけでは知能ではない。よって多層な表現と各層毎の思考は高度に構造化された知能を可能にするのである。

(9) Killzone 2 AI のまとめ

Killzone 2 のオンラインモードにおける AI の仕様についてまとめて来た。オンラインゲームは複数のプレイヤーが相手であるから、AI システムは、常に敵・味方問わず、プレイヤー全体の動向を把握しながら意思決定を行う必要がある。そこで、戦略マップによって戦局をモニターしながらスカッドの大局的行動を決定し、個々の AI は HTN プラナーによって局所的な戦闘の思考を精緻に作りこむという方法を取った。実際、オンラインの長期的な戦闘に耐えるためには、プラナーによる行動のスケジューリングなしに人間らしい行動を構築することは難しい。Killzone 2 は、前作の Killzone の上に、これまで伝統的に使われて来た影響マップの手法を新しい形で応用すること、各 AI に新しくプラナーを導入することで、全体として柔軟かつ堅牢な AI システムを作り上げた。

6.2.3 The Sims の AI の技術

The Sims というゲームは先の Killzone 2 と対照的に、平和な街の中で自律的に動作するたくさんの人（AI）がインタラクションしている様子を見ながら、時々、命令したり、オブジェクトを配置したりする、所謂、箱庭ゲームである。そこで必要とされる AI 技術も自然、FPS と違ったものとなり、また、同じ技術でも用い方が全く違って来る。そういった意味で、この両極にある 2 つのゲームの AI 技術は鋭い対照を成す。この対照を解説することは、ゲーム AI 技術全般に示唆を与えるものである。以下では、GDC2010 における講演[2]を踏まえて、The Sims の AI システムをシリーズを通した総合的な解説を行う。

(1) The Sims が求める AI

街や家の中で社会的な生活を営む AI の街を再現する、そして、プレイヤーがそれを観察しながら、上から影響を及ぼす、それが The Sims というゲームである。AI は、0.1 秒やわずか 10cm で生死を分かつ FPS の戦場ではなく、ゆったりとした時間の中で、様々な人（別の AI）や物とインタラクションしながら生活を営む。「AI の人（他の AI）や物とインタラクション」を実現することが、The Sims の AI の本質と言っていいだろう。FPS の作る戦場という極限状態では、単に遮蔽物としてしか意味を持たなかった様々な構造物も、或いは敵・味方としてしか意味を持たなかった他の AI も、The Sims では、多様なインタラクション、多様な人間関係の中で、新しく多様な意味を持ち始める。そういった人や物との多様な関係性を積み重ねて行くところに、The Sims のゲームとしての奥深さがあるのである。

そしてゲームとしては、このような AI によって「街の中で自然発生的にさまざまなドラマが産まれる」ようダイナミクスを実現することを目標にしている。

(2) The Sims AI の原理

The Sims 1,2 AI の原理は昨年の報告書でも解説しているが[17]、The Sims 3 への変化を見るために、もう一度総合的に解説しておこう。

FPS の AI は殺人衝動と保守本能という極限化された内面状況によって駆動しているが、日常系の AI は、多様な内部パラメーター(動機パラメーターとも呼ばれるが、ここでは内面パラメーターと呼ぶことにする)が動的に変化するシステムとして内面を形成している。この内面は、精神的（メンタル、mental）パラメーター（楽しみ、部屋の快適さ、社交、エネルギー)、生理的（フィジカル、Physical）なパラメーター（尿意、空腹、衛生、快適さ）の 8 つで構成されている。The Sims の AI は、これらが変動パラメーターとして保持しており、環境、行動、時間がこれらのパラメーターを変化させる。直接このパラメーターを変化させることは出来ないが（ちょうど人間が自分の感情を自由にコントロールできないように)、AI は行為を通じて自分の内面の状態を変化させることが出来る。例えば、風呂に入れば衛生パラメーターが上がる、などである。また、自律的に変化するパラメーター（空腹）や外部環境によって変化するパラメーター（快適さ）がある。これらのパラメーターにウェイトをかけて足し合わせた値（Mood）を総合幸福度として、AI は基本的に、この総合幸福度を最大化する方向の行動を選択して行うのである(Motive Engine)。

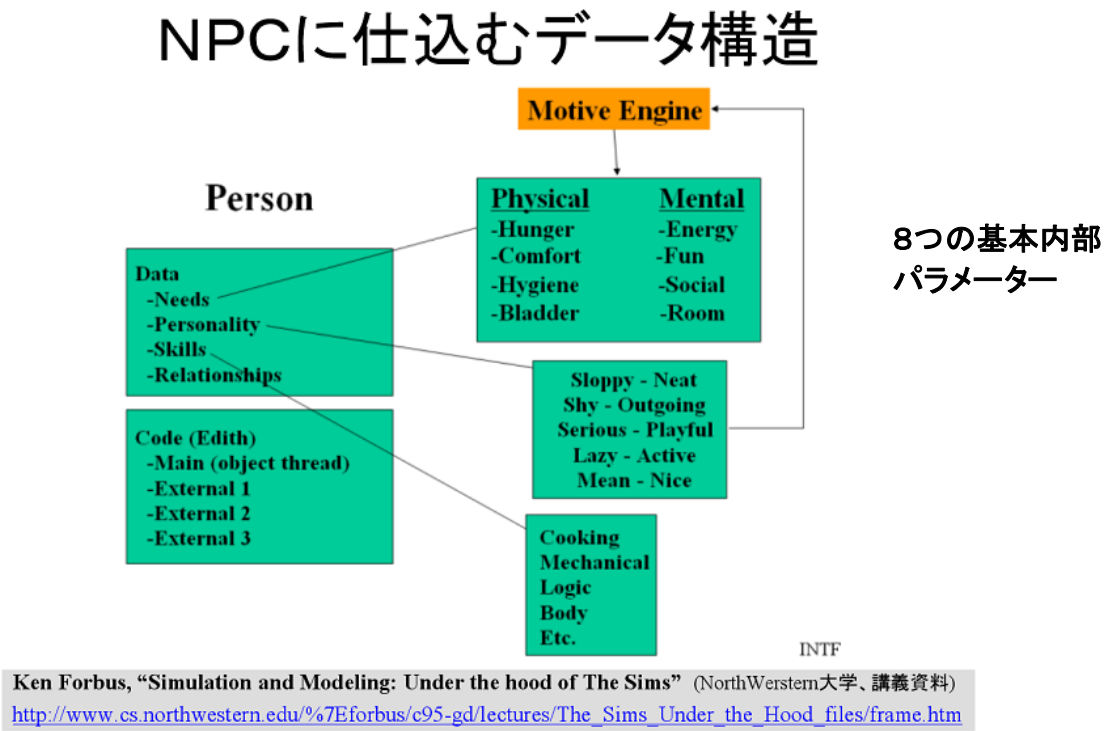


図 6.2-25 The Sims 1,2 における AI の内部の仕組み[18]

8 つの内面パラメーター(生理的な欲求と精神的な欲求、各 4 つからなる)の変動によって駆動する(Motive Engine）。この基本構造は The Sims 3 でも同様である。

(3)　The Sims の行動原理

The Sims の AI の行動は、一つの関係で結ばれている「内部パラメーター」「AI の行為」「環境のオブジェクトは」からなるシステムによって駆動される。これを解説する。

各オブジェクトには、そのオブジェクトに対して行い得る行為（アフォーダンスされた行為）の情報と、その行為による内部パラメーターの変化が定義されている。例えば、「風呂」オブジェクトには「洗う」「入る」などの行為リストが付随されており、そういった行為が上記の 8 つの内面パラメーターをどう変化するかが設定されている。つまり、「風呂」(Bathtub)に「入る」(Take Bath)ことで、内面パラメーターが変化し、その結果、Mood の値が変化する。

AI は常に周囲の複数のオブジェクトをリストアップし、さらに、その一つ一つのオブジェクトに関係する行動（例えば風呂ならば、洗うと入る）をリストアップし、この中から、Mood を最大化する行為を選択する。つまり、AI は、周囲のオブジェクトに対して行い得る行為とその効果を全て事前に計算した上で、Mood を最大化する行為（つまり最大のプラス変化をもたらす行為）を選択するのである。

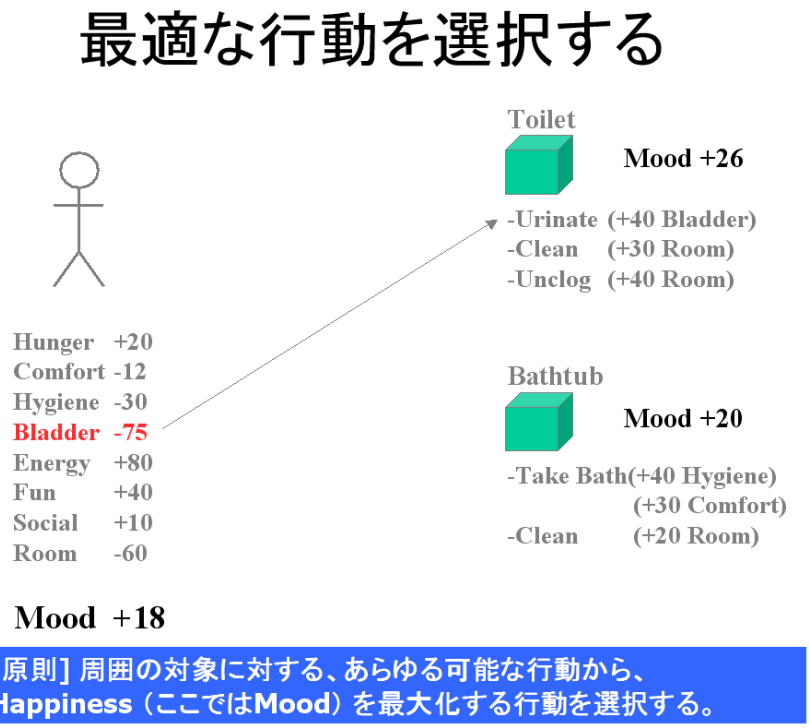


図 6.2-26　The Sims における最適な行動の選択原理[18]

オブジェクトには AI がそのオブジェクトに取りえる行動がリストされており、各行動は、AI の内面パラメーターを変動させる。その中で、Mood を一番上昇させる行為を選択する。ここでは、+40Bladder（尿意解消） の変化をもたらす Urinate(排尿する)が、総合幸福度を+26 にするため、この行動が選択される。

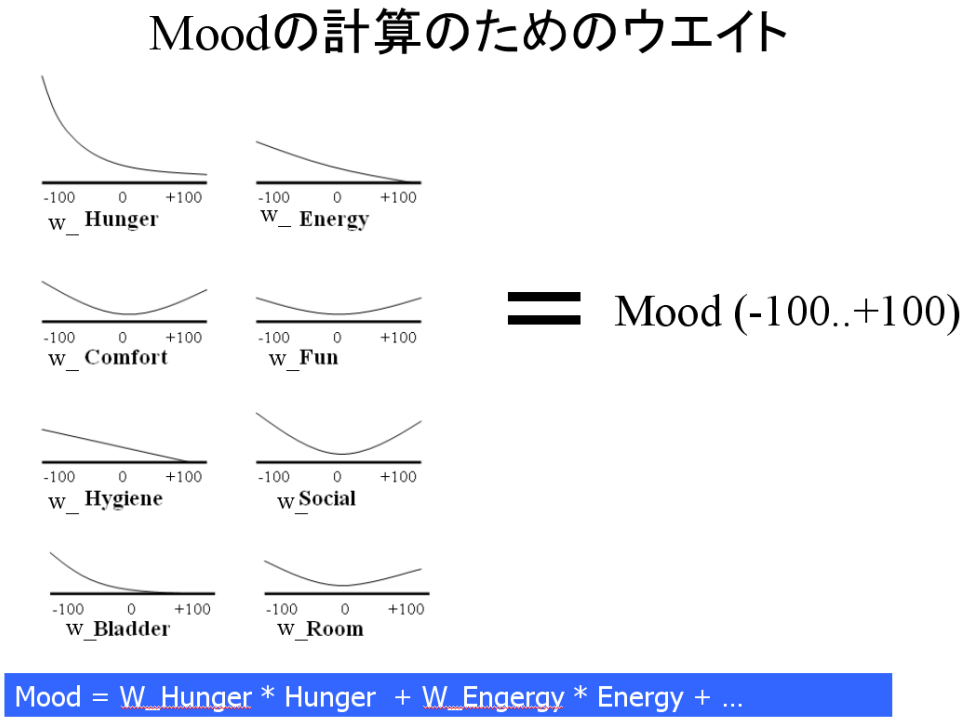


図 6.2-27　Mood 計算とウェイト[18]

上記の 8 つのグラフは、8 つの内面パラメーターが各値においてどれだけ Mood に貢献するかを表す。例えば、Social パラメーターは、W_Social というグラフは Social の値が X_Social の時、W_Social(X_Social)＊X_Social だけ Mood に貢献する、 Mood は同様に計算した全てのパラメーターの足し合わせである。

さて、ここでウェイトのグラフに注目して頂きたい。このウェイトのグラフは、各内部パラメーターが、Mood に対してどれぐらいの度合いで貢献するかを表している。例えば、Hunger パラメーターの値が今 70 とすれば、70 の時のウェイトグラフの値を W_Hunger(70)として、W_Hunger(70)*70 が Mood に貢献する値である。つまり、

Mood = W_Hunger(X_Hunger)*X_Hunger + W_Energy(X_Energy)*X_Energy + ...

という値が Mood の値になるのである(X_... は各パラメーターの現在の値とする)。Bladder、Energy, Hunger などは、値が大きくなると、ウェイトの値が 0 に近くになっている。つまり、こういった欲求は満たされれば気にしないし（Mood に貢献しない)、不足するとひどく幸福度を損なう（Mood にマイナスの作用をもたらす）ように、ウェイトグラフが設計されているのである。Hygiene(衛生)も同様である。一方、Fun(楽しみ)、Social(社交)といった精神的な楽しみは、満たされれば満たされるほど大きな幸福度を生み出すようにウェイトグラフが設計されているのである。こういった設計は人間の感情的性質を表現するように表現されているのである。

(4) 限界効用逓減（ていげん）の法則

また、ここからもう一つわかることは、一旦、あるパラメーターが満たされてしまうと、そのパラメーターを上昇させても、幸福度にさして変化がないということである。例えば、Social が 60 から 90 による変化は

Δ ＝ W_Social(90)*90 - W_Social(60)*60

である。一方、満たされていないパラメーターを満たす方が、幸福度は大きくアップする。例えば、Hunger が -80 から -50 になったとすれば、

Δ´＝ W_Hunger(-50)*(-50) - W_Hunger(-80)*(-80)

の変化分が Mood をアップさせる。つまり、一旦、あるパラメーターがある程度満たされると、そのパラメーターよりは、満たされていないパラメーターを上昇させる方が、全体の幸福度（Mood）に貢献する度合いが大きいということである。つまり、満たされていないパラメーターを変化させる行為が選択される傾向が強い。これも人間の性格をよく現している。最初のビールの一杯は、常にそれ以降の二杯目、三杯目よりも一番おいしいのである。これを経済学では、限界効用逓減の法則と呼ぶ。

(5) シミュレーションの負荷低減の方法

ここからはプログラミングのテクニカルなところであるが、この原理だと、AI の周囲にある全てのオブジェクトに対する Mood 変化率を計算しないと、行為を選択できない。実際、The Sims1,2 では、あらゆる計算を行っていた。しかし、殆どのパラメーターは普段、ある程度満たされている（らしい）。そういったパラメーターを変化させても、それほど Mood は変化しない。そこで、満たされていないパラメーターを上昇させる（Mood が大きく変化する）行為のみ選択して評価する、という仕組みを作れば、わざわざ、満たされてしまったパラメーターを変化させる行為を評価する計算の手間を省くことが出来る。

こういった計算の効率化を行うため、The Sims 3 からは、内面パラメーターと行為の対応表をあらかじめ作っておき、変化させるべきパラメーター（満たされていないパラメーター）から行為を検索できる仕組みを導入している。この検索によって、満たされていないパラメーターから選ばれた行為の集合を評価することで、最も幸福度をアップさせる行為を効率よく見つけ出すことが出来るのである。

物と欲求の相互作用マップ

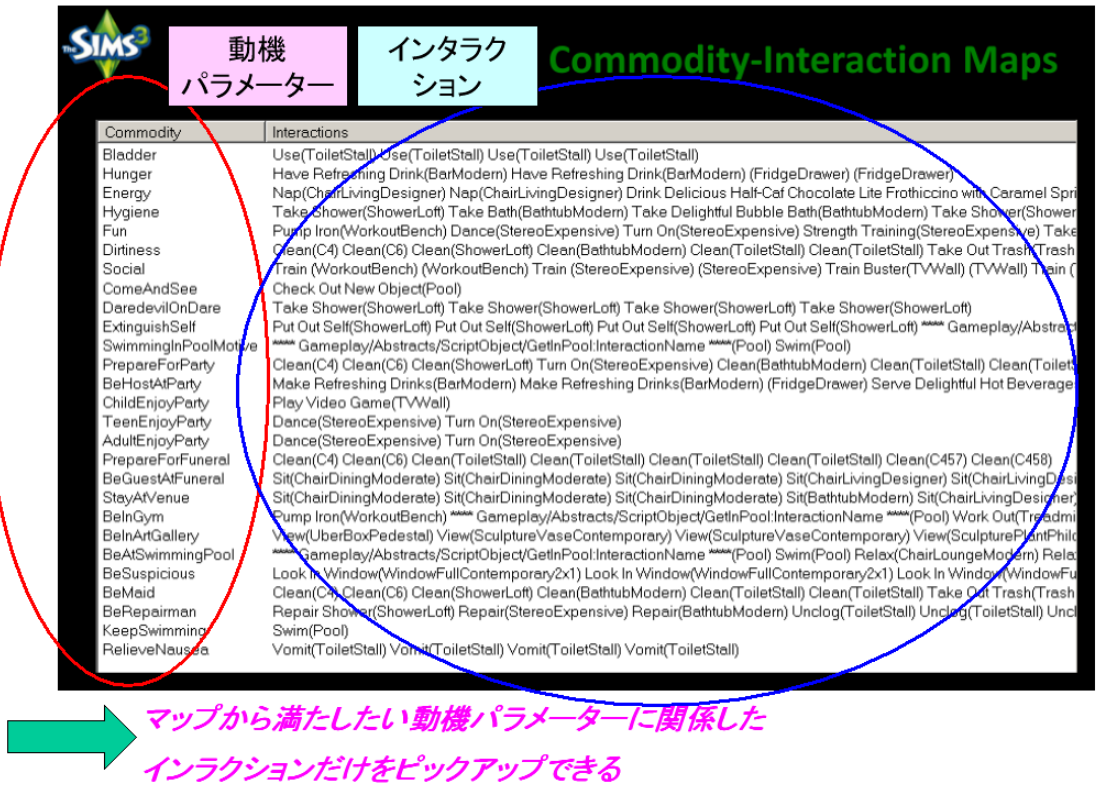


図 6.2-28　The Sims 3 における内面パラメーターと行為の対応表[2]

The Sims では、経験的に、ある種のパラメーターは殆どの場合、大きく満たされている。つまり、そういったパラメーターを変化させる行為を行っても、Mood を上昇させることは僅かである。そこで、満たされていないパラメーターがあれば、そのパラメーターを変化させた方が Mood は大きく変化するのであるから、そういったパラメーターのみの着目して、行為を評価すればよい。そこで、内面パラメーターと行為の対応表から、そういった行為のみをピックアップして、周囲のオブジェクトからそういった行為のみをまず評価して見るのである。

また The Sims 3 からは、Hunger パラメーターなど、自明な生理的欲求は、ある程度欲求が強くなると、自動的に満たすような処理(Auto-satisfaction)が施されている。これによって毎回の食事のようなゲーム内で何度も繰り返さなければならない行為に煩わされないで済む。

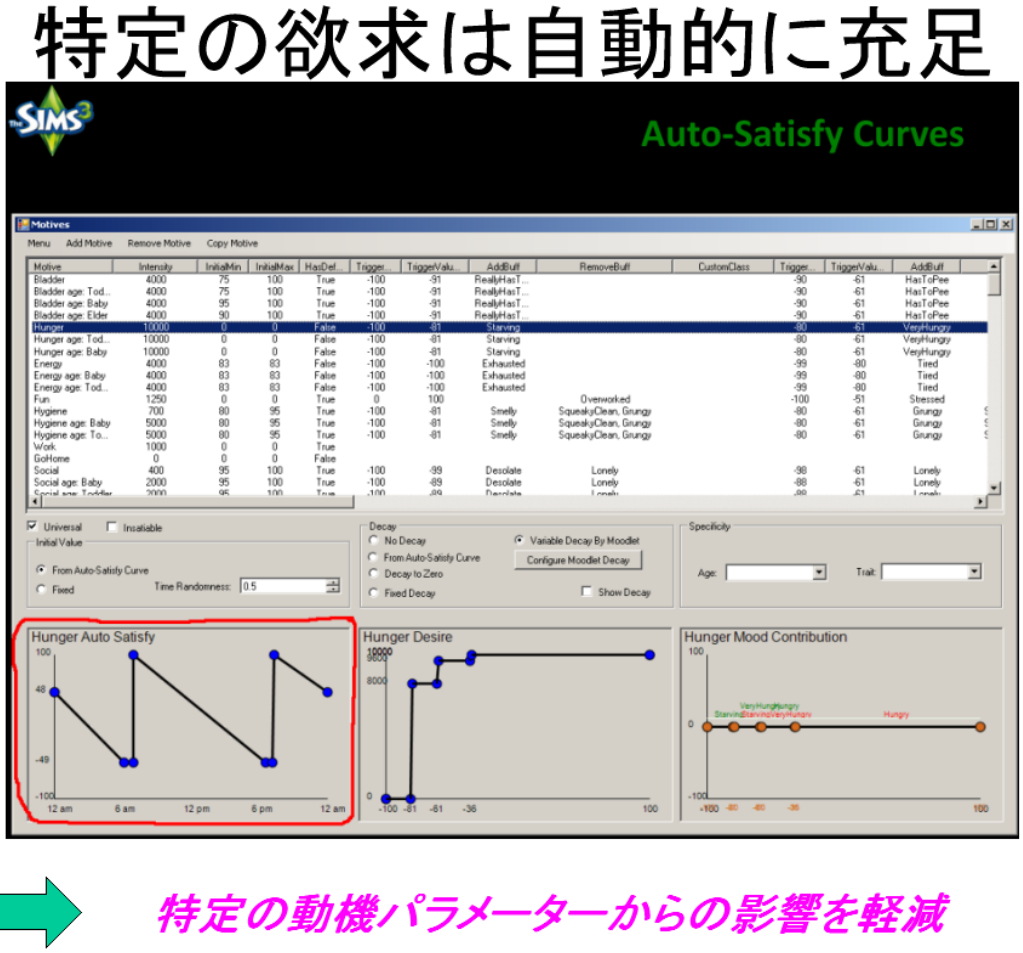


図 6.2-29　Hunger パラメーターの自動充足機能[2]

Hunger パラメーターは AI に食事を取らせようとするが、ゲーム的に一日何度も食事をするのは煩わしいので、自動充足されている。

(6)　AI 全体の制御アルゴリズム

The Sims 3 は、The Sims 1,2 より、広大な世界におけるシミュレーションを目指している。そこで、The Sims 3 では、AI シミュレーションにかかる計算負荷を軽減するため、以下のような工夫を行っている。The Sims 1,2 では、対象とするエリアで、各エージェント(AI)に対して、あらゆるインタラクションを考慮してシミュレーションしていた。しかし、The Sims では、あるエリアのエージェントに対して、そのエージェントをインタラクションの対象とするエージェントをピックアップして、そのインタラクションをシミュレーションする。このように、The Sims 3 のエージェント同士のインラクション全体を階層的に効率化している。

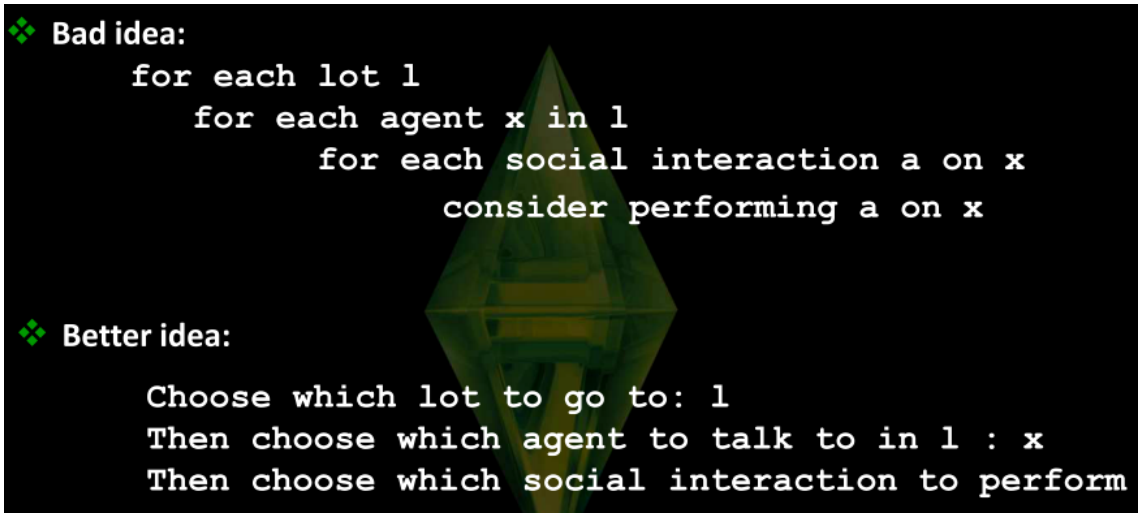


図 6.2-30　The Sims 1,2 から The Sims 3 へのインタラクションのシミュレーションアルゴリズムの変化[2]

階層化され効率化されている

AI全体の制御アルゴリズム

階層型プランニング

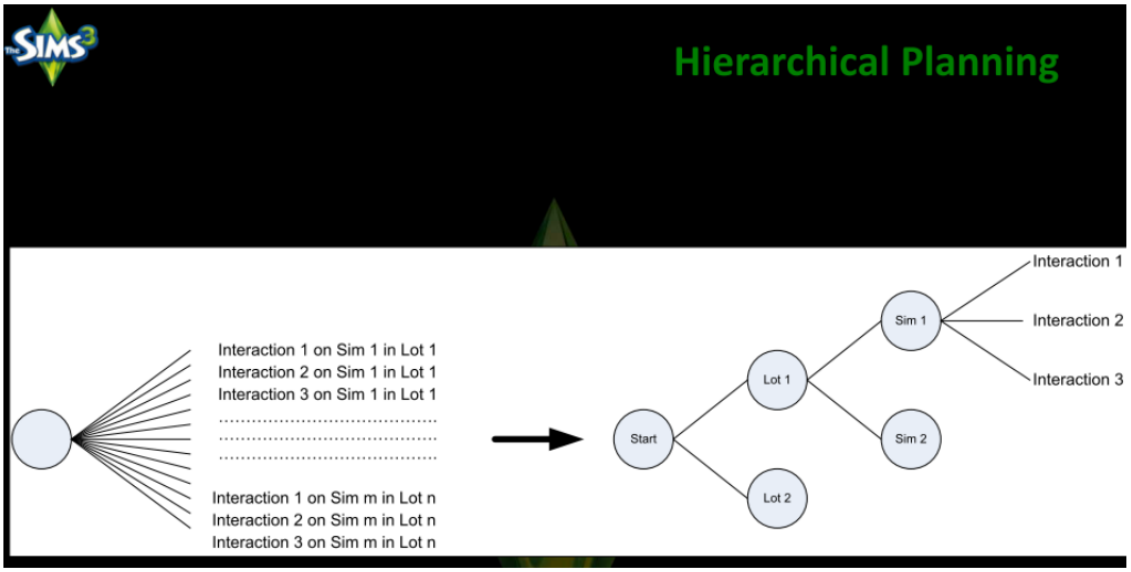


図 6.2-31　The Sims における階層型プランニング[2]

上記のアルゴリズムはエージェント全体のシミュレーション制御を階層型プランニングを用いて効率化していると言える。全体の状況を局所に分解してシミュレーションして行くのがコツである。

(7)　The Sims AI　の発展、個性化へ

The Sims は人間の社会的なインタラクションをエンターテインメントすることに踏み込んだタイトルである。プレイヤー（人間）が人間（AI）社会を見て楽しむのであるから、各 AI には、人間的な社会的欲求のリアリティーが求められる。The Sims 3 の講演では、こういった人間の欲求を明確に捉えるため、「Maslow の欲求階層図」が解説のために提示された。「Maslow の欲求階層図」は人間の欲求は段階的な階層があるという説に基づき、下層から生理的欲求、完全欲求、社交的欲求、個性的欲求（自己実現、尊敬を受けたい）、道徳的欲求といった階層構造を為す図である。つまり、下層に行くほど肉体的な欲求であり、上層へ行くほど社会的な欲求、或いは抽象度の高い欲求となっている。The Sims の AI も、この下層から上層へ向けて順番に実装されて来たと推測できる。そして、The Sims 3 の目標とは、「AI に特徴（Trait）を与えること」「長期的な目標を与えること」である。つまり、Maslow の図で言えば、第二段の欲求を AI に実装することとなる。

図 6.2-32　Maslow の欲求階層図[2]

人間の持つ欲求が階層的に表現されている。下位の欲求が満たされれば一つ上の階層の欲求を満たされたくなる。Trait Satisfaction （個性的満足）とは、自己の個性や特徴を社会や人に認めて貰いたい、ということである。

The Sims 3 においてそれぞれの AI に特徴を与えるのは以下のような方法である。まず、個性化とは、ゲームの場合、ユーザーがそれを見て分かるようなものでなければ意味がない。そこで、個性化を「普遍的な内面の構造」と「社交（人との係わり合い）における行動」に分けて実装することにする。

まず「普遍的な内面の構造」の特徴付けを解説する。80 個ほどの特徴(trait)を準備する。これらの特徴は例えば、「盗癖がある」「ロマンティックな TV を見る」「インターネットフォーラムでチャットする」とか、そういったユーザーから見て行為的な特徴のことである。そして、これらの特徴に対応する、動機（内面パラメーター）を設定する（これは先に解説した 8 つの基本的な内部パラメーターと行為の対応と同様である）。例えば、「盗む」「ロマンティックなＴＶを見る」パラメーターなどである。各 AI を特徴付けを行うときには、この 80 個の中からランダムに 5 個の特徴を選んで、そのキャラクターの特徴とする。これまで、The Sims 1,2 では、あらゆる AI がこれまで解説して来たような 8 個の共通の内面パラメーターのみを持っていたところに、5 つのパラメーターが増設されることになる。また、こういった追加的な特徴パラメーターは、特に発動する必要のない時間帯があるため、一日のうちで、追加したパラメーターを取り除いて動作させる時間帯を設置している。例えば、「ポテトチップスを食べながらＴＶを見る」という特徴は、休日の昼間や夜以外はふさわしい行為ではない。つまり、時間帯によっても AI の特徴が変化する仕組みになっている。

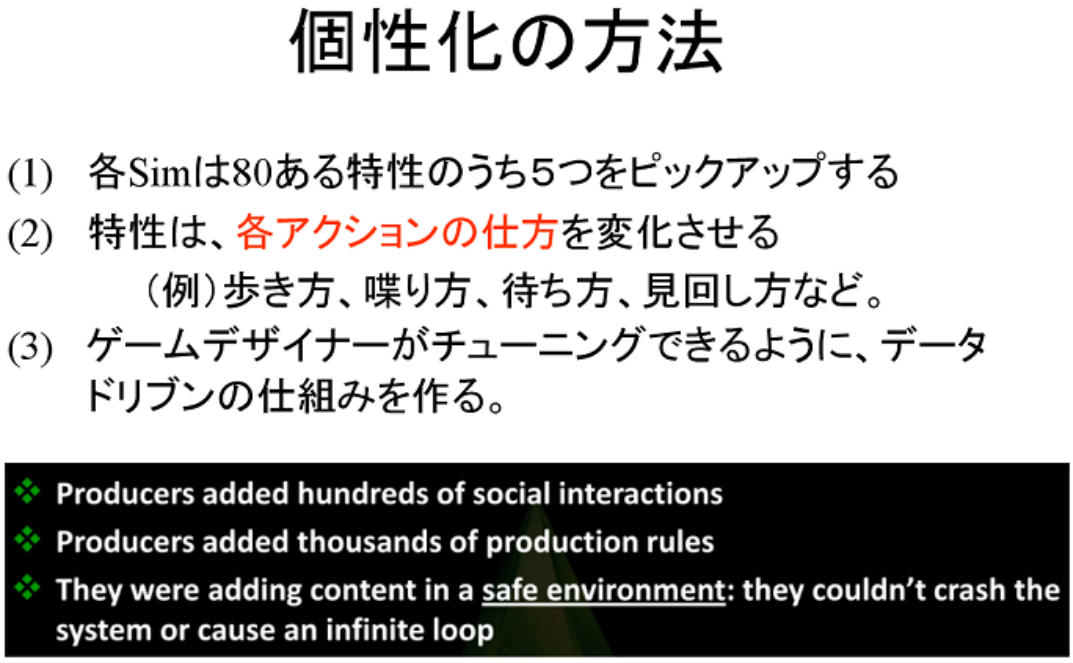


図 6.2-33　The Sims 3 における個性化の方法[2]

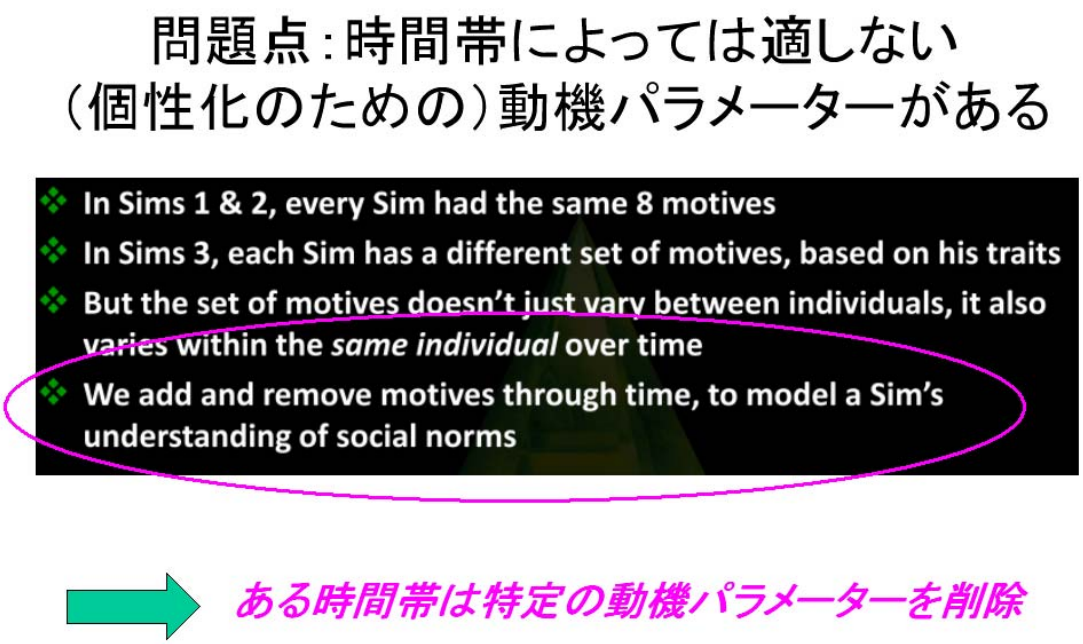


図 6.2-34　時間帯によっては特徴付けの内面パラメーターは除かれる[2]

さて、もう一つの「社交における行動」の各 AI におけるカスタマイズであるが、これはプロダクション・ルールを用いて行う。プロダクション・ルールとはここでは、「もし～だったら～する」といったルールの集合である。こういったプロダクション・ルールの全体は、データとしてプログラムから読み取らせるようになっている。つまり、コードを変えることなく、このルールのデータを読み取ることで駆動される仕組みになっている（データ駆動型システム）。このため、非プログラマーであるゲームデザイナーやプロデューサーがプログラマーの手を煩わせることなく、自由にルールを追加することが出来る。このような分業体制のもとで、数百個のインタラクション（行為）と数千のプロダクション・ルールのデータがプロデューサーによって作成された。

データドリブンなプロダクション・ルール

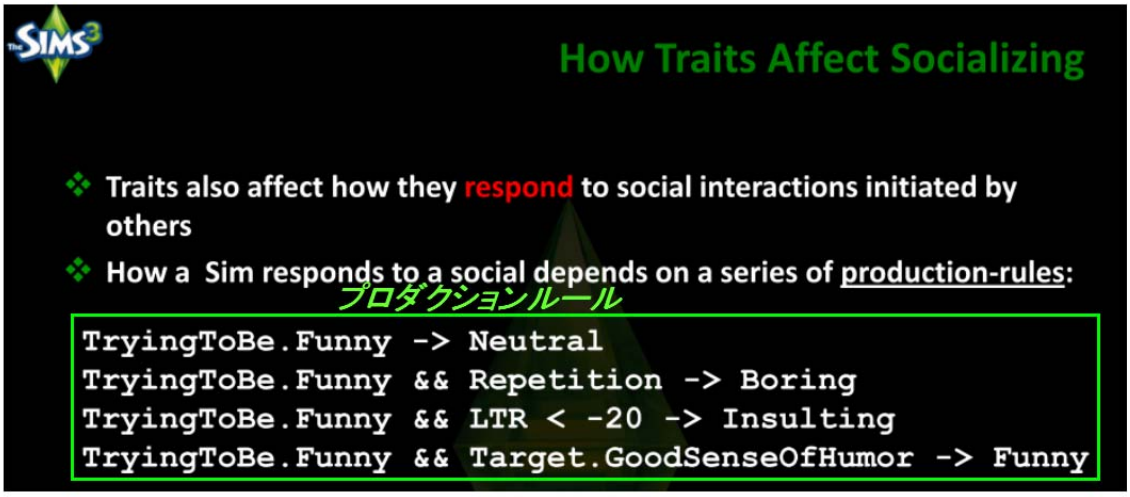
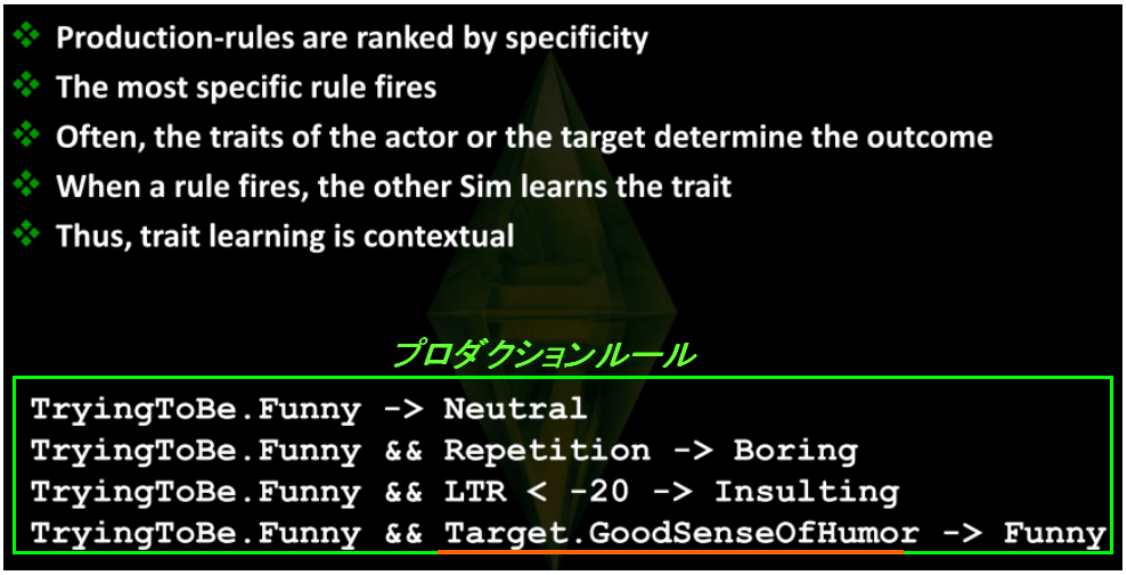


図 6.2-35　社交における行為をカスタマイズするプロダクション・ルール[2]

データドリブンなプロダクション・ルール



プロダクションルールは種類によってランク付けされていて、
最もランクの高いルールが発火する。
ルールが発火すると他のSimは、その特性を理解する。

図 6.2-36　プロダクション・ルールの全体の仕組み[2]

(8)　街全体を制御する AI

The Sims 3 では、各 AI とは独立に、街全体を統制する AI も実装されており、NPC を増やしたり消したりし、雇用をしたり職を奪ったりしながら、雇用率や男女比をコントロールしている。このコントロールは、街全体で様々な人間が点在し、様々なドラマが創発されるような状況を作り出すためである。

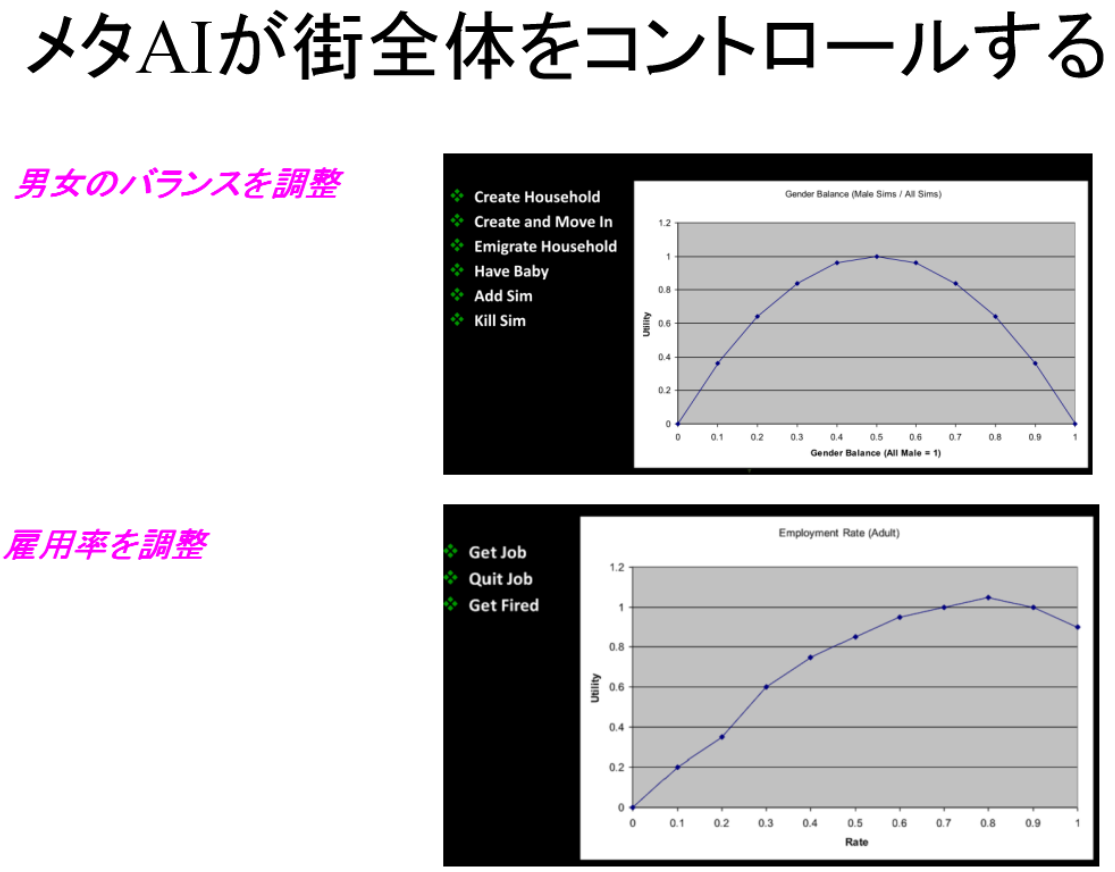


図 6.2-37　メタ AI が街全体の状況をコントロールする[2]

6.2.4　FPS(戦闘) ⇔ 日常系の AI の比較

以上が、Killzone 2 と The Sims 3 の AI の解説である。FPS と日常系 AI という対照的な AI を解説したが、ここでは幾つかのファクターに分けて 2 つの AI 技術の相違をより明確に対比したい。

(1)　AI の内面構造

まず AI の内面構造であるが、FPS の AI は思考が複雑であっても、内面構造自体は非常にシンプルである。戦場という極限状況において、感情や周囲との関係性は極めて単純

な形に還元されている。「敵を倒す」「仲間を助ける」「自己保全」という 3 つの動機にほぼ集約できると言ってよい。また、構造物も、敵に射線が通るか、敵の視点から隠れているか、という障害物としての機能に還元されているのである。

一方、日常系の AI の内面は、複数の内面パラメーターから構成され、その競合の中から行動が決定される。合理性よりは人間性のエミュレーションを目標としており、環境との多様な関係の中で自己の行為を決定して行く。周囲の物、一つ一つが多様な意味と可能な行為の集合を保持しており、またそれが内面パラメーターと深く結びつくことで、AI と物、また AI と AI 同士の多様なインタラクションを産み出すのである。

図 6.2-38　FPS（戦闘）と日常系ゲームにいおいて必要とされる AI 技術の比較[3]

(2)　思考

次に思考であるが、FPS における AI の思考は極めて高度である。動機が敵の殲滅と自己の保持にあるとはいえ、その分、生死をかけた繊細さで、状況を極めて緻密に解析しつつ行動を瞬時に決定する必要がある。敵勢力、味方勢力、自分の状態、敵の状態、残り弾

薬数、地形の状態、など、様々なファクターが、生死の上に極めてセンシティブな意味と効果を持つのが戦場である。こういった状況を全て把握、そして未知の情報は推測した上で、意思決定を行うのである。

一方、日常系の AI にとって、世界はどこまで行っても日常的な空間であり、状況というものには、それほどセンシティブではない（特に The Sims の街は、少々の無礼が許される平和的な街である）。道路や公園、周囲の人、人の家、自分の家、と大まかな状況しかない。どちらかと言えば、自分の内面的なパラメーターの力学によって行為が選択されており、意思決定の原理(Motive Engine)自体は極めて単純である。

(3) 集団としての AI

また集合としての AI を考えてみる。Killzone 2 では、戦場の組織として極めて厳密な階層的な構造が形成されていた。また、当然のことであるが、FPS では戦場の雰囲気を再現するように AI が動作させられている。

一方、The Sims では、大勢の人が緩やかで多様な関係で結ばれている。The Sims では、人間ドラマを産み出すようなシステムが根底に敷くことを目的としているのである。

(4) スコープ時間

スコープ時間とはここでは、AI が活動の単位とする時間の目安である。或いは、ユーザーから見て首尾一貫した知性的な振る舞いを維持できる時間である。

FPS は初期のオンライン専用ゲームであった頃には、部屋に固定された AI がプレイヤーがステージを通り過ぎる極めて短時間の思考が出来ればよかった。しかし、ステージが拡大し、オンラインに AI が参戦するような現代では、数分から十数分といった極めて継続的な時間を知的に活動をし続ける必要が発生したのである。

一方、日常系の AI も、かつての、簡単な Motive Engine によって周囲の人、オブジェクトとインタラクションを重ねていたところから、より長期的な目標や夢を持って日々を組み立てるように実装されており、長期的に見ても人間らしい活動を行える方向が目指されている。

(5) 学術分野との関係

また他の学術分野との関係を考えたとき、FPS の AI は、意思決定の問題を極めてアルゴリスティックに解決するために、人工知能、及び制御工学の伝統的な分野（プランニング、有限状態機械、プロダクション・ルールなど）との結びつきが強い。

一方、日常系の AI は、どちらかと言えば、人間の社会性・心理を穿ったシステムの構

築を目指すために、心理学・認知科学的な分野との結びつきや引用が多いのが特徴である。

このように、FPS と日常系の AI は極めて対照的な特徴を持っており、重要なことは、この二極のゲーム AI の間には両極の特徴を兼ね備えた多様な AI のフィールドがあり、そこには、まだ、様々なゲーム AI の可能性、ひいてはゲームデザインの可能性が眠っているということである。こういった可能性を一つ一つ取り出し、発展させ、ゲーム AI の広大な領野を耕し、ゲーム AI のポテンシャルを上げ、知識と方法を蓄積して行くことが、今後 10 年のゲーム AI の発展に必要な仕事である。

図 6.2-39　ゲーム AI マップ

ゲーム AI は広大で多様な分野であるが、FPS の AI と日常系 AI は、このようなフィールドの良い指標となるだろう

6.2.5　AI における階層構造

「Killzone 2」「The Sims」に共通する特徴はその階層型構造にあった。ここでは、なぜゲーム AI にとって階層構造が大切なのかを解説する。

人間の知性は様々な階層的な世界の捉え方を無意識に行っている。その仲で、主な二つの階層化は「空間階層化」と「時間階層化」である。他の階層化は、この二つのメタファーとして産み出されると言ってよい。

空間階層化は、空間を様々なスケールで切り取ることで、自分の周囲だけの空間、街全体の空間、地域、国、宇宙、或いは翻って蟻の巣穴のスケールなど、思考しやすい単位の空間を切り取って、それぞれの段階で思考する能力を持っている。かつ、それらをスケールに沿って階層的に捉える能力を持っている。広大な仮想世界で活動する AI にも、これらの能力は必須である。

時間階層化は、瞬時の判断から、中期、長期、一生に渡るようなプランまで、活動するべき時間を様々なスケールで切り取って、各段階で思考している。長時間の活動（数分～十数分）が必要とされて来たゲーム AI にとってもこの能力は必要である。そして、こういった階層化構造を導入することで、AI は、一つレベルの高い AI に発展する土台を得るのである。階層型 AI が導入されたゲームの例は多く、また最近では、一つの設計方針として頭に入れておくべきものである。

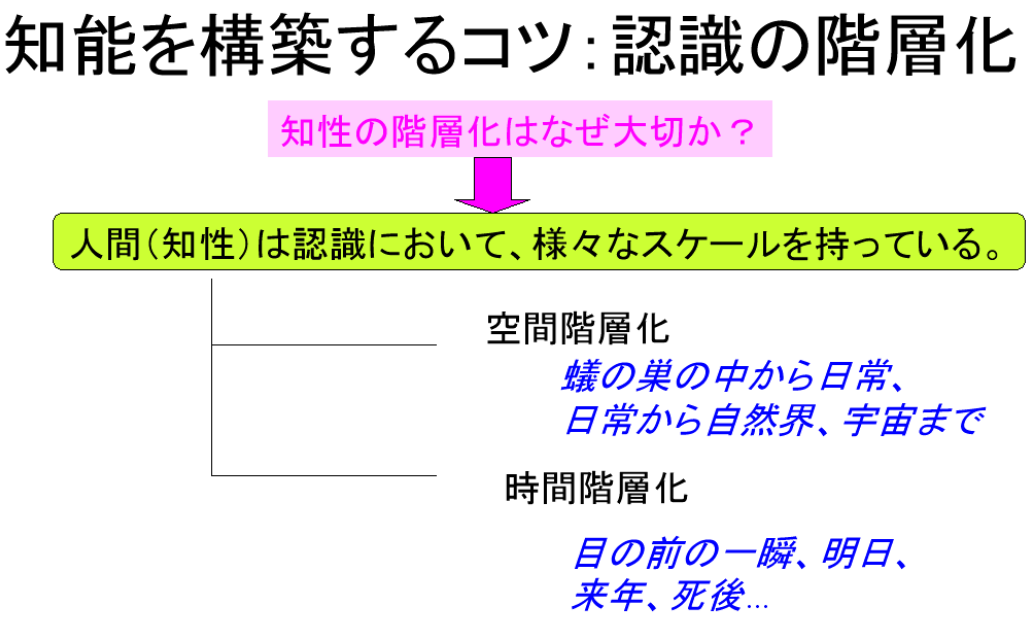


図 6.2-40　知性の二つの階層構造[3]

空間階層化、時間階層化は、ゲーム AI の設計の基本ポリシーと言ってよいだろう。広大なマップを自由に動き回る AI とは、長時間・長距離を支配すると同時に、局所的戦闘、瞬時の判断を行う能力を持っていなければならない。そこで、瞬時の判断から長期行動プランまでを、階層的な思考として持っておくこと、或いは空間の把握においても戦略的に地形全体を捉えるところから、数十 m 四方の局所戦闘思考まで、多様な階層レベルの思考を実現することが重要になって来るのである。

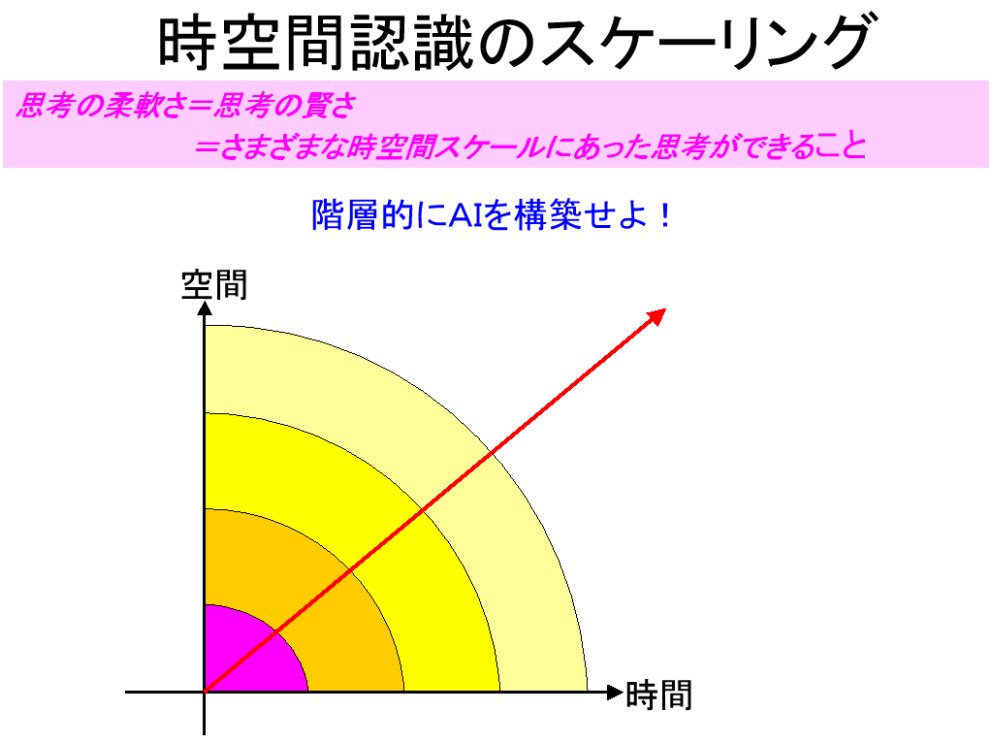


図 6.2-41　時空間認識のスケーリング[3]

人間は無意識のうちに世界を階層的に捉えている。人間の場合、その階層はより精緻で複雑なものであろうが、AI の思考には、時間、空間に渡る明確な階層性を導入することで、柔軟な意思決定機構を築くことが出来る。

階層型思考を持つ人間から、階層型思考を持たない AI を見た場合、それはちょうど上から下を見下ろすように、拙い存在に見える。例えば、短期的な思考しか持てない AI には、長期的戦略によって勝つことが出来るし、局所戦闘しか頭にない AI には、戦略的な罠や、退路を断って攻略すればよい。つまり、自分の作成している AI が何か足りないと思ったら、まず階層構造で足りていないかを検証するべきである（詳細は以下の文献[19]にまとめているので参照されたい）。

6.2.6　00 年代から 10 年代のゲーム AI へ

(1)　00 年代のゲーム AI

00 年代は、ゲーム AI がゲームエンジンから独立した一つのシステムとなろうとした時期であり、独立したシステムとして発展すると同時に、ゲームエンジン本体との関係を模索した時期でもあった。

欧米では FPS を通じて学術から AI 技術が流入し、ゲーム AI 思考を高度に発展させ、パス検索用のデータは大規模化・階層化され、それらをまとめるフレームとしてエージェ

ント・アーキテクチャが導入された。つまり、00 年代という年代はゲーム AI がひとり立ちを始めると同時に、さまざまに必要になりそうな技術が試されながら、だんだんと荒削りながらも、ゲーム AI という分野の全体像が浮かび上がって来た時期である。

この主要な項目を挙げると、以下のようになるだろう。

- エージェント・アーキテクチャ
- パスプランニング
- パスデータ階層化
- 階層型思考
- アニメーションと意思決定
- プランニング

これらは、GDC や学会で議論され、数多くの英語の文献として実例やアイデアとして蓄積されている[20]。日本は、だいたいにおいてこの流れの外にあったと言ってよい。その理由は複数あるが、

- 第一は日本のゲーム産業が開発情報を極めてクローズドにして来たこと
- 第二はそういった技術を発展する英語圏のコミュニケーションが出来なかったこと
- 第三に日本のゲームデザインと欧米のゲームデザインに隔たりがあり、自然、求める AI の方向が違ったこと
- 第四に、方向が違ったなら自分たちで築けば良かったが、そういった技術的積み上げの基盤がなかったこと
- 第五に、ゲームデザインが必ずゲーム開発技術に優先するべきという思い込みがあったこと（現在でも）

である。日本が技術的にも、ゲームデザインとしても、市場規模の拡大としても行き詰っている間に、欧米のゲーム AI 分野は急速にその土台を築いて行った。

(2)　00 年代のゲーム AI 最大の成果

さて、こういった 00 年代の流れの中で、欧米の各ゲーム AI 開発者、各ゲーム開発企業は、ゲーム AI の開発の力点とも言うべきものを、それぞれ見出し始めた。以前は、見当はずれな、或いは、工程ばかりが増えて行くポイントに力を入れていたことも多かった。そして、ごてごてとしたメンテナンスのしにくい AI システムが出来上がって行った。しかし、そう言った、さまざまな失敗例と、幾つかの成功例を研究し、AI の開発の力点を何処に集中するべきかを学習した。そして欧米では、そういったノウハウが、各種ゲームカ

ンファレンスや学会を通して、共通に認識する技術やノウハウとして行った。さらに共通の認識とすることで、同じ言葉、同じ技術を、ゲーム産業全体で研究し応用し議論し合い発展させるという体制を作って行った。このことは、00 年代における欧米ゲーム産業の AI 分野最大の成果と言ってよい。

「パスシステムを設計し質の高いパスデータを階層化して築いておくこと」「エージェント・アーキテクチャのフレームの中で、センサー、意思決定、身体制御、意思決定部分に導入する技術を検討し」「各部分の関係性を検討すること」、こういった、ゲーム AI 製作の基本姿勢も、この 10 年を通して、次第に明確になって来たことであった。

(3) 10 年代のゲーム AI へ

そして、今年の GDC2010 では、ゲーム AI のこの 10 年の発展から次の 10 年の発展へ向けての発展の兆しが垣間見られた。静的なパスシステム（オブジェクトがマップ上に落ちても対応できない）から、次世代のパスシステムとして動的オパスシステム（動的なオブジェクトに対応するパスシステム。例えば落下したオブジェクトに合わせてメッシュが再分割されて変化する）への移行が複数のタイトルで実現されていた(「Sprinter Cell: Conviction」(Ubisoft,2010)、各種 AI ミドルウェア)。思考部分では、プランニング技術がほぼ基礎技術として、複数のタイトルに導入されていた(「Killzone 2」「The Sims」など)。

おそらく、これからの 10 年は、各種プラナー（プランニング・アルゴリズム）の開発が、AI 思考技術の中核となると予想される。Killzone 2 の AI が 1992 年の学術論文を基礎にしたように、30 年以上に及ぶ学術におけるプラナーの研究成果がゲーム AI に応用されて行くと考えられる。プラナーはちょうど CG 分野の「シェーダ」と同じように、AI 分野の共通フレームとしての役割をもって AI 技術の標準的な技術として定着して行くと予想される。

このように 10 年代は手探りで探り当てたゲーム AI 技術分野の全体の領野が、一つ一つ本格的に探求され、成果を産む時代がこれからの 10 年である。その発展には学術と技術双方のバックグランドを持つ優秀な人材が必要であり、既に専門化しつつあるゲーム AI 分野は、欧米では、そういった人々を数多く引き寄せ、さらなる発展を遂げるだろう。しかし、残念ながら、まだ日本にはこの基盤が出来てない。

(5) 日本のゲーム AI の未来のために

00 年代は、ゲーム AI 発展の流れの完全な外にあった日本である。しかし、これまでのゲーム AI の各分野の探求の荒削りの結果が、膨大な英語の文献や論文に記されている（そ

ういったゲーム AI のこれまでの歴史や参考文献は、これまでの報告書に記して来た[6][17])。そして、それらは、十分に探求されずに放置されたゲーム AI 技術の萌芽である。幾つかの芽は、欧米の FPS やシミュレーションゲームで大きな成長を遂げた。しかし、幾つかの芽は、日本のゲームデザインの土壌でこそ大きく発展を遂げるものであろう。

こういった文献を一つ一つ読み解き、各技術事項の持つ可能性を十分に吟味し、我々が持つ独自で多様なゲームデザインの上に新しく応用の展開し応を見出すとき、日本はこの 10 年の遅れを取り戻し、さらに、そこから新しいゲーム AI の発展の形を築くことができる。そして、欧米で育まれたゲーム AI の流れと、我々が築くゲーム AI 技術の流れが合流を果たすことで、全体として新しいゲーム AI の広がりを産みだし、世界のゲーム AI の発展の流れに大きく貢献することになるだろう。

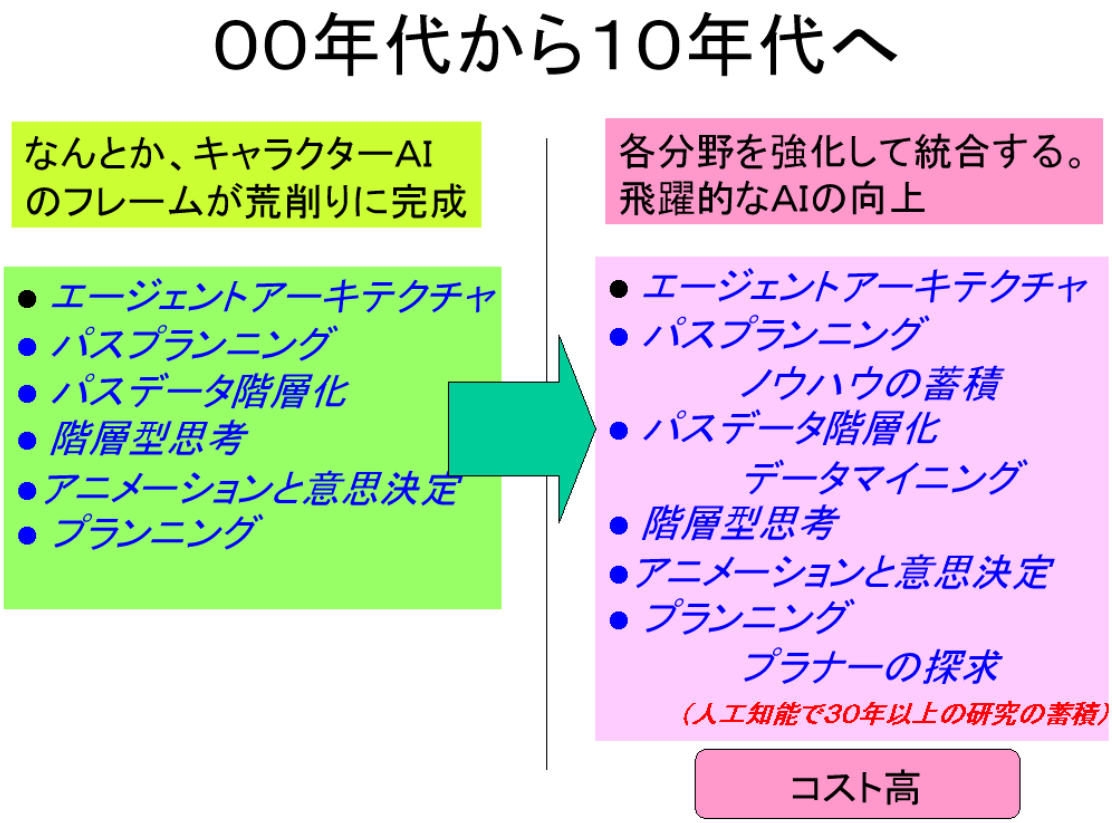


図 6.2-42　ゲーム AI の 00 年代から 10 年代への発展・変化[3]

参考文献：

[1] Alex Champandard, Tim Verweij, Remco Straatman, Killzone 2 Multiplayer Bots, Game AI Conference, Paris, 2009　(GDC2010でも同様の資料が用いられた) http://files.aigamedev.com/coverage/GAIC09_Killzone2Bots_StraatmanChampandard.pdf

[2] Richard Evans, Modeling Individual Personalities in The Sims 3, GDC2010, 2010 http://cmpmedia.vo.llnwd.net/o1/vault/gdc10/slides/Evans_Richard_ModelingIndividualPersonalitiesInTheSims3.pdf

[3] 三宅陽一郎, "「00年代から10年代へ ゲームAIとプロシージャル手法の進化」", GDC2010報告会(IGDA日本) , 2010 http://igda.sakura.ne.jp/sblo_files/ai-igdajp/GDC2010/GDC_Report_2010_YMiyake_0403.pdf

[4] "AI Summit '10: Slides, Notes, Highlights and Photos", AIGameDev, 2010 http://aigamedev.com/open/coverage/gdc10-slides-highlights/

[5] 三宅　陽一郎, 「GDCにおける海外のゲーム関連技術についての調査」, 財団法人デジタルコンテンツ協会「デジタルコンテンツ制作の先端技術応用に関する調査研究報告書」(2008年度), pp.331-369, 2009 http://www.dcaj.org/report/2008/data/dc_08_03.pdf

[6] 三宅　陽一郎, 「ゲームAI分野」, 財団法人デジタルコンテンツ協会「デジタルコンテンツ制作の先端技術応用に関する調査研究報告書」(2007年度), pp.37-113, 2008 http://www.dcaj.org/report/2007/data/dc08_07.pdf

[7] スチュワート ラッセル、ピーター ノーヴィグ,「エージェントアプローチ 人工知能」, 共立出版, 1997

[8] 生天目 章, 「マルチエージェントと複雑系」, 森北出版, 1998

[9] 三宅陽一郎, IGDA日本 ゲームAI連続セミナー第2回「F.E.A.R におけるゴール指向型アクションプランニング」資料, 2007　http://blogai.igda.jp/article/33936286.html

[10] 三宅陽一郎, IGDA日本 ゲームAI連続セミナー第3回「Chrome Hounds におけるチームAI」資料, 2007　http://blogai.igda.jp/article/33936286.html

[11] Nau, D. et al., "SHOP2:　An HTN Planning System", Journal of Artificial Intelligence Research, vol.20 pp.379-404, 2003 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.188&rep=rep1&type=pdf

[12] Dana S. Nau, "Hierarchical Task Network Planning, Lecture slides for Automated Planning: Theory and Practice", http://www.cs.umd.edu/~nau/cmsc722/notes/chapter11.pdf

[13] Damian Isla, "Building a Better Battle: HALO 3 AI Objectives", GDC2008 , 2008 http://www.bungie.net/Inside/publications.aspx

[14] Dave C. Pottinger, "Terrain Analysis in Realtime Strategy Games", http://zeniroy.springnote.com/pages/481669/attachments/212469

[15] 多摩 豊, "ウィル・ライトが明かすシムシティーのすべて", 角川書店 , 1990

[16] Maurice Bergsma and Peter Meij, "Influence Maps", Student Lecture GAI , 2006 http://www.slideshare.net/mobius.cn/influence-map

[17] 三宅　陽一郎, 「プログラミングAI」, 財団法人デジタルコンテンツ協会「デジタルコンテンツ制作の先端技術応用に関する調査研究報告書」(2008年度), pp.73-137, 2009　http://www.dcaj.org/report/2008/data/dc_08_03.pdf

[18] Ken Forbus, "Simulation and Modeling: Under the hood of The Sims", http://www.cs.northwestern.edu/%7Eforbus/c95-gd/lectures/The_Sims_Under_the_Hood_files/frame.htm

[19] 三宅陽一郎, "ゲームAIにおける4つの階層", y_miyake のゲームAI千夜一夜 , 2009 http://blogai.igda.jp/article/32493117.html

[20] 三宅陽一郎, IGDA日本 ゲームAI連続セミナー資料集, http://blogai.igda.jp/article/33936286.html

7. 日本と世界のゲーム産業支援の現状と課題

馬場　章（東京大学大学院）

7.1 調査委員会の成果

7.1.1 2007 年度・2008 年度を振り返って

私たちの委員会は 2007 年度から 3 年間にわたって、日本と世界のゲーム関連技術の動向を追究してきた。3 年間という期間が短いか長いかは見方が分かれるところであるが、この 3 年間の間にも、日本と世界のゲームをめぐる状況は大きく変化してきた。そのすべてを本委員会で解明しえたわけではないが、各年度における特徴を把握することに努めてきた。今、それらを振り返ってみると、以下のような調査のポイントが存在したことを指摘出来る。

(1) 2007 年度

① 国内外のゲーム開発技術の動向を俯瞰するために、CEDEC、DiGRA、GDC、インタラクティブ東京に注目し、それらで発表されたゲーム開発技術を紹介した。
② 論文検索手法により、国内外の学会におけるゲーム関連技術の研究動向を明らかにした。
③ 技術分野としては、とくにゲーム AI 技術とゲーム・プロシージャル技術に注目して詳細な解説を行った。
④ コンテンツ管理技術としてのコンテンツ・パイプラインに注目して解説を行った。
⑤ ゲーム開発やゲーム教育をめぐる国内外の産学連携の成功事例について調査し、紹介した。
⑥ 北米と日本における MOD およびオープンコミュニティの実態を解説した。

(2) 2008 年度

① ゲームプラットホームの歴史的変遷を調査し、技術の進化を明らかにした。
② 素材作成、プログラミング、タスクシステム、ネットワーク通信を中心に、ゲーム開発技術のロードマップを策定した。
③ ゲーム会社に対するインタビューなどを通じて、日本におけるゲーム開発者の人材育成をめぐる実態と問題点を解明し、提言を行った。
④ 海外のゲーム開発者教育で行われているカリキュラムフレームワークを紹介した。
⑤ GDC において紹介された SPU、プロシージャル、AI 技術を中心に解説した。

それぞれの具体的な内容については、各年度の報告書を参照していただきたいが、本調査研究では、ゲーム開発技術の動向を調査して紹介するだけでなく、それらの技術を活用してゲームを開発するゲーム開発者という人材や、ゲーム開発技術を研究する体制として産学連携を重視してきていることが分かる。つまり、ゲーム産業やゲーム開発技術それ自体の紹介にとどめるのではなく、それらが寄って立つ社会的基盤にまで視野を広げ、その相互関係の解明と強化にも努めてきた。

7.1.2 2009 年度の調査を振り返って

前節で述べてきたような 2 年間の蓄積と、2009 年度のゲーム産業を取り巻く変化をふまえて、今年度の報告書では、以下のような点にポイントを絞って調査した。

① 2008 年度のゲーム開発技術ロードマップの範囲をシナリオ、ミドルウェア、ユーザーインターフェイス、工程管理、ローカライズなどにも拡大し、さらに充実させた。
② ゲーム開発をイノベーションの観点から社会的に捉え直す試みを行った。
③ ゲームユーザーの分析を試みた。
④ 国内のゲーム開発者教育の事例を紹介するとともに、ゲーム開発者の実態調査を行って、その特徴を明らかにするとともに、問題点を解決するための提言を行った。
⑤ GDC のセッションを俯瞰して、産業の趨勢と技術の動向を明らかにし、紹介した。

2007 年度・2008 年度の 2 年間で確立したゲーム開発技術と産業・社会という観点は、今年度、さらに一層推し進められ、ゲーム開発者の実態解明やゲームユーザーの実態分析にまで進んだ。それらの調査・分析の結果は、日本のゲームとゲーム産業の真の姿を実証的に浮き彫りにし、新たな知見を加えただけでなく、問題点の掘り起こしと解決策の提言に結びついている。

本調査研究の原点は、ゲーム開発技術の国内外の動向を調査し紹介して、日本のゲーム開発に貢献することにあった。3 年間の調査研究で、それが十全に行いえたとは言えないかも知れないが、ゲーム開発の特徴点の洗い出しとその推移の解明とゲーム開発技術のロードマップは、本調査研究の本来の目的に対応する成果にほかならない。

それに加えて、本調査研究では、産業・社会という観点からゲーム開発技術をながめ、人材育成の提言やユーザー分析、産学連携の情報提供なども行ってきている。

今年度の報告書でも産業と技術の両面から取り上げた GDC2010 では、ゲーム教育のセッションに、アフリカのガーナから教員が始めて参加し、同国においてゲーム開発者教育が始まったことを報告した。それを欧米のゲーム教育の教員は歓迎の大拍手で迎えた光景が記憶に鮮明に残っている。今や成熟したゲーム大国である日本のゲーム産業が、ガーナ

のような瑞々しさを取り戻すために、ゲーム開発技術の基盤である社会や人材から洗い出し、ゲーム開発の歴史と動向を調査し紹介した本報告書が役立つことを期待したい。

7.2 世界のゲーム産業振興政策

7.2.1 ゲーム産業力の公表

2007年にイギリスの貿易総省が行った調査[20]によると、2005年までと2006年以降とで、世界の「ゲーム産業力」のランキングに大きな変動が生じたという。ここでいう「ゲーム産業力」とは、行政によるゲーム産業振興政策、ゲーム会社の資本構成、ゲーム開発者の人材育成と待遇、ゲームソフトの開発・販売タイトル数、ゲームソフト売り上げなどを総合して評価した指標である。この「ゲーム産業力」はゲームソフトを中心にした見方で、ゲーム機やアーケードゲームが考慮されていないという問題があるが、世界のゲーム産業の趨勢を見る尺度のひとつにはなるであろう。

イギリス貿易総省によると、2005年までは、第1位アメリカ合衆国、第2位日本、第3位イギリス、第4位カナダ、第5位韓国という順位であったが、2006年に第3位と第4位の順位が入れ替わって、第1位アメリカ合衆国、第2位日本、第3位カナダ、第4位イギリス、第5位韓国になったという。つまり、イギリスにとっては、カナダに抜かれてランクをひとつ落としたことが、大変動ということになる。

さらに、イギリス貿易総省は、上記のランキング発表にあわせて三つのコメントを加えている。すなわち、1）日本の順位は変わらないが、相対的な地位は低下していること、2）韓国の順位は変わらないが、相対的な地位は向上していること、そして、3）カナダ躍進の要因を分析してイギリスも早急に対策を打つ必要があること、の3点である。

イギリス貿易総省が発表した「ゲーム産業力」のランキング自体は、実感としても概ね首肯しうる。とすれば、私たちは、それに付されたコメントに注目し、危機感を抱かなければならない。つまり、世界市場における日本のゲーム産業の地位の低下と韓国の躍進という、私たちが日常的に抱いていた不安が、第三者の客観的な立場から冷静に観察されているという事実である。そして、このままで推移すれば、日本の相対的地位が低下し続けて、おそらく、日本のゲーム産業のランキングは第3位以下に降下し、日本が世界に対して優位を保っていた産業のひとつを失うことになるのである。

[20] イギリス貿易総省の報告は以下を参照のこと
“Playing for Keeps–Challenges to sustaining a world-class UK game sector”
COUNTRY PROFILES / INTELLECTUAL PROPERTY UK TRADE&INVESTIMENT
https://www.uktradeinvest.gov.uk/ukti/fileDownload/Finalpdf(LR)AD2695Mono1Games.pdf?cid=410528
https://www.uktradeinvest.gov.uk/ukti/fileDownload/Finalpdf(LR)AD2695Mono2Games.pdf?cid=410529

7.2.2　カナダのゲーム産業振興政策

イギリス貿易総省の調査によれば、カナダが「ゲーム産業力」のランキングを上げた要因は、連邦政府や州政府による手厚い保護政策と産官学連携の成功に集約される。その結果、モントリオール、トロント、バンクーバーの 3 都市を中心に、それぞれにゲーム産業の地域クラスターを形成し、国全体として「ゲーム産業力」を向上させ、ランキングの逆転をもたらしたとされる。

カナダの連邦政府と州政府のゲーム産業振興政策は、経済的優遇政策に特徴がある。すなわち、1）法人税・所得税の優遇、2）人件費の補填、3）開発経費の支援、である。

カナダでは、国内のゲーム会社の法人税とゲーム会社に働く開発者の所得税が軽減されるとともに、開発者の人件費の一部を公費で負担する制度がある。それに加えて、ゲーム会社がゲーム開発の技術の調査・開発に経費を支出した場合には、その分が法人税から控除され、加えて、公費の直接支出によって技術開発の経済的支援を行っている。このような経済的な優遇政策に好感をもって、カナダ国内に開発拠点を設置したゲーム会社は少なくない。

さらに、それらに加えてカナダのゲーム産業の躍進を支えているのは、産官学の連携とゲーム開発者の人材育成策である。とくに、産学の連携は上記の地域クラスターの核となっており、例えば、ブリティッシュ・コロンビア州では、ブリティッシュ・コロンビア大学やサイモン・フレイザー大学などバンクーバーとその周辺の大学がコンソーシアムを組織し、さらに、エレクトロニック・アーツ社などのゲーム会社との強固な連携が成立している。その成果として、ゲームの技術研究やビジネス研究が飛躍的に進展した。また、各大学は学生をゲーム会社に対しインターンとして派遣し、学生の社会的スキルの向上に寄与するとともに、ゲーム産業の即戦力養成を実現しているのである。

ただし、注意しなければならないのは、上記のようなカナダの産業振興政策は、ゲーム産業だけに限ったことではない。カナダでは、産業によって程度の差はあるものの、同様の振興政策を採用している産業は少なくない。むしろ、カナダの産業振興政策が成功した分野のひとつがゲーム産業なのである。

ゲーム産業に対するカナダの振興政策が成功した最大の要因は、社会のゲームやゲーム産業に対する理解度や受容度の高さにある。つまり、ゲーム産業に対して行政などの公的機関が積極的に支援することを、社会が強く支持したという事情がある。それを可能にしたのは、カナダの公教育で先進的に行われてきたメディア・リテラシー教育である。カナダでは、隣国アメリカ合衆国のメディアを通じた文化進出に対する対抗として、メディア・リテラシーの必要が痛感され、その実践が進められて来た。そして、その一貫として、近年ではゲーム・リテラシー教育が重視されてきたのである。

ゲーム・リテラシーとは、ゲームを最新メディアのひとつと捉え、そのメリットとデメリットを踏まえて本質を理解し、ゲームを使いこなす技術的、倫理的、教育的能力である。

したがって、具体的には、ゲーム開発の技術や工程の知識の獲得やゲーム・レーティングの意義の理解などが、ゲーム・リテラシー教育のポイントとなる。カナダでは、ゲーム・リテラシー教育が実を結び、ゲーム産業に対する理解が深まっており、ゲームに対する社会的な受容が進んでいるのである。そのような土壌の上に、行政によるゲーム産業振興政策が支持され、ゲーム会社の事業が好感を持って迎えられることになったのである。

7.2.3　イギリスのゲーム産業振興政策

イギリスでは、他のヨーロッパ諸国でもそうであるように、ゲーム産業はクリエイティブ産業の一分野として捉えられてきた。そして、他のヨーロッパ諸国やカナダに対して、長く産業としての優位性を保ってきた。しかしながら、前述した貿易総省の「ゲーム産業力」のランキング報告は、イギリス政府に少なからず衝撃を与えることになった。その結果、イギリス政府はゲーム産業に対する支援強化を迫られることになったのである。
下表は、イギリス貿易総省が、ゲーム産業の振興政策を再構築するに際して行った調査結果のひとつである。ここでは、ゲーム先進国を対象に、ゲーム産業に対する支援の種類と有無が調査されている。調査項目は、国税・地方税の優遇措置、国や地方の公的ファンド、民間ファイナンス、そしてＲ＆Ｄに対する税控除である。表中の◎は非常に充実、○は充実、△は存在するが不十分、×は不存在を意味する（◎○△×は筆者が評価を記号に置き換えた）。

表 7.2-01　先進国のゲーム産業支援策

	アメリカ	イギリス	カナダ	フランス	オーストラリア
国税のゲーム制作優遇措置	×	×	×	○	○
地方税のゲーム制作優遇措置	○	×	○	×	○
国のゲーム IP ファンド	×	×	○	○	○
地方のゲーム IP ファンド	△	△	◎	◎	◎
ゲームの R&D 税額控除	△	△	◎	○	○
民間ファイナンス	◎	△	◎	◎	△

調査対象に日本がはいっていないのは、イギリスにはアジアが視野に入っていなかったのか、それとも、日本にはゲーム産業に対する経済的支援が存在していないという判断によるものかは分からないが、皮肉な結果である。

表を一見すれば明らかなように、調査対象とした先進国の中で支援策を総合的に見て、もっとも充実しているのがカナダであり、もっとも不十分なのがイギリスである。また、従来、ゲーム産業に対する支援が存在しないといわれていたアメリカ合衆国にも税制上の

優遇措置があることが分かる。また、フランスやオーストラリアなど「ゲーム産業力」が高いと認識されてきた国々では、支援策が手厚く存在していることも見て取れる。

イギリスではこのような調査結果から、ゲーム産業振興政策を再構築することとした。まず、従来はクリエイティブ産業のひとつとして位置付けてきたゲームを IT 産業として位置付け直し、それがもつ技術開発と人材育成の重要性を強く打ち出すこととなった。その上で、カナダを手本として、1）公的支援の強化、2）産官学の連携の強化、3）地域産業クラスターの形成、などの政策を打ち出している。

これらの政策のうち産官学の連携に基づく地域クラスターの形成がいち早く実を結び、これまでゲーム産業の中心であったロンドンだけでなく、スコットランドにもゲーム産業の地域クラスターが形成され、また、イングランド中西部でも、コベントリー大学がウェストミッドランズ地域開発公社から支援を受けてシリアスゲーム・インスティテュート（SGI）を設立し、周辺のゲーム会社を巻き込んで、シリアスゲームによる地域産業振興の研究・開発拠点が形成されている。SGI は、3 年間で 310 万ポンド（約 6.5 億円）の予算規模の研究開発拠点であり、今後シリアスゲームに関連した応用研究や事業開発の支援を行い、既存の地域産業だけでなく、海外企業の事業参入や外国人研究者の支援、国際学術会議の招致も積極的に行う計画である。

7.2.4 韓国のゲーム産業振興政策

韓国のゲーム産業では、オンラインゲームが中心となっていることは周知である。しかしながら、韓国のオンラインゲーム業界は 2003～2004 年に韓国国内市場の売り上げが鈍化し、海外市場開拓を余儀なくされた。しかし、早期に海外進出に乗り出したことが、世界市場の先占効果を生み出し、オンラインゲームといえば韓国というように、オンラインゲームは韓国の国家イメージのひとつにまで成長した。

韓国のオンラインゲームの利益率は平均 40%ほどで、自動車や携帯電話といった韓国の主力製造業よりも高い。なぜなら、オンラインゲームは、部品や材料を輸入する必要もなく、韓国のゲーム開発者のアイデアと人材だけで作り上げることが可能なデジタルコンテンツであるからにほかならない。そのため韓国政府は、オンラインゲーム産業に対する積極的な支援策を次々に発表してきた。

韓国では、2006 年 5 月に、韓国内の経済学・経営学の研究者らを動員して、韓国ゲーム産業振興政策を発表している。その主要な内容は、1）2012 年までに韓国を世界三大ゲーム強国入りさせる、2）ゲーム開発技術に対する経済支援を行う、3）ゲームビジネスの海外展開に対する経済支援を行う、4）ゲームアカデミーを中心にゲーム開発者の高度な人材育成を進める、5）オンラインゲームだけでなくコンソールゲームの開発も進める、などというものであった。

そして、それまでふたつに分かれて実施されていた韓国内のゲームショーを、Gスター

として統一して行うようになり、また、韓国内のゲーム会社が海外のゲームの見本市に出展する際にも政府が強力に支援するようになった。その背景には、ゲーム産業を担当する官庁が、文化観光部と情報通信部に分裂していたのを改めて、両者が共同でゲーム産業の振興に当たることとし、文化観光部の管轄の下にあったゲーム開発振興院（のちにゲーム産業振興院）が実質的にその任に当たることとした点が大きい。その後、李明博政権の誕生によって、情報通信部が廃止され、ゲーム産業は文化観光部から変わった文化体育観光部が担当することになり、ゲーム産業振興院は、他のアニメ、映画、キャラクターなどの振興院とともに、韓国文化コンテンツ振興院（KOCCA）のもとに再編成されることとなった。ここに、韓国のゲーム産業振興は、他のコンテンツ産業の振興と一体のものとして、行政による振興政策の相乗効果を発揮するようになったのである。

文化体育観光部の柳仁村（ユ・インチョン）長官は 2008 年 12 月にソウルで「ゲーム産業振興第 2 回中長期計画（2008 年～2012 年）」を発表し、ゲーム産業を新成長動力に位置づけ、その振興のために大規模な支援政策を明らかにした。

柳長官は、2012 年までに韓国の世界三大ゲーム大国入りを目指し、3500 億ウォン（約 222 億円）を投入することを明らかにした。ここでいう三大ゲーム大国とは、アメリカ合衆国、日本、そして韓国を指しており、前述したイギリス貿易総省の「ゲーム産業力」ランキングにおいて、相対的地位を向上させている第 5 位の韓国が、現在第 4 位のイギリスと第 3 位のカナダを追い抜くことを意味している。

韓国文化体育観光部は 2012 年までに国内ゲーム市場規模を 10 兆ウォン（約 8,200 億円）、輸出規模は 36 億ドル（約 3,299 億円）達成を目標とし、それに向けて、ゲーム産業の戦略的輸出支援を大幅に強化し、例えば、独立開発スタジオを設立するなどして、ゲーム産業を先進型産業構造に転換、定着させる方針である。

この中期計画の柱は、1）戦略的な国際市場進出、2）次世代ゲーム製作基盤の造成、3）未来型創意人材と先導技術の確保、4）ゲーム文化の価値創造、5）流通環境の先進化、6）世界の e スポーツの先導、7）融合環境制度、8）政策の体系化、である。予算規模は国庫から 2,500 億（約 205 億円）ウォン、民間資本や地方自治体、基金などから 1,000 億ウォン（約 82 億円）が投じられる。さらに 2,000 億ウォン（約 164 億円）規模のファンド造成を進めることにしている。

一方、ゲームの社会的受容度を高めるために、ゲームソフトに対して取り締まりや規制でなく産業支援中心に据えるよう政策転換を進めて、レーティングシステムの制度改善に着手し、従来、政府直轄のゲーム・レーティング・ボード（Game Rating Board）が行っていたレーティングのワンストップシステムを構築し、長期的にはゲーム業界による自主審議基盤を設ける計画を掲げている。また、レーティングのみならず、ゲーム産業に関連する法律を産業振興に益するよう、体系的に改正することとしている。

韓国のゲーム産業は 2001 年以降、年平均 10%以上の高度成長を遂げ、国内の文化コンテンツ産業における輸出の半分を占めているという。また、輸出額を見てみると、2001

年に 1 億 3000 万ドル（約 119 億円）、2005 年に 5 億 6400 万ドル（約 515 億円）、2007 年に 7 億 8000 万ドル（約 715 億円）、2008 年には 10 億 6000 万ドル（約 971 億円）と順調に拡大を続けている。韓国でオンラインゲームがブームになってから 10 年経たないうちに、当初目標よりも 2 年早く 10 億ドル突破を達成したため、韓国政府は次の目標を「2012 年に 36 億ドル（約 3,300 億円）輸出」と大幅に引き上げている。

韓国のオンラインゲーム輸出は東南アジア、欧米、中国に続き、新興国の南米、ロシア、アフリカにも広がっており、輸出相手国は順調に拡大している。また、一時不況の影響を受けた国内の市場も順調に回復して、今後は 20%台の成長が見込まれている。

ただし、ゲーム産業全体が成長する中で、韓国国内のゲーム会社間の競争は激化しており、かつては韓国を代表していた NEXON、NCSoft、Gravity などに代わって、NHN が急速に業績を伸ばしている。その要因は、人気ゲームタイトルの帰趨にもよるが、ゲームソフト開発で培った技術とビジネスのノウハウを生かして、今や韓国随一のポータルサイトとなった NAVER の運営などを通じて総合 IT サービス企業への脱皮に成功したことが大きい。

7.2.5 ドバイのゲーム産業振興政策

アラブ首長国連邦の首長国のひとつあるドバイは、1966 年アラビア湾の油田発見を契機に、原油の採掘・販売によって得た利益をもとに、経済発展をとげた。しかしながら、他の産油国に比べて埋蔵原油量が少ないと言われているドバイは、石油資源が枯渇すると言われる 21 世紀半ば以前に資源を失う可能性があったために、早期に経済転換を遂げる必要があった。そこで注目されたのが、金融機能と観光資源である。金融・流通・観光を基幹産業とする都市国家として近年急速に経済を成長させてきた。

とくに 2003 年以降の発展は凄まじく、2004 年の後半から続く原油高がその発展を後押しした。2005 年度の経済成長率は 16%と高い成長率を見せており、2007 年の実質 GDP は 1,980 億ディルハム（約 6 兆円）に達した。人口も 220 万人を超え、ドバイは世界一の高さを誇るブルジュ・ハリファに代表される摩天楼の連なる幻惑的な都市国家として、中東において随一の繁栄を誇ってきた。

しかしながら、金融・流通・観光という新基幹産業の育成だけでは、経済基盤の脆弱性を克服することは難しい。そこでドバイが注目したのがゲーム産業である。王族のひとりが無類のゲーム好きという事情があったとも言われているが、ドバイは、次なる産業多角化の目玉としてゲーム産業に注目し、ゲーム産業に対する積極的な支援策を採っている。

もっとも代表的な支援策が、ドバイ国際展示場を会場とする Dubai World Game Expo（DWGE）の開催である。DWGE の目的は、まだ国内に十分な産業基盤を有しないゲーム産業に対して、海外からの投資を促し、また海外のバイヤーを招いて商談を成立させ、国内の開発者人材育成を進める、ドバイ発ゲームの海外進出のために多額の資金支援する、

などの役割が中心となっている。DWGE は 2008 年に第 1 回目を開催し、以後、毎年開催している。

2008 年秋のいわゆるリーマン・ショックを遠因として、2009 年 11 月には、政府が欧米系の金融機関に対し、政府系不動産開発会社のナキール社とその持ち株会社であるドバイ・ワールド社の債務約 590 億ドルについて支払い猶予を求めると発表したため、ドバイ・ワールド社と政府自体の債務不履行が懸念される状況となり、ユーロが売られ、ドバイ金融市場の株価も大幅に下落して、いわゆるドバイ・ショックを招いた。ドバイ・ショックはドバイ経済の脆弱性の一端を示すこととなり、苦しい資金繰りの中で、産業構造転換の象徴として、いっそうゲーム産業への支援が強化されることとなったのである。

ドバイのゲーム産業支援の最大の柱は人材の育成である。ドバイ人口の 7～8 割は外国人であり、ドバイの産業を支えているのも外国人労働である。しかしながら、ゲーム産業においては、ドバイ人（ドバイ出身のアラビア人）による純血主義とも言える政策が採られ、それまで海外に留学していた IT 系の学生を帰国させ、彼らを指導者として、国内の開発者育成を進めている。また、それとは反対に、優秀な学生を日本や欧米の大学に留学させ、当地でのゲーム開発の経験を積ませてから将来帰国させるための先行投資も並行して行っている。

また、ゲーム流通の集散地としての機能強化も狙っており、それは、前述した DWGE の開催に端的に現れている。もともと、ドバイ国内には、ゲームの多様な流通ルートが存在し、日本を始めとして世界各国の正規版のパッケージゲームが流通していた。それは、ドバイを経由して欧米とアジアを結ぶ流通ルートを利用してドバイ国内にもたらされたものである。ドバイの地の利や従来の実績を利用して、そのゲーム流通ルートをさらに太く強固なものにしていこうという政策である。

さらに、ドバイへゲームとその関連会社の誘致を積極的に進めている。これまで、他分野においても企業誘致は積極的に進められてきたが、ゲームに関しても、国内の起業を奨励するばかりでなく、税制面での優遇措置を通じて、海外からの企業誘致を進めている。以上のような取り組みの結果、すでにドバイで開発されたゲームソフトが誕生し、市場に流通するようになった。ただし、国内で開発されるゲームソフトは PC 用が中心となっており、マルチプラットホーム展開や海外進出は、今後、さらに強化するべき課題とされている。

7.3 日本の新ゲーム産業戦略の策定に向けて

7.3.1 ゲーム産業戦略（2006 年）の検証の必要性

日本では、2006 年 8 月に経済産業省商務情報政策局文化情報関連産業課（通称メディアコンテンツ課）を事務局として、産業界・学術界の委員からなる「ゲーム産業戦略研究

会」が組織され、ゲーム産業戦略が策定された[21]。このゲーム産業戦略は、2000 年ごろから日本国内でも強化されてきたアニメ、映画、コミックなどいわゆるコンテンツ産業の振興政策の一環として立案されたものである。新たに登場してきた次世代ゲーム機（Wii、プレイステーション 3、Xbox360 とニンテンドーDS、プレイステーション・ポータブル）の展開という状況の下で、2010 年までの日本のゲーム産業を展望し、持続的な成長を目指して描いた戦略である。本年はその目標年に当たる。

このゲーム産業戦略では、まず、ゲーム産業の意義、歴史、現状と課題をまとめている。ゲーム産業の意義では、ゲーム産業を次の 4 点から意義付けた。

① コンテンツ産業で最大の輸出産業であり、日本のソフトパワーの海外展開に貢献している産業である。
② 今後のマーケット拡大の可能性が大きい産業である。
③ コンテンツ産業を技術面で牽引している産業である。
④ ユビキタスネット社会の実現に貢献している産業である。

次にゲーム産業の歴史においては、1970 年以前と 70 年以降を 10 年毎に区切って、プラットホームの変遷を中心に振り返っている。そして、2006 年現在の状況を、1）国内市場、2）海外市場、3）人材育成、4）社会環境に分けて分析している。

それらに基づいて、2006 年のゲーム産業戦略では、以下のような二つの未来像を 2010 年までにゲーム産業の目指すべき姿として描いている。

(a) 日本のゲーム産業が、世界に誇るエンターテインメントコンテンツとして、また、日本ブランドを支える存在として、今後とも競争力を維持、強化し、輸出産業として発展し続け、世界をリードしていくこと。
(b) 日本のゲーム産業が青少年健全育成等の課題に積極的に対応し、教育・学習、医療・福祉等のエンターテインメント以外の分野においてもポテンシャルを発揮するなど、これまで以上に社会や国民に広く支持を受けること。

である。そして、そのために、1）ゲームの創造・開発力を強化するための開発戦略、2）海外市場や新しい環境への展開を強化するためのビジネス戦略、そして 3）社会とのコミュニケーションを強化するためのコミュニケーション戦略、という三つの戦略を提唱した。

私たちの 2007、2008、2009 年度の 3 年間に渡る調査を開始した契機は、まさに 2006 年のゲーム産業戦略にあり、戦略を深化させ推進させるためであった。しかしながら、2006

[21] ゲーム産業戦略（2006 年）は以下の URL を参照のこと。
経済産業省「『ゲーム産業戦略～ゲーム産業の発展と未来像～』の公表について』
http://www.meti.go.jp/press/20060824005/20060824005.html

年のゲーム産業戦略の策定以来、日本のゲーム産業を取り巻く環境はドラスティックに変化した。私たちの調査では、この間の環境の変化を踏まえて、ゲーム開発のための技術動向を中心にまとめてきたつもりである。また、韓国をはじめとする諸外国が、ゲーム産業の本質を IT と捉え、積極的に行政からの支援策を展開する中で、開発技術の問題だけでなく、とくに、産官学の連携と人材育成の重要性を指摘してきた。

しかし、それにもかかわらず、日本のゲーム開発をめぐる状況は厳しさを増しており、日本の国際的な地位は相対的に下がるばかりである。これまでは、コンテンツ産業の輸出を牽引し、技術や経済の波及効果の上で、コンテンツ産業の優等生と言われてきた日本のゲーム産業は、2010 年現在、大きな岐路に立たされている。

日本がこれまで、かろうじて優位を保ってきたゲーム産業の将来像をどのように描くのか、そして、そのために、ゲーム開発者、ゲーム会社、そして、行政や研究者が何をしなければならないのか、ゲーム産業戦略（2006 年）の目標年である 2010 年の今こそ、過去の 4 年間を振り返って検証し、新たな戦略を策定すべきときである。

7.3.2 ゲーム産業の経済波及効果への期待

2008 年 10 月に、コンピュータエンターテインメント協会（CESA）が「ゲーム産業の分析と波及効果調査」を発表した[22]（調査は日本総合研究所に委託）。発表内容の要点は以下の 2 点である。

① ソフトや関連書籍、アニメーション、キャラクターグッズなどの周辺投資を含めた日本国内での経済波及効果は約 1 兆円であること。
② ゲーム産業をほかの産業と比較すると、成長性、収益性、国際性の 3 点が強みであること、さらなる発展のためにはグローバル化への対応、プロデューサーの人材育成が課題であること。

前者について見てみると、国内では年間約 1,000 本の新タイトルが発売され、その市場規模は 3,823 億円だが（2007 年度実績「2008CESA ゲーム白書」より）、同調査での経済波及効果とは、それらすべてのタイトルについて、過去の開発・販売への投資額と、将来の販売やアフターフォローなどへの投資額まで含めたものであり、ハード関連の経済効果は含まれていない。そして、1 兆円の内訳は、プログラムやデザインなどの制作活動や広告宣伝などゲーム開発・販売による経済波及効果が 7,411 億円、書籍やキャラクターグッ

[22] CESA の「ゲーム産業の分析と波及効果調査」については以下の URL を参照のこと。
「ゲーム産業の分析と経済波及効果（速報版）」
http://research.cesa.or.jp/pdf/20081001.pdf
「ゲームソフトウェア産業の分析と波及効果に関する調査報告書」
http://www.cesa.or.jp/uploads/info_081128.pdf

ズ、衣料など関連商品の製造・販売、ライセンシーによる周辺投資がもたらす経済波及効果が 2,850 億円となっている。

つまり、ゲームソフトウェア産業の産業連関乗数は 2.3 である。下表は、他の産業の産業連関乗数を示したものである。これを他の産業と比較してみると、ゲーム産業の産業連関乗数は、情報通信業（ICT 産業）の 2.0～2.3 の上位、かつ、サービス産業（エンターテインメント産業を含む）の 1.9～2.3 の上位と等しいことが分かる。ゲーム産業の本質を考えてみると、ゲーム産業は、まさに、情報通信産業とサービス産業の両方の側面を有する産業にほかならないから、それらの産業と比較して上位に位置するということは、同一あるいは類似産業の中でも、他産業に多大な経済波及効果を有する産業であるとみなしてよいのである。ちなみに、製造業において最大の産業連関乗数 3.5 を示しているのは自動車産業である。

表 7.3-01　他の産業の産業連関乗数

製造業	2.4～3.5
情報通信業（ICT 産業）	2.0～2.3
サービス産業（エンターテインメント産業を含む）	1.9～2.3
医療産業	1.9～2.0

次に、後者については、日本の世界シェアが高い産業としてのデジタルカメラ産業(デジタルカメラを内蔵する製品を含む)、アジアでのブランド力を高めている化粧品産業、同じエンターテインメント産業で市場規模が同程度の映画産業、国際競争力が高い日本の代表産業である自動車産業を比較対象としている。そして、過去 5 年間の成長率や収益性、輸出割合や資本構成から見る国際性などを比較している。その結果、過去 5 年間の成長率、収益率はゲーム産業がトップであり、輸出割合ではデジタルカメラ産業に次いでゲーム産業は 2 位となったが、過去 5 年間の輸出割合の成長率ではゲーム産業がトップであった。以上より、ゲーム産業は日本を代表する産業の中でもとりわけ優秀な産業であることが証明されたと言える。

もちろん、経済不況時においては、経済波及効果がマイナスに働くことは、先のリーマン・ショックの際の自動車産業の例が端的に示すところである。しかし、好況時や回復時においては、他の産業を牽引していく重要な基幹産業として位置付けられる。現今の日本の経済状況においては、まさにゲーム産業に対して、コンテンツ産業、情報通信業、そしてエンターテインメント産業の経済回復の中心的な役割を期待しうるのである。

しかし、現実には、日本のゲーム産業には、ただちに如上の立場を占めるだけの力があるとは言えないだろう。CESA の「ゲーム産業の分析と波及効果調査」にも述べられている通り、国内市場の成熟に伴うグローバル化の推進、開発費高騰への対応の必要性、クロスメディア展開の促進が喫緊の課題として存在しており、ヒット作を生み出すクリエイタ

ーに加え、クロスメディア展開を含めた資金調達などゲームをビジネスとして戦略的に行えるプロデューサー人材を育成する必要があるからである。

7.3.3 新ゲーム産業戦略の視点

2006 年に策定されたゲーム産業戦略を検証し、新たなゲーム産業戦略を策定する上で、ポイントとなる項目と方向性について列挙する。とくに、前回のゲーム産業戦略と比較して、ゲーム産業を取り巻く現在の状況に鑑み、新たに留意すべきポイントである。また、新ゲーム産業戦略の策定に際しては、2006 年の日本のゲーム産業戦略と相前後して策定された諸外国のゲーム産業振興政策を参照することも希望する。

(1) 開発技術支援

ゲーム開発における技術の重要性は、3 年間の本調査研究の基本であり、それぞれの年度の報告書において、つねに具体的に主張してきたところである。ゲーム開発において、先端技術の利用はゲームの進化と多様化を実現する要素として重要である。しかしながら、ゲーム開発技術のベクトルは、つねに最新技術のゲームへの応用、つまり垂直方向にのみあるのではない。故横井軍平氏の「枯れた技術の水平思考」の言辞に象徴的に示されるように、他分野の安定した技術のゲームへの転用、すなわち平行方向のベクトルもまた重要である。そのような、垂直・平行の両方向に向ってゲーム技術の展開を実現する空間（環境）の確保が課題である。それは、欧米の技術動向に関する情報提供、技術研究のための資金、そしてそれを推進する人材の確保を意味する。また、直接ゲーム開発に関わる技術ではないが、開発工程を管理する技術、商品としてのゲームソフトの品質を管理するための技術は、従来、日本が苦手とする分野であり、今後の強化が必要である。

(2) ビジネス支援

ニンテンドーDS と DS 用ゲームソフトは、国内市場において、従来ゲーム機やゲームソフトに接する機会の少なかった女性や高齢者に広く受け入れられ、国内市場を拡大した。しかし、それも 2009 年には終息し、国内市場は再びシュリンクしようとしている。国内にはいまだ潜在的市場が存在すると思われるが、今後、それが急速に顕在化し飛躍的に拡大することは期待できない。とすれば、当然のことながら、海外市場のさらなる開拓が、今までにもまして今後の大きな課題となる。欧米やオセアニアなど、ゲームの普及国だけではなく、南米やアフリカをも視野に入れた、大胆な海外進出の戦略が必要となるであろう。とくに、中国は巨大な市場であることが認識されつつも、ゲーム産業は他産業に比較して出遅れている市場である。各国・各地域の実情にあわせてビジネス戦略を練り、それ

を実践するためのノウハウの共有と人材の確保が肝要である。

(3) 知的財産の保護

「ゲームは特許の塊」と言われるほど、ゲームと知的財産は密接な関係にある。しかしながら、ニンテンドーDS が発売された 2004 年以降、マジコン（マジックコンピュータ）を利用して違法複製ゲームソフトをプレイする行為が広く行われるようになり、ゲーム会社に深刻な経済的打撃を与えるようになった。違法複製ゲームソフトは、インターネットにより国境を越えて交換・共有されている。このような事態に対しては、ゲーム会社が一致して対応する必要があり、同時に、行政による国内法制度の整備や国際的対応が早急に求められる。ゲーム開発者が創意工夫を凝らして開発するゲームデザインの保護から、ゲーム会社の経済的利益の保護にいたるまで、広範な対策が必要である。同時に、不正行為の根絶のための啓発・啓蒙活動のさらなる展開も重要である。

(4) 中小規模のゲーム会社への支援

日本のゲーム会社は、コンテンツ産業の中でも、大企業から中小規模の企業まで多様な規模の企業が存在しているという特徴がある。ゲーム産業の分業が進むにつれて、中小規模の企業の存在感は大きくなっており、日本のゲーム産業の存立にとって重要な役割を果たしている。また、ゲーム文化の多様性という観点から見ても、個性的なゲームソフトを提供する中小規模のゲーム会社は不可欠である。しかしながら、資金調達や人材育成などの側面において、中小規模の企業は不利な立場にある。また、海外進出においてもノウハウの蓄積が少ないのが現状である。そのような中小規模の会社に対して、個別の支援ではなく、ワンストップサービスを行えるシステムの設置が求められる。

(5) ゲーム開発人材の育成と就業支援

ゲーム産業への就職を希望する学生に対して、多様な教育機関が専門教育を実施してきた。しかしながら、未だに教育機関とゲーム産業の距離、とくに学生とゲーム会社の距離が縮まったとは言えない。つまり、学生や教育機関に対して、ゲーム産業に関する正確な情報が十分に伝わっているとは言えず、就職・雇用のミスマッチは後を絶たない。学生と会社の接点を多様な形態で設定し、また、インターンシップによって学生が事前にゲーム会社の業務などを体験出来る機会を増やすべきである。また、ゲーム開発技術の高度化・専門化に応じて、効率的なカリキュラムや教授内容を実現するために、教育機関とゲーム会社は一層協力関係を強化しなければならない。ゲーム開発者教育の国際化、教材の整備と教員の養成も重要な課題である。

(6)　産官学連携の更なる強化

技術開発や人材育成などの個別具体的な課題においては、産学・産官あるいは官学の連携は進展した。しかしながら、それらの課題を総合的に、かつ、産官学がともに取り組むことの出来る共通プラットホームは未だに存在しない。海外の事例や教訓を分析して、産官学がそれぞれの役割に応じて共同で取り組むことの出来るプラットホームの構築が求められる。

(7)　ゲームの社会的受容の促進

2006 年のゲーム産業戦略においては、ゲーム産業界と社会とのコミュニケーション戦略として位置付けられていた課題である。その後、社会に有用なシリアスゲームの普及などによって、ゲームに対する社会の見方は変化しつつある。しかしながら、未だにゲーム・リテラシーが十分に社会のものになっているとは言いがたい。コミュニケーションから理解と受容の段階へ進める必要がある。それは、ゲーム産業の発展のための社会的基盤の構築としてばかりでなく、さらに進化するであろうデジタル社会に対して、人々が正しく対応していくための人類史的課題とも言える。

以上、2006 年のゲーム産業戦略の検証と新たなゲーム産業戦略の策定のために、重要と思われる視点と方向性について述べてきた。繰り返しになるが、ゲーム産業戦略の目標年である 2010 年の今こそ、ゲーム産業界、行政、そして研究者が一体となって、今後の日本のゲーム産業の発展の可能性と方向性について真摯に議論すべきときである。それが、世界のゲーム産業の趨勢に取り残されること無く、ゲーム産業を起爆剤として、日本の社会と経済が持続的発展をとげるための唯一の方策である。

附録：「ゲーム開発者の実態調査」単純集計

Q1 メールアドレス

（非公開）

Q2 性別

1. 男	416（87%）
2. 女	61（12%）

Q3 平均年齢

33.8 歳

Q4 最終学歴

1. 中学校卒業	2（0%）
2. 高等学校卒業	51（10%）
3. 各種学校・専修学校（専門学校）卒業	144（30%）
4. 高等専門学校・短期大学卒業	24（5%）
5. 大学卒業	210（44%）
6. 大学院修士課程修了	44（9%）
7. 大学院博士課程修了	2（0%）

Q5 最終学歴の学問系統

1. 法律・政治系	8（1%）
2. 経済・経営・商学系	35（7%）
3. 社会・情報系	45（9%）
4. 国際関係系	2（0%）
5. 文学・人文・人間系	34（7%）
6. 外国語系	3（0%）
7. 教育・福祉系	5（1%）
8. 家政・生活系	4（0%）
9. 芸術・表現系	117（24%）
10. 体育・健康系	0（0%）
11. 教養・総合系	5（1%）
12. 理・工系	157（32%）
13. 農・獣・畜産・水産系	3（0%）

14. 医・歯・薬系	0（0%）
15. 看護・保健・衛生系	0（0%）
16. その他	59（12%）

Q6 配属関係

1. 未婚	280（58%）
2. 既婚	194（40%）
3. 離死別	3（0%）

Q7 子どもの有無

1. あり	97（20%）
2. なし	380（79%）

Q8 あなたの家庭での主たる生計の維持者（世帯で最も生活費を負担している方）

1. 本人	389（81%）
2. 本人以外	45（9%）
3. その他（生計費は折半等）	43（9%）

Q9 平均年収

5,184,995 円

Q10 現在の役職

1. 一般	296（62%）
2. 主任・係長クラス	79（16%）
3. 課長クラス	36（7%）
4. 部長クラス以上	66（13%）

Q.11 現在の職種

1. プロデューサー	53（11%）
2. ディレクター	43（9%）
3. プログラマー	131（27%）
4. グラフィッカー	103（21%）
5. プランナー	55（11%）
6. サウンド	16（3%）
7. デバッガー	6（1%）

8. ネットワーク・エンジニア	8（1%）
9. その他	62（12%）

Q12 あなたの仕事への関わり方

1. 後方支援	128（26%）
2. 企画開発	349（73%）

Q13 現在のゲーム会社の所在地

北海道地方	4（1%）
東北地方	1（0%）
関東地方	418（88%）
中部地方	11（2%）
近畿地方	37（8%）
中国地方	0（0%）
四国地方	0（0%）
九州地方	5（1%）
沖縄地方	1（0%）

Q14 現在のゲーム会社の勤続年数

6.6 年

Q15 ゲーム産業の経験年数

10.2 年

Q16 あなたは現在どのような事業に関わっていますか。

1. 据置型ゲーム事業	292（61%）
2. 携帯型ゲーム事業	226（47%）
3. モバイル・コンテンツ事業	117（24%）
4. オンラインゲーム事業	93（19%）
5. PC ソフトウェア事業	74（15%）
5. アミューズメント機器事業	82（17%）
6. アミューズメント施設事業	14（2%）
7. パチンコ・パチスロ事業	40（8%）
8. トイホビー事業	8（1%）
9. その他	34（7%）

Q17 あなたはなぜゲーム業界に入ろうと思いましたか。

1. ゲームが好きだから	356（74%）
2. コンピューターが好きだから	169（35%）
3. 自分の能力・個性を活かせるから	275（57%）
4. 技術を覚えられるから	82（17%）
5. 将来性を感じたから	92（19%）
6. 給料が高いから	9（1%）
7. 仕事が楽しそうだから	248（51%）
8. 社会に貢献できそうだから	32（6%）
9. 友人・知人に勧められたから	22（4%）
10. その他	37（7%）

Q18 あなたはなぜ現在のゲーム会社に就職しましたか。

1. 自分の能力・個性を活かせるから	258（54%）
2. 仕事が面白いから	163（34%）
3. 技術を覚えられるから	126（26%）
4. 会社の将来性を考えて	96（20%）
5. 一流会社だから	93（19%）
6. 経営者に魅力を感じたから	45（9%）
7. 地理的条件が良いから	79（16%）
8. 給料が高いから	20（4%）
9. 実力主義の会社だから	40（8%）
10. 福利厚生が充実しているから	26（5%）
11. 労働時間が短く、休日が多いから	15（3%）
12. 先輩・知人が多いから	29（6%）
13. どこも行くところがなく、やむなく	54（11%）
14. その他	96（20%）

Q19 あなたはどのようにして現在の仕事を得ましたか。

1. 民間の求人ウェブサイトを通じて	66（13%）
2. 公共職業安所（ハローワーク）を通じて	3（0%）
3. 企業の求人情報を通じて	151（31%）
4. 求人情報誌を通じて	19（3%）
5. 新聞の求人情報を通じて	2（0%）
6. 折り込み広告を通じて	0（0%）

7. 友人を通じて	91（19%）
8. 起業	38（7%）
9. 紹介予定派遣	32（6%）
10. 雇用形態の転換制度を利用して	4（0%）
11. その他	71（14%）

Q20 あなたが現在の仕事を続けている理由は何ですか。

1. 社会の一員として務めを果たすため	103（21%）
2. 生きがいのひとつ	163（34%）
3. お金を得るため	337（70%）
4. 社会で自分が所属する場所として	145（30%）
5. 自分が自分らしく生きられる仕事だから	179（37%）
6. この仕事が楽しいから	258（54%）
7. 現在の仕事が好きだから	258（54%）
8. 自分の才能や能力を発揮するため	253（53%）
9. 作品を通じて自分の考え方を伝えるため	77（16%）
10. 自分にとっては一番楽な仕事だから	62（12%）
11. なんとなく	35（7%）
12. 他にできることがないので仕方なく	53（11%）
13. その他	26（5%）

Q21 あなたはこれまでにゲーム開発の仕事を解雇されたことがありますか。

1. ある	73（15%）
2. ない	404（84%）

Q22 ゲーム開発の仕事を解雇された理由は何ですか。

1. プロジェクトが途中でキャンセルされたため	10（13%）
2. 会社が倒産したため	14（19%）
3. 事業部門が閉鎖されたため	12（16%）
4. 退職勧奨のため	21（28%）
5. その他	16（21%）

Q23 あなたはプロジェクトの途中で辞めたことがありますか。

1. ある	141（29%）
2. ない	334（70%）

Q24 プロジェクトの途中で辞めた理由は何ですか。

1. プロジェクトが失敗に陥ることを感じたため	11（7%）
2. 管理者または協働者との対立のため	28（20%）
3. プロジェクトを続けるには疲弊していたため	38（27%）
4. 他によりよい仕事を見つけたため	28（20%）
5. その他	35（25%）

Q25 あなたは転職をしたことがありますか。

1. ある	282（59%）
2. ない	195（40%）

Q26 あなたは何回転職をしましたか。

1. 1 回	92（32%）
2. 2 回	73（25%）
3. 3 回	43（15%）
4. 4 回	22（7%）
5. 5 回以上	51（18%）

Q27 あなたの前職（直近の仕事）はゲーム産業でしたか。

1. ゲーム産業	188（66%）
2. ゲーム産業以外	94（33%）

Q28 あなたはなぜ転職しましたか。

1. 自分の能力・個性を活かせなかったから	94（33%）
2. 仕事が面白くなかったから	82（29%）
3. 技術を身につけることができなかったから	46（16%）
4. 会社の将来性を考えて	97（34%）
5. 会社に魅力を感じなかったから	102（36%）
6. 経営者に魅力を感じなかったから	73（26%）
7. 地理的条件が悪かったから	14（5%）
8. 給料が低いから	71（25%）
9. 年功序列の会社だから	11（3%）
10. 福利厚生が不十分だったから	32（11%）
11. 労働時間が長く、休日が少なかったから	43（15%）
12. 人間関係が悪かったから	57（20%）

13. どこも行くところがなく、やむなく	9（3%）
14. その他	101（36%）

Q29 あなたの現在の仕事の満足度についてお伺いします。

1. 大変満足	38（7%）
2. 満足	185（38%）
3. 普通	134（28%）
4. 不満	86（18%）
5. 大変不満	34（7%）

Q30 あなたはどのくらいの期間、ゲーム産業で働く計画ですか。

1. 2 年以内に他の産業で仕事を探す	36（7%）
2. 2-5 年の間ゲーム産業で働き、他の産業で仕事を探す	83（17%）
3. 5-10 年の間ゲーム産業で働き、他の産業で仕事を探す	73（15%）
4. 生涯、ゲーム産業で働く	285（59%）

Q31 あなたがゲーム開発のキャリアを変えられるとしたら、それは何ですか。

1. より面白いプロジェクトで働く	219（45%）
2. より多くの収入を得る	83（17%）
3. より大きな責任を引き受ける	61（12%）
4. より短い時間で働く	39（8%）
5. 仕事安定性を獲得する	64（13%）
6. 何も変えられない	11（2%）

Q32 あなたがストレスを引き起こしているとしたら、その主要な原因は何ですか。

1. 官僚的な組織	67（14%）
2. 管理者と開発者の関係悪化	83（17%）
3. 協働者との関係悪化	57（11%）
4. 会社とパブリッシャーの関係悪化	15（3%）
5. 会社の財政難	98（20%）
6. 次のプロジェクトがどこから来るのかわからない	23（4%）
7. すべてはうまくいっている	26（5%）
8. その他	108（22%）

Q33 あなたの現在の就労形態についてお伺いします。

1. 正社員	343（71%）
2. 契約社員	37（7%）
3. 常用型派遣社員	11（2%）
4. 登録型派遣社員	20（4%）
5. アルバイト・パート	10（2%）
6. フリーランス・独立開発者	19（3%）
7. 経営者	34（7%）
8. その他	3（0%）

Q34 あなたの労働時間からみた勤務形態は、次のうちどれに近いですか。

1. 始・就業時間が一定している通常勤務	143（29%）
2. フレックスタイム勤務	119（24%）
3. 短時間勤務・短日数勤務	1（0%）
4. 裁量労働制	201（42%）
5. 交代勤務・変則勤務	0（0%）
6. その他	13（2%）

Q35 あなたは一週間でどのくらいの時間働いていますか。

1. 35 時間未満	6（1%）
2. 35-45 時間未満	151（31%）
3. 45-55 時間未満	189（39%）
4. 55 時間以上	131（27%）

Q36 現在のゲーム会社では、どのように時間外を補償していますか。

1. 従業員の通常の 1 時間ごとの率で、時間外が支払われている	52（10%）
2. 割増料金で時間外が支払われている	121（25%）
3. 時間外が計算されて、時間外と同等の時間を休暇で補償されている	18（3%）
4. プロジェクトの終了時に休暇で補償されるが、時間外は計算されない	34（7%）
5. プロジェクト報奨金（一時金）が支払われている	5（1%）
6. 時間外は補償されない	179（37%）
7. その他	68（14%）

Q37 あなたが関わった直近のプロジェクトでは、どのくらいの繁忙期がありますか。

1. 繁忙期はない	17（3%）
2. 最終的なベータ・テストの時	106（22%）
3. あらゆるマイルストーンの前	254（53%）
4. 毎月	73（15%）
5. その他	27（5%）

Q38 あなたが関わった直近のプロジェクトでは、どのくらいの長さの繁忙期がありますか。

1. 1 週間未満	39（8%）
2. 1-2 週間未満	44（9%）
3. 2 週-1 ヶ月間未満	79（16%）
4. 1 ヶ月-2 ヶ月間未満	107（22%）
5. 2 ヶ月-3 ヶ月間未満	74（15%）
6. 3 ヶ月以上	134（28%）

Q39 繁忙期の週の間、あなたはどのくらいの時間働いていますか。

1. 35 時間未満	2（0%）
2. 35-45 時間未満	23（4%）
3. 45-55 時間未満	73（15%）
4. 55-65 時間未満	152（31%）
5. 65-80 時間未満	134（28%）
6. 80 時間以上	93（19%）

Q40 繁忙期に関して、あなたが関わった直近のプロジェクトの方針についてお伺いします。

1. 何も問題がなくても誰もが繁忙期を経験する	222（46%）
2. すべての開発者は長時間働くが、管理はされない	151（31%）
3. 管理者は、各自に与えられた仕事が終了したかどうかについて関与しない	66（13%）
4. 個人や家族の理由のために繁忙期を拒否することは可能である	178（37%）
5. 管理者は、ゲーム産業におけるビジネスでは繁忙期を一般的なものと考えている	312（65%）
6. 管理者は、繁忙期を必要悪とみなして、従業員への影響を最小にしようとする	156（32%）
7. 管理者は、週 40 時間労働のような繁忙期の廃止方針を適用する	24（5%）
8. その他	33（6%）

Q41 あなたの現在のゲーム会社では、開発者の育成（OJT、Off-JT、キャリア開発支援）をどの程度実施していますか。

	1. 十分実施している	2. ある程度実施している	3. あまり実施していない	4. 全く実施していない
a. OJT	48（10%）	191（40%）	137（28%）	101（21%）
b. Off-JT	27（5%）	148（31%）	178（37%）	124（25%）
c. キャリア開発支援	11（2%）	109（22%）	180（37%）	177（37%）

Q42 あなたの現在のゲーム会社では、現在、開発者の Off-JT について具体的にどのような取組みを行っていますか。

1. 新入社員研修	282（59%）
2. 階層別研修	119（24%）
3. 職種別研修	74（15%）
4. 語学研修（英語）	22（4%）
5. 社内勉強会	117（24%）
6. 国内教育機関への派遣	22（4%）
7. 海外教育機関への派遣	6（1%）
8. 国内研修機関・研究会等への派遣	65（13%）
-a.CESA デベロッパーズカンファレンス（CEDEC）	256（53%）
-b.オンラインゲームカンファレンス（OGC）	30（6%）
-c.デジタルコンテンツ協会（DCAJ）	28（5%）
-d.国際ゲーム開発者協会（IGDA Japan）	53（11%）
-e.ブロードバンド推進協議会（BBA）	2（0%）
-f.ゲーム学会（GAS）	4（0%）
-g.日本シミュレーション&ゲーミング学会研究会（JASAG）	2（0%）
-h.情報処理学会（IPS）	4（0%）
-i.日本デジタルゲーム学会（DiGRA JAPAN）	24（5%）
9. 海外研修機関・研究会等への派遣	39（8%）
-a.Game Developers Conference（GDC）	150（31%）
-b.SIGGRAPH	135（28%）
-c.GC Developers Conference（GCDC）	14（2%）
-d.Korea Games Conference（KGC）	1（0%）
10. 専門誌の閲覧	193（40%）
-a.週刊ファミ通	217（45%）
-b.Game Developers Magazine	23（4%）

-c.専門書（日本語）	131（27%）
-d.専門書（英語）	50（10%）
11. 特に行っていない	99（20%）
12. その他	11（2%）

Q43 あなたの現在のゲーム会社では、開発者のキャリアをどのように支援していますか。

1. キャリア開発に関する情報提供	95（19%）
2. キャリア開発の費用援助	52（10%）
3. 資格取得のための支援	76（15%）
4. 資格取得後の一時金支給	74（15%）
5. キャリア開発のための時間確保	26（5%）
6. 自己申告制度の導入（自律的なキャリア開発に基づく異動希望を申告する制度）	63（13%）
7. 社内公募制度の導入（プロジェクト等で必要な人材を社内公募する制度）	102（21%）
8. キャリア・カウンセリングの実施	56（11%）
9. 心身の健康管理に対する支援	166（34%）
10. 特別な支援はしていない	187（39%）
11. 自己負担で自己研鑽に努めている	135（28%）
12. その他	10（2%）

Q44 あなたの現在のゲーム会社では、開発者がキャリアを形成するために、どのような情報を提示していますか。

1. 企業が開発者に求める人材要件に関する情報	71（14%）
2. 能力を修得した後の処遇に関する情報	39（8%）
3. 社内の教育訓練、キャリア開発に関する情報	91（19%）
4. 社外の教育訓練、キャリア開発に関する情報	56（11%）
5. 各職掌および求められる技能や能力に関する情報	46（9%）
6. 将来的なキャリア・パスに関する情報	45（9%）
7. 将来的なキャリア開発方法に関する情報	24（5%）
8. キャリア開発について相談できる機関に関する情報	22（4%）
9. 特に提示していない	301（63%）
10. その他	10（2%）

Q45 評価や処遇に対するあなたの現在のゲーム会社の考え方についてお伺いします。

	1. 大変そう思う	2. そう思う	3. あまりそう思わない	4. 全くそう思わない
過去の成功や実績に基づいた意見や提案が評価される	87（18%）	238（49%）	121（25%）	31（6%）
給与制度は、成果や業績的な要素が大きい	71（14%）	212（44%）	145（30%）	49（10%）
個人の業績に連動して、報奨金等の一時金で処遇されている	16（3%）	121（25%）	188（39%）	152（31%）
個人の業績に連動して、昇進や昇格等のポストで処遇されている	22（4%）	189（39%）	179（37%）	87（18%）
評価結果に基づく給与ダウンは頻繁に発生している	29（6%）	76（15%）	221（46%）	151（31%）
評価結果に基づく降格人事は頻繁に行われている	19（3%）	73（15%）	211（44%）	174（36%）
評価基準は開発者に公開されている	21（4%）	123（25%）	158（33%）	175（36%）
評価結果・評価理由は本人に十分開示されている	25（5%）	157（32%）	177（37%）	118（24%）
評価に関する苦情処理の仕組みがある	12（2%）	86（18%）	184（38%）	195（40%）

Q46 あなたの給与体系はどのような仕組みになっていますか。

1. 年俸制	241（50%）
2. 定期昇給	127（26%）
3. 年齢給	61（12%）
4. 業績給・成果給	73（15%）
5. 職務給・役割給	155（32%）
6. 能力給	154（32%）
7. 退職金の基本給組み入れ	25（5%）
8. 個人業績に連動した賞与・一時金	51（10%）
9. 開発チーム別業績賞与・一時金	43（9%）
10. 開発タイトル報奨金	63（13%）
11. 企業業績に連動した賞与・一時金	122（25%）
12. ストックオプション	40（8%）
13. 通勤手当	339（71%）

14. 時間外手当	174（36%）
15. 扶養手当	111（23%）
16. 住宅手当	98（20%）
17. 資格手当	36（7%）
18. 役職手当	119（24%）
19. その他	9（1%）

Q47 あなたの現在のゲーム会社（ゲーム事業部門）についてお伺いします。

1. 同時に 4 つ以上のプロジェクトに取り組んでいるスタジオ	319（66%）
2. 同時に 2-3 のプロジェクトに取り組んでいるスタジオ	103（21%）
3. 1 つのプロジェクトに取り組んでいるスタジオ	21（4%）
4. 独立チーム	9（1%）
5. フリーランサーまたは契約労働者	16（3%）
6. その他	9（1%）

Q48 あなたの現在のゲーム会社の資本金についてお伺いします。

1. 500 万円未満	36（7%）
2. 500 万-1000 万円未満	37（7%）
3. 1000 万-3000 万円未満	70（14%）
4. 3000 万-1 億円未満	64（13%）
5. 1 億-10 億円未満	70（14%）
6. 10 億-50 億円未満	37（7%）
7. 50 億円以上	163（34%）

Q49 あなたの現在のゲーム会社の従業員数（正社員・契約社員・派遣社員・アルバイト・パートを含む）についてお伺いします。

1. 1 人	12（2%）
2. 2-5 人	17（3%）
3. 6-10 人	22（4%）
4. 11-20 人	25（5%）
5. 21-50 人	64（13%）
6. 51-100 人	43（9%）
7. 101-300 人	75（15%）
8. 301-500 人	22（4%）
9. 501-1000 人	22（4%）

10. 1001-3000 人	129（27%）
11. 3001-5000 人	33（6%）
12. 5001 人以上	13（2%）

Q50 あなたの現在のゲーム会社の業績は、前年度と比べてどのように推移していますか。

1. 増益	78（16%）
2. 横ばい	134（28%）
3. 減益	265（55%）

Q51 あなたの現在のゲーム会社の従業員数は、前年度と比べてどのように推移していますか。

1. 増加	139（29%）
2. 横ばい	178（37%）
3. 減少	160（33%）

Q52 あなたの現在のゲーム会社の事業についてお伺いします。

1. デベロッパー	181（37%）
2. パブリッシャー	34（7%）
3. デベロッパー・パブリッシャー	249（52%）
4. その他	13（2%）

財　団　法　人　Ｊ　Ｋ　Ａ

平成 21 年度デジタルコンテンツの保護・活用に関する調査研究等補助事業

デジタルコンテンツ制作の
先端技術応用に関する調査研究
報　告　書

発　行　平成 22 年 3 月

発行者　財団法人デジタルコンテンツ協会
〒102-0082　東京都千代田区一番町 23－3
日本生命一番町ビル LB
TEL.03(3512)3900
FAX.03(3512)3908

不許複製　禁無断転載