る。

３. プログラマのスキル

3-1. スキル表各項目の説明

（1）能力

プログラム能力に関するスキル

① コンピュータ言語

プログラマにとってのコンピュータ言語は必須のスキルである。ゲームプログラムは初期にはアセンブリ言語などの低級言語で作成するのが普通であったが、近年ゲーム機の高性能化に伴い、ほとんど C 言語、C++言語などの高級言語で作成されている。しかしながら、プログラムチューニングやサブプロセッサプログラム開発におけるアセンブリ言語、携帯アプリケーション開発における JAVA 言語、開発環境を構成するスクリプト作成など、それ以外の様々な言語も状況に応じて使用されているため、C、C++に加えて複数の言語に関してスキルを持つことが望ましい。

また、言語がオブジェクト指向言語へとシフトしていくなかで、ゲームプログラムの設計段階からオブジェクト指向の考え方を取り入れるようになった。そのため、オブジェクト指向分析・設計の考え方を理解し、UML などを用いてそれを表現できるスキルが必要になる。

② アルゴリズム

アルゴリズムとは問題の解き方であり、コンピュータに対しアルゴリズムを示すのがプログラムである。既に過去のプログラム中で多くのアルゴリズムが示されてきているため、プログラマはプログラムを作成する際、まったく新しいアルゴリズムを開発するよりも過去のアルゴリズムを応用するほうが多い。また、新規アルゴリズムを必要とする場面においても、過去のアルゴリズムに関する知識を持つことで発想がより容易になる。そのため、アルゴリズムに付いてのスキルを持つことはプログラマにとって大きな強みとなる。

③ Windows プログラミング

Windows が大きなシェアを占める現在では、プログラマにとっても Windows プログラミングスキルの必要性が高くなっている。例えば Windows 環境で動作するターゲット（PC、ゲーム機）の場合そのプログラムスキルは必須である。また、開発環境が Windows 環境であれば、開発ツール作成といった開発環境構築にあたってスキルが必要になる。

Windows プログラムでは、DirectX をはじめとする WindowsAPI の利用が必要だが、低水準な API であるため、MFC を利用することも多い。Windows プログラミングスキルはこれらに関する知識と応用である。

④ ネットワークプログラミング

PC については勿論、家庭用ゲーム機にもネットワーク対応ゲームが増えている。当然開発者にもネットワークプログラムのスキルが要求される。またほとんどの場合、開発環境はネットワークを利用しているので環境構築・ツール作成の面でもネットワークプログラミングスキルは必要になっている。

サーバ･クライアントの機能やプロトコルに関する知識、インターネット・イントラネットの基礎技術などに関しての幅広い知識が含まれる。

⑤ 開発環境に関する知識と技能

開発環境はターゲットによって様々だが、いずれも基本機能は共通している部分が多い。また家庭用ゲーム機や PC で共通の統合開発環境を提供しているケースもあり、プログラミングするにあたってひとつの開発環境に精通すれば他の環境でも応用が利く。

開発環境を構成するのは、作成（エディタ）・実行形式変換（コンパイラ）・実行モニタ（デバッガ）の三機能で、統合開発環境では、これらを GUI 上でリンクさせて様々な機能を提供している。いずれの機能もプログラミングに重要な機能で、スキルを身につけることが生産性の向上につながる。

⑥ 各種フォーマットの知識

プログラムがデータを処理するものである以上、データのフォーマットに関する知識もプログラマにとって重要なスキルである。特にデータの一次加工をする開発ツール作成など、開発環境プログラミングにおいては、フォーマットに関する知識が必須のスキルである。

また、新たなフォーマットの策定を行う場合でも、既に存在するフォーマットに関する知識があれば、容易により良いフォーマットをデザインすることが出来る。

⑦ その他プログラミング手法

上記以外にも様々なプログラミングに関するスキルが存在する。

a．テスト駆動式開発（TDD）

小モジュールに対し最初にテスト・実装を繰り返し、テストにパスしてからソースの整理（リファクタリング）を行う開発手法。結合前に細かくテストを行うことで一つ一つの機能を確認でき、結合後の修正作業を減らし効率化が出来る。

b．セキュアプログラミング知識

セキュリティホールを埋め込まない安全なプログラムを作成するための知識。どのような、コードが悪用されるのか、安全性の高いコードを作成するノウハウなど。ネットワーク対応などのオープンな環境で使用されるゲームソフトが増えていく中で重要なスキルになりつつある。

c．Gnu-binutil の理解

オブジェクトファイルを扱う為のプログラミングツールで、アセンブル･リンク・オブジェクトファイルコピー・ダンプ・シンボル情報の操作など、汎用性が高い非常に強力なツール群。Linux 環境での開発では必須スキル。

d．デバッグ手法

プログラム内に埋め込まれたバグを発見し特定するスキル。ウオッチ・ブレーク・トレースなどデバッガの機能に対する理解とその応用に関する知識、過去のバグフィックス経験などによって構成される。

e．C/C++ABI の理解

コンパイラ・リンカの動作・機能に対する知識と理解。あるプログラムが、コンパイルしてもエラーが出力されないのにかかわらず、期待した動作をしないことがある。そういったケースでは、プログラマがコンパイラ･リンカの動作･機能に関して正しく理解していない。また、こういったケースでは何が間違って動作しないのかが容易に発見できないことが多い。

f．インストラクションセットの理解

ターゲット CPU の持つ命令セットを理解することはそのターゲットの動作を理解する上でも重要である。またコンパイラの動作によって引き起こされるバグ（上記ｅ．コンパイラに関する知識不足から発生するものを含む）など、C 言語のソースレベルでは発見できないバグを特定する為にも必要になるので、インストラクションセットに関する知識も有用なスキルである。

g．データ圧縮

データ容量の増加が、ターゲットハードウェアのメモリ増加を上回る速度で進むゲームプログラムにおいて、メモリを有効利用する上で、またデータ転送速度を上げる上でもデータ圧縮技術が大きなポイントとなってきた。とくに高速な伸張が可能な効率の良い圧縮技術が求められている。必ずしもプログラマ全員が身につける必要はないが、ゲーム基礎技術として必須のスキルである。

(2) 技術

プログラマとして求められる技術知識のうち、主に専門的な知識の部分の項目について以下に述べる。

プログラマの作業は細分化が進んでおり、以下の項目すべてを網羅することは必要ではなく、担当する部分の項目（たとえばグラフィックプログラマであれば、グラフィックプログラムの項目）が必要になる。また、必要とされている項目内の小項目においても、すべてが必要という訳ではない。担当するゲームの内容によって必要とされる技術は様々であるため、この小項目ではショウケース的に技術を提示する方針とした。スキル表を見るときは小項目一つ一つを見るのではなく、関連する小項目を含む中項目全体でのスキルレベルの把握を前提としている。

① マルチコアプロセッサ

近年、PC 用のプロセッサ、家庭用ゲーム機のプロセッサにおいて、マルチコア化やハイパース

レッディング対応が進んできている。マルチコアプロセッサで性能を出すためには、旧来の単一スレッドで記述していたプログラミングスタイルとは違う考え方が必要になる。プログラム面から見た場合、ほとんどはスレッドと呼ばれるテクニックを使って平行に動作するプログラムを記述する。マルチスレッドでは、同時に複数の仕事を実行するため、メモリなど共有リソースの排他制御や、同期処理、スレッド間通信など、旧来のシングルスレッドでは必要ではなかったテクニックが必須となる。

② 物理シミュレーション

主に３D ゲームにおいて、画面内のオブジェクトをより現実世界に近い挙動にさせるために、物理現象をシミュレーションする手法である。

工学分野でシミュレータとして研究されていた技術を転用していることが多いが、ゲーム特有の条件としてリアルタイム性を出すため、正確さより計算にかかる時間を少なくすることを優先する必要があり、ゲーム業界で特有の手法も出てきている。

シミュレーションの対象によって、以下のような分類が可能である。

a. 剛体運動シミュレーション

力学に乗っ取って、物体（剛体）同士の衝突や接触を取り扱う。一般的に物理シミュレーションといった場合はこれを指すことが多い。大学レベルの数学・力学の知識を必要とする。車の挙動等に用いられることが多かったが、近年ではゲームの中のオブジェクト全てに対して物理シミュレーションを行い、よりリアルな世界を再現しようとするゲームも多くなってきている。

b. 流体運動シミュレーション

液体や気体などをシミュレーションする。流体力学を基礎とし、一般に剛体シミュレーションよりも高い負荷を必要とする。

c. 弾性体（毛髪・布）シミュレーション

服や毛髪などをシミュレートする。方法としては、バネをメッシュ状に接続したものとして近似的に扱い、弾性体シミュレーションと呼ぶ。

d. 燃焼シミュレーション

炎や爆発時の煙などをシミュレートする。前述の流体運動シミュレーションで行う場合もあるが、より近似的な方法で行われることも多い。

e. 人体運動シミュレーション

人間の体の関節等をシミュレートする。手の位置から肘の関節の曲がり方を求める等、末端の位置から関節角度を決めるのはインバースキネマティクスとよばれ、様々な解法がある。最近では、攻撃や爆発で吹っ飛ばされた人体の挙動をゴム人形と近似して行うラグドールと呼ばれる手法も広く使われている。

③ エフェクト技術

ゲーム中で表示される火花や煙、破片等、映画で言うところの特殊効果に当たるものをエフェクトと呼び、専用のプログラマが割り当てられることが多い。また、２D のゲームにおいても、ゲームの見た目をよくするために、エフェクト技術が使用される。

代表的な処理に以下のようなものがある。

a. 高速パーティクル処理

火花や煙、破片等、小さな物体をパーティクルと呼び、その挙動を作成するには数が多くなりすぎて上述の物理シミュレーション法を使うと処理が重くなりすぎる。そこで、パーティクルの処理では、多少めり込むなど物理的におかしくても高速な処理が行われることが多い。どう処理を省き、それらしく見える物にするかということが必要になる。

④ AI シミュレーション

ゲームにおけるキャラクタの挙動について､それを制御する技術を AI シミュレーションと呼ぶ。たとえば、戦闘ゲームやスポーツゲームにおける、敵キャラクタや仲間キャラクタを、プレイヤーの動きに合わせてどう動かすか、などである。ゲームのおもしろさが、この AI に依っている割合は非常に大きい。AI に必要な処理としては、まず、経路探索・空間分割のような状況把握アルゴリズムが基本にあり、それを用いて、プレイヤーを攻撃したり、援護したりする行動を取ることになる。

主として NPC（Non Player Character）を制御することが多いが、それ以外の地形やゲームの流れ全体を制御することもある。

このカテゴリに必要な知識として大まかに以下のような物が考えられる。

a. 経路探索アルゴリズム

AI 処理に必要な基本的な処理のひとつ。ステージ上のある点からある点へどういう経路を取れば移動できるかを調べる。A*など、代表的なアルゴリズムがいくつかあるが、それぞれのゲームの性質によって使いわけたり改良したりすることが一般的である。

b. 空間分割アルゴリズム

ステージ上の空間を、小部屋や通路の連続として分解し、プレイヤーや他の NPC との関係を理解する処理を出来るだけ簡単にするためのアルゴリズム。BSP 法が主に用いられるが、これもゲームの性質によって、様々な方法が用いられる。

c. 戦闘アルゴリズム

プレイヤーをどう攻撃するかのアルゴリズム。ゲームの基本部分のひとつである。

d. 集団アルゴリズム

敵や味方が複数出てくる場合、どう集団として協調するかというアルゴリズム。

e. NPC 制御アルゴリズム

プレイヤー以外の敵や味方などをどう制御するかというアルゴリズム。オンラインゲーム等ではいかに違和感なく人間の操作するプレイヤーに似せるかどうかも必要になってくる。

⑤ ユーザインターフェース

昨今、パッドなどの入力デバイスやディスプレイなどの出力デバイスに手を入れることで、ゲームのおもしろさを拡張する動きが出てきている。家庭用ゲーム機においても例外ではなく、今後の技術の動向には気を配らなくてはならない。

簡単に例を挙げると、以下のような物がある。

a. 多機能コントローラ

今までの方向キー＋複数のボタンだけではなく、たとえば感圧センサーや傾きセンサー、ポインティングデバイス、タッチパネル等、様々な物が入力デバイスとして採用されている。入力だけではなく、コントローラ自体が振動したり、スピーカーがついたりと同時に出力機能が実装される場合も多くなってきた。これらに関してはそれぞれのセンサー等の理解と、各ゲーム機メーカが出してくるライブラリの実装の仕方の理解が必要になる。また、各ゲーム機に標準的に USB のコネクタがつくようになってきたため、ゲーム個別にデバイスを作ることも多々あり、その USB のドライバをプログラマ側で作成することもある。この場合は USB の仕様の理解も必要になってくる。

b. 画像認識技術

USB 等を経由してゲーム機側にカメラからの映像を取り込み、それを使うゲームも出てきている。たとえば撮った人間の動きを検出してゲームの入力として扱ったり、カードを撮ってどの種類のカードなのかを認識する技術等がある。携帯など別の業界では、顔認識で誰かなのかを判定する技術や、表情認識等の技術もあり、これらがゲームに応用される可能性も大いにあるだろう。

c. 光学技術（立体映像・ドーム映像）

主に業務用の技術として表示系を一般の TV/ディスプレイに限らず、大きなドーム型に投影したり、立体映像として出力したりするゲームも出てきている。ハードウェアだけではなく、ソフトウェアについても、それぞれについて特性を理解し、それに合わせた形で画像を構成する技術が必要になる。

d. マルチチャンネルサラウンド

DVD や HDTV の普及に伴い、家庭にも 5.1ch や 7.1ch のアンプが普及し始めている。ゲーム機にもそれに対応した出力機能がつくようになってきており、徐々に対応することが当たり前な状態になりつつある。

e. 音声認識・合成

携帯型ゲーム機において顕著であるが、音声認識を使ったゲームが徐々に増えてきている。また、料理レシピ等実用ソフトも近年ゲームとして出てきており、そこで大量のデータをゲーム内で音声としてしゃべらせるための音声合成技術も必要となってきている。この分野では家電系メーカが先行しており、そこからミドルウェアとして供給されることも多い。

⑥ システムプログラム

ファイル管理やメモリ管理、実行単位の管理等を総称してシステムプログラムと呼ぶ。ゲーム機メーカ提供のライブラリをよりそのタイトルに使いやすくするための処理を作成することが多い。また、デバッグのための機構なども含み、この部分の作りの善し悪しで開発効率も変わる、ハードウェアの深い知識と、高いノウハウが必要になる。

また、新しいデバイス等に対応するのもこの領域で行われることが多い。その場合は USB などのインターフェースの知識が必要になる。

あるいは、ゲームの流れの部分の処理や、AI 処理などを、C、C++などのプログラム言語ではなく、スクリプト言語で記述することも近年多くなってきている。この場合、それを解釈する部分をやはりシステムプログラマが作成することになる。

a. DVD,BD 等ディスクメディア操作

家庭用ゲーム機では、その媒体に DVD、BD といったディスクメディアが使用される。PC などでよく使われる HDD などとは違い、シークが遅く、リードエラー処理が必要になるなど、気をつけなければならない点は多い。シークを出来るだけ少なくし、高速にファイルを読み込むために、読み込み順にファイルを並べて 1 ファイルにパッキング化したり、圧縮したりするなどのテクニックが必要になる。

また、近年では BGM などはすべてゲーム中にディスクから逐次少しずつデータを読み込んでいくストリーミング再生を行うことが多くなってきており、さらにそれと平行して同時にデータを読み込む処理を行うマルチストリーム処理などが必要になってきている。

b. USB 等デバイス操作

家庭用ゲーム機では、ユーザのデータが保存されるメモリーカード等の外部保存デバイスの扱いに関して、ゲーム機メーカのチェックが厳しく、毎回気をつけなければならないポイントになる。このあたりは各社のノウハウとしての蓄積が必要になる。また、それ以外でも、USB を使った独自デバイスを作成することもあり、その場合は USB 規格の知識が必要になる。

c. メモリ管理技術

通常のメモリ管理関数はメーカ製ライブラリとして提供されるが、速度面やデバッグ効率の面で問題があり、独自にメモリ管理関数を書くことも多い。確保するメモリの粒度によって管理方法を変え、速度を改善したり、どのオブジェクトが確保したメモリなのかを記録するようしたりしてデバッグ効率を上げることがある。

d. 組み込み用スクリプト言語

ゲームが大型化するにつれ、ゲームの流れや敵などの配置処理の部分をインタプリタ型のスクリプト言語で記述することも多く行われるようになってきた。トライアンドエラーのサイクルを短くし、デバッグの手間を少なくすることが出来る。独自の言語を作成する場合もあれば、既存のオープンソースな言語を用いることもある。後者の場合は、lua や lisp、Javascript 等が考えられる。

⑦ グラフィックプログラム

ゲームの見栄えの部分を司る、非常に重要なカテゴリである。近年、ハードウェアの進歩に伴って、シェーダと呼ばれるグラフィックスプロセッサ用のプログラミング言語が多く用いられるようになってきた。表現できることに自由度が増え、いろいろなことが出来るようになった反面、速度面での見積もりが難しくなってきており、リアルタイムが必須とされるゲームのプログラムでは見栄えと速度のバランスがもっとも大きな問題になってきている。

a. シェーダを用いた CG 表現手法

頂点シェーダ(vertex shader)、ピクセルシェーダ（pixel shader。フラグメントシェーダ fragment shader とも呼ばれる）を使って、頂点ごと、ピクセルごとに演算処理を行い、最終的にそのピクセルの色を設定する手法が一般的になってきている。自由度が高い反面、いろいろ処理を入れるとすぐに処理速度がかかるようになってしまう。すでにいろいろな CG 手法が開発されてきており、それらを参考に独自のシェーダを作成していくことが多い。

b. HLSL,Cg 等でのシェーダプログラミング

シェーダをより高レベルな言語で書けるように、C ライクな構文の、HLSL(High Level Shader Language) ,Cg といった言語が登場し、現在ではほとんどこれらを使って開発している。

c. グラフィック描画パスの構築

最近の CG 手法では、単純に順番にモデルを描画するだけではなく、影を書くために光源方向から見たモデルを別に描画したり、高速化のために手前から書いたり、半透明のために奥から書いたり、それぞれ違う処理を同じモデル群に対して順番に行って行く必要がある。これらひとつひとつの処理を描画パスと呼ぶ。描画プログラム中で、このパスをどういう順番でどう実行していくかが、問題になることが多い。

d. ターゲットゲーム機の処理特性の理解

家庭用ゲーム機ではそのグラフィックチップによって、全く特性が異なり、あるゲーム機で速度を出すための手法が、違うゲーム機では全く役に立たないこともよくある。それぞれのゲーム機の特性をよく知らないと最適化は出来ない。

e. オブジェクトカリング手法

出来るだけ高速に描画するために、少しでも描くオブジェクトの量を減らす必要がある。その減らす処理をカリングと呼ぶ。基本的なのは、カメラから見えない物は表示しないという視錐体カリングだが、それ以外にも Z バッファを使う場合の Z カリングや、その場所から見えない物をカリングする PVS カリングなど、様々なカリング手法がある。

f. シーングラフ手法

各オブジェクト間を管理するデータ構造をシーングラフと呼び、それをどう管理するかということが重要になってくる。いくつかの代表的な構造を元に各タイトルに合わせてカスタマイズしていくことが多い。前述したグラフィックの描画パスやカリング手法とも絡み、どう効率よく描画、カリングできるかがこのデータ構造に関わってくる。

g. アニメーション（モーション）処理

物体をデザインツールで指定したとおりに動かすことをアニメーション（モーション）処理と呼び、主に人体の場合に、各間接の角度を時間軸で変更していく処理として実装される。単純にデータを再生するだけではなく、よりなめらかに見せるために 2 つ以上のモーションを補完したり、上半身と下半身など、一部分だけ別のモーションを再生したりといろいろな処理が行われる。また、目的の位置に末端を持って行くための関節角度を計算する IK（Inverse Kinematics）等と組み合わされることも多い。

⑧ オンラインゲームクライアント技術

オンラインゲームのクライアントにおいては、スタンドアローンのゲームとは違ったノウハウが必要になってくる。本当にそのサービスの会員であるかを認証する必要があること、ネットワークのレイテンシ問題、WAN につないでいることによるチートやアタック対策などである。

a. ネットワークのレイテンシ、スループット制御

ネットワークの伝送速度は、現実には 10～100ms のオーダーであり、これは、60 フレームでのゲームでは数フレーム程度の遅れを意味する。これは相手の入力とそれに伴う挙動がそれだけ遅く伝わると言うことであり、完全に同期したゲームは作れないということである。主に操作情報そのものを送るのではなく、各オブジェクトの位置や動きを送ったり、操作によって起こる挙動イベントを別に送ったりする手法が用いられる。また、一度に送れるデータ量に関しても制限があり、それをどううまく制御するかがノウハウになる。

b. 認証技術

ゲームのユーザ ID の管理として、サーバ側と協調して行うことが多い。あるいは、サーバがなりすまされていないかどうかを確認するためにも必要となる。

c. チートアタック制御

クライアントのプログラムやデータそのものを書き換えられたりすると、それ以外の一般のユーザに対して圧倒的な優位でゲームを進めることができてしまい、ゲームの構造そのものがなりたたなくなることもある。また、サーバやクライアントに対して DoS などのアタックがかけられることも多々あるので、防御機構は必須になる。

⑨ モバイルデバイスプログラム

近年の携帯電話のゲーム環境の拡大は著しく、一昔前の家庭用ゲーム機並みの画面が出るようになってきている。開発環境は Java や Brew といった独特の環境が使われ、また端末ごとの性能差をどう埋めていくかなど、家庭用ゲーム機とは全く違ったノウハウが必要になる。

a. Java, Brew プログラミング

携帯電話では Java が主に使われ、一部で Brew が使われている。いずれにしても使用できるメモリ量が限られているため、通常のプログラムと異なり、独特の工夫が求められる。

b. 各種クラスライブラリの知識

言語は同じ Java であるが、各キャリアから微妙に異なるクラスライブラリが提供され、そ

の対応が必要になる。ほとんどの場合、主なキャリアすべてに対応する必要が出てくるため移植性に気を遣ったプログラムを書かなければならない。

⑩ サーバプログラム

オンラインゲームのサーバプログラムでは、ゲーム本体のプログラミングと全く違うノウハウが必要になる。ゲームのタイプにより若干は異なるが、ある程度は Web サイトやショッピングサイト等に共通に必要な知識がある。以下にその主な領域をあげる。

a. ネットワークハードウェアの理解
   サーバ側でどのようなハードウェア構成を取るかということ、まず一番重要な検討点となることが多く、そのあたりのハードウェア知識は必須となる。主には負荷分散のために冗長な構成を取り、それをコントロールする機器が必要となる。
b. スケーラブルなゲームサーバ、Web サーバ等の理解
   オンラインゲームのサーバは、ゲームサーバ複数と Web サーバ等で構成されることが多い。耐障害性や、ユーザの増加に対応するために、簡単に数を変更できる冗長構成をとらなければならない。
c. サーバネットワークプログラミング
   ゲームサーバは、C や Java,Perl 等で構築され、TCP や UDP を使ったネットワークアプリケーションとして構築される。当然、多人数をさばくためにマルチスレッドやマルチプロセスといった手法を使い、出来るだけ軽く多数のデータをさばいていかなければならない。
d. データベースプログラミング、（SQL 言語）
   オンラインゲームの肝となる部分はユーザごとのデータを扱い、それを検索・更新する部分である。これには一般的に RDB（リレーショナルデータベース）が使われ、SQL 言語によって操作される。データベースの操作には特有のノウハウが必要になる。
e. Perl、PHP 等、CGI 言語プログラミング
   ゲームサーバ以外にも、Web 系の知識は必要になる。ランキング情報やログイン処理、アカウント管理などは、Web サーバが使われることが多く、それは CGI 処理として実装される。一般的な CGI 言語の知識が必要とされる。

⑪ サウンドプログラミング

サウンドプログラミングは、ゲームプログラムとは別のチームとして扱われることが多い。作曲者などのサウンドアーティストと同じ部署で、どのゲームでも共通して使われるように作られることが普通である。BGM には昔は Midi シーケンス等が用いられていたが、近年ではディスクからストリーミング処理でならすことが主流になってきている。また、5.1ch などのマルチチャンネルサラウンドも普及し始めているため、その対応も必要になることがある。

a. 音響工学

近年 CPU の処理能力が上がってきたため、専用のサウンドハードウェアを使わず、すべて CPU で波形処理することが多くなってきている。リバーブやピッチ変換といった基本的な処理も、CPU で行うため、自由度が上がっている反面、作業量が増えてしまっている。

b. ３D 音響モデリング

３D ゲームで、さらに臨場感を持った音を出すため、単純にパンとボリュームだけではなく、壁の反射や材質等も考慮した音響モデリングを行うことも始まってきている。マルチチャンネルサラウンドと組み合わせて、よりリアルな音を目指す。

c. 音声圧縮フォーマット

BGM や音声をディスクからストリームできるようになったが、無圧縮の音声だと大きすぎるので、何らかの圧縮をかけることが多い。ADPCM 等が多いが、近年では MP3 や Ogg Vorbis、ATRAC のような処理は重いが高圧縮の圧縮形式も用いられるようになってきた。それぞれの特性を理解して使うことが重要である。

⑫ 開発環境制作

直接ゲームのプログラムには含まれないが、ゲームを開発する環境としてのツール群を作る必要はある。特に大きなチームになると日々のルーチンワークをいかに効率よくこなす環境を構築するかで、チーム全体の作業効率向上に大きく寄与することが出来る。主としてデザイナが使うモデル・アニメーションなどのプレビューツールや、ゲームデザイナが使うイベント設定ツールなどが作られることが多い。そのほか、３D ツールが出力するデータを実機用のデータにコンバートするツールや、膨大な作成データを管理するためのツール、開発チームの情報共有用の Web ページ、掲示板の制作など、その業務は多岐にわたる。そのため、その担当者も広い知識が求められることが多い。

a. Windows プログラミング

デザイナやゲームデザイナが使うツールは主として Windows アプリケーションとして作成されることが多く、Windows 上でのプログラムノウハウが必要となる。

b. CGI プログラミング

昨今では、チームの情報共有に Web がよく使われ、簡単な CGI を使って掲示板等を作ることも多い。

c. PHP,Perl 等を使った Web アプリケーション

CGI プログラミングを一歩進めて、たとえばデータ管理や進捗管理などで、PHP,Perl を使った本格的な Web アプリケーションを作ることもある。

d. Perl、Ruby 等 LightWeight 言語でのツール制作

テキストコンバータや一連の作業の自動化などで、perl や ruby といった言語を使って簡単なツールを作ることは多い。一連のルーチンワークをこういう言語を使って簡単にツール化すれば、かなりの時間の短縮が期待できる。

e. Excel 等の Office アプリケーション

ゲーム内の各種パラメータや配置データなど、数値情報は Excel を使って管理されるケースも多く見られる。マクロ等を駆使して、ひとつのアプリケーションのようなデータシートを仕上げているケースもある。

f. 外部のツール、フリーソフトウエアの知識、情報

開発環境制作においては、外販するわけではないので、外部のツールやフリーソフト等を使ってより効率よく優れた物を作ることを求められる。ネットから幅広く情報を集めて、それを生かすことが求められる。

g. 一般的な画像形式、データフォーマットの理解

テクスチャやモデルデータ等、一般的な形式でオーサリングツールから出力されたデータを、独自形式のデータに変換することは良く行われる。よく使われるフォーマットについては、どうすればプログラムに取り込めるのかを知っておく必要がある。

h. XML などによるデータ交換手法

昨今では、XML によってデータが出力されることが多くなってきている。独自フォーマットもバイナリ形式でなく、XML 等を使うことで、相互運用性や、未来に渡って再利用しやすくなることが期待できる。

i. XSI,Maya,3DStuidioMax 等の３D ツール知識

デザイナがモデルやアニメーションを作る環境である、３D ツールに関する知識は必要である。ツールによっては、スクリプト等を使って一連の作業を自動化できる物もあり、これらスクリプトをうまく整備することで開発効率を上げることが出来る。

（3）知識

プログラムに関連する知識に関するスキル

① OS・基本ソフトウェアの学習

プログラミングがＰＣ上でを行われ、入力・コンパイル・デバッグをアプリケーションによって行う以上、プログラマにとって OS・基本ソフトウェアに関する知識は前提である。

一般に開発環境には Windows 環境と Unix/Linux 環境があり、ターゲットに応じて、また処理内容に応じて使い分けるので、それぞれについて最低限の知識が必要とされる。

② コンピュータグラフィックス

プログラマが直接グラフィックを作成することはないが、グラフィックデザイナの作成した素材をターゲット上で処理するために、グラフィックに関するスキルが必要とされる。ハードウェアの進化とともに必要とされるレベルが上がり続けていて、単純なフレームバッファの仕組みから質感などの高度な３Ｄ表現を可能にする為の処理まで様々な段階がある。シェーダプログラムなどを作成する為には、当然高レベルのスキルが必要とされる。

③ ゲーム物理

ゲーム内で物理現象をシミュレートしたり、より高度な画像表現を実現する為に、数学や物理に関する知識が必要とされる。かつては単純なものしか扱えなかったが、ゲーム機の高性能化によって、本格的なシミュレートを行うことが可能になるなどスキルの幅がどんどん広がっている。また、ゲームでは良く採られる簡易モデル化（処理的に正確なシミュレートが不可能なため、近い結果が得られるような簡素化したモデルを考案する）といった場合でも、当然対象に対する知識が必要とされる。

④　ゲームハードウェア

ハードウェアに関する知識は、担当業務に応じて必要なレベルが異なる。内部処理プログラムを作成する為には最低レベルでも可能であるが、デバイスプログラムとなれば当然高いレベルが要求される。理想はデバイスプログラムやライブラリレベルでハードウェアの相違を完全に吸収してしまうことである。デバイス・ライブラリ部分を含め、ゲームプログラムにおいてはしばしばハードウェアの特性を考慮したプログラムが要求されるので、重要性の高いスキルのひとつといえる。

⑤　データベースプログラミング

データベースに関するスキルは特にサーバプログラマに必要とされる。様々なデータを集積し、時々刻々移りかわる状態を反映し外部の要求に応じて送信するデータベースの処理はゲーム上でも必要になる。多人数ネットワークゲームのデータなどはその典型といえる。ネットワークに関する知識・情報セキュリティに関する知識と密接に関係しているので、同時のそれらのスキルも身についているのが望ましい。

⑥　情報セキュリティ

ネット環境での開発が当たり前になった現在では、環境構築や運用に当ってセキュリティの基礎的な考え方が必要とされる。また、サーバプログラム・ネットワークゲームプログラムなどでは情報セキュリティスキルはより高いレベルが要求される。サーバプログラマには特に必要なスキルである。

⑦　ネットワーク基礎

現在では開発環境がネットワーク上にあるのが前提であるため、ネットワークに関する基礎的な知識は環境構築・運用に必須である。小規模なＬＡＮを構成したり、簡単なネットワークエラーは自力で解決できることが望ましい。また、ネットワークに対応したゲームの場合ではさらに重要性は高くなる。

⑧ 基礎知識

プログラマに必要とされる基礎知識としては，代数幾何（ベクトル・行列、2 次元・3 次元空間），微分積分，確率統計などの数学，計算論，アルゴリズム、言語、アーキテクチャなどの計算機科学，視覚，聴覚，触覚についての心理・生理学，計測・制御学，があげられる。これらの基礎知識は，個別の開発業務に対応するものではないが，複数のスキル・知識にまたがって必要となる前提知識を含んでいる。これらを一から学ぶことは時間的にも業務への効果を考えても効率的ではないが，各スキルの習得に難しいと感じたときには，これらの基礎知識に関する本などを調べ，解決できることが望ましい．たとえば，スキル表の各スキルに対応する知識を挙げると次のようになる。

アルゴリズム：計算機科学(計算論・アルゴリズム・言語・アーキテクチャ)

ゲーム物理：数学（代数幾何，微分積分，計算幾何学），力学，ロボット工学，制御学

エフェクト技術：数学（微分積分）

AI シミュレーション：数学（確率統計，代数幾何），制御学，計算機科学（計算論・アルゴリズム・言語）

ユーザインターフェース　：数学（代数幾何，微分積分），力学，計測・制御学，心理・生理学)

グラフィックプログラム： 3 次元空間，線形代数，微分積分，計算幾何学，計算機科学

計算機科学はアルゴリズムに対して処理時間の見当をつけたり，高速化の見当をつけたりする際に必要な知識であり，実際のアルゴリズムを開発しながら習得するとよい．

心理・生理学は，心・体についての知識である。たとえば，人間の反応速度や，視力の限界など，プログラムの性能への要求について，見通しを立てるために役立つ。また，ゲーム機に用いられるユーザインターフェースの進化に伴い，人間側の基礎知識として，心理・生理学の重要性が増している．また，計測側の基礎知識としては，各種センサ(加速度・光・音・画像処理など)や入力情報の処理といった用途から計測・制御学の重要性も増している．

3-2. キャリアパスの現状

入社後、開発者各自が自らのキャリアパスを認識し、業務に従事しているかを今回実施の『ゲーム産業における経営戦略と人材マネジメントに関する総合調査（以下アンケート調査結果と称す)』より考査した。(Q10)『将来的なキャリアパスに関する情報』の提示は 6.7%と、ほとんどの企業で実施されていない状況にあるように見受けられ、開発者に対するキャリア・パスが作られていない状況にあると思われる。またそれを裏付ける他の要因として、(Q12)『評価基準を開発者に公開している』に対して『どちらともいえない』～『公開していない』の占める割合が（62.1%）半数以上の企業であり、この状況からも開発者に対する評価目標を認識させることが困難な状況であると思われる。

また、上記以外の要素として、ゲーム業界はまだ若い産業であり、社員平均年齢も 30 代前半で

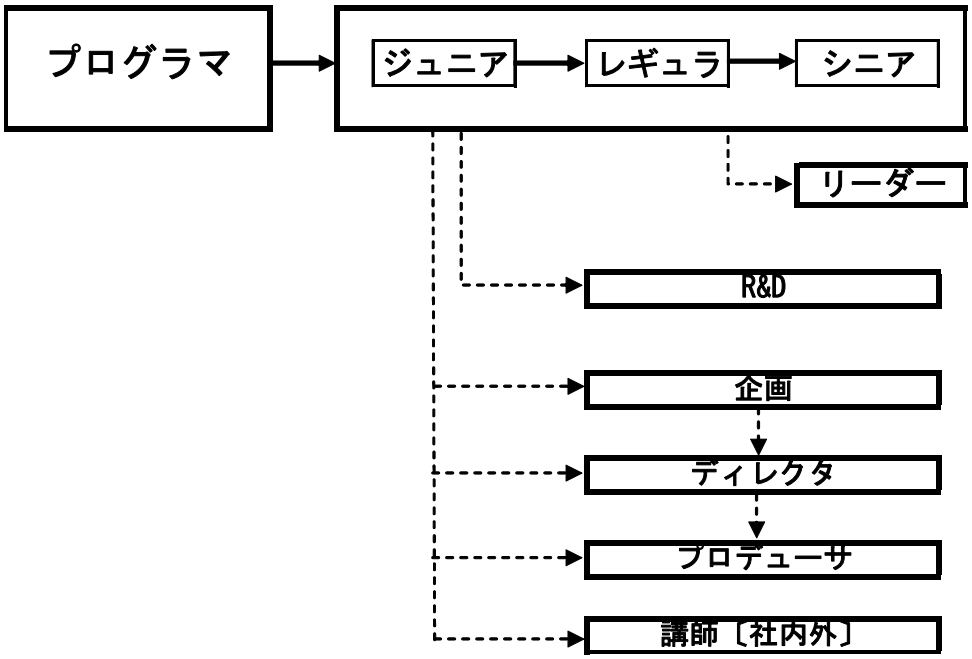
あり、キャリアパスの実例を開発者が見るケースが少なく、中々認識しづらい状況にあると推測される。

ただし、現状のキャリアパスとしては、以下記載の通りのパターンがあると考えられる。

キャリアの積み方として、以下のようなケースが考えられる。

- ・ある特定分野の専門家として、ジュニアクラスからシニアクラスにスキルアップすること（例えば物理シミュレーションの専門家という意味)。
- ・ある特定分野の専門家であってもシニアクラスになってくるとチーム運営の一端を担うケースもあり、専門分野のセクションのリーダーとしてのマネジメントスキルが期待される
- ・プログラム全般の専門家として、幅広い能力発揮をする。チーム全体のシステムの方向性を決める
- ・他分野におけるプログラマ（サーバプログラマから開発環境制作プログラマ）への転換
- ・プログラマからゲームデザイナ（プランナ）への職種転換し、ゲーム企画をする
- ・チーム全体のディレクタとして、プロジェクトを制作面から引っ張る
- ・プロデューサとして、総合的にチームを仕切る
- ・経験・知識を生かし、プログラム技術研究を行う（例えば、最新ゲーム機向けに共通技術の開発・研究）
- ・経験・知識を生かし、学校の講師等

【ゲームプログラマキャリアパス例】



＊各プログラマの職種間の異動も当然キャリアパスとして行われる

＊実線は標準的なキャリアパスであり、破線はまれなケースでのキャリアパスである。

3-3. 在職者の教育カリキュラムの現状

アンケート調査結果（Q6）より、現状の育成活動において『OJT』中心（96.6%）の指導がほとんどであり、Off-JT（44.8%）、キャリア開発支援（34.5%）を実施している企業は半数を大きく割る状況であった。しかし今回のアンケートの効果とも考えられるが、今後の育成活動に関しては、Off-JT（79.3%）、キャリア開発支援（75.9%）と 8 割近くの企業にて『OJT』以外の育成活動も積極的に実施いていく考えを表明している。

これはアンケート調査結果（Q5）の開発者（プログラマ）の活用に対する企業の考え方として、『ゲーム開発には、組織に蓄積したノウハウが重要であるため、当該人材を長期間雇用する』との回答企業が圧倒的に主流を占めていることからも納得のできる結果であると考える。

しかしながら、別のアンケート調査結果（Q10）では、『社内の教育訓練、キャリア開発に関する情報』、『社外の教育訓練、キャリア開発に関する情報』の提供が今後においても全体の５割弱の企業であり、上記育成活動の改善伸び率と比較すると、まだまだ具体的に考えている状況であるとは言いがたい状況にあると推測する。

社外での『開発者の Off-JT』の手法としては、『CEDEC』（57.1%）が活用状況としては最も高いが、それ以外の国内・海外での研修機関・研究会の活用はほとんどされていない状況であり、なかなか外部研修の活用もうまく出来ていないのではと推測する。

3-4. プログラマを育成するためには

プログラマの仕事は、企画が作成したゲーム仕様をもとに、グラフィックデザイナが制作した映像素材やサウンドデザイナが制作した音素材を組み合わせて、ゲーム機上でインタラクティブに動くゲームとしてまとめていくのが仕事であるが、ゲーム仕様を具体的な設計仕様に落とし込み、次に設計仕様にもとづいたプログラミングを行い、最後にデバッグをしてゲームの完成までの一連の業務をこなすのが役目である。そのような業務のために、「3Dシェーダ」、「共有ライブラリ」、「サーバプログラム」、「アプリケーションプログラム」、「ビジュアル作業支援ツール」、「サウンド支援ツール」、「データ作成用ツール」、「データベース」、「開発環境の整備（ネットワーク）」などの関連プログラムの開発も行うため、その仕事内容は非常に多岐にわたっている。

また、企画からあがってきたゲーム仕様に従ってプログラミングするだけでなく、ゲームユーザの立場で遊びやすさや使いやすさを考えてゲーム開発をすることが要求される。そのため、プログラマの資質として、開発業務をスムースに進めるコミュニケーション能力、ゲームユーザのニーズを捉える感性や柔軟な頭、より良い製品にしていくプロ意識が求められている。

以上の多岐にわたる仕事を立派に果たすプログラマを育成する方法は、単純な方法ではなく種々の方法の組み合わせが考えられるが、ここでは、2つの視点（個人育成または組織育成、企業内育成または業界内育成）からプログラマを育成する方法を検討した。次表には、検討結果のプログラマ育成マップを示す。

プログラマ育成マップ

	企業内育成	業界内育成
個人育成	① キャリアパスの明確化： 職務についてどのような技術をマスターすればよいのか明示することが重要であり、さらに、上位の仕事内容を見えるようにすることも必要である。 ② 評価育成の導入： 業績連動型賞与や発明考案の実績褒賞制度を導入し、技術者のモチベーションをアップさせることが積極的に行われつつあるが、技術能力の育成として、ゲーム技術の社内論文制度や社内コンペティションを導入し、ゲーム技術者が喜んで自らの技術を開示し、お互いに技術を切磋琢磨する環境作りが考えられる。また、開示された技術を評価し人事制度に反映させるで、モチベーションを高めることも方法のひとつとして考えられる。	① キャリアパスの明確化： 企業の垣根を越えて、同一職種の技術者がどのような能力を有するのかの基準作りを行い、また、資格制度を検討することも考えられる。 ② 評価育成の導入： 従来、ゲームの社会的評価は、作品としての評価であるため、プロデューサやグラフィックデザイナなどが主な評価対象であり、プログラマが直接評価されることがなかった。ゲーム作品の裏方としてどのような新しい技術やアルゴリズムを考案したのかを競う国内や国際的コンペティションを導入することにより、ゲーム技術者が喜んで自らの技術を開示し、お互いに技術を切磋琢磨する環境作りが図られる。技術者が応募するに当たっては技術開示を条件にすれば、個人の育成ばかりでなく、業界全体の技術者育成にもつながる。
組織育成	① 技術情報の共有と伝承： 従来のゲーム制作工程においては、開発した技術情報の共有、伝承にあまり時間をかける余裕がなかった歴史がある。しかし、ゲーム開発期間の長期化と開発費用のアップを考えると、同じ失敗を繰り返さない効率的な開発体制を築く必要がある。そのためには、ゲーム制作工程の中に業務として、技術データやドキュメントを整理させ、共有と伝承する仕組みの確立が考えられる。	① 技術情報の共有と伝承： CESA のような業界団体において、技術情報の共有と伝承の仕組みを構築する。 ・ ハンドブックや技術便覧を業界と大学研究所で共同整備 ・ 各企業からの技術者と大学研究者による特定プロジェクト（最新シェーダ、ゲームエンジン、物理エンジンの共同開発など）の共同推進。 ・ 若手技術者のための交流の場（手軽にミーティングができ、必要な技術図書の閲覧もできる場）

	② 社内技術者による技術者教育： 日進月歩の発展を遂げる先端ゲーム技術を目指す組織を充実することは重要であるが、それだけでなく、技術を底辺からアップすることにより組織としての技術能力アップすることが重要である。そのためには、OJT だけでなく、先端技術内容を咀嚼できる優秀な技術者による社内技術者に対する教育が有効である。さらに、指導役の技術者に対しては人事上の評価があるなど、モチベーションアップも期待できる。	② 業界、大学研究所関係者による技術者教育： 業界の指導的な技術者や大学研究者による各技術分野の基礎から先端までを開設する定期的な技術シンポジウムの開催 （例：IGDA主催ゲーム開発者セミナー、SIG-GTやACM SIGGRAPH Course、LA SIGGRAPHなど）