2. 新卒プログラマ向けスキル

本項では、ゲーム企業への就職を希望する学生、もしくは学生を養成している教育機関に向けて、新卒プログラマに求められるスキルをスキル表に基づいて具体的に説明します。

2-1.　新卒プログラマ向けスキル表

(1)　重要度について

スキル重要度は１～５までの５段階で構成され、数字が大きいスキルほど入社時に重視されることになります。重要度の各段階は以下のようにわけられます。

重要度１：参考として知っているとよいスキル
重要度２：将来的に身に着けていくべきスキル
重要度３：習得しておくことが望ましいスキル
重要度４：優先的に習得すべきスキル
重要度５：すべての基礎となる最重要スキル

また、表の一番左に位置する項目が大項目となり、その中での具体的な能力が中項目やスキル項目になります。さらにプログラミング言語に関して、小項目という形で言語ごとにその重要性をまとめてあるので参考にしてください。

(2)　スキル表に関する注意点

スキル表に関する注意として、ゲーム企業へのアンケート結果を集計し、その分析をもとに作成している点です。メーカーによっては重視する点に差異がある可能性もあります。さらに注意すべきは、重要度が低いからといって全くその項目に関して知識・技術が必要ないということではない点です。表の重要度は新卒採用時点での重要度であり、いずれのスキルも入社後には必須となる項目や、入社後の職務の幅を広げるための重要なスキルになります。

最後になりますが、入社後は各々の得意な分野を確立しエキスパートとして成長していくことがゲーム企業のキャリアパスとして含まれています。このため重要度の高低に関わらず自己の得意分野を伸ばしていくことも忘れてはなりません。

(3)　新卒ゲームプログラマ向けスキル表

重要度は５段階評価で、５が最も重要なスキルです。

大項目	中項目	小項目	重要度
個人のセンス	プログラムセンス	読みやすいコード、ソフトウェアアーキテクチャ	5
	解析能力	他者制作のプログラムを解析できるか?	3
	アイデア面のセンス	ゲームの状態遷移、ゲームバランス設計	3
	画面設計のセンス	グラフィックス、UI デザイン	1
全般	グループワーク	集団への適応、経験の有無	5
	ゲーム制作	ゲーム制作の経験と総合力と得意分野	3
基礎技術面	計算	数学要素(ベクトル、行列など）、物理要素（速度、加速度、力学）	5
	言語理解	C	5
		C++	5
		JAVA	3
		アセンブラ	2
	データ管理とアルゴリズム	データ構造、基本アルゴリズム、オブジェクト指向	4
オプション技術面	システム	メモリ管理、ファイル、プロセス管理、割り込み、スレッド	3
	AI	経路探索、戦闘アルゴリズム、群集アルゴリズム	2
	ネットワーク	クライアント、サーバ、データベース	2
	グラフィックス	2D（スプライト、画像フォーマット、エフェクト）	2
		3D 表示（ライティング、シェーダ）	2
		3D アニメーション（キーフレーム、モーションブレンディング）	2
	ハードウェア技術	デジタル電子回路や機械工学	1

※オプション技術面は新卒採用後、担当業務に応じて必要になる重要なスキルです。

2-2.　スキル項目及び習熟度に関する説明

以下にコンピュータゲームプログラマを目指す人の習得すべきスキル項目を挙げます。プログラマにおいて技術に関しては基礎技術面を可能な限り充実させることが最優先課題となるため、各項目に対しての習熟度を示すことは控えます。基礎技術の習得を優先し、オプション技術面はその後にとりかかるのがよいでしょう。

「新卒プログラマ向けスキル表」には、これからゲーム制作においてのプログラマを目指す方に向けて共通して必要となるスキルとその重要度を目安として提示します。また、スキル項目の詳細について以下に記述します。

(1)　個人のセンス

「センス」とは微妙さを感じ取る能力や感覚を指します。決して一部の特殊な人々だけがもっているものではありません。

①　プログラムセンス

この言葉を広い意味で捉えると、それは非常に重要でありながら、それ単体として磨くのはとても難しいものです。それはプログラマ個々人のプログラムに対する知識や経験、情熱さえもが相まって最終的にセンスという形となるのですから、以下に続くスキル項目をすべて包含した結果ともいえるでしょう。

そのため私たちはこの項目を最重要であると位置づけ、リストの筆頭に置いています。あくまでもこの言葉のもつ意味を全体像として捉え、言葉自体の響きにあまり拘らないでください。

コンピュータゲームに限らず、ソフトウェアの論理的構成はアーキテクチャと呼ばれます。物理的な建築物や建築様式、構造を指すものとして古くから使われてきたこの言葉は、ソフトウェアも一種の建築物とみなすことができることを示しています。

砂場の小さな砂山は建築学の知識などがなくても作ることができますが、人の住む家や超高層ビルを建てる際には、設計者と施工者それぞれの様々な知識と経験がなくては成立しないことは明白でしょう。「自然に崩壊しない」という基本的なところから、「住みやすいか」「修理しやすいか」など、考えなくてはならないことは多くなります。

同様にコンピュータプログラムでも、「目的通り動作する」ことを大前提として、どのように「使いやすく」「高速に」「再利用可能に」「メンテナンスを容易に」などの複数の要件を満たし、場合によっては一部を放棄し、全体の構成を決定しなくてはなりません。そのために最重要となるのは、制作者の知識と経験に裏打ちされた、全体を見渡し最適を見出すことのできるセンスに他なりません。

プログラムの書き方次第では可読性が低く流れが理解できないプログラムや、ちょっとした修正によってバグが発生するプログラムになってしまうことがあります。現在のプログラム開発は、規模が拡大してプログラマが増加する傾向が一部にはあるため、難解で高速なプログラムではなく、すっきりしていて可読性が高くメンテナンスしやすい高速なプログラムが求められています。読みやすく理解しやすい、メンテナンスの容易な、いわゆる「綺麗な」プログラムを作成する技術もとても重要となります。

１つのサブルーチンと大規模ソフトウェアの全体構成とでは、考慮しなくてはならない事柄に違いがあることはもちろんです。しかし規模の差こそあれ、その根底を流れる思考法は同一であ

ると考えてください。
意識をもってプログラム経験を重ねることにより、確実にセンスは身に付いていきます。

② 解析能力

解析とは細かくデータを分析し、理論で究明していくことです。

他の人の考え方で書かれたプログラムの内容・意図を読み取る能力は、代表的な解析能力といえるでしょう。自分のプログラム技術を高める早道でもありますし、集団開発では必要性も高いので、ぜひ身につけてほしい能力です。しかし当然ながら、前提としてプログラムや言語に関するある程度の知識が必要となります。

そして他人のプログラムを理解する能力としてだけでなく、バグなど様々な事象について結果から原因を推定するといった、より幅広い問題解決能力として発展させていくことが重要となります。これもある種のセンスが必要ですが、前項と同じく経験がセンスに結びついていきます。

③ アイデア面のセンス

プログラマは実際にプログラムを書く時に、自分の様々なアイデアを盛り込むことが可能です。現実問題として実装に対する指示がそれほど細かくなされない場合などに、自分で細部を補うためにもアイデアは必要になります。プログラムが目的通り動作するのは大前提ですが、自分なりのプラスアルファを付け加えることができれば、作成したプログラムの価値は更に上がるでしょう。

またこのアイデア面のセンスは、他の担当者が思いつかなかった新たな楽しさに繋がっていくことも多々あります。

プログラマというと理論だけですべてを構築していくイメージがありますが、決してそうではありません。理論はもちろん最重要ですが、度を過ぎて頭が固くなってしまってはよいプログラムも書けなくなってしまいます。

ゲーム作りに携わる者にとって、「アイデア」はいつも頭の片隅に置いておきたい言葉です。

④ 画面設計のセンス

画面設計に関しては、大概の場合グラフィックス担当が存在します。そのためプログラマは普段からあまり意識する要素ではありませんが、特に動きや画面遷移などプログラマの実装に大きく依存する項目もあるのも事実です。

また、小規模なゲーム開発やツール開発では、しばしば画面設計の細部がプログラマに委ねられることがあります。優れたプログラマはしばしば優れたユーザインターフェイスをデザインすることもまた事実です。

前項と同じように、「スムースさ」や「ストレスのなさ」にはいつも意識を向けておいてください。

(2) 全般

ここでは技術的切り口とは違った角度から、ゲームプログラマに必要なスキルについて述べます。

① グループワーク

プログラムセンスの項目でも触れましたが、コンピュータゲームの制作スタイルは、一方では更なる大規模化が進んでいるため、集団作業への適応性・経験の有無が非常に重要な能力になってきています。相手の意見をしっかりと理解できる能力、無用な摩擦を生まずに自分の意見が主張できる能力なども必要とされています。

また一方ではごく少人数構成で短期間に完成させるなど小規模な制作スタイルも存在します。しかし制作スタイルに依らず、自分と他者という存在の把握と理解は必要不可欠です。

二人の人間の共同作業であっても、１つの物事に対する意見が違うことを認識するところから始まり、その違いの元となるかもしれない目標の違いから解決方法の選択に至るまで、同一の目的の中でさえも自分と他者がいかに違う存在なのかを学ぶことができます。集団になればさらに複雑さを増すその中で、自分がどのようにそこに適応していけばいいのかを模索しなくてはなりません。

ゲームプログラマに限ったことではありませんが、他者と正面から向き合って目を逸らさず、他の人を理解し、時には説得しながら複数の人間で１つの目的を成し遂げるという経験は、経験者のその後に大きな力となって残るはずです。

② ゲーム制作

「おもしろさ」というあいまいな目標をもつゲーム制作は通常のソフトウェア開発に比べて特殊な作り方になる傾向もあります。おもしろさを模索する中で、ゲームの内容や実装方法などについて記述された仕様書の変更が追いつかないこともありますし、時としては仕様書が存在しないこともあります。こうした特殊な環境において、スケジュールを守り品質を高めるためには、目標を見失わず、追及すべきところと切り捨てるところをバランスよく判断することが重要です。ゲームの制作経験はそういった机上の学習では身につけることのできない部分を習得する絶好の機会となり得ます。

上記のように、ゲームの制作はプログラマにとってより具体的・実践的なテーマとなり得ますが、ただ表層的にのみ「ゲーム」を作ることにはあまり意味がありません。次の項目で述べる数学的な基礎やコーディングの基礎をしっかり整えてから、実践的な領域にチャレンジするようにしてください。性急にならないことが大切です。

(3) 基礎技術面

コンピュータゲームのプログラムというと、グラフィックス表現技術などを真っ先に思い浮かべる人が多いに違いありません。しかしそれらの前に、基礎技術の知識と、それに対する深い理解がなくては、他のどのような技術も軽薄なものとなってしまいます。自分の興味のある領域を出発点に、下記の基礎技術の習得を進めてください。

① 計算

プログラムはすべて計算ですし、プログラマはゲーム内のすべての事象を数学的・物理的に表現します。

三角関数、微積分、空間内の光線や視線の表現に不可欠なベクトルの概念、それらの空間表現に有利な行列計算、モノに実在感を与える様々な物理法則に基づく計算など、求めたい解にたどり着くためにどのような計算を行えばいいのかを知っていることは、プログラムをする以前に必

要なことです。そのためには、暗記ではなく理解が必要となります。

同一の解を求めるための計算方法を複数知っていること、より簡易な計算方法を利用することができること、それらのことは実際にプログラムをするための部品を多くもっていることと同じです。ブロック工作の例えを出すまでもなく、部品を多くもっていることは複雑な物作りに非常に有利です。

まずは基本的な数学・物理からはじめ、新しい問題の解き方に出会った時のわくわくするような感覚を経験してください。

そして、様々な問題の解法を知っていれば、あとはそれを特定のプログラム言語で記述するだけです。計算手法の習得は、プログラム言語を理解する前にも可能ですし、実際にプログラムに落とし込む前にこそしっかりとした基礎を作ることができるといえます。

② 言語理解

前項では計算に対する理解の重要性について述べましたが、理解を表現するためには「言葉」が必要となります。

私たちは他の人に説明する場合、自然言語である日本語を使っています。同じように、コンピュータに自分の頭の中を伝えるためにはプログラミング言語を使わなくてはなりません。

現在多くのゲームプログラマが最も触れる機会の多い言語は C/C++ 、次いで JAVA でしょう。各言語にはそれぞれ得意分野があります。

(a) アセンブラ

コンピュータが解釈し実行する形態に最も近い言語です。現在、アセンブラでのプログラミングはあまりされませんが、部分最適化を行う場合やコンパイル依存・言語仕様に絡むバグの修正等、理解していて役に立つ局面も少なくありません。どのような高級言語でも、最終的には機械語に変換されています。エンジンの構造を知らなくても車の運転はできますが、知っていればより効率的な走行ができるように、アセンブラはコンピュータの構造を理解した上でしかできない問題解決を行うために身につけた方がよい能力です。

(b) C

かつてはプログラマの入門口でしたが、最近その座を C++に譲りつつあります。しかし C は見逃せない強みをもっています。高級言語でありながら、アセンブラのようにコンピュータに近いプログラムの作成が可能なのです。

C++に進んでいくことを前提としても、ある程度までは C で学ぶことができます。アセンブラより生産性に富むことはもちろんですが、C++、JAVA などに比べて抽象化が進んでいないため学びやすく、将来 C++でプログラミングする際にも C への理解はより高度なプログラムを作成する手助けをしてくれるでしょう。

(c) C++

C++は現在ゲーム制作で最も普及している言語でしょう。C よりも大規模なコードに向き、様々な有用な言語拡張が施されています。クラス・継承などオブジェクト指向の拡張は、ルールを守って使えばプログラムの構造化に大変大きな力になります。C++を学ぶのであれば、とことん使いこなせるようになってください。

(d) JAVA

環境に依存しないオブジェクト指向言語でメモリ管理の簡便さには定評があります。モバイルアプリケーションやサーバプログラム開発では使用頻度が高い言語であるため、そういった方向への興味があるのであればあらかじめ身につけておくべきです。コンシューマゲーム機ではあまり使用されていません。

それぞれ利用価値と魅力のある各言語ですが、これからゲームプログラマを目指す人にはCからはじめて C++という順番をお勧めします。現在最も使われているという理由が一番ですが、Cを一応でも理解してから C++に進むことによって、コンピュータの「核心」に近づけると考えるためです。

この項の最後に、もう一度くりかえしておきます。言語理解はとても重要なスキルですが、言語だけが使えてもあまり意味がありません。プログラミング言語はコンピュータに自分の頭の中にある複雑な計算処理を理解してもらうための「伝達手段」です。伝えたい事柄、つまり問題の解決のために計算させたい数式があるからこそ言語が必要なのだということを忘れないでください。

③ データ管理とアルゴリズム

アルゴリズムとは問題を解決するための定型的な手法のことです。その意味では「計算」の項で述べたことは大きな意味でアルゴリズムの習得と理解です。

しかしコンピュータ言語上でのアルゴリズムという言葉は、より具体的な演算を指示する手法を意味するように使われることが多々あります。データをどのように格納しておき、それをどのような演算手法で効率的に処理させるかなど、同じ計算式のプログラムでも、コンピュータで処理させるために効率的な工夫が多種多様に存在します。そのため、目的に最適なデータの構造を選択すること、最適なアルゴリズムを用いることは、質の高いプログラムを作る上で必須の要素ですし、過去生み出されてきたデータ構造やアルゴリズムを知ることは、プログラマの選択の幅を広げ、独自のひらめきを付加するための土台となります。

また、オブジェクト指向はプログラムに取り入れることで、安全でメンテナンスしやすい強固なプログラム、可読性の高いプログラムを作成することができます。ゲーム開発プログラムのボリュームが増大し複雑化する中で非常に重要な考え方です。

これらの手法も市販の書籍などを参考に、状況に応じて選択することのできる幅をもって、一般的なものから身につけていくようにしてください。

(4) オプション技術面

この大項目は実制作に携わる時に、担当毎に必要になる項目です。これまで述べてきた事柄は、大概においてどのような領域を担当しても必要となるスキルでしたが、これから挙げる項目は上記基礎技術面に対しての代表的な発展型で、しっかりした基礎技術の上にしか成り立たないものです。

① システム

メモリ管理、ファイル、プロセス管理、割り込み、スレッドなど、いずれも OS が実装している部分です。近年のゲーム開発ではシステムについて知らなくてもゲームが作れるというレベル

まで隠蔽されています。しかし、メモリ管理は効率的なプログラムを作る際、そしてしばしばデバッグでも必要な知識ですし、それ以外でもプログラムを作成する際に知識が必要になることがあります。

② AI

ゲーム開発の特徴ともいえる技術です。コンピュータゲームは実際の対戦相手がいらないゲームとして発達してきたため、AI については古くから様々なものが作り出されてきました。古くはコンピュータの性能が低かったため、考えていないコンピュータをいかに考え深く見せるかという点が重要でしたが、コンピュータの進化とともに、いかに人間の思考をトレースするか、いかに人間らしく見せるかといった本来の方向へと進化しています。また、単純に対戦相手だけでなく、街の人がどのように行動するかなど、ゲーム内の世界を表現するために使用されています。

③ ネットワーク

コンピュータゲームでネットワークを利用するケースは増大の一途をたどっています。TCP/IP の各種プロトコル、サーバ・クライアント通信、P2P などの知識は、習得していれば大きな力となるでしょう。

また、サーバ側の各種既存ソフトウェアの有効利用、各種データベースを使用してのデータ管理、サーバ OS に対する知識など、サーバ側でのゲームプログラムを行う上で必要となる知識もあります。サーバ側でのプログラムは、負荷分散などクライアント側とはまた違った観点からのアプローチが必要となるため、それらの技術・経験に対する需要は多いといえます。

④ グラフィックス

(a) 2D（スプライト、画像フォーマット、エフェクト）

2D は平面的な画像の重ね合わせによる表現技術です。ゲームに 3D 技術が導入されてから、それ以前のスプライトや BG といった表示方式を総称して 2D と呼びます。現在でも、3D 表現の必要がない表示物（メニュー画面、プレイ情報表示やシステムメッセージなど）の表示や、2D でそれらしく見える表示物（エフェクト表示）などは 2D で表現されることがあります。この分野の業務を担当するためには、2D 独自の表示方式に関する知識、画像フォーマットに関する知識が必要となります。

(b) 3D 表示（ライティング、シェーダ）

3D 表現による表示技術です。ゲーム機内部で仮想空間を設定し、仮想カメラに対しての様々な物体の位置関係と形状、材質、光の影響を考慮して、仮想カメラがとらえた平面画像を計算で作り出します。ゲームで利用されはじめた当初には単純な形状とライト程度しか処理できませんでしたが、ハードウェアの進化と表現技術の進化によって非常にリアルな画像が作れるようになりました。現在では画像処理チップやチップ内での処理手順、メモリ構造などハードウェアに関する知識と、マッピングなどの表現技術に関する知識の両方が必要ですが、すべてがソフトウェアで行われる日ももうすぐでしょう。

(c) 3D アニメーション（キーフレーム、モーションブレンディング）

3D 表示のアニメーションは、表示物の形状変更と位置変更が中心です。形状情報と位置情報を間引いてもち、その間を演算によって補完する技術であるキーフレーム法や、別々のアニメーションを演算によって合成して滑らかに連続させ新しい動きを作り出すモーションブレンディングなど、演算によって情報を加工することが中心となります。数学と物理の知識が非常に重要です。

⑤ ハードウェア技術

デジタル電子回路に対する知識や機械工学への理解は、プログラムをする上で有利に働くことはもちろんです。しかし重要度としてはそれほど高くありません。

この項で挙げた各技術は、それぞれ非常に高度な技術力が要求されるにも関わらず、コンピュータゲームで実際に目につくため、基礎をないがしろにしたまま一足飛びに手をつけてしまいがちです。高度な技術は確固たる基礎の上にこそ成り立つことを再確認し、基礎技術面を少しでも充実させることを最優先に取り組んでください。

これらのオプション技術に関しては、自分の興味のある分野、得意な分野をスキルアップ対象に選ぶなど、１つ先の技術課題として捉えるのがいいでしょう。

2-3. ゲームプログラミングの基礎となる基本的な能力について

近年、様々なソフトウェア開発の中でも、特にゲームプログラマには数学や物理の素養が必要だといわれています。そこでまず、ゲームプラットフォーム・インタフェースの高度化と数学・物理との関係を説明し、ゲームプログラムに必要となる数学・物理が基礎となるスキルの習得方法について解説します。

次にプログラミングのセンスなどと呼ばれる、わかりやすく効率のよいプログラムを組むために必要なスキルについて説明します。

(1) ゲーム世界と物理学・数学

ゲームを行うためには何らかの場が必要となります。例えば RPG ならば主人公が冒険を繰り広げる街や広野、シューティングやパズルならば自機や敵機が現れたり、ブロックが詰まれたりする画面が必要となります。このようなゲームを行う場のことをここではゲーム世界と呼びます。

プレイヤがゲーム世界を理解するためには、ゲーム世界がある程度プレイヤが慣れ親しんだ世界＝プレイヤが日常生活している現実世界と似ている必要があります。プレイヤは類似点を元に画面が何を意味しているのか類推してゲームのルールを理解することができます。

そうでなければ、プレイヤはゲーム画面が何を意味しているのかわからず、ゲームのルールを理解することができません。ブロック崩しを例に挙げれば、ボールが跳ね返る、ボールが当たるとブロックが崩れて消えるなど、現実世界と似た点があります。

昔のゲームプラットフォームでは、シンプルなゲーム世界しか作ることができませんでした。このため、プログラマはキャラクタやものの動き、映像を直感で作ることができました。近年では、ゲームプラットフォームの進化により、ゲーム世界のリアリティが高く複雑になり、またコントローラの進化により自然な操作ができるようになりました。このため、ゲーム世界のキャラ

クタやものを色々な角度から見たり、様々に動かしたりするためのプログラムが必要になりました。映像や動きは非常に複雑で多様になるため、プログラマが直感だけでプログラムしてゲーム世界を作ることはとても難しくなってしまいました。

そこで、最近のゲームでは、現実世界についての知識（＝物理学）を用いて、ゲーム世界が現実世界と同じように動くようにプログラムを作ることが多いです。このため、現実世界がどうなっているかを調べ、プログラムで処理しやすい数式の形でまとめる学問である物理学がゲームプログラマの素養の１つとなりました。また、物理学に出てくる様々な数式をプログラムにする必要があるため、数学も必要になりました。以下、ゲーム世界を作り表現するために必要な技術と物理数学の関係を見ていきます。

(2)　コンピュータグラフィックスの物理と数学

自由な視点からゲーム世界を見回すためには、ゲーム世界に登場する物や人(キャラクタ)の 3 次元形状を用意し、視点にあわせて画面に表示する必要があります。この処理は 3 次元コンピュータグラフィックスと呼ばれます。実世界では、太陽や電灯といった光源から出た光が物体に当たって反射し、反射した光が目に入って網膜に結像することで物体が見えます。3 次元コンピュータグラフィックスはこの像を計算して画面に表示しています。

この計算をするためには、まず光線の反射・屈折といった現象を数式で表して、光源から出た光線がどのように視点に届くか計算します。次に、視点から少し離れた場所に画面があると考えて、画面と光線の交点を求め、その画素の色を光線の色にします。こうして全部の画素を塗りつぶせばゲーム世界の画面ができ上がります。

3 次元空間上に画面や光線を考えてその交点を求めたり、光線の反射を考えたりする部分では、3 次元図形、ベクトル、行列といった空間を表すための数学が活躍します。

(3)　３次元音響の物理と数学

人は左右の耳に音が届く時間の差から、音源が右にあるのか左にあるのかを区別することができます。また、前後や上下の位置も頭や耳たぶなどに反射する音の音色の変化からある程度わかります。また、音源の動きはドップラー効果によってはっきりとわかります。

音波の性質に基づいて、左右の耳に届く音波を計算することで、これらの効果を再現することができます。この処理は 3 次元音響と呼ばれ、例えば、頭のすぐそばを弾が通った時の音響を作り出すために利用されています。この処理では、波と波源の速度と周波数の関係、波の重ね合わせ、反射や回折といった波動についての性質を計算することになり、ベクトル行列に加えて、三角関数が活躍します。

(4)　物理シミュレーションの物理と数学

近年では、ゲーム世界の物体の動きを力学に従って計算して作り出す物理シミュレーションがしばしば行われるようになりました。物体だけでなく、人間や動物のようなゲームに登場するキャラクタの動きも力学に基づいて生成されることもあります。

物体は運動方程式に従って動きます。運動方程式を計算して様々な状況に置かれた物体がどのように動くかを考えます。これをプログラムにすることで、様々な状況の物体の動きを作り出すことを物理シミュレーションといいます。運動方程式は簡単な微分方程式ですが、微分方程式は非常にたくさんのことを表現できます（たくさんの解をもちます)。また、水や空気のような流

体の動きや物体変形のシミュレーションでは、運動方程式といくつかの微分方程式をあわせて計算します。このように、物理シミュレーションでは、微分方程式の計算が基本になるため、微分積分の概念が重要になります。

また、物体の運動を考えるためには、物体に加わる力を求める必要があります。物体同士が接触している時に働く接触力を計算するためには、物体同士がどのように接触しているのかを詳細に知る必要があるため、接触判定処理とよばれる３次元物体同士がどのように接触しているかを調べる計算が必要になります。そのため、３次元空間や図形のベクトル表記などの数学が活躍します。

(5)　ゲーム世界と物理数学の単元の対応

以上のように、リアリティの高いゲーム世界を作り出すために、物理を基に数学を活用したプログラムが作られています。下の図はゲーム世界を作るための技術と、中学・高校で習う数学・物理の単元や大学の授業の対応を表しています。

目的	ゲーム世界の表現		
支える技術	美しいコンピュータグラフィックス	迫力があり、気持ちのよい音響	キャラクタや物体のリアリティの高い動き
	3DCG： 光源から出た光が、反射をくりかえして目に入るまでを計算	３次元音響： 音源から出た音波が耳に届くまでを計算	物理シミュレーション： ゲーム世界の物体やキャラクタの動きを物理法則に基づいて計算
物理	光学、３次元空間 波動、図形	空間、波動	力学、３次元空間、図形
数学	力学や波動で使う数学：三角関数、微分積分 空間や図形の表現に使う数学：行列・ベクトル、線形代数		
数学の基礎	分数、方程式、1 次式、2 次式、三角関数、指数関数 ...		

例えば、アクションゲームではボタンを押すとキャラクタがジャンプする場面がよく見られますが、このジャンプの動きを中学の力学で習う投げ上げ運動を再現するようにプログラムすると、現実の物体と同じような滑らかで気持ちのよいジャンプの動きを作り出すことができます。投げ上げ運動の式は、中学の数学で習う２次関数になります。また、ジャンプしたキャラクタが地面の高さまで落ちてきた時の位置を求められれば、キャラクタが穴に落ちてしまうか無事着地できるかがわかりますが、この位置は２次式の解になります。

また、3D コンピュータグラフィックスを用いて、３次元空間内に作られたゲーム世界からゲーム画面を作るゲームが増えています。ゲーム世界内にカメラを置いて撮影した映像を見ているようなものです。この時、ゲーム世界内でのカメラの向きや位置は、位置を表す３次元のベクトルと回転を表す行列を用いて設定します。また陽だまりや影を表現するためには、光がどのように差し、それにより物の色がどう変わるかを計算する必要がありますが、これは光の反射を考えることでわかります。このように３次元コンピュータグラフィックスは、高校の数学で習うベクトル・行列や物理で習う光の法則が高度に組み合わさったものだといえます。

(6)　数学・物理の学習法

数学や物理などの理科系の科目には、2 年生で習う物理を理解するために 1 年生で習う物理と数学が必要であったり、3 年生で習う数学に 1 年生で習う数学が必要であったりと依存関係があります。本当は依存関係が絡み合っていますが、授業や教科書はできるだけ絡み合いがないよう

に考えて並べてあります。このため、教科書や授業の順番に勉強するのが一番わかりやすく、理解するための近道であることが多いです。

一方、理解を深めるためには、絡み合った依存関係をいろいろとたどってみるとよいです。ある内容と別の内容の関係を自由にたどっていけるようになることが望ましいです。このためには、教科書から離れて難しい問題に取り組むことや、疑問に思ったことをじっくり考えてみることも必要になります。例えば、計算して、図やグラフを書いたり、プログラムを書いてシミュレーションしてみたりといった実験を自分やってみることもよい練習になります。自分の興味のある内容、例えば、ゲーム世界の表現に直接関係する内容からスタートしてその周辺をたどりながら、図やグラフを書いたりプログラムを書いたりするとよいです。

(7) プログラミングのセンスとスキル

プログラムは、何らかの機能を実現するための手順を書いたものだといえます。ゲームのプログラムでは、例えば、ゲーム画面の表示、操作に応じた動作、ゲームルールの実現といった機能の手順が書かれています。

機能を実現するための手続きを書くためには、まず実現したい機能がどのような仕組み、構造で実現できるかを考える必要があります。次にその仕組み、構造がどのように動作すれば機能が実現するかを考え、そのための手順を書けばよいです。機能を実現するための仕組み、構造を考えることは、物事を捉え分析することにつながります。このような能力を養うためには、物事についてじっくり考えて分析することが好きになるのがよいです。例えば、ややこしい問題をじっくり考えること、不思議なこと疑問に思ったことを「なぜ、どうして」と考え追求すること、スポーツやゲームの上手いプレイを見て、どうしたらそのような上手いプレイができるのかを追求すること、複雑なことをわかりやすく説明することなどを楽しむことができるとよいです。

次に、仕組みや構造の動作を考え、その手順を書き下すことが必要になります。これは、順序を考えて段取りを組むことであり、自分で設計して物を作ったり、皆で行く旅行や、大きなイベントの計画を立てたりする経験が役立つと考えられます。

また、複雑な機能を実現するためには、いくつかの場合にわけて考えることが役立つことが多いです。順列・組み合わせ、確率のような数学の分野では、どのような場合があるかを考えすべての場合を尽くすことが必要ですが、プログラミングでも同じことが必要になります。

2-4.　スキルの時系列習得について

ゲームソフトのプログラマを目指す学生・生徒にとって、プログラミング言語やグラフィックス技術を学ぶことも必要ですが、その前に基礎的な知識と能力をしっかりと身に付けておくことが最も重要なことです。

プログラミング能力やグラフィクス API についての知識などは、経験を積むことにより熟達していきますが、その前提として、プログラマとしての基礎的な知識・能力（論理的な思考力、分析力、構成力、数学の理解）がしっかりと身についていることが必要です。

また、これら基礎的な技術・知識の習得にはじっくりと取り組む時間が必要となるので、学生時に習得しておくことが望まれます。

図「ゲームプログラマ時系列スキル習得例」に、具体的なスキル習得順序の例を示します。

この図は目安を示すための一例であり、図の順序通りに習得する必要はありません。

ゲームプログラマ時系列スキル習得例

ゲームプログラマへのステップ

大学・専門学校

高校生まで

「ゲームメーカへの就職」
それはゴールではなく、ゲームプログラマのスタート！

ゲームメーカに入社しても、プログラマとしては漸くスタートラインに立ったに過ぎません。学生時代に培った知識や技術を、たくさんの人が関わる実際の商品開発の中で応用し、活かすことが出来て、初めてプロのゲームプログラマと言えるでしょう。お互いのスキル、長所、性格を知り活かしながらの商品の開発は、他のことでは代え難い貴重な経験となるでしょう。
一方実務のあいまに、現場から視点を離して自分の仕事を客観的に見たり、将来を見据えて勉強する余裕も必要です。

基本技術の習得

魅力的なゲームを作るには、基本技術の習得も必要です。

＜プログラムの基本技術＞

アルゴリズムとデータ構造：検索、ソート、ツリー、ヒープ、計算量、FSM
コンピュータの動作原理：プロセッサ、バスの仕組み
プログラミング言語：母国語(C、C++)＋スクリプト言語
＋視野を広げるために他の言語もやりましょう。
ソフトウェア工学：チーム開発、テスト、再利用、メンテナンス性
オブジェクト指向、構造化、デザインパターン

＜ゲーム世界構築のための基本技術、基本知識＞

3次元世界の扱い：ベクトル、行列
コンピュータグラフィックス
物理シミュレーション：剛体、弾性体、流体の力学、制御工学
生理学心理学：プレイヤ（＝人間）のことを考えてゲームを作るため

またこれらを身に着けるためには、小さなゲームやゲームの1シーン、1場面を作る実践も重要です。

基礎能力の養成

プログラマとして必要な基礎的な能力を身に付けることが重要です。

ゲームプログラマは、プログラミングすることでゲーム世界を構築しますが、それぞれに必要なことがあります。

＜プログラミングに役立つ論理的思考＞

コンピュータを使って、様々な表現を実現するのがゲームプログラムです。
コンピュータはとても簡単な作業しかできませんので、プログラミングには難しい問題を簡単な問題に分けて考える分析力と、簡単な作業を積み重ねて大きな仕事を実現する方法を考える構成力が必要です。
数学、物理、化学、…の難しい問題を解いたり、文章を論理的に読解したり、物事を分かりやすく説明したりすることで、これらの能力を身に着けましょう。

＜ゲーム世界の構築に必要な物理法則を使いこなす技術＞

リアリティのあるゲーム世界を作るためには、現実世界の法則である物理法則をプログラムに組み込むのが良い方法です。物理と物理を使いこなすために必要な数学をしっかり学び、コンピュータの中に世界を作る方法を学びましょう。プログラムができるならば、物理の問題をプログラムで解くことにも挑戦してみましょう。

もの作りの経験

プログラミングには「もの作りの経験」と
「なぜ、どうしての追求」が大切です。

＜もの作りの経験＞

図工や技術、家庭の授業で習うような物を作る経験、夏休みの自由研究のような自分で考えて何かをやってみる経験が大切です。

＜なぜ、どうしての追求＞

また、不思議だと思うこと、不思議だと思ったことを「なぜ、どうして」と聞き返し、調べ、考えてみる経験も大切です。どうしてキリンの首は長いの？空は、なぜ青いの？といった素朴な疑問を大切にしましょう。分からなくても不思議だなと思い続けることも大切です。

こういった経験はプログラマに必要な「物事を論理的に考える」、「問題解決のための手順を考える」といった能力を育てます。

2-5. プログラマ採用における現状の問題点

ゲーム産業は、日本が誇るエンターテインメント産業です。世界の舞台で競争するため、開発者は新たなハードウェアの登場に伴い、より面白い、斬新なソフトウェアを作成すべく、新たな技術を習得してきました。

日本のゲーム産業は国内のコンテンツ産業の中でも、とりわけ、世界の多くのユーザに支持されております。当産業では世界を舞台に活躍できる新しい人材を必要としております。

ここでは新卒採用の問題点を挙げ、就職希望者、企業、教育機関の 3 者が、円滑に採用（就職）活動を行えるようになり、今後のゲーム産業の発展に繋がることを期待しています。

(1) 採用時のプログラム作品について

新卒プログラマの採用時にはグラフィックデザイナほどではありませんが、多くのゲーム企業では作品審査があります。その審査において提出される作品が共同制作や用意されたライブラリを使用している場合があり、採用希望者の制作した部分がわからないという問題点があります。

この問題に対しては、就職希望者や教育機関による対策が必要になります。教育機関は学生の作品制作環境としてライブラリを提供しているか、そのライブラリの内容やレベルについて明確にし、また、採用希望者が作品のどの部分を制作したかが客観的にわかるような文書を発行するのが望ましいと思われます。

作品審査の対策として、コメントを書き込むなどの読みやすいソースコードを意識して制作するような指導を実施している教育機関が増えていますが徹底されていないのが事実です。また、就職を希望する学生の自主性にも期待したいところです。

また、教育機関の現場では学生のやる気を引き出すためのアプローチとして、グラフィックや表現などの見た目を重視した作品制作の指導がなされている場合が多くありますが、プレイヤにわかりやすいメニュー作りやチュートリアルなどのユーザインターフェイスを制作できているかも企業側は見ています。

(2) 採用時における学校成績について

採用時において、多くのゲーム企業では「学校の成績」だけで決めるのではなく、現時点だけではなく将来も含めて業務をこなせる素養や技術があるかどうかを確認しています。

そこで多くのゲーム企業では少なくとも高校レベルの数学、物理の基礎知識がプログラマにおいては必要であると考えており、筆記試験で確認しています。

大学、大学院生については数学、物理に限らず、高度な専門知識（例えば、生命工学、建築など）を習得していることも重要と考えています。ゲーム制作に直接関係のない分野であっても、その専門知識を体系的にまとめる力やその研究の実績を評価して採用する傾向も見られます。

専門学校生においては、数学、物理の基礎知識に加え、即戦力として期待しているところからもプログラムやコンピュータの基礎についてそれぞれのゲーム企業が筆記試験を実施して確認しています。

採用時において「学校の成績」は重要視されていないのが事実ですが、それぞれの教育機関において重要視される面が異なり、それぞれの教育機関における授業や研究が重要視されていないわけではありません。昨今、数学や情報工学は苦手とする学生も多いかもしれませんが、積極的に勉学に励んでくれることを望んでいます。

(3)　教育機関の成績評価に対する問題点

(2)　で述べているとおり、採用時の学校成績評価は重視されておりません。各教育機関においてはこの事実を問題として認識し、対策を実施していかなければなりません。

この問題は、設置科目、科目内容、レベルが学校によって異なっていることや、異なる学校間で評価の基準が共通化されていないことなどから、企業側が学校の成績が採用時に考慮する項目として適正を欠いていると判断した結果だと思われます。

これに対して、各学校で設置している科目やその内容及び難易度、成績や評価について共通化する必要があるとの意見もありますが、現状では各学校や学科での目標や特色が異なることや、それぞれの教育ノウハウをオープンにすることができないなどの問題から困難なことのように思われます。

情報処理（プログラム）教育は約30年の歴史の中で、必要とされる科目やその評価の基準が共通化、適性化されていきましたが、ゲームのプログラム技術においては変化や成長が早いため必要なスキルの特定が難しいのが現状です。

企業側にとって就職希望者の実力を明示した評価を作成し提示することが現在の学校のとれる対策であると思われます。

(4)　採用時における情報処理系、言語等の資格取得の有効性

採用時において情報処理やプログラム言語等の資格取得を考慮していないゲーム企業が多数あります。この結果に対して、「採用担当者（ゲーム業界側）に資格に対する知識や理解が薄いのではないか」との意見がある一方、ゲームプログラマに求められる技術や素養が多岐に渡ることや、既存の資格がゲームプログラマの特性にマッチしているのかどうかという意見もあります。

しかし、資格は評価の基準が全国で共通化されたものであり、学習の経験や知識の有無、特に基礎力の判断ができることから、資格の内容によっては、新卒プログラマの採用に際しては有効な判断材料の１つとして見直してもよいのかもしれません。

また、上記の資格とは別にゲームの専門技術に係る資格検定制度を考えてはどうかとの意見もありますが、変化、成長のスピードが早いこと、質の保証が難しいこと、また、資格習得の対策が先行して本来の必要な能力の習得を目指す意義が希薄になる可能性、検定合格者が必ず採用になるわけではないことなどの問題点が多いと思われます。

(5)　専門分野の研究機関、研究者の不足

これは採用というよりも人材育成に対する問題点に近いのですが、米国においてはゲーム制作だけではなく、ソフトウェア開発における体系付けや効率化などといった開発工程を科学するなどの幅広い分野で研究が行われています。日本においては、ソフトウェア工学の講義で知識として習得するまでのものが多く、より実践的な研究として実施しているところは少ないようです。

ゲーム分野においては企業側からのニーズの開示が少ない部分もありますが、日本の教育機関においても様々な分野での実践的な研究がゲーム企業とともに進められることが望ましいと思われます。

(6)　就職活動時期に関する問題

当産業に限らず、産業界全体の問題ではありますが、よい人を早めに採用し、確保するといった採用の早期化が進んでおります。大学では３年から就職活動が始まることから、卒業研究が固

まっておらず、企業側は専門知識の習得状況を判断するのが困難な状況です。

また、ゲーム業界を目指していた学生が、プログラマやシステムエンジニア（SE）を求めるIT業界の採用活動の早期化や人材不足などから比較的採用されやすい環境であることから、IT業界に就職しているケースが見られます。特に優秀な学生ほど早期にIT業界に内定が取れ方向転換する傾向が強く見られます。

IT業界への方向転換の理由としては「ゲーム業界への就職は難しい」「作品の完成までに時間がかかることからゲーム業界への就職活動時期が長期化する」などです。

ゲーム業界においては、一部ではありますが優秀な人材がゲーム業界に就職してこないことを認識し、対策を考えていかなければならないのではないでしょうか。