オープンソースゲーム開発環境の現状理解と今後の展望

永 江 孝 規
メディアアート表現学科

A Survey on the Open Source Game Development Environment

NAGAE Takanori
Department of Media Art
(Received November 10, 2006 ; Accepted January 9, 2007)

1. はじめに

マイクロソフト社は2006年4月19日、同社独自の、ホビーストや教育機関、学生向けの開発環境であるVisual Studio 2005 Express Editionを永久に無償配布するとアナウンスした。しばらくして同年8月30日には、Visual C# 2005 Express Editionに基づく無償ゲーム開発環境XNA Game Studio Expressのβ版の提供を開始し、同年12月12日には正式版をリリースした[5~40]。DirectX（マイクロソフト独自のグラフィックス、サウンド、入出力API）の開発環境はすでに無償で配布されていた。これらのことはすなわち、DirectXを使ったマルチメディアプログラミング、特に3Dゲームプログラミングが、マイクロソフト純正の開発環境を用いて無償で行えるようになったことを意味する。マイクロソフトは大胆にも、この無償開発環境を次期OSであるWindows Vistaにバンドルしようとした[1)]。この戦略はかつてマイクロソフトが本丸のOSやビジネスソフトを有料のまま保持しつつWebブラウザを無償化して「抱き合わせ」的に圧倒的シェアを掌握したやり方を思い起こさせる。実際には種々の理由でバンドルを断念したものの、マイクロソフトがただの思いつきでVisual StudioやXNA Game Studioを無償にしたわけではなく、長期的な戦略の上での取り組みであることが伺える。

大学教員はおおむね特定の企業に依存しない中立な立場を好むゆえにオープンソースを愛好する。大学教育もまたオープンソースと親和性が高い。これまで、オープンソース運動を支えてきた無償開発環境は非常に限定されたものであり、事実上二つしかなかった。その一つはRichard Stallman率いるGNUが提供する開発環境（gccなど）であり、もう一つはSunによって完全に無償でサポートされるJavaであった。

マイクロソフトにとってはOSやXbox 360やビジネスソフトで利益が出ればよく、ホビースト向けの開発環境を無償提供することにはおそらくさほどのためらいはなかっただろう。SunやBorlandやGNUなどが無償の開発環境を提供している中、マイクロソフト側の体勢が整い次第無償化する計画だったのではないか。事実、C#は2003年4月にはISOで標準化されており、2006年3月にはJISで標準化された。C#がJavaに類似しているとはよく言われるが、Javaを参考にC#を開発する当初から、C#はJavaのように標準化する予定だったのだろう。Javaの開発にもSunの打算があったようにC#にもマイクロソフトなりの現実的な思惑があるのは当然であって、我々がその利益を享受できるのであれば拒む理由はない。

Visual Studio Express EditionはGNUやJavaに匹敵する影響をオープンソースの世界に与えるだろう。ネットワークやデータベース業務などであればJavaテクノロジーの優位性がゆらぐ気配は見えないが、ゲームなどのインタラクティブコンテンツにはマイクロソフトのDirectXが必要不可欠であって、マイクロソフトから純正の無償開発環境が提供されるということには非常に重大な意味がある。

本稿では、筆者が現時点で開発環境としてXNAを選択した理由について述べる。簡単に結論を言ってしまえば、XNAはDirectXを利用できる無償でかつ純正の開発環境であるからである。XNAはゲーム以外にも広くメディアアート制作にも利用可能である。将来OpenGLなどのオープンなグラフィクスAPIとJavaのようなフリーな開発環境が、何かの強いリーダーシップの下、あるいは何かの必然性によって結びついて、XNAを凌駕するようなことがあるかもしれない。しかし今のところ、XNA以外の選択肢はないように思われる。

以後本稿では、2章で研究の背景についてのべる。3章は本稿の主要なテーマであるXNAとJava、およびそ

の周辺について述べる。4章はデジタルメディア教育のための開発環境について考察する。最後の5章には考察と結論を記す。

2. 研究の背景

2.1 アマチュアとコミュニティ

世の中には無数のアマチュアプログラマがいる。彼ら一人一人が社会に与える影響は非常に小さく、かつインターネットが存在しなかった時代にはばらばらに孤立していた。しかしインターネットによって彼らは世界的規模のコミュニティを作り、共同作業と集積によって大きな影響を与えることができるようになった。電子工作やプラモデルなどの物理的実体をネットワークごしに共有することは難しい。しかし、プログラミングは容易に海を越え、国を超えて、ほとんど無償で、協働できる。

オープンソースは単なる同人活動や日曜プログラミングと明確に異なる。オープンソースが成立するには、常時オンラインで活動を続ける世界的なコミュニティの存在が不可欠である。オープンソースとはつまり単にソースが公開されているという静的なモノや状態のことではなくて、その本質はオンライン上の世界的規模のコミュニティなのである。従って「オープンソース」イコール「オープンソースコミュニティ」のことだと言って間違いない。

このようなオンライン上のコミュニティは、歴史的には、誰かが作ろうと意図してできあがるのではなく、何らかの必要に迫られて、自発的な相互扶助の結果としてできあがるものである。たとえば一例としてLinuxのコミュニティを取り上げよう。ここにはLinus Torvaldsという中心人物がいるが、彼がすべてを取り仕切っているわけではない。また彼の周りに特別に重要な人物がいるわけでもない。従って、マイクロソフトがLinuxの優秀なエンジニアを数十人数百人引き抜いてもLinuxコミュニティには何の影響もない。BorlandやNetscapeなどの会社組織であれば、有能な一部のリーダーをヘッドハンティングされることによって痛手を受けることもあるが、オープンソースコミュニティでは無名の人物によって常に代わりの人材が補充される。またそうでなくては、コミュニティは持続していかない。

Linuxが普及した背景には、無料で利用可能なUnixクローンOSに対する潜在的な需要があった。有料のUnix、無料だが窮屈なUnix、コミュニティが育たない不毛なUnixよりも、人々は未完成で不完全だが、自由なUnixであるLinuxを選んだ。いったんLinuxというコミュニティが形成され、その将来性が嘱望されるようになると、IBM、HP、Oracleなどの大手ハードウェア・ソフトウェア企業がLinuxを商業目的に利用するため、Linuxをサポートするようになった。IBMなどは気前よく、自前の有償OSであるAIXから多くのソースコードをLinuxコミュニティに寄贈し、また常勤のエンジニアも開発に投入した。このようにしてLinuxは大手企業のサポートを受け、デファクトスタンダードの地位を勝ち取るまでになった。

ただ単に自分の感性の赴くままになにかプログラムを組んで、それをフリーソフトウェアライセンスで公開してもそのことがオープンソースなのではない。同人ゲームを作って同人ショップや同人マーケットで小銭を稼いだりしても、そのこと自体が商売でないのと同じである。同人活動は商業活動の形をとった趣味にすぎない。おなじようにゲーム開発、ゲーム販売などといっても採算を度外視した活動は趣味と違わない。

世の中には「オープンソースゲーム開発」と呼ばれるものがすでにたくさんある。しかし、それらのほとんどすべては「オープンソース」を自称しているだけであり、事実上のオープンソースの条件を満たしていない。つまり、

- 潜在的需要に応えている。
- 世界的規模のオンラインコミュニティを持つ。
- 無数のアマチュアメンバーのささやかなボランティア活動によって支えられている。
- 商業利用によってデファクトスタンダードと見なされている。

といった項目が該当していなければいけない。もし、

- 需要がないのにあると信じ込んで自分の趣味の世界を正当化している。
- 世間にほとんど知られてないのは世間に見る目がないせいだと思っている。
- 特定少数の人に負担が偏っていて、それらの人がいないと成立しない。
- 商業的に成立しない。
- 標準化はおろかドキュメントすらない。

といった項目が該当するようであれば、明らかにそれはオープンソースではない。形式的にはオープンソースではあるかもしれないが、Web 2.0的ではない。「Web 2.0的」[1]とはここではオンライン上で多くの人々のわずかな労力が積み重なることによって、旧来の組織やメディアを超える質・量の生産物を作り出すことを言っている。オープンソースは、Web 2.0的でなければ発展のしよう

がない。

Shawn Hargreaves は、オープンソースゲーム開発についての先駆的な論文の中で以下のように問題提起している[2]：

> ハッカーは、クールなことに取り組むのを好む。退屈で義務付けられた仕事もこなすことも多いが、それは結局のところより興味を起こさせる選択肢がない場合のみの話である…。ゲームはクールなものであり、大抵のプログラマはそのコードを楽しんで書くが、それなのにゲームこそが Linux が Windows プラットフォームで入手できる市販製品からずっと遅れをとっている領域なのである。一見したところ、納得のいく話ではない。

筆者もまた一人のオープンソース愛好家として、これまでなぜオープンソースなゲーム開発環境が一般化しないのかと、現状を憂えつつも、その理由は思い付かないでいた。確かにゲーム開発はハッカーにとって非常に魅力的な分野であって、これまでも無数の試みが繰り返されてきた。

オープンソースとハックとは非常に密接な関係にあるが、オープンソースは個人から大企業まで、世界中のあらゆる組織や企業が参加できる場であることに本質がある。ハッカーはその中の一要素にすぎない。ゲームはクールであるがこれまでオープンソースにあまりなじまなかった。少なくとも最適な分野ではなかった。

オープンソース開発はクローンやエミュレータが目立つ。たとえば FreeCiv は Civilization という商業ゲームのクローンとして成功している。Jake は Quake を Java で移植したクローンである。Linux も元はといえば Unix クローンであった。また Apache も a patch が語源となっているように、NSCA httpd から分かれたものである。オープンソース開発は、少なくともそのスタートの段階では、既存の商業製品のまねであって、同人開発となんら区別がつかない。最初はクローンでもやがてはオリジナルを超え、オリジナル以上のオリジナリティを獲得して成長していくものが本当のオープンソースである。

オープンソース運動、少なくとも、数年間継続して人々にも知られるようになったものは、非常に多くの萌芽的な試みの中で、たまたまうまく行って生き残ったものなのである。従ってそうはならなかったα版やβ版のオープンソースプロジェクトが山のようにあってもおかしくはない。それにしてもなぜゲーム開発の世界で Linux や Wikipedia や Apache や Firefox に匹敵するような優れたオープンソースソフトウェアがいまだに生まれないのだろうか。

同人は「二次創作」「二次制作」と呼ばれることもある。パロディとも違う。いかなるオリジナル作品もなんらかの影響の下に作られるということを肯定すれば、パロディは単にその既存の作品の影響を明示し誇張することだろう。二次制作は模倣の殻から出てこず、自分の妄想で好きなようにいじり回しているだけのことが多い。同人であり続けるには商業的な成功をあきらめねばならないこともあるだろうし、商業的に成功したければ同人的なアングラ意識から抜け出さなくてはならない。アングラ、クローン、エミュレータ、リバースエンジニアリング、違法改造から始めても、コミュニティが健全に育ちさえすればいずれの日にかは世界的なオープンソース運動に昇華させることができるはずである。

同人開発がオープンソース開発に成長できるかどうかは法則性があるようでもあり、偶然に左右されるようにも見える。Linux と Apache はほぼ順調にクローンからメジャーへの道を歩み続けた。一方 Mozilla は挫折の連続であった。1998年に AOL が Netscape を買収して事実上 Netscape が IE に敗れ去ったのちオープンソースの Mozilla が生まれたが、しばらく開発は停滞していた。2002年に Mozilla スイートから Firefox を分離し、ソースの簡素化・軽量化を図ったことによってやっと開発が加速してきた。Mozilla のようにかつてはうまくいってないと思われていたプロジェクトも、何かのきっかけで発展していくこともある。2D グラフィクスの世界では、Photoshop に代わる Gimp が古くから知られていたが、Illustrator に代わるフリーソフトがなかなか現れなかった。ところが最近になって急速に Inkscape というクオリティの高いドロー系のオープンソースソフトが成長しつつある。かと思えば、かつては非常に盛んだった SETI at home などのプロジェクトは今ではまったく忘れ去られてしまった。どのようなプロジェクトに将来性や持続性があるのか、あるいはないのかを判断するのは難しい。

PC ゲームの世界ではオープンソース運動と連動した変化が起きていた。1990年代後半から2000年にかけて id Software の John Carmack らによって Doom、Quake、Quake2などのゲームのソースコードが GPL ライセンスで公開された。Quake から派生した Half-Life、Half-Life からさらに派生した Counter-Strike などは PC ゲーム特有の自由な雰囲気の中ではぐくまれ発展してきたものだ。このことからもゲーム開発の中では特に PC、PC ゲームの中でも特に FPS（first person shooter; 一人称視点シューティング）に、オープンソース運動との連動性が高い。

2.2 OpenGLとJava

多くのハッカーたちはクロスプラットフォームでかつ無償であるという点でOpenGLとJavaの組み合わせを好んだ。LWJGL（Lightweight Java Game Library）[41]などがその成果であるといえる。LWJGLはOpenGLやOpenALなどのAPIを用いたJavaによるゲームライブラリであり、2007年1月7日現在でバージョン1.0 RC1がリリースされている。OpenALはOpen Audio Libraryの意味であって、Javaゲーム開発のためのサウンドライブラリとして開発されているのがJOAL（Java bindings for OpenAL）である。同様にJOGLはJavaでOpenGLを扱うためのAPIである。JOGLはSunが直接開発しており、2006年12月にリリースされるJavaの新しい開発環境Java SE 6でも正式に採用される。かつて同じくSunが直接開発していたJavaの3Dグラフィクスライブラリに Java 3Dというものがあったが、バージョン1.3.2以後はSunとは半ば独立したコミュニティに開発が移り、現在のバージョンは1.4.0である。

Java3Dは前世紀のVRML時代にその基本ができあがっており、プログラマブルシェーダを多用するGPUの進化に追いつけているとは言い難い。JOGLはJava3Dに比べると低レベルのライブラリだが、比較的最近OpenGLに加わったGLSL（OpenGL Shading Language）にも対応しており、現在のGPUの性能を活かすことができるという意味において、これからはJava 3Dよりも使われる機会が増えると考えられる。

思うに、Java 3Dはまだ GPUが一般化していなかった時代に、CADやVR（virtual reality）といった特殊な最先端技術のために開発され、それゆえに一般ユーザにはオプション扱いだった。ところが、Windows VistaやMacOSX 10.5、およびLinux版のGLXのように、OSのGUIが3D化していく傾向がある。Sunとしては純正GUIコンポーネントであるSwingを急遽最新GPUに適合させる必要が生じたが、Java3Dではその用途には重すぎて、複雑すぎた。そこでOpenGLを直接制御する比較的低レベルのAPIであるJOGLをSun自ら開発し、Java SE 6に間に合わせたということだろうと思われる。結果的にようやくJavaでも3DCGがメインのパッケージにバンドルされるようになり、2Dグラフィクスと3D CGが統合されることになった。

Jake2はGPLライセンスで公開されているQuake2のソースコードを元にLWJGL、JOGL、そしてJOALを使って作られていて、その完成度はきわめて高い。Quake2は1997年にリリースされており、その当時の最先端のFPSが今日のJavaで十分高速に実行できることを証明している。Jake2以外でLWJGLを用いて開発されたゲームにmonstrumoというオリジナルのFPSがある。だがその完成度はまだ高いとは言えない。その他のプロジェクトに至ってはまだα版かβ版のレベルにとどまっている。

LWJGL自体が未だ正式版がリリースされていないことと、Javaがいまだに数世代前のFPSしかサポートし得ないという事実を私たちは深刻に受け止めなくてはならないと思う。LWJGLは最初から商業ゲームを開発するために設計された。このような世界に将来販売会社と開発会社の分業体制や、ディレクターとプロデューサーの役割分担が確立された正真正銘のゲーム産業が生まれるのだろうか。それとも、ただハッカーたちが自分の衝動の赴くままにゲーム作りをしている、そんな状態がこれからも続くのであろうか。

2.3 ゲーム開発コストの高騰

ゲーム開発の初期は、開発コストは非常に小さなものであった。システムには汎用性はほとんどなく、コードはアセンブラで記述されるのが当たり前であった。せいぜい数人が数ヶ月程度で一つの作品を制作した。したがってソフトウェアの開発費用はせいぜい1千万円程度だったと推測される（ハードウェアの製造費や広告費などはのぞく）。ところが最近のゲームだと数百人のスタッフが数年がかりで開発することもざらであり、人件費だけで数十億円から数百億円の費用がかかっていることになる。実際、現時点ですでにゲーム開発コストは1作品あたり3億US$から6億US$程度（約500億円）、PS3やXbox360のゲーム開発には10億US$から20億US$（約1500億円）かかると言われている。1999年にリリースされたシェンムーの開発費が70億円で、世界でもっとも開発費がかかったビデオゲームとしてギネスに認定されたのが、遠い昔の話のようだ。Halo 2の開発にはスタッフが190人以上、3年がかりで総制作費が40億US$（約5000億円）かかったという。これは人気ハリウッド映画の制作費に相当する。

PC上の開発がCやC++に移行しても長い間ゲーム機のゲームはアセンブラで開発されてきた。PS2もグラフィクスはアセンブラで書かれる。一方でゲームはますます複雑化し、開発コストは膨大になり、アセンブラやC++などの比較的低レベルの言語で開発しているとその人的時間的コストがバカにならなくなってきた。このままアセンブラで開発を続けて行けば遠くない将来に破綻することは誰にもわかるようになった。

またWii、Xbox360、PS3などの最近の家庭用ゲーム機では、CPUはPowerPCベースの独自仕様のマルチコアCPUを搭載し、またGPUもNvidiaやATIのGPUに基

づくものの、PCなどに使われているものとは同じではなく、独自に手が加えられている。当然、開発環境や言語やOSやAPIなどもばらばらである。サードパーティの販売元（publisher）や開発元（developer）はこれらのゲーム機のためのクロスプラットフォーム開発環境を用意するだけで相当の先行投資が必要だが、ゲーム機のライフスパンは5年間で、その後はまた別の開発環境を再構築しなくてはならない。

ゲーム開発コストはますます高騰していく。このままでは独自仕様のCPUやGPU、独自のハードウェアとOS、独自の開発環境と独自のグラフィクスAPIでは、ゲーム開発自体が不可能になるかもしれない。ハリウッドの映画制作会社くらいの巨大資本ならともかくとして、日本の販売元や開発元にそのような体力はあるのだろうか。

PCの世界では、IBM PCが1990年前半に完全に一人勝ちしてしまった。ハードウェアの規格自体は徐々に進化していくが、同時には標準規格が一つだけしか存在しない。その一つの標準規格に対して、マイクロソフトやBorlandなどの複数のソフトウェア開発会社がしのぎを削るという構造になっている。1990年代後半からは、企業だけではなくオープンソースコミュニティもこの世界に参戦してくる。家庭用ゲーム機の世界のように、ハードウェアメーカーがいくつか並立して、それぞれが独自の仕様で独自の開発環境を用意するというものではなく、非常にオープンに競争が行われる。その結果新しい言語や開発環境、クロスプラットフォームやフレームワークなどが次々に考案されて、古い仕組みはまたたくまに捨て去られていく。

PCの世界に住んでいる人間にとってはゲーム機の世界はまどろっこしくて仕方ない。この世界でまっとうなソフトウェア開発会社といえば、新参のマイクロソフトしかいない。マイクロソフト以外は、ソフトウェア開発という意味で10年以上遅れているのである。ソニーや任天堂は少なくともソフトウェア開発という世界でまともに戦った経験がないのだ。開発費の高騰、マイクロソフトの参入、そしてオープンソース運動の導入によって、ゲーム開発の世界にももっと自由でオープンな競争や協働がもたらされることになるのだろうか。

2.4 アマチュアプログラマとは何か

アマチュアは単なる素人とは限らない。特に技術職やクリエータ職の趣味は本業と密接に関係していることが多い。たとえば本業が役所の事務職員であり、趣味が囲碁だとしよう。事務処理と囲碁の関わり合いはほとんどなく、囲碁は単なる息抜きだと考えてよかろう。しかし、プログラミングの場合、本業はSEで、仕事ではCOBOLやFORTRANの保守をやっているが、趣味でCやJavaのゲームプログラミングをやっている、というようなことは十分にありえる。仕事もプログラミング、趣味もプログラミングというパターンであって、プログラマーにはこういうタイプの人が非常にたくさんいる。たとえばゲームのテクニカルディレクターが趣味でプラモデルを作ったりフィギュアを作ったりする。これも本業と趣味がきわめて近接している例である。趣味と実益をかねている場合もあるだろう。ある人がLinuxの保守を生業としていて、趣味でカーネルをハックしているとすれば、その趣味の知識は十分にシステム管理に役立つだろう。

オープンソースやWeb 2.0によらずとも、市場規模が拡大することによって同人活動が商業的に成立することもあり得る。コミックマーケットのような巨大な同人市場を作り上げる分野では、プロと同人の中間に位置する「プロ同人」が生まれている。つまりプロの漫画家か漫画家のアシスタントが同人マンガを描いていたり、同人マンガから商業誌に進出したり、あるいはあえて逆に商業誌で名前を売って実利を同人で得るなど、プロとアマチュアの間に連続的で多様な形態が生まれつつある。このような現象はインターネットの世界と同時並行的に進行しており、本業と同人の間の線引きはますます曖昧になってきている。

趣味のプログラミングとは往々にして、本業のプログラミングではできない、自由な、自分のやりたいようにできるプログラミングなのであり、従ってそれは損得の問題ではないし、かつプログラミングスキルが低いことを意味しない。同様のことは同人マンガや同人アニメなどにも言える。

オープンソースコミュニティという仕組みは、そのようなセミプロのプログラマやクリエータをうまく組織して、かなり高水準の、ときとして世界最高水準の作品を作り出す。同じことをゲーム開発に適用してみよう。もし、ゲーム開発環境のオープンソース化、クロスプラットフォーム化が成立すれば、ゲームユーザーの中で多少のプログラミングやゲームデザインができる多くの人たちがゲーム作品を改良してくれるかもしれない。オープンソースコミュニティにプログラミングやデザインを一部アウトソーシングすることによって開発費用は劇的に安くなるはずだ。またそのような開発コミュニティからインディーズやベンチャー会社が立ち上がって、本当のゲーム開発会社に成長するかもしれない。このようなことはPCゲームの世界ではすでに前例がある。QuakeからHalf-Lifeが生まれ、Half-LifeからCounter-Strikeが

生まれるように、PC ゲームではユーザ有志が MOD（改変、modification の意味）を作ることが一般化しつつある。

3. ゲーム開発環境

3.1 統合開発環境

エディタ・コンパイラ・リンカなどを統合した統合開発環境（IDE; Integrated Development Environment）を最初に実現したのは1983年にリリースされた Borland 社の Turbo Pascal である。Borland はその後クロスプラットフォームの統合開発環境 Delphi を1995年にリリースし、Object Pascal、C++ Builder、JBuilder、Kylix などが、VCL（Visual Component Library）という単一のフレームワークの上で開発され、実行された。マイクロソフトはこの VCL のエンジニアを大量に Borland から引き抜くことによって自社のクロスプラットフォームフレームワーク .NET Framework を立ち上げたと言われている。

コンパイラ、リンカ、統合開発環境などを独自に開発できる会社は非常に限られている。Borland、Sun、IBM、Intel、Oracle、マイクロソフト、そして GNU くらいである。この中で将来独自の言語、独自のフレームワークを構築できる（あるいは構築に影響力を行使できる）体力を持つ会社は、マイクロソフトと、おそらく IBM しかいない。この戦いに日本企業はまったく参戦していない。そしておそらく今後のソフト開発産業界は、Java と Linux を全面的にサポートする IBM が提供する Eclipse と、PC 界の覇者マイクロソフトがサポートする Visual Studio の最終戦争に突入することになろう。GNU の影響力はますます小さくなるだろう。もし GNU が GPLv3（GNU パブリックライセンス、バージョン 3。次期 GPL ライセンス）で gcc などの開発環境を提供し始めれば、GPLv3に否定的な Linux コミュニティは独自の言語や独自の開発環境を模索することになるかもしれない。オープンソースコミュニティ全体を巻き込む大問題に発展する可能性もある。

3.2 Java と .NET Framework

フレームワークは IDE と同義に使われることもあるが、本稿ではもう少し広く、言語や実行環境や開発環境やプラットフォームやネットワークなどをひとくくりにまとめた概念であるとして議論を進める。

言語やプラットフォームごとに開発環境を作っていては同じような手間が何度もかかることになる。まず多言語対応でクロスプラットフォームな仮想化されたフレームワークというものを作っておいて、新しい言語も新しいプラットフォームもみなその共通の仮想化したフレームワークの一機能として実装することで対応すればよい。このフレームワークという考え方は言語や開発環境というものが発展してきて必然的に生まれてきたものであり、ゲーム開発環境も当然数年後にはこのようなフレームワークという概念の中に吸収され統合されることになるだろう。

PC 上での多言語共通のクロスプラットフォームを最初に築き上げたのは Borland であったが、一方で、機種依存しないネットワークを介したクロスプラットフォームを実現したのが Sun の Java である。

インターネットが漸く普及しつつあった1995年に Sun から Java がリリースされた。Borland やその他のライバルたちを蹴散らして、ビジネスソフト界で覇者となったマイクロソフトが当時次に狙っていたのはインターネットの覇権であり、ねらいをつけたのは Java だった。マイクロソフトは Java ごとインターネットをまるごと我がものにしようとしたが、Java を勝手にいじくり回すことを Sun に拒まれた（1998年の Java Compatible 商標使用仮差止命令と、マイクロソフトによる Windows での Java VM サポート中止の件）。このときからマイクロソフトは Java に取って代わる独自言語を開発しようと考えたに違いない。それが C#であった。

Java はメモリの動的管理（GC、ガベージコレクション）や例外処理やセキュリティの管理などを行う仮想化されたクロスプラットフォームの実行環境である仮想マシンの上で動く（厳密には、実行環境のことは JRE（Java Runtime Environment）と言い、仮想マシンのことは Java VM（Virtual Machine）と言う）。Java をまねて新しい言語を作っただけでは意味はなく、Java 同様な仮想実行環境を作らなくてはならない。Java VM に相当するマイクロソフトの実行環境が.NET Framework であり、また CLR（共通言語実行環境、Common Language Runtime）ともいう。Common Language と謳っているのは、この環境で Visual C#だけではなく Visual BASIC なども動くからである。.NET Framework は2001年にはじめてリリースされた。Borland の Delphi、Sun の Java に遅れること約 5 年であった。

よく知られているように、PC ゲームのデファクトスタンダード開発環境である Visual C++ / DirectX で開発されたゲームは仮想化された実行環境で動くのではない。C++では伝統的に動的メモリ割り当てやメモリ解放などのメモリ管理を、実行環境ではなくてプログラム自身が明示的に行ってきた。GC はメモリを解放する必要が生じると自動的に実行される。しかしながらリアルタイムプログラムは、フレームレートを維持するために

コンピュータのリソースをすべて制御する必要があり、GCのようないつ動き出していつ終わるかもわからない不確定要素を組み入れることは不適当だと考えられてきた。

Visual C++ / DirectXのソースコードは見るもおぞましい低レベルの記述に満ちている。このようなコーディングは、LinuxであればカーネルソースかシステムコールかデバイスドライバをCでごりごりと書くときくらいしか使われない。

たとえば、Javaならば「構造体へのポインタの領域を確保」して、「構造体にデータを格納してデバイスに渡す」などという記述を決してしない。「マウスデバイスの初期化」のコードを書くことはあり得ないし、「マウスデバイスからの出力をポインタに入れて受け渡す」こともない。どのようなマウスイベントを扱うか、そのイベントが発生したときにどのような処理を行うかを記述するだけである。

VisualC++ / DirectXのような「低級言語」で、ますます複雑化するゲームプログラミングを行うことは、次第に困難になりつつあったが、同時にCPUやGPUの能力はますます向上してきた。10年後、20年後を予測した場合、いつまでもVisual C++のようなコーディングをするとも思えなくなってきた。となるとC/C++の次に来るべき言語はやはりJavaしかない。オープンソースなOpenGLがJavaと組んだとき、Visual C++/DirectXでは将来勝ち目がない、ましてVisual BASICではJavaには太刀打ちできない、とマイクロソフトは考えただろう。そこでマイクロソフトがJavaに対抗して開発してきたのが、.NET Framework / Visual C# / Managed DirectXなのである。

本来DirectXは.NET Frameworkでは扱われないが、C#は.NET Frameworkで動く。C#でDirectXを使うために、.NET Frameworkに合わせたDirectXがManaged DirectXなのである（Visual C++もManaged C++=C++/CLIから.NET FrameworkやManaged DirectXにアクセスできる）。Managed DirectXでは、デバイスの初期化などの低レベルの記述のほとんどは隠蔽されている。

Managed DirectXの初期のバージョンMDX1.1ではかつてVisual BASICでサポートされていた2次元グラフィクスAPIであるDirectDrawもサポートされていた。Java Graphicsクラスに相当するいろいろな描画処理ができていたのである。また、Direct3Dもサポートされていたので、3次元図形も描画できたが、同様のことはJava3Dですでにできていた。MDX1.1+C#は一事が万事、Javaの焼き直しのような印象があった。C#は言語仕様もJavaに酷似しているしやっていることもJavaと同じとあっては、世間からC#はマイクロソフトのJavaクローンだと言われても仕方のない状態であった。

ところがMDX1.1を引き継いだMDX2.0βは、2006年10月6日をもって突然無効化された。Managed DirectX / C#の正当な後継としてXNA Game Studio Expressが位置づけられたためである。XNAはもはやDirectDrawも2次元グラフィクスもサポートしていない。また、固定シェーダのサポートをやめ、プログラマブルシェーダだけに対応した。固定シェーダとは、拡散反射光や鏡面反射光、環境光などのごくふつうのシェーダのことで、従来はハードウェア的に標準実装された。最近のGPUは固定シェーダを省略し、シェーダはすべてプログラマブルシェーダとして扱われる。プログラマブルシェーダは統合シェーダ、共用シェーダなどとも呼ばれ、（マルチコアの）複数同一のシェーダを頂点シェーダに使ったりピクセルシェーダに使ったりして融通を利かせることによってGPU全体に負荷を分散させ稼働率を上げる（ロードバランシング）というねらいがあるものと思われる。また、実際のゲームプログラミングを考えたとき物体表面にはほぼ間違いなく何かのテクスチャが貼られる。単純な固定シェーダが使われる場面はそれほど多くない。

2Dグラフィクスも多くの場合はテクスチャやスプライトで解決される。もしどうしても2D描画処理が必要であればソフトウェアで十分だし、もしPostScript並みの高機能な2Dグラフィクスを実現しようとするとGPUにもそれなりの負担がかかる。GPUは高度な3Dに特化したということだろう。プログラマから見ればデフォルトのシェーダが廃止されたので、シェーダをすべてプログラムしなくてはならなくなった。

DirectXはOpenGLを模倣したり、Java Appletをまねるためにあるのではない。そのようなものは、未来のグラフィクスには必要ないか取るに足りない要素である。だから切り捨ててしまおう、というのが、MDX2.0βの無効化、XNAの登場として現れたものと思われる。ここに至ってXNAはJava + OpenGLとははっきりと違う独自の方向性を持ち始めた。

3.3 DirectXとOpenGL ES

最近、PS3のグラフィクスAPIにOpenGL ES 2.0（OpenGL for Embedded System 2.0）が採用されたことが話題になった。OpenGLはもともと汎用3Dグラフィクス APIであり、2Dグラフィクス APIも兼ねている。またCADなどゲームにはあまり必要とされない仕様を多く含んでいる。その割にはGPUのプログラマブルシェーダへの対応が十分でなかった（OpenGL2.0用にシェーディ

ング言語 Cg が提唱されたのは2002年 2 月だったが、OpenGL 2.0のリリースと GLSL の正式サポートは2004年 8 月である。一方マイクロソフトの HLSL は2002年12月に発表された)。また PS3を開発するにあたってソニーとしてはいつまでも GPU をアセンブラレベルでコーディングしていては開発コストが爆発するとようやく気づいたのであろう。何かもっと高級なグラフィクス API は無いかと模索したが OpenGL では図体がでかすぎるし、DirectX はマイクロソフトの独自仕様だし、これは OpenGL ES にてこ入れして新たにグラフィクス API を作るしかないと考えたのではないか。

OpenGL ES 1.0は確かに組み込みシステム用の OpenGL として考案されたのかもしれないが、ES 2.0は主にソニーの PS3用にスペックが決定されたのではないか、と筆者は考えている。もちろん PS3も家庭用ゲーム機なのであるから、組み込みシステムの一種である。しかし、OpenGL ES 2.0の真の意図は、GPU プログラミングに最適化した DirectX のクローンというところにあるのではないか。

OpenGL 1.0は1992年に生まれたが、当時リアルタイム3DCG アニメーションは一部の CAD システムやフライトシミュレータ、シリコングラフィクスのワークステーションなどに限られており、まだ PC の世界には降りてきてはいなかった。一方で DirectX は Nvidia や ATI の GPU とともに急速に進化して、少なくともシェアにおいて OpenGL を抜き去ってしまった。DirectX（厳密には Direct3D）は最初 OpenGL の PC 用簡易版のような位置づけだったが、GPU の進化と同調していち早くピクセルシェーダ、頂点シェーダなどのプログラマブルシェーダを導入した。これに対して OpenGL は長い間 GPU と無関係な機能が多く付随したままだった。

ところで OpenGL ES を開発したのは Khronos という企業グループだが、OpenGL の規格を決定する OpenGL ARB（OpenGL Architecture Review Board）は OpenGL の仕様管理を Khronos に移管することが2006年の SIGGRAPH で発表された。OpenGL ARB にはもともとマイクロソフトが含まれていたが2003年に去っている。すなわち、マイクロソフトはもはや OpenGL をサポートする意志はなく、DirectX 一本で OpenGL に対抗していこうと考えているのであろう。一方で Khronos はマイクロソフトに対抗して迅速かつ柔軟に OpenGL の仕様決定が行えるような組織として作られたのだろうと推測される。

3.4 Java が XNA に勝てる可能性

Java は Sun が単独でサポートしているのではなく、多くの開発コミュニティと多くの企業によって支えられている。Sun 本体はサーバやワークステーション、開発環境などすべての部門で徐々に弱体化しつつあり、代わりに IBM や Oracle、HP などの巨大企業が、主にマイクロソフトに対抗するために、Java を支持している。Sun が盟主となって IBM や Oracle や HP を率いているというよりは、IBM に主導権を奪われても抵抗できないほど、Sun の影響力が小さくなってきていると言える。IBM などの産業界の巨人に支えられているという意味においては Java も Linux も大きな違いはない。

Java は特定のプラットフォームに依存しないよう設計されている。特定のプラットフォームに依存する API を Java から利用するには、まずその API のための拡張 API を C か C++などで作らなくてはならない。Sun が元気な頃は Sun 自身が基本的な拡張 API をすべて提供しただろう。しかし今では Sun にとって戦略的に特に重要な部分をのぞいて、拡張 API の開発は「開発コミュニティ」という Sun のコントロールが及ばない第三者任せになっている。Java の開発コミュニティとは拡張 API を作ってくれる個人や企業のことで、API ごとに非常にたくさん存在していて、プロジェクトも構想段階で形のないものやβテスト段階のものなど、さまざまである。Java 用の IDE も Sun 以外に IBM や Oracle、Borland、マイクロソフトなどの多くの企業がサポートしている。

たとえて言えば Sun は室町幕府末期の将軍家のような名前だけの飾りになってしまい、IBM や Oracle やマイクロソフトなどの戦国大名がそれぞれの領国を実行支配している。XNA はマイクロソフトという大大名の領国の中だけで通用するローカルなもので、ローカルなだけに効率が良く実用的で整合性は高い。しかも領国が非常に拡大した結果、まるでマイクロソフトが全国を統治しているようにも見える。Sun と IBM の関係は鎌倉幕府時代の将軍と執権職にも似ているかもしれない。

Web サーバやデータベース部門などの商業的利用価値が高い部分は、Sun 自身や IBM などの大企業が開発コミュニティを支援するので、非常に発達している。しかしそうではない分野ではまともな API が存在しなかったり、存在していたがサポートされなくなっていたり、一部のプラットフォームにしか対応していなかったりする。たとえば、Windows 環境でいえば、DirectX や DirectShow や DirectSound などをまともにサポートする拡張 API がない。サウンドの不具合など、昔からある現象もいつまでたっても改善されない。JMF（Java Media Framework）など久しくアップデートされておらず、ビデオカメラ一つまともに制御できない。USB や

RS232C などの非常に基本的な部分にもまともに動く API がない。このように特にマルチメディア部門に使えない API が多くあることは、ゲーム開発やメディアアート制作には致命的であると言える。

一方で Visual C#や XNA はマイクロソフトがまるごとサポートしているものであり、いろいろな企業や開発コミュニティのきまぐれや理想主義には左右されない。マイクロソフトが今日のような隆盛を極めたのは、標準化などの手順を踏まず、一社独占の利を最大限に生かして迅速かつ緻密にサポート態勢を整え、ユーザを囲い込んで来たからであろう。企業コンソーシアムが仲良しサロンと化し、利権や既得権益で保守化することで、技術的な発展が局所解で停滞することはあり得るが、マイクロソフトが常に暴走を続けることによって、そのような怠惰な状況は起こりえず、結果として技術革新を加速し、エンドユーザに恩恵をもたらしたことは事実である。オープンソース運動も、先端技術を有する企業が安穏とすることを許さぬ勢力として、マイクロソフトと同じような役割を果たしている。

マイクロソフト環境のマルチメディア関連部分は Java とはうってかわって手厚くサポートされており、至れり尽くせりである。メディアアート作家がこのような現状を知れば皆 Java を捨てて Visual C#に転向することだろう。ただしメディアアーティストは Windows ユーザではなく、なぜか Mac ユーザであることが多く、Mac 環境では Visual Studio などの開発環境は利用できず、その代わり Java の環境は相対的に非常に良くできている。Mac ユーザにとっては Java や Java から派生した Processing の方が便利かもしれない。しかし、Windows 環境では Java は決して優れた環境ではない。Mac でまともに動く Java プログラムも Windows では動かないことも多い。

Mac や Linux に比べれば Windows のシェアは大きく、Windows 環境の重要性が将来も大きいままならばおそらく Java は勝てない。しかし、Windows 以外の環境、たとえば PS3 + Linux 環境が重要な意味をもってくれば、Java にも勝てる余地がある。Java が PS3環境で勝利するためには、ソニーがゲーム開発における Java の重要性に目覚め、IBM などの巨大企業をいくつも巻き込んでマイクロソフトに対抗する必要がある。しかしソニーは開発環境音痴と言っても良いくらいで、それに対してマイクロソフトは Borland などの多くのライバルを蹴落とてきた世界最強のソフト会社なのである。おそらくソニーには無理なのではないか。

あるいは Java 陣営の IBM がソニーと手を組んで PS3 を救済するか。PS3に搭載される Cell は IBM の技術であり、PS3が好調に推移すれば、Cell のために IBM が OS や開発環境を提供するということも、あり得なくはない。Cell は先進的分散コンピューティング環境であると同時に仮想マシンだとソニーは言っている。つまり、Cell そのものが Java テクノロジーや.NET Framework に相当すると言いたいわけである。ソニーは PS3はどうせ売れるものと高をくくっている。売れるのであればさらに新技術 Cell によって一気に Windows + Intel のパラダイムを覆そうとしている。

ソニーや IBM でなく第三者が、PS3のために Java に基づくクロスプラットフォームでオープンソースな開発環境を作ってくれるだろうか。

言語や API の差は優劣にそれほど大きく影響しない。Java と C#の違いなどは誤差に過ぎない。Java もこれまで仕様変更で大きく変わってきた。OpenGL も OpenGL ES となってまったく違う意味合いをもってきた。Java や OpenGL が変わり、PS3も変わり、ソニーも変わり、ソニーを取り巻く環境が変化すれば、もしかすると XNA に勝てるかもしれない。

3.5 ゲーム専用機と PC

ゲーム専用機はライフスパンが5年間程度と固定されており、独自スペックのハードウェアを使う。

PC は固定したライフスパンはなくハードウェアは連続に進化していく。もちろん、ライフスパンが固定されることと独自スペックであることは独立事象ではない。もしライフスパンがなく、システムが連続に進化していけば、独自スペックのハードウェアはどんなに優れていてもたちまち淘汰されてしまう。独自であることのリスクがきわめて大きい。しかし家庭用ゲーム機のように、ある程度の年月ハードウェアを更新する必要性が無い場合、ハードウェアはしばしば特殊な独自仕様となり、開発環境もまた特殊化し、組み込みシステム化する。システムは汎用的でもデファクトスタンダードでもなくてよい。オルタナティブな世界を築き上げればそれでよい。任天堂の生き残り戦略はこれだろう。しかし PC の世界では常にデファクトスタンダードを維持していかなくては生き残れない。

最高スペックのシステムを組むには PC の方が優れているが、ゲーム専用機が PC よりも世間一般は受け入れやすい。このことはつまりシステムとして進化しているよりも、コストを下げて「ゲーム家電」化し、ゲーム内容も「家庭用」化した方が大量に受け入れやすいということであり、そうでなければとうの昔に人々はみな PC でゲームをするようになっただろう。

事実、もし常に最新の PC ゲームで遊びたければ毎年

GPU と CPU を買い換えなくてはならず、ハードウェア本体に毎年10万円以上の支出をしなくてはならないだろう。結果的にこのような遊び方をするゲーマーはごく一部となり、それ以外のゲーマーは5年間ならば5年間という固定した期間、単一の家庭用ゲーム機で遊ぶことになる。もっとも高額な PS3も減価償却5年と考えれば年で1～2万円の出費にすぎず、PC に比べればずっと割安である。

ゲーム機と PC の立場が逆転するには PC がますます安くなり、あるいは PC がますます便利になって誰もが所有するようになる必要がある。PC は家庭にあって当たり前であって、ゲーム機として特別に家計から支出する必要がなくなれば、ゲーム専用機を買う方が割高となり、ゲーム機は自然と売れなくなるだろう。しかしそのように PC が家電化する時代はくるのだろうか。

もう一つの可能性としては、先に述べたように、ゲームの開発コストがあまりにも高くなりすぎて、固定したライフスパンがあろうとなかろうと、開発システムを PC と同様な標準的な汎用システムにせざるを得なくなるというシナリオが考えられる。あるいはフレームワークの仮想化がさらに進んでゲーム専用機と PC を分け隔てする必要性が将来無くなるかもしれない。つまり PC とゲーム機は統合して家電になることになる。組み込みシステムはあまりに高度化すると汎用化する。携帯電話も PDA も、今は組み込みシステムだが、機能がさらに高度化すれば形態はともかくとして中身は PC と同じになってしまうかもしれない。

かつてマイクロコンピュータと呼ばれたものが、技術の進歩によっていつの間にかワークステーションと同じもの、PC になってしまった。しかし、一方では当時の Z80系の CPU も未だに組み込みシステムの中では生き残っているのであり、技術の進歩によってすべてが汎用化するのではない。ゲーム機が汎用化していくのか、組み込みシステムとしてさらに特殊化し分岐していくのかは予測つきがたい。

3.6 サードパーティとオープンソースコミュニティ

サードパーティとオープンソースコミュニティはまったく異質なものであり、ゲーム業界ではこれまでオープンソースの影響はほとんどみられなかった。Jeremy Reimer は「Cross-platform game development and the next generation of consoles」[4] において次のような予測をしている。

- ゲーム開発コストが高騰するに従ってゲーム販売会社はクロスプラットフォームのゲーム開発を要求するようになる。
- クロスプラットフォームゲーム開発が増加するにつれてハードウェアの性能に依存しない最小公倍数的スペックしか要求しないゲーム開発が行われるようになる。
- その結果、ゲーム専用機の中ではもっとも安価なプラットフォームが利益を得て、高価なプラットフォームが損をする。PC ゲームもまた利益を得る。

彼の説は要するに「ゲーム機は安ければ安いほど良い」と言っているだけであり、確かにニンテンドー DS があれほど売れた理由にはなるだろう。しかし本稿では携帯ゲームについてはこれ以上ふれない。次世代ゲーム機とはここでは Xbox360、Wii、そして PS3である。サードパーティによらない純正品とはマイクロソフト、任天堂、SCE が作るゲームのことである。クロスプラットフォーム開発が可能なサードパーティとはスクウェアエニックス、バンダイナムコ、コナミ、カプコン、コーエー、セガなどの大手ゲーム販売会社のことだが、海外にも Electronic Arts、Activision、Ubisoft、Epic Games、Valve など PC ゲームと家庭用ゲームのクロスプラットフォーム開発・販売が可能な巨大企業がいくつか存在する。

Jeremy Reimer の予測はかつての PS、PS2全盛期のように良質なゲームがサードパーティから大量に供給される時代には正しいかもしれない。もちろんこれからも体力のあるサードパーティは Wii、Xbox360、PS3、PC すべてのプラットフォームの開発環境をそろえ、クロスプラットフォーム環境を自力で開発し、すべてのゲーム機に同じゲームを供給し続けるかもしれない。同じ内容のゲームならば一番安いプラットフォームで遊んだ方が得であるから、みんな Wii を買うだろう。

しかし、任天堂はこれまで純正品で利益を確保してきたメーカーである。今更ただ単にゲーム機本体が安いからという理由だけで、サードパーティが押しかけるだろうか。海外のメーカーには確かにそのような傾向が見られなくもないが日本のメーカーがそんなことをするだろうか。もしそうならすでに Game Cube のときにそうなっていたのではないか。

確実に言えることは、ゲームの開発費はこれからも高騰し続けるということである。これまでゲームの開発費を削減するのに有効だったのは、東欧や旧ソ連の中小デベロッパーにアウトソーシングするという方法であった。日本であれば韓国や東南アジアやインドなどであろうか。チェコやクロアチア、ウクライナ、ルーマニア、ロシアなどのデベロッパーの躍進はめざましく、FPS の世界ではすでに日本を完全に追い抜いている。東欧や旧ソ連圏が資本主義化することによって優秀な人材がゲーム開発産業に流れたこともあるだろうし、西欧諸国のゲーム

発売元が大規模な投資をしたということもあるだろう。例を挙げればVietcongを開発したチェコPterodon、Serious Samを開発したクロアチアのなどである。日本や東アジアにこれほど個性的で元気なゲームデベロッパーが育っているだろうか。日本に健全なベンチャーキャピタルは育っているだろうか。

ゲーム機メーカーはこれまでサードパーティやオープンソースコミュニティの開発環境の整備などということはほとんど考慮してこなかった。任天堂のように純正品に依存する比率が高いところは特にそうである。ソニーはもともと家電メーカーであって組み込みシステムを作ってきた会社である。ソフトウェア開発会社でもなければ言語や開発環境を作る会社ではない。サードパーティを囲い込むために積極的に支援してきた形跡はない。

ソニーや任天堂に比べれば、オープンソースにもっとも近いところに居るのはマイクロソフトである。皮肉なことにソフトウェアの世界ではマイクロソフトはもっともオープンソースから遠い存在だとみなされてきた。同時にオープンソースとの戦いに真っ向から取り組んできたのもマイクロソフトである。マイクロソフトはこれまでの戦いで、オープンソースとはなんであるかを熟知しているだろう。つまりマイクロソフトはその気になりさえすれば、オープンソースコミュニティをうまく操ることができる位置にいるのである。マイクロソフト自身もそろそろそのことに気づいたに違いない。マイクロソフトは2006年11月になって、SUSE Linuxを販売するNovellと提携した。SUSEは3D GUI環境GLXを開発したところである。マイクロソフトは「オープンソース・ソフトと独自ソース・ソフトの分断状況に橋を架けるための努力」を行うという。非常に意味深長な言葉である。マイクロソフトは自社開発した言語と実行環境と開発環境を持っている。オープンソースコミュニティの力を借りて一気に劣勢を挽回しようと考えているのではないか。

マイクロソフトは比較優位なDirectXというすぐれたグラフィクスAPIを持っている。無償ゲーム開発環境を既存の大手サードパーティの頭越しに、大学などの教育機関や中小のデベロッパー、オープンソースコミュニティに提供することによって、一気に体勢を挽回できるのではないかという戦略をマイクロソフトは思い付いたのではないか。その答えがVisual Studio 2005 Express EditionとXNA Game Studio Expressである。

日本ではPCゲームはほとんど注目されない。PCゲームのほとんどはいわゆる洋ゲーである。一方ヨーロッパや北米ではPCゲームの方が主流の地域もある。オンラインゲームに限ればPCゲームの比率はさらに高まる。オンラインゲームはこれからさらに需要が高まるだろう。また、家電やゲーム機がインターネットにつながるよりもさらにPCは普及していくだろう。XNAはPCとXbox360のクロスプラットフォーム開発環境である。XNAはまず北米やヨーロッパのオープンソースコミュニティに急速に受け入れられていく可能性がある。

ゲームをプレイするのはゲーム機であっても開発するのはみなPCである。OSはほとんどの場合Windowsだろう。XNAが普及してオープンソースコミュニティの支援が得られれば開発コストはものすごい勢いで下がるだろう。コストが下がるだけでなくその中からFFやDQに匹敵するキラーアプリが生まれる可能性がある。いったんオープンソース化したソフトウェア業界が元の独占体質に戻ることはあり得ない。XNAをきっかけにゲーム業界自体が一気にオープンソース化する可能性すらあり得る（任天堂は自社開発を貫くかもしれないが)。

ゲームに新しい技術が取り入れられていくのもまたPCゲームである。ValveのSteamやSource Engine、Half-Life2などの例を挙げるまでもあるまい。オープンソースはまた大学などのアカデミズムとの親和性が高い。最先端技術を駆使したゲームがオープンソースゲーム開発環境で次々に作られていけば、旧来のゲーム業界の仕組みは簡単に変わってしまうかもしれない。

3.7 プロトタイプ開発

XNA Game Studio Expressは間口を広げるために日曜プログラマや同人ゲーム制作者や学生のために作られてはいるものの、XNAそのものは必ずしも素人のためのものではない。XNAはどちらかといえばプロのゲーム開発の現場でのプロトタイプ工程を効率化するために作られているといえる。XNA開発の総責任者J Allard氏は「開発者ははじめの4分の3の期間を、開発のやり方やどう進めていくかを考え出すことに費やします。しかしその間アーティストには何の刺激もなく、アーティストはユーザにとって重要であるゲームデザインそのものにまったく集中できません。したがってゲームデザイン自体の期間が短縮され、悪い結果に終わります」と言っている[40]。

ゲームデザインの初期の段階ではじっくり時間をかけてコンセプトを練らなくてはならないが、デザインは紙とペンだけでできるのでなく、なにかプロトタイプを作ってみて、実際にプレイしてみなくてはならない。ここにXNAが使われる。

本格的な開発に先立ち、数人程度のチームが数週間程度でプロトタイプを作る。まずいところがあれば作り直す。納得できるまで何度も作り直す。ここまでがXNAの役割。せいぜい数百万円か数千万円のコストで済むレ

ベルの世界である。それでおもしろいゲームが出来そうだとなればそこで初めて数十億数百億円の投資をする。それだけのお金があれば開発環境がどんなものかというのことあまり意味はない。XNA であってもそうでなくても良いだろう。ゲームの開発費用がたかだか数千万円だった時代には、プロトタイプと本番の違いもなくて済んだ。いくつもゲームを出してそのうち何割か利益がでればそれで良かった。今は数年がかりで数百人規模のスタッフでゲームが開発される。ゲーム開発のリスクも非常に高くなっている。そのリスクをできるだけ回避するためにも、プロトタイプの段階で良い企画良いアイディアを選別しなくてはならず、そのため XNA のようなコンパクトな開発環境が必要になるわけだ。

これまでゲーム開発は素人にはあまりにも敷居が高すぎた。いきなり難解な Visual C++のコードを読まされ、親切とは言い難い DirectX の仕様を理解させられ、さらには仕様変更に悩まされた。XNA はゲーム開発の世界で Visual BASIC のような役割を果たすことが望まれているという。Visual BASIC（あるいは N88-BASIC でも良いが）からプログラミングの世界に入ったひともいるだろうし、業務の現場で今も Visual BASIC を使うひともいるだろうし、Excel などのマクロに VBScript を使うひともいるだろう。

マイクロソフトの開発環境の中では Visual BASIC が一番簡単だったし、今も BASIC から入門する人たちが少なからずいる。マイクロソフトにこだわらなければ Flash でも ActionScript でも JavaScript でもよい。いまやファミコン時代のゲームは Flash でも十分作ることができる。これらの簡易スクリプト系の言語で、Web ブラウザ上のプレイヤーで動作するというとっつきやすさは確かに魅力だ。

しかし XNA や C#との比較対象と成り得るのは Java と OpenGL である。JavaScript や ActionScript ではない。個人の趣味のプログラミングにも使えるが、大人数の大規模開発に利用するには Java や C#でなくてはならない。また XNA は DirectX の機能を十分に引き出せることに意味があるのであって、簡単に絵が描けたり動かせたりすることに意味があるのではない。そういう意味では XNA は Flash や Java Graphics2D API などとはまったく競合しない。

Flash や ActionScript や JavaScript、あるいはノベルゲーの開発ツールなどは確かに XNA よりもしきいは低いだろう。しかしできることは限られている。XNA は残念ながら Flash よりも難しい。Java をアート用途にカスタマイズした Processing よりも難しい。その代わり汎用性や拡張性があって、そうとう複雑なことまでできる。大規模開発のプロトタイピングに最適だと思う。

4. デジタルメディア教育とプログラミング

これまでに縷々として述べてきたように、プログラミングやゲーム開発、あるいはメディアアート制作や Web コンテンツデザインなど、これまではこまごまと分かれてきた多くの分野が、仮想化された単一のフレームワークの中で扱われるようになるだろう。その理由としては、その方が効率的でより少ない予算でより大規模な開発ができて、マルチプラットフォームに適しているからだ。

フレームワークとして今後生き残るのは Visual Studio と Eclipse の二つ、主要な言語は Visual Studio では C# で、Eclipse では Java。ゲーム開発もこの二つのどちらかに収束していくのではないか。ただフレームワークというものがそうなっていくというより、これまでは細分化されていた映像、CG、ゲームなどの分野がデジタルコンテンツという一つの大きな総合分野に統合していくと考えるべきだろう。

フレームワークの中でも、また大学における情報教育の中でも今後もっとも重要性が高いのは Web コンテンツ制作とネットワーク業務であって、言語の進化もやはりまずここから起こり、それがゲーム開発やメディアアート作品制作などのマルチメディアコンテンツ分野に数年遅れで波及する形になるだろう。この傾向はおそらくこれから10年間は変化しないだろう。XNA のようなゲームに特化した開発環境、Processing や ActionScript、Max/MSP のようにメディアアートに特化した言語を適宜使うのはよい。しかしメインの汎用言語は何か、汎用用途は何かということを了解した上でそれらの派生言語や派生用途を利用すべきだ。C や C++や Java や C#などのそれぞれの時代を代表する汎用言語にはそれぞれに意味と価値がある。さまざまな言語の利点を集約して一つのスタンダードを築き、その後に現れる言語に広く影響を与えた言語は習得すべきだ。そうでなくてはいつまでたっても言語というものがわからない。

芸術学部のカリキュラムで、募集から就職までの全課程において、ゲーム教育と Web 教育を切り離して考えることは決して適切であるとは言えない。Web のための知識やスキルは当然ゲームにも活かせるのであり、その逆も成り立つ。多くの高校生はプログラミングの経験がない。あっても高校までに体験しているのは Web 構築か Web コンテンツ制作程度であり、ゲーム制作に適性があるかどうかは未知数である。無謀なことに、高校までにゲームをまったく作ったことの無い人間がゲームプログラマーやゲームクリエーターをめざし、同様に

CGを作ったことがない人間がCGを作ろうとする。そして多くの学生がCGやゲーム開発から脱落し、映像やデザイン、企画等の分野に移っていく。実に不思議な光景であると言わねばならない。

専門学校の短期教育であればゲームあるいはWebなどに特化したカリキュラムは有効であろう。しかし四年制大学、大学院まで含めた六年間の高等教育には必ずしもなじまない。プログラミングの初歩から始めて徐々に専門性に分かれていき、卒業制作の段階で明確にWebやゲームに分かれるという構成をとるべきであろう。

同じことはCGやアニメや映像について言える。高校生は漠然とCGやアニメや映像にあこがれて大学に入ってくるが、高校までに制作の経験があるものはまれである。美術予備校に相当するものもない。従って大学四年間で志望が変わり、ゼミ配属前と後でもやはり変わってしまう。その心変わりの余地を許さない学科構成や学部構成は非効率的にならざるを得ない。もしWebならWeb、ゲームならゲームと決めたコースからリタイアすればただちにドロップアウト＝大学中退とならざるを得ない。それほど潔く腹をくくった学生ばかりではない。むしろ自分のやりたいことがわかってないから安易にCGやゲーム制作を志望するのである。それよりももう少し大きなくくりで、情報学部またはメディア学部のような中で、学年があがるにつれて徐々に専門性を絞り込んでいき、出口では最終的にスペシャリストとなるようなカリキュラムを組むべきではないか。

5. 考察とまとめ

5.1 オープンソースゲーム開発は成立し得るか

本稿の一番大きな関心事は、オープンソースゲーム開発がそもそも成立しうるかどうかということである。そのためのサーベイと考察を行った。

オープンソースとかWeb 2.0など言っていることの本質は個人から巨大企業まで、あらゆる組織が関わり合って世界に一つのオンラインのコミュニティを作り上げることにある。一部のゲーマーやハッカーが、気が狂ったように働いたからどうにかなるものではない。ある種の熱狂というよりは、何かの必然性によって静かに継続的に成長するものだ。個人が参加しても楽しくて役に立ち、大企業にとっても旨味があり、誰にとってもそれぞれにメリットがある、そういうオープンソースコミュニティをゲーム開発でも実現できればよく、その予兆はすでにあると言ってよいと思う。

5.2 どのような形で成立し得るか

次の関心事は、ではオープンソースゲーム開発はどのような形で実現し得るのかということだが、おそらくその鍵を握っているのはIBMかマイクロソフトだろうと思う。

最近、2006年11月になってRedHat Enterprise Linux（RHEL）をOracleがサポートするという話が物議を醸した。オープンソースが機能しない事例として典型的だと思われるので、少々長いがここに紹介する。

RedHatは自社のオープンソースなFedora Core Linuxで得られた最新の成果を有償のRHELに応用するという体勢を取っている。RHELライセンスのサポートには毎年10万円近くかかり、マイクロソフトWindows Serverなどと比べて決して安いわけではない。かつ、現行バージョンの不具合は次期バージョンで解消されるのが普通であり、現行バージョンのまま何年間もサポートされるのではない。RHELユーザはRedHatに毎年ライセンス料を払いつつ、半年か一年ごとにシステムを更新していかなくてはならない。

RHELのユーザである企業にとって、毎年の保守費用はともかくとして、システムの更新と仕様変更についていくのは難しい。仕様が変更するたびに担当のエンジニアは新しいシステムを学習しなくてはならず、また新しいシステムに古いアプリケーションを移植しなくてはならない。オープンソースの不具合は、オープンソースであるがゆえに常に公知のものとなるので、一度オープンソースシステムを導入すれば常に最新の状態にシステムを保ち続けなくてはならない。古いまま残しておくという選択肢がとれなかったのである。

OracleはRHELの古いバージョンもRedHatに代わってサポートしますよ、と言ったのである。逆の言い方をすればRedHatはこれまでRHELの古いバージョンのサポートを非常に短い期間で打ち切っていたのである。RHELと同じことがMacOSXサーバについても言えると筆者は考えている。もともとオープンソースなシステムを有償なシステムに導入した場合、オープンソースであったときと同じように頻繁な保守や仕様変更が伴う。ところが多くの場合、経営者はそのような煩雑な手間ヒマを嫌ってオープンソースではなく企業が丸抱えしてくれるシステムを買っているのである。お金を出したのに相変わらず手間もかかるのではなんのためお金を出したかわからないわけである。

オープンソースは自ら管理ができる人にとっては安価で便利だが、ソリューションとして購入する場合には結局オープンソースでないものと同じくらいに手間もお金もかかるということだ。また適切に管理しなければもちろん、管理していてもオープンソースだからと言って安全なわけでもない。マイクロソフトがこれまで主張して

きた通りである。

オープンソースがうまく行くのは、結局のところ世界最大の消費者である「大衆」と、世界最大の企業であるIBMもしくはマイクロソフトがうまく連携がとれたとき以外あり得ないのである。身も蓋もない言い方だが事実だ。我々は、いずれにしても、巨大資本を巻き込まないことにはオープンソースは成功しないということをここ10年近く見て気づいている。一人の天才の力で一から世界を変えられるわけではない。オープンソースはそのような意味において夢もロマンも少ない。

IBMはオープンソースコミュニティのパトロン役としてすでに成功している。IBMはLinuxをうまく飼い慣らした。IBMはLinuxなどのコミュニティから決しておそれられてもいなければいやがられてもいない。IBMがやっていることはベンチャーキャピタルと同じで、既存の優れたコミュニティに必要な援助をして、自由に活動させた。そしていつの間にかじんわりと主導権を握ってしまう。Javaについても同様である。

マイクロソフトはIBMとは違うアプローチをとろうとしているように見える。クローズドで独占的な技術を無償提供することによって、自分の周りにオープンソースコミュニティを作りだそうとしているのだ。マイクロソフトは次世代のオープンソースコミュニティをうまく操って、XNAのために無料で奉仕するプログラマーたちを育てられるだろうか。

おそらくオープンソースにとってもっとも幸福な未来はPS3上でEclipseをベースに完全な形のオープンソースの開発環境が構築されるというものだろう。あるいはVisual Studioがオープンソースではないがほとんど無償で自由に使えるような世界が到来するかもしれない。D言語など全く新しい言語が普及するかもしれないし、JavaやC#もさらに仕様変更で変わっていくかもしれない。どのような言語であるかということはこの際それほど大きな問題ではない。いずれにせよ、この先マルチプラットフォームではない、クローズドで組み込みシステム的な開発環境が長く生き残るとは予想しにくい。

5.3 現時点での最適解は何か

将来オープンソースゲーム開発が成立し得るとして、現時点で私たちはどのような開発環境を利用し、また将来のためどのような準備をするべきか、これが三番目の関心事である。答えとしては少なくとも数年間はXNAに専念すれば良い。

現時点のXNAはMDXをXbox 360とのクロスプラットフォームに作り替えただけのものであり、PC上で実行するだけであればMDXと大差ない。むしろ、ゲーム以外の要素を切り捨ててしまった分、MDXよりも使いにくくなった部分もある。今回のマイクロソフトの試みが単なる度重なる仕様変更の一つに過ぎない可能性もある。しかしそれらの不安要素を差し引いてもXNAは非常に優れたゲーム開発環境だし、将来の改良の余地もある。Javaの素養があればC#を覚えるのはそれほど難しくはないし、C#を使うのがたとえ数年間であったとしても、覚えておいて損はないと思う。ともかく現時点ではEclipseにせよXNAにせよ複数の環境を平行して利用していけば良いのではないか。

参考文献

1）梅田 望夫著「ウェブ進化論」筑摩書房新書2006
2）Darryl K. Taft 著「Microsoft Won"t Put VS Express in Vista; OEMs May」August 11, 2006
http://www.eweek.com/article2/0, 1895,2002522,00.asp
3）Shawn Hargreaves 著、yomoyomo 訳「オープンソース・ゲームをプレイする」
http://www.yamdas.org/column/ technique/gamej_ind.html
4）Jeremy Reimer: Cross-platform game development and the next generation of consoles, November 07, 2005
http://arstechnica.com/articles/paedia/hardware/crossplatform.ars
5）馬場 章、西川善司、田代昭博：「ゲーム制作に関する最新動向」の特別セミナーレポート、2007.01.24、
http://www.4gamer.net/news.php?url=/news/history/2007.01/20070124224940detail.html
6）後藤弘茂：第2世代Xbox 360とXNA Frameworkの狙い、2007.01.15、
http://pc.watch.impress.co.jp/docs/2007/0115/kaigai329.htm
7）NyaRuRu：Xbox 360で.NETとC#による自作ゲームを動かそう、2007.01.10、
http://www.atmarkit.co.jp/fdotnet/directxworld/ xnaanniversary01/xnaanniversary01_01.html
8）新 清士：ユーザがつくったデータが「ゲーム」を盛り上げる、2006.12.25、
http://www.rbbtoday.com/news/20061225/37198.html
http://headlines.yahoo.co.jp/hl?a=20061225-00000030-rbb-sci
9）西川善司：3Dゲームファンのための「XNA Game Studio Express」講座 手持ちのパソコンでオリジナルのXbox 360ゲームが作れてしまう衝撃、2006.12.19、
http://www.watch.impress.co.jp/game/docs/20061219/3dxna.htm
10）IGDA：マイクロソフト、「XNA（TM）Game Studio Express」最終版の提供開始および「XNA クリエイターズ クラブ」開設を発表
http://www.igda.jp/modules/ news/article.php?storyid=1041
11）マイクロソフト、WindowsとXbox 360で同一のゲーム開発環境を可能にする「XNA Game Studio Express」最終版を提供開始、2006.12.13、
http://www.watch.impress.co.jp/game/docs/20061213/xna.htm
http://journal.mycom.co.jp/news/2006/12/13/462.html
http://www.itmedia.co.jp/news/articles/0612/12/news019.html
061212
http://ascii24.com/news/i/soft/article/2006/12/13/666540-000.html
http://www.famitsu.com/game/news/2006/12/13/103,1166001073,64427,0,0.html
http://www.dengekionline.com/data/news/2006/12/13/4668cd8fc

884a358f28bb537cc7695ed.html
http://itpro.nikkeibp.co.jp/article/NEWS /20061213/256873/
http://japan.cnet.com/news/ent/story/0,2000056022,20338428,00.htm
http://journal.mycom.co.jp/news/2006 /12/12/100.html
http://plusd.itmedia.co.jp/games/articles/0612/13/news074.html
http://japanese.engadget.com/2006/12/12/xna-game-studio-express/
http://japan.cnet.com/news/tech/story/0,2000056025,20338708,00.htm
http://itpro.nikkeibp.co.jp/article/USNEWS/20061212/256614/
http://www.xbox.com/ja-JP/press/release /20061213.htm

12) NyaRuRu：ゲーム開発者にとってのビルド作業とは？ —Content Pipeline を使ってみる、2006.12.06、
http://www.atmarkit.co.jp/fdotnet/directxworld/directxworld05/directxworld05_01.html

13) マイクロソフト、「XNA Game Studio Express Beta 2」を発、2006.11.08、
http://plusd.itmedia.co.jp/games/articles/ 0611/08/news085.html

14) NyaRuRu: XNA Game Studio Express を触ってみよう！、2006.11.01、
http://www.atmarkit.co.jp/fdotnet/directxworld/directxworld04/directxworld04_01.html

15) 自分で作ったゲームが Xbox 360で動く——「2006年 秋のシステム アップデート」、2006.10.30、
http://plusd.itmedia.co.jp/games/articles/0610/30/news083.html

16) Xbox 360™ 向けゲーム開発ツール　国内 4 大学が XNA™ Game Studio Express 採用決定、2006.09.20、
http://www.xbox.com/ja-JP/press/release/20060920-4.htm
http://www.digrajapan.org/modules/news/article.php?storyid=70
http://www.famitsu.com/game/news/2006/09/20/103,1158739907,60350,0,0.html
http://www.gpara.com/article/cms_show.php?c_id=448&c_num=14

17) 米田　聡：Microsoft に訊く、XNA Game Studio Express の狙い、2006.09、
http://www.4gamer.net/news.php?url=/specials/xna_interview/xna_interview_01.shtml

18) 泉水　敬：【CEDEC2006】マイクロソフトが提供するゲームプラットフォームの世界、2006.09.05、
http://www.rbbtoday.com/news/20060905/33655.html
http://www.watch.impress.co.jp/game/docs/20060901/ms.htm

19) マイクロソフト、Xbox 360™ 向けゲーム開発ツールを 一般に初めて開放、2006.08.16、
http://www.xbox.com/ja-JP/press/release/20060816.htm

20) クリス・サッチェル：GDC 06 次世代に向けて動き出す XNA、2006.04.10、
http://www.rbbtoday.com/news/20060410/30174.html

21) Microsoft、GDC で XNA ベースの開発ツール群「XNA Studio」のプレリリースバージョンを配布、2006.03.24、
http://plusd.itmedia.co.jp/games/articles/0603/24/news079.html

22) マイクロソフトの次世代開発環境 XNA Studio の体験版が配布、2006.03.21、
http://www.famitsu.com/game/news/2006/03/21/103, 1142918858,50321,0,0.html

23) マイクロソフト、「XNA™ Framework」を発表し、XNA ツール群のプレリリースバージョンを配布することによりゲームの革新化に寄与、2006.03.22、
http://www.xbox.com/ja-JP/press/release/ 20060322.htm

24) CEDEC2005 - マイクロソフト、Game Developers Day で「XNA」を詳細解説、2005.09.30、
http://journal.mycom.co.jp/articles/2005/09/30/cedec/

25) Xbox 360のキーパースン J Allard 氏に聞く、2005.06.02、
http://pc.watch.impress.co.jp/docs/2005/0602/kaigai184.htm

26) E3 2005 Xbox 360は日本で必ず成功する —J アラード氏インタビュー、2005.05.23、
http://www.rbbtoday.com/news/20050523/22905.html

27) Microsoft 社の開発責任者 J Allard 氏が語る次世代 Xbox、2005.03.21、
http://techon.nikkeibp.co.jp/article/NEWS/20050321/102809/

28) マイクロソフト、次世代 Xbox の詳細の一部を初めて発表～チーフ XNA™ アーキテクトが HD（高解像度）時代のゲームのビジョンを語る～、2005.03.10、
http://www.xbox.com/ja-JP/press/ release/20050310.htm
http://www.watch.impress.co.jp/game/docs/20050310/gdc_xb.htm
http://techon.nikkeibp.co.jp/article/NEWS/20050310/102557/
http://www.watch.impress.co.jp/game/docs/20050314/gdc_xb.htm

29) マイクロソフト、ゲーム開発需要の増大に対応～「XNA Studio」でチームを統合し、ゲーム制作をスピードアップ～、2006.03.09、
http://www.xbox.com/ja-JP/press/release/20050309.htm
http://journal.mycom.co.jp/news/2005/03/08/102.html

30) Microsoft talks Longhorn, XNA, and Xbox 2、2004.09.22、
http://www.gamespot.com/news/2004/09/22/news_6108247.html

31) 平澤寿康、伊藤雅俊：XNA で “Windows にとっての VB” のようなゲーム開発環境を提供　—MS チーフ XNA アーキテクト J アラード来日、2004.07.06、
http://www.rbbtoday.com/news/20040706/17420.html
http://watch.impress.co.jp/game%2Fdocs/20040706/xna.htm

32) 小野憲史：携帯ゲーム機、携帯電話のゲーム市場はこれから～GDC2004に見る潮流、2004.04.02、
http://www.rbbtoday.com/column/report/20040402/page3.html

33) 後藤弘茂：見えてきた Microsoft の次世代 Xbox 戦略、2004.03.31、
http://pc.watch.impress.co.jp/docs/2004/0401/kaigai080.htm
http://pc.watch.impress.co.jp/docs/2004/0331/kaigai079.htm

34) 船津　稔：Microsoft、次世代ゲームプラットフォーム「XNA」発表 Windows と Xbox、モバイル機器も共通化、2004.03.24、
http://watch.impress.co.jp/game%2Fdocs/20040324/xna.htm

35) 後藤弘茂：次世代 Xbox のためのゲーム開発プラットフォーム「XNA」、2004.03.25、
http://pc.watch.impress.co.jp/docs/2004/0325/kaigai078.htm

36) 後藤弘茂：次期ゲーム機のカギを握るのは OS とネットワーク、2004.03.23、
http://pc.watch.impress.co.jp/docs/2004/0323/kaigai076.htm

37) Yoichi Yamashita: GDC 2004 - 次世代 Xbox は新開発プラットフォーム「XNA」から始動、2004.03.25、
http://journal.mycom.co.jp/articles/2004/03/25/gdc/

38) マイクロソフト、ゲーム開発需要の増大に対応～「XNA Studio」でチームを統合し、ゲーム制作をスピードアップ～、2004.03.24、
http://www.xbox.com/ja-JP/press/release/20040324-1.htm

39) J・アラード氏へのインタビュー　HD 時代について、
http://www.xbox-news.com/game/interview/allardint02.html

40) J Allard Microsoft XNA インタビュー、
http://www.xbox-news.com/game/interview/xnaint.html

41) LWJGL: Lightweight Java Game Library, http://lwjgl.org/