少人数・個人開発者向けオンラインゲームフレームワークの提供

～オンラインゲーム開発「３つの壁」に立ち向かう～

藤井大輔　　澤口徹
Daisuke Fujii　Toru Sawaguchi
RED-ZONE SOFT　株式会社

概要

昨今、ネットワーク通信を利用したオンラインゲームが大きく広まりつつある。しかしその開発にあたっては「技術」「資金」「時間」など多面にわたる制約から、大手ベンダーの少数プロジェクトに限られている。本論文ではこれらの問題を解決するフレームワークを提案し、少数・個人による開発の道を開くことを試みる。特にフレームワーク構築にあたって「通信情報量、計算負荷の節約」「ロジックを組む過程の分かりやすさ」の２点に焦点を当てて述べる。

１章：はじめに

オンラインゲームと呼ばれるゲームが世に出て広まり始めている。しかしながら、実際に商用のサービスとして浸透している作品の数は、スタンドアローン（非オンライン）のそれに比べてまだまだ少ない。また少人数・個人単位で制作・運営されているオンラインゲームは皆無に近い。

この事実に筆者は危機感を感じている。オンラインゲーム開発のリスクが高いことが、新しいゲームコンセプトの創造を躊躇させている。オンラインゲーム市場は成長期にあることは誰も疑わないにもかかわらず、あたかも成熟してしまった市場であるかのように「安全な選択肢」に資源が集中して投下されている。　この先、どのようなことが起こるか、オンラインゲーム市場は、その可能性を開花させぬまま、縮小に転じるのではないだろうか。

問題の解決には、新しいゲームコンセプトを創造できる人材を育て、発表の機会を与える環境が必要である。そう筆者は考え、そのために開発のトップレベルにあるベンダーとは真逆の、少人数・個人単位でもオンラインゲームが開発運営可能なフレームワークを開発しようと試みている。

２章：オンラインゲーム開発「３つの壁」

実際に「少人数・個人」がオンラインゲームの開発にあたり、どのような壁に直面しているかを述べる。

２－１．「技術」の壁

ゲーム制作のプログラムに関する技術、その詳細については実はオンラインであるから特に困難が大きいとは筆者は考えていない。一度、自分が出来ることが明らかになれば、あとはモチベーションと相談しつつ、自発的に技術を向上していくことができると考えるからだ。

しかしながら、順調な成長曲線に乗るまでに挫折してしまうことがオンラインゲームにはあると思う。

例えば、一番小さいスタンドアローンゲームプログラムを書くためには、必要な知識として「言語の基本文法」と、「コンパイルや実行の手順」が最低限あれば、いいはずだ。（自分が作りたいゲームを開発するための優れたフレームワークが用意されているならば、それすら知らなくても良い。）

その一方で、一番小さいオンラインゲームを開発するに当たって必要となる知識は遥かに多い。以下、自分が Java 言語でオンラインゲームを開発するにあたって最低限必要と思われる知識を列挙すると「オブジェクトの直列化、入出力 API、サーバ稼動、IP アドレス、ポート番号、同期、マルチスレッド処理」などが挙げら

れる。これら全てがオンラインゲームを制作する前提となるならば、(人にもよるが) ゼロから初めてJavaに関する60時間程度の演習が必要となるだろう。体感で自動車の免許を取得するのと同じくらいと思う。

この初めの取りかかりで多くの人が「挫折」を感じる。これが、1つめの「壁」である。(ソースの記述からコンパイルまでに 10 個の追加手順を踏まなければならないとしたら、あなたは人生最初のプログラムを書く気になっただろうか?)

2－2．「資金」の壁

首尾よくオンラインゲームを制作することが出来たとしても、まだ困難は続く。そのオンラインゲームの稼動にあたって、インターネットに接続したサーバ内にプログラムを走らせて、各個クライアントプログラムと通信させるのはよくある方式だ。

しかしながら、この構成の実現のためには、サーバマシンに多数からのアクセスに耐えうる通信回線と計算処理能力が必要となる。そもそもそのサーバは自分が今所持しているPCに役割を持たせるのか?もう1つサブPCを購入るのか?あるいは市販のレンタルサーバサービスを利用するのか?

いずれにしても、オンラインゲームを作ってみようと一念奮起した学生が、小遣い出してまでサーバ環境を整えようと思うまでには障害がある。

これが2つめの「資金」の壁である。

2－3．「時間」の壁

最後に、「時間」の壁である

なんとか先の2つの壁を乗り越えてオンラインゲームのサービスを稼動することができたとしても問題が残る。それは「保守」に関する問題である。

一般に、公開したゲームについて、そのゲームを面白くするための工夫に苦労を厭わないのが、ゲーム制作者というものである。モチベーションが伴う仕事には疲れが無い。

しかしながら、オンラインゲームにはオンラインゲーム特有の保守作業が必要となる。例えば、サーバセキュリティの確保、ゲームバランスの監視、ユーザ同士のコミュニケーショントラブルなどがある。これら保守のためにかかる作業はモチベーションが沸きにくいわりに、労力を割かねばならない時間が多く、開発者に過度の負担を与える。

この保守に関する作業に対する体力的・心理的負担こそが、3つめ「時間」の壁である。

3章：通信情報量、計算負荷の節約

一般的なサーバ／クライアント型のネットワークゲームを開発する場合に、どこで、どのタイミングで、どのように計算・通信すれば資源を節約しながら実用的なパフォーマンスが実現できるかを述べる。発表では、実際に試作した小さなネットワークゲームについて計算方法を変えながら、実際に節約されている様を見せ、一般的な家庭用回線でも工夫すれば千人程度を相手にしたゲームが実現可能であることを示す。さらに、そのような節約の工夫をフレームワークが透過的に適用する試みについても述べる。

3－1．最も単純なアプローチから

さて、どんな処理がオンラインゲームに求められるのだろうか。単純なモデルから考え始める。筆者はまずスタンドアローンのゲームを複数のPCで同時プレイするところから考え始めた。コンシューマゲーム機(いわゆるファミコン・プレステなど)にコントローラを2つつないで遊ぶ「2Pプレイ」を、遠隔地にいる両人がネットワークを介して遊ぶモデルを考える。

3－2．コントロールスルー・モデル

これを実現するのに最も適したサーバのモデルを考える。両者のキー入力だけを素朴に集計し、素朴に全員に返すモデルを考える、このサーバモデルをその特徴から「コントロールスルー・モデル」と呼ぶことにする。

このモデルの優れている点は、情報通信量・サーバの計算付加ともに軽量であることだ。コントローラに一般で使われている 16 ボタンのゲームパッドを使用したとして、人間がコントローラを通じてサーバに送信することの出来る情報量は多くても秒間1KBを越えない。

また、サーバの計算負担も送信されたキー情報を両ユーザにそのままスルーし返すだけでい

いので、計算量も微々たるもの。またゲームロジックを全く含んでいないため、サーバの実装も非常に楽である。

しかしながらこのモデルは一見上手くいきそうで重大な問題を抱えている。両クライアントの状態が各々どうなっているかの情報を全くサーバが知らないために、両者の間で起こっている現象が同じであることを保証しない。オンラインゲームでは個々のゲーム状態が異なるのでそれはうまくいかない。

３－３．ザ・サーバワールド・モデル

逆に別のサーバモデルを考える。ほぼ全てのゲームロジックをサーバに持たせ、クライアントはキー情報をサーバに送信し、描画情報をサーバから受信するモデルだ。これには「ザ・サーバワールド・モデル」という名前を付ける。

このモデルでは、サーバが、ゲームに必要な情報の全てを知っている。それゆえ先ほどの「コントロールスルー・モデル」でおこる同期の問題は解決する。

しかしながら、このモデルでは、クライアントで処理されていた計算のほとんどをサーバに任せるため、処理が大きくなることが容易に予想される。また、描画情報はキー入力と違ってデータ量が大きい。ラスタ形式を送るような素朴な方式ならば通信量が膨大になってしまうので、どこの画像をどこに表示するといった形式で送信するなどの工夫が必要となる。

３－４．サーバモデルの最適化

さて、この両極端な２モデルには、それぞれに長所と短所がある。どちらもそのままでは商用のサービスとして耐えうる品質は持ち得ない。一般に「コントロールスルー・モデル」のように「重いクライアント」と「必要最低限の軽いサーバ」でサービスを実現する方式を「クライアント主導方式」、その逆に「重いサーバ」と「必要最低限の軽いクライアント」を用いるアプローチを「サーバ主導方式」という。両モデルの特徴を捉え、ゲームロジックを加味した上でオンラインゲームのプログラムを筋良く書くためには卓越した設計能力・バランス感覚が求められる。いわば達人・匠の技術だ。

なんとかこの、達人のパターンを再利用できないだろうか。あるいは、サーバモデルの最適化を自動化できないだろうか。今回は「ザ・サーバワールド・モデル」を元に簡単なアクションゲームを実装し、サーバ主導方式から徐々に処理をクライアントに移行するアプローチを持ってサーバモデルの最適化をまず行う。

その後、同じゲームロジックの処理をクライアント・サーバの双方どちらでも状況に応じてリアルタイムに変更できる仕組みを作り。サイバの負荷状況や通信回線の使用状況によって、サーバ・クライアントのモデルをやはりリアルタイムに最適化する仕組みを作る。

作る試作モデルは以下の３通り。これらの処理についてパフォーマンスの検証などを行い比較検討する。（結果は講演にて発表予定）

＜試作モデル１（素朴モデル）＞

簡単なアクションゲームを作ることが可能なフレームワークを制作。キー入力と、グラフィック出力命令を通信可能なオブジェクトとして定義する、このオブジェクトにクライアントPC を特定する情報を付加してゲームロジックの存在するサーバに転送する。
これによりスタンドアローンのゲームを、ネットワークゲームにすることができる。（「ザ・サーバワールド・モデル」による実装）

＜試作モデル２（手動最適化モデル）＞

キー入力と、グラフィック出力とは別に、ゲームロジックの中間に出現するゲームロジックを通信データとして用い、クライアントへの処理移行を果たす。

＜試作モデル３（自動最適化モデル）＞

キー入力・グラフィック出力方式、中間データ形式、これら２種の方式をリアルタイムに切り替え可能とする仕組みを作り。これらをサーバ・クライアント双方の付加などを加味して適正な通信方式を判断させる。

４章：ロジックを組む過程の分かりやすさ

ゲームロジックとはゲームのルール体系のことであり、変数と関数の組み合わせで表現することが出来る。その点ではゲームロジックはプログラムとよく似ている。しかし、制作者の考え

るゲームロジックに比べて、大抵ソースコードの方がより多く書かなければならない。しかし、全てのゲーム開発者が、そのソースコードの全てを書く必要は無い。例えば複数 PC 間での同期など、初心者には手の余る機能は抽象化して提供してあげる方が望ましい。それだけでなく、ゲーム開発者の最も関心とこだわりがあるのは画面に表示されるモノとその動きである。「口と紙で説明した方が早い」ことを手軽にコード化するためのサービスがあれば、開発者はロジック自身の制作に集中することが出来る。以下、ゲームロジックの定義のために用意した機能を列挙する。

＜物体エディタ＞
- ゲームに登場する「物体」一般を定義することができる。物体の外見・動きなどを定義でき、その作業を GUI でサポートする。
- スクリーンに登場する物体の外見・動きを定義するソフトはアニメーション制作用のもの（例えば FLASH など）などにも存在するが、今フレームワークではゲームに特化した特徴ある動き（自機を追尾する敵機の動きなど）をプリセットとして用意することで差別化を図る。

＜システムオブジェクトエディタ＞
- ゲームで良くあるシステム用オブジェクトが簡単に制作できるようになっている。システム用オブジェクトとは、例えばシューティングゲームで言えば、ゲームの得点を表すための数値カウンター、自機の体力を表すためのゲージバー、自機の残りを個数として表すマーカー、などのことである。
- これらのオブジェクトの作り方はパターン化されているため、
- これらのシステム用オブジェクトはゲームロジック内で用意されている数値をゲームプレイヤーのために知らしめるためのものである。

＜グラフィックレイヤー＞
- 描画レイヤー機能を備えている。描画レイヤー機能とは、2D で描かれた画像の表示に際して奥行きの概念を擬似的に実現するものである。
- レイヤー層は 4 種の属性を持っている（最背面層・背景層・キャラクタ層・システム層）、それぞれ背景・キャラクター・システムグラフィックスを描画するのに特化した機能を持つ。例えば、キャラクタ層は、自機がスクリーン内に表示されるようその動きを追尾するカメラワーク機能を有している。

＜サーバモデル最適化との連動＞
- 物体のアニメーションパターンは、描画用の中間データ形式として機能する。アニメーションパターンの変更時にのみ通信で命令を送ることにすれば、「ザ・サーバワールド・モデル」と比較して大きく通信量を節約できる。

＜お手軽チャットウィンドウ＞
- クライアント同士が短いメッセージを送りあうチャットシステムは、オンラインゲームでもよくある実装で、フレームワークに対しても要求が大きいと見込まれている。
- チャットシステムをパフォーマンスよく装するためには、メッセージの入力情報を適正に加工する必要がある。例えば、編集中の文書は送信する必要が無い。
- 現状のモデルではキー情報は全てサーバに送信されることとなっているが、今回、チャットシステムに関しては、文字の編集中はキー情報を送信しないという例外を設けることにより実装をする。
- チャットの例のほかにも、他のプレイヤーに影響を与えないことが確保されている場合はそのデータをクライアントで一時保持し、送信しなくてもいいはずであり、さらなる研究の余地がある。

５章：結論

フレームワークの提供により、少人数・個人によるオンラインゲーム開発の環境を整えることが出来れば、新しいゲームコンセプトの創出につながり、オンラインゲーム市場の再活性化が見込まれる。