プログラミング教育と３Ｄコンピュータゲーム開発

Programming Education and Development of 3D-Computer Games

玉真 昭男*, 小松 隆$, 青木 悠#

Teruo TAMAMA, Takashi KOMATSU and Yu AOKI

Abstract: In the programming education, what is important are not only mastering the programming language grammar and solving many exercises, but also development experience of a program over 3000 lines. 3D-games are good targets for that purpose. This year, two students developed a shooting game and an F1-racing game. Each of them developed an over-10,000 line program and found how difficult a 3D-game development was. They also knew the necessity of knowledge in mathematics and physics, which proved that game development is very good target for the programming education as well as basic subjects education.

１．はじめに

情報処理学会は、情報処理の優れた題材であるゲームを対象とした研究を世界の中心となって押し進めることが日本の情報処理技術の発展に貢献するとの考えから、ゲームに関連した研究領域を新たに「ゲーム情報学」と名付け、1999 年に「ゲーム情報学研究会」を発足させた。その設立目的の中でも述べられているように、ゲームはルールが明確で評価し易い、それ自体が興味深い、制作目標が達せられる度に、更に拡張しようという次の目標を自然に思いつく、などの特徴を有しているため、情報処理やプログラミングの例題として非常に適している。外国では、特にチェスを人工知能分野での探索、データベース、機械学習、専用アーキテクチャなどの研究材料として、またパズル全般をアルゴリズムの研究開発題材として、大いに活用してきた。

しかし、「日本では、ゲームやパズルを扱った研究は少なく、情報処理において世界に遅れをとる一要因ともなっていた。」という。

「ゲーム情報学研究会」発足以来、将棋、囲碁、チェス、あるいは Amazons など、主にボードゲームのアルゴリズムに関する研究が中心であるが、日本でもゲームの研究が盛んになってきた。一方、レースゲームやシューティングゲームなどのアクションゲームに関しては、研究材料として取り上げた例は少ない。機能的にも、ビジュアル的にも、優れた市販ゲームが多く出回っているため、研究として取り上げ難いというのが理由であろうが、シューティングゲーム分野で敵機の動作に知性を持たせることで、新たなゲーム性を追求している研究[1)]などもある。

しかし、今の子供達にとって、最も身近で誰もが一度は試したことのある遊びは、将棋・囲碁ではなく、圧倒的に「コンピュータゲーム」であろう。大学の情報処理系学科に入学してくる学生にも、このような体験からコンピュータゲームを作ってみたいと希望する学生は多い。ゲーム作りは、出来栄えを自分で評価できる、何か１つ作るとアイデアが次々に湧いてもっと作りたくなる、更に高度な機能を作りこみたくなる、といった自己拡張性があり、完成したときの達成感も大きいので、アルゴリズム考案やプログラミングといった「知的もの造り」教育の題材として非常に優れている。

将来、プログラマーやＳＥ（システムエンジニア）を目指す学生には、プログラミング言語の文法を理解し、多くの演習問題を解くだけでは不十分で、卒業研究などで、例えば 3000 行以上の大規模プログラミング開発の体

2007 年 3 月 6 日受理

* 理工学部　情報システム学科

$ 現　デンソーテクノ株式会社

現　SSB ソリューション株式会社

験が必要である。その課題として、出来栄えを自分で評価でき、アイデアが次々に湧いてもっと作りたくなる「ゲーム」は格好の題材である。

では、大学でどんなゲーム作りを目指すかであるが、ゲーム情報学研究会はまたその「目的」の中で、次のように示唆している。

「ゲームの強いプログラムを目指すだけでなく、コンピュータを用いてゲーム自体の歴史的あるいは社会的な研究を行なうのも興味深いことです。たとえば、『どういうゲームが世間に受けるのか』をコンピュータを用いて分析することは応用の側面からも重要と考えられます。」

主な研究分野として14分野を挙げているが、そこで我々はその中の次の３つを目指すことにした。

・ゲームプレイングプログラム
・インターネット上のゲーム
・コンピュータを用いた新しいゲーム開発

大学でゲームを開発するもう一つの意義は、大学祭、オープンキャンパスなどのイベントで、展示の目玉として宣伝に使えることである。子供、小中高校生、大学生から社会人まで、長く居座ってゲームを楽しむ姿が見受けられ、最も人気のある会場の一つとなる。特に高校生は、大学進学後自分でもこのようなものを作ってみたいと思うからであろうが、全てのゲームを試した上で、どうやって作るのか、作成にはどれ位時間が掛かるのかなど、熱心に質問する人が多い。

このような背景から、Windows用のC/C++コンパイラであるVisual C++.NET[TR]と３Ｄグラフィック・ライブラリDirectX[TR]を駆使して、３Ｄゲーム作りを行ってきた。我々は、アルゴリズムよりはゲームシステムそのものの実現と三次元の映像表現技術に重点を置き，シューティングゲーム、レースゲーム、ロールプレイングゲームなど、これまでに約10種類の３Ｄゲームを開発してきた。

今回、シューティングゲームとレースゲームの２つのジャンルで、新規性も含めて一定の成果を上げることが出来たので報告する。特にＦ１レースゲームに於いては、物理効果の導入、フォースフィードバックの掛かる操縦席とのドッキングにより、Ｆ１カーの加速性能を体験できるレーシングシミュレータを開発することが出来た[2)]。

２．使用したソフトウェア

開発環境としてWindows用のC/C++コンパイラVisual C++.NET[TR] 2003、３Ｄゲーム開発用ライブラリとしてDirectX[TR] 9.0c、モデリングソフトとしてMetasequoia[TR] Ver2.4.0を使用した。DirectXは、Windows用のゲームを開発するために必要な、高速グラフィックス描画処理、３Ｄ演算、サウンド・ミュージックの再生、ネットワーク通信機能などをまとめたコンポーネントである[3)]。

３．　シューティングゲーム

３．１　ゲームのジャンル

シューティングゲームにもいろいろな種類があり、自機や画面がどのように動くかで「固定画面シューティング」、「縦スクロールシューティング」、「横スクロールシューティング」などと分類されている。これらのゲームはいずれも実質的には２Ｄで表現されている。完全3D表現によって作られているゲームには「フライトシミュレータ」や「フライト・シューティング」というものがあるが、いずれも奥行きや空間が把握しにくく、誰もが簡単に遊べるレベルのゲームではない。

画像は３Ｄでありながら、２Ｄのような操作感覚で簡単に遊ぶことのできるゲーム性を持ったものに「奥スクロールシューティング」と言う分類が存在する。これは機体を画面中央に半固定し、決まったルートを走行しつつ、その先々で出てくる敵機を破壊して行くというゲームである。プレーヤは奥行きなどを気にすることなく敵を撃つことができ、さらに３Ｄグラフィックスによる奥行き表現で全体の見栄えを確保することができる。

今回、シューティングゲームとレースゲームをもとにそれらを結合させることで新しいゲーム創りに挑戦した。そのジャンルの名称をここでは「レース・フライト・シューティングゲーム」と呼ぶことにする。これはレースゲームの「走る」要素とフライト・シューティングの「撃つ」要素を取り入れたものである。ゲーム内容は、決められたコースの中を自由飛行し、規定時間内にどれだけ多くの敵を攻撃・破壊できたか、コースをどれだけ周回できたかを競うものである。シューティングゲームの「敵を撃って倒すこと」とレースゲームの「コースを高速で

正確に走ること」のいずれの要素も取り入れることに成功した。「奥スクロールシューティング」が決まったコース上を走るのに比べ、コース内を自由に飛行できる点が違っている。コース周辺の壁に激突しないように自機を高速に操縦しつつ敵を攻撃する、という二重の難しさとスリルのあるゲームである。

３．２　モデリング

前述したモデリングソフト Metasequoia TR Ver2.4.0 を用いて自機のモデリングを行い、これにテクスチャを UV マッピングし、自機として Fig. 3.1 を完成させた。UV マッピングとはテクスチャをポリゴン毎に座標を指定して貼ることのできる技術で、これを使用することでより精密なモデリングを行うことができる。モデリングソフトを用いて好きなキャラクタを作る楽しさも体験できる。完成したモデルは、Metasequoia のエクスポート機能を使い、X ファイル形式(.x)で出力する。DirectX は D3DXLoadMeshFromX 関数を使って、その X ファイルを３Ｄ画面内に描画する。

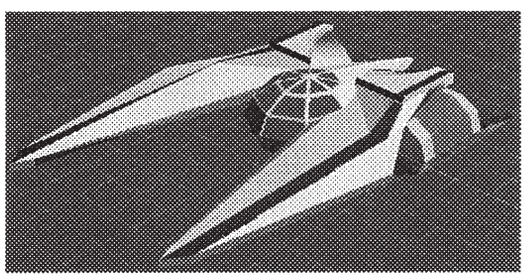

Fig. 3.1　完成した自機のモデル

３．３　当たり判定

シューティングゲームのプログラミングで最もやっかいなものが「当たり判定」である。これには弾丸と敵機、または自機との当たり判定、自機とコース周辺との衝突判定が含まれる。弾丸、敵機、自機などのキャラクタをポリゴンの組合せで作っている場合は、一方のポリゴンの各頂点と他方のポリゴン面の重なり判定を数学的に行えば良い。しかし、複雑な形状になるとポリゴン数が増加するため、関係する全てのポリゴン同士の重なり判定計算にしてしまっては処理が重くなるなどの問題があった。

一方、3Ｄゲーム開発用のライブラリである DirectX は X ファイルを用いたキャラクタ同士の衝突判定に便利な処理方法やそのための関数を用意している。バウンディングスフィア（Bounding Sphere）やバウンディングボックス（Bounding Box）[4]と呼ばれるものがそれである。前者は球体どうしの当たり判定であり、後者は直方体の箱（Box）で当たり判定を処理する。ボックスの生成方法は D3DXComputeBoundingBox 関数を使いオブジェクトの左下隅と右上隅の頂点データを取得する。これを対角線として直方体を生成する。バウンディングスフィアに比べ直方体の当たり判定なので球体のようなモデル形状との著しいアンマッチは減らすことができる。バウンディングボックスの当たり判定には幾つかの方法があるが、ここでは OBB（Oriented Bounding Box）法を用いた。

OBB 法は自由に回転可能なバウンディングボックスのことで、オブジェクトの角度が変化してもバウンディングボックスは変化せず、モデルを追いかけるように角度を変えて、バウンディングボックスが移動する。処理としては重くなるが、オブジェクトとのフィット率が高いため、不自然な当たり判定を避けることが出来る。

３．４　完成したシューティングゲーム

ゲームの特徴は以下のとおりである。

- シューティングゲームとレースゲームの融合
- コースの自由飛行
- 複数の敵の登場と複雑な動き
- 時間制限性

敵の挙動に工夫を凝らし滑らかで変化のある動きをさせている。スプライン曲線に沿った飛行をさせることで半自動的に滑らかな動きを作り出すことに成功した。

本ゲーム制作で苦労した点は、立体的にコース内を自由飛行させることで、コースをはみ出さずにいかにして飛行させるかという点に最も時間が掛かった。コース壁面への衝突を判定・表示するために「レイ」という直線とポリゴンの交差を利用する処理をすることで解決することができた。本ゲーム遊戯中の一画面を Fig. 3.2 に示す。

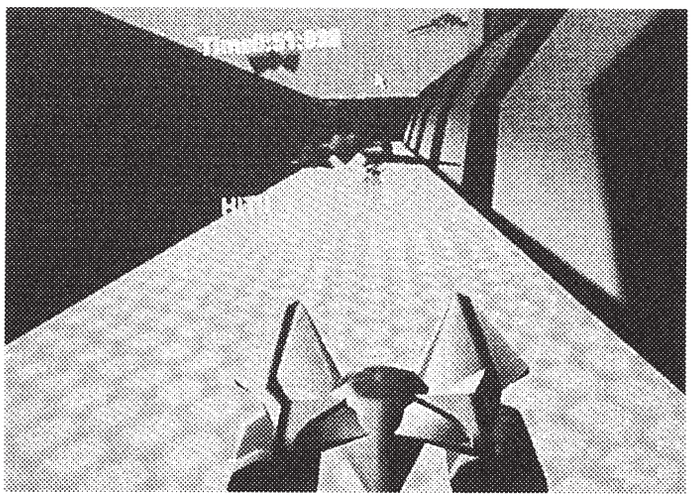

Fig. 3.2 ゲーム遊戯中の一画面

４．Ｆ１レースゲーム

モーターカーレースＦ１の迫力を体感できる３Ｄレーシングシミュレータの実現を目指して研究を進めて来た。物理モデル[5]を取り入れることによって、ゲームに登場する 3D オブジェクトに物理法則に従った運動や挙動をさせることが可能となる。また、ゲームと連動させて椅子の角度を変化させることができる専用操縦席を利用し、操縦席の傾きによってプレーヤにＦ１の加速度を擬似体感させることを目標とした[2]。

４．１ ゲームプログラミング

ゲームプログラムの全体を示すために、ゲームプログラムに含まれる主要クラスのいくつかを UML クラス図で表す。この線図では３つの規約を使う。１番目は、四角の中の名前でクラスを表す。２番目は、ダイヤの付いた線で複合クラス関係を表す。３番目は、複合線の終わりに星をつけて、「所有者」クラスがもう一方のクラスのインスタンスを２つ以上持てることを示す。本ゲームプログラムに含まれる主要クラスをこのクラス図で表すと Fig. 4.1 のようになる。中心となるクラスは CStageMain、CStageOP、CStageED の各ステージクラスである。ステージクラスが CObj3d や CSprite、CSound などのインスタンスを持ちそれぞれを動作させることによって各ステージの振る舞いが決まる。そして各ステージのインスタンスは Main が持ち、ゲームの進行にあわせて Main プログラムの中でステージを切り替える。

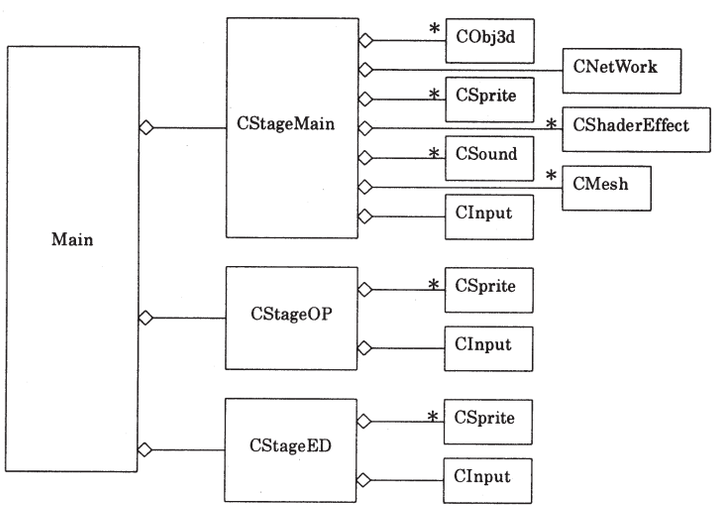

Fig. 4.1 作成したプログラムのクラス図

４．１．１　CObj3d クラス

３Ｄゲームでは車や建物、動物といった３Ｄのオブジェクトが複数登場する。また、それらをフィールド上で動かすことも必要になってくる。３Ｄの物体は基本的に位置、角度、大きさの３つの変化があるが、プログラムを書く上で３Ｄの物体にこれらの振る舞いをさせるクラスが必要になり CObj3d クラスを作成した。位置、角度、大きさをセットしたり、それらの状態を取得したりするミューテータとアクセッサ関数で構成している。自分自身の描画も CObj3d クラスで行う。

また、この CObj3d クラスはゲーム中に出てくるいろいろな３Ｄのオブジェクトに対してそれぞれのクラスを派生させる目的で作成した。CObj3dCar クラスは CObj3d クラスを継承し、車両を加速させたりストップさせたりするアクセル関数やブレーキ関数などを付け加えた。

４．１．２　CStage クラス

CStage クラスはゲームプログラムの中で中心となるクラスであり、ゲームのオープニングやエンディング、メインのゲームステージを作成するための基底クラスである。CStage クラスから CStageOP、CStageMain、CStageED などのクラスを派生させている。

CStageOP クラスや CStageMain クラス、CStageED クラスは CStage クラスを継承しているため、それぞれ Draw、Move 関数を持ちそれぞれのクラスによって違うバージョンの関数を持つことになるが、ここでポリモーフィズムのテクニックが使える。

４．２　状態遷移

ゲームプログラミングをしていると、オープニングやエンディング、選択画面、メインゲームステージなどいろいろなゲームステージの段階が存在する。ゲームをしていて常にオープニングから始まり、選択画面へ移りメインゲームステージ、エンディングという流れでゲームが進行すれば問題ないが、途中でオープニング画面に戻ったり、選択画面に飛んだりすることが多々ある。そういった時にいちいち if 文で場合分けをしていると分かり難くなり、新しいステージを間に挿入したくなった時に修正がきかなくなってしまうという問題がある。そこで状態遷移プログラムを取り入れた。状態遷移関数は以下のような型で表現する。

```
typedef int(*STFuncPtr)(void *);
```

状態遷移関数の戻り値は整数になっているが、これが状態遷移図の相関を表すパラメータである。

４．３　物理モデル

ゲームに登場するキャラクタや背景を３Ｄグラフィックスでリアルに表現できたとすると、次はその動きのリアルさが問題になる。ゲームに出てくる「物」が物理法則にしたがっていなければ、ボール１つ投げるのにも不自然な表現になってしまう。物理を理解していなくても、ボールの動きを観察して何となく真似ることはできるかもしれないが、インタラクティブ性が要求されるゲームの世界ではプレーヤの操作に合わせて動きを変えなければならないため、形ばかりの真似では追いつくことができない。今日のようにコンピュータの性能や３D ライブラリなどの条件が整ってくると「現実感」を表現するためにいかに物理効果を表現できるかが重要になってくる。

４．３．１　物理エンジン[5)]

物理エンジンといえばゲーム業界では”Havok”が有名だが、最近では AGEIA 社の”PhysX”がクローズアップされてきている。それは”PhysX”の物理エンジンに対する考え方がかなり次世代であることから来ている。また、PhysX API は非商用であれば無料で使用することができるため PhysX を利用して研究を進めることにした。

４．３．２　PhysX について

ここで言う「物理」は「ニュートン物理」をベースとした運動「物理」を指す。ゲームにおける「物理」エンジンといった場合も、一般的には 3D オブジェクトの運動/挙動を司るものということになっている。Havok も AGEIA もこの運動物理のエンジンメーカーである。

現在、AGEIA では、同社の Web サイトにて、フリー版の SDK やデモソフトなどをアップロードしている。一般ユーザーでも入手可能で、SDK を利用して開発を行うことができる。

PhysX でサポートされる物理演算は剛体物理、有限要素解

析、軟体物理、流体物理、毛髪シミュレーション、布シミュレーションなど、多岐に渡る。

4．3．3　PhysXの使用方法

PhysXでは、物理モデルを適用するオブジェクトのことをアクター（Actor）と呼ぶ。アクターには様々なステータスを割り当てることができ、様々な物体、場面を表現することができる。他にも、アクター同士をつなげるジョイント（Joint）がある。ジョイントにも様々なステータスを割り当てられ、壊れるジョイントなどさまざまな場面に対応できる。

PhysXのもうひとつ重要な概念がシーン（Scene）である。シーンとは、世界にどのような決まり（重力など）を与えるかを決めるものである。例えば重力が無い宇宙と、重力がある地球上は別のシーンであるといえる。PhysXでは、必ず1つシーンを作成しなければならない。

4．3．4　自動車型のアクター作成

F1カーの動作を再現するにはF1カー型のアクターを作成する必要がる。しかしそのようなアクターを作るのは非常に複雑で困難であるため、車両全般で使用できるFig.4.2のような車両型アクターを作成した。ボディの大きさやタイヤの直径は3Dモデルの大きさに合わせる必要がある。タイヤとボディはジョイントでつながっており、タイヤにたいしてトルクを加えることによって車両型のアクターを動かすことができる。これによりFig. 4.3のように、壁との衝突時に車体の衝突応答を表現したり、タイヤの地面に設置している部分の静止摩擦係数や動摩擦係数を変化させることによって車の横滑りなどの表現が可能になる。

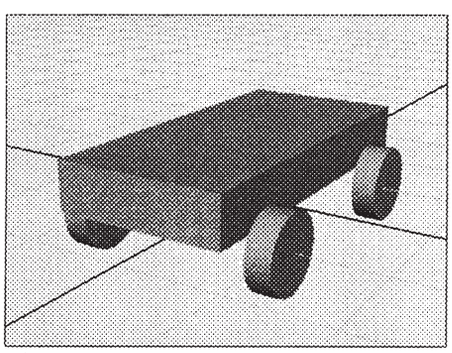

Fig. 4.2　車両型アクター

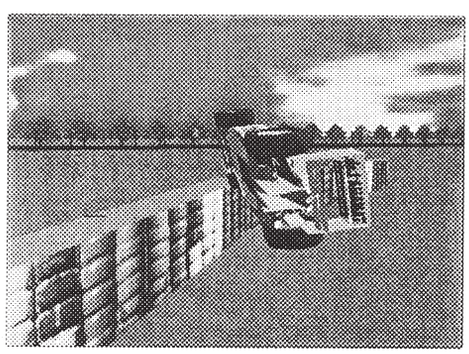

Fig. 4.3　ゲーム中で壁との衝突時の画像

4．4　専用操縦席

専用操縦席の土台はフライトシミュレータ用モーションベース（日本BTA, FSチェアー[6]）で、ロール方向とピッチ方向の2軸で角度を変えることができる椅子である。本学と共同研究を行っている静岡文化芸術大学の宮田圭介教授はこのFSチェアーをベースに機能的な車の操縦席を開発中である[7]。今回その操縦席を借用してドッキングし、椅子のロール角、ピッチ角を変化させることによりプレーヤがF1の加速度を体感できるのではないかと考えた。傾斜時の運転者に加わる重力加速度を用いて並進加速度を模擬するように設計した。

4．4．1　フォースフィードバック機能

F1カーの加速度を本操縦席で体感できるようにするためにまず、レーシングゲーム中のF1カーが実際のF1カーと同じような加速をするようにプログラムすることから始めた。あるF1チームWEBサイト[8]に、F1カーに関するデータが掲載されているが、それによると実在するあるF1カーの加速は、静止時から100km/hまで加速するのに3.7秒かかり、100km/hから200km/hまで加速するのに1.5秒かかるということが分かった。このデータを基にFig. 4.4のような加速度曲線（折れ線近似）を作成し、ゲーム中のF1カーが作成した加速度曲線と同じように加速していくようプログラム側で調整をした。

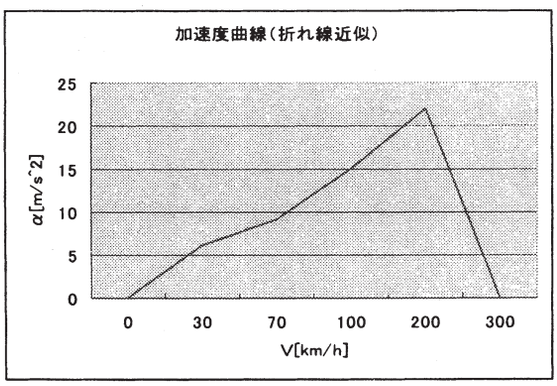

Fig. 4.4 加速度曲線（折れ線近似）

そして、Table 4.1 に示すように、それぞれの加速時、ブレーキ時の操縦席のピッチ角を設定した。またF1マシンにかかる横方向の G と操縦席のロール角を対応させてプレーヤがF1の横方向のGを体感できるようにした。そのために、F1 マシンを操縦している人間に働く遠心力を計算する必要があるが、実際のF1マシンのコーナリング時に操縦者に働く遠心力は次の式で表される。

$$F = m \times \frac{V^2}{R} \qquad (8.1)$$

m ： 運転者の質量， V ： 車速， R ： 旋回半径

簡単のため、コーナー部では等速円運動になると近似した。この式を使ってコーナリング時にF1マシンを操縦している人間にかかる横方向の G をプログラム側で計算するようにした。そしてF1マシンの操縦者に働く横方向のGに合わせて操縦席のロール角を設定した。Table 4.2 に横Gと操縦席の傾斜角との対応を示す。

Table 4.1 ピッチ傾斜角とF1カー速度との対応

速度領域 [km/h]	<100	100～200	200～300
加速時(後)	5°	10°	5°
ブレーキ時(前)	5°	7°	10°

Table 4.2 F1カー加速度とロール傾斜角と対応

加速度	<1G	1G～2G	2G～3G	≦3G
角度	3°	7°	10°	12°

4．4．2 評価

Fig. 4.5 にゲーム操縦風景、Fig. 4.6 ゲーム画面を示す。椅子の前後の動きに関しては、擬似的ではあるがうまく再現することができた。

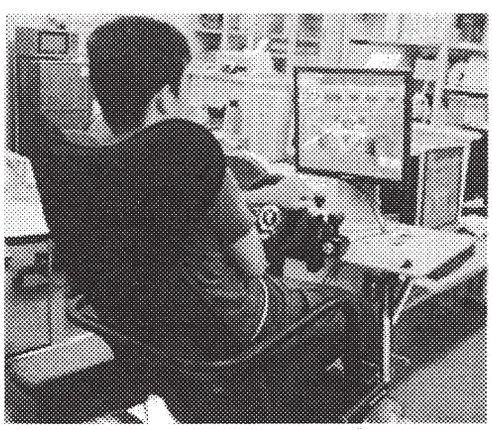

Fig. 4.5 ゲーム操縦風景

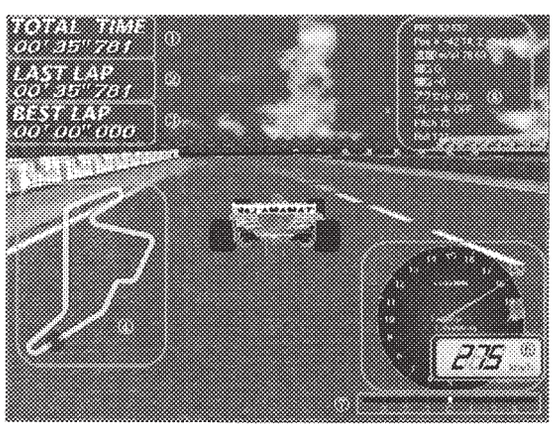

Fig. 4.6 ゲーム画面

5．考察

３Ｄ空間でのレースゲームとシューティングゲームを結合した新しいジャンルのゲームを作成することが出来た。また、物理モデルとフォースフィードバック付操縦席導入によって迫力のあるＦ１レーシングシミュレータを開発することができた。それぞれのプログラム規模、すなわちモデリングで作られたＸファイルの個数・サイズ、プログラム（ソースコード）行数をTable 5.1 に示す。いずれも１万行以上の巨大プログラムである。

前者はまだ出来て間もないが、後者はすでにオープンキャンパス、大学祭、産学官連携フォーラム等のイベントでデモに供され、安定動作が確認されている。いずれも、開発者達は所期の目標をほぼ達成したとして、満足感を持って研究を終えている。１万行以上の動作するプログラムを開発したことで、プログラミング教育の最終段階での目的を十分に達成したと言えるであろう。

Table 5.1 開発したプログラム規模

	Xファイル		
プログラム名	個数	総ファイルサイズ(MB)	ソースコード行数
シューティングゲーム	42	1.67	10,127
F1レーシングシミュレータ	20	6.17	12,938

6．まとめ

本学に於けるプログラミング教育の最終段階として、卒研生１人と院生１人に、Visual C++.NET 2003 と DirectX 9.0c を使って、３Ｄシューティングゲームとレースゲームを開発させた。それぞれに以下の点で新規性を発揮することも出来た。

（1）新しいジャンルでのゲーム創り

（2）物理モデルとフォースフィードバック付操縦席導入により、0～300km/h までの加速性能の模擬体験が出来るＦ１シミュレータの開発

開発者達は、本研究を通してゲーム開発がどれだけ大変なのかを理解することが出来た。プログラミング技術だけではなく、3D モデリング技術や数学・物理の知識など、多方面にわたる技術や知識が必要であることを体験し、自らの力で会得する努力も出来た。ゲームという「もの」造りを行わせることによって、本学が目指す「ものから入り、のちに基礎知識の必要性を自覚させる」教育の成果を十分に達成したと言うことが出来よう。

参考文献

1) 川野洋：シューティングゲームの敵機動作および攻撃弾発射アルゴリズムに関する考察,第 16 回ゲーム情報学研究会, 9, 2006/06/30.

2) 小松隆, 玉真昭男, 宮田圭介(静岡文芸大) ：DirectX を活用した 3D レーシングシミュレータの作成,情報処理北海道シンポジウム 2006, ポスターセッション E-8、2006. 10. 13.

3) 登大遊: DirectX9.0 3D アクションゲーム・プログラミング, 2003, 工学社

4) 斉藤 和邦：“DirectX 逆引き大全 500 の極意”, 秀和システム.

5) PhysX by Ageia ホームページ, http://www.ageia.com/, 2006.

6) FS チェアーホームページ: http://www.device21.com/FSChair.htm, 2006.

7) 宮田圭介(静岡文芸大), 小松隆, 玉真昭男：運転操作系デザイン検討用モーションベースの開発, 人間工学会東海支部研究大会, 3B2, pp. 68-69, 2006. 10. 28.