2008－CE－97　(5)
2008／12／20

３Ｄコンピュータゲーム開発を課題としたプログラミング教育

玉真 昭男
静岡理工科大学

抄録

プログラミング教育において重要なことは、文法マスターと数多くの問題演習に加えて、数千行以上のプログラムの開発体験を積ませることである。３Ｄゲーム開発は格好の課題となる。これまでに、Visual C++とDirectXを組み合わせ、３Ｄ描画技術を用いた各種のゲームを作成してきた。物理モデルを取り入れ、ゲームに登場する３Ｄオブジェクトに物理法則に従ったリアルな運動や挙動をさせることも出来た。今回、シューティングゲームとレースゲームの２つの成果を報告する。後者に於いては、運転再現モード等を追加して、運転の評価・分析が行えるシステムに拡張した。これを「全日本学生フォーミュラ大会」に向けたドライバーの運転練習に活用し、総合成績アップにつながる成果が得られた。

Programming Education Based on Development of 3D-Computer Games

Teruo TAMAMA
Shizuoka Institute of Science and Technology

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. The F1-racing game has been developed to a racing simulator for the "*Student Formula SAE™ Competition 2008*" and helped the driver train his driving skill off the course.

１．はじめに

1999 年に発足した「ゲーム情報学研究会」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。」

同研究会発足以来、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グ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。

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が主な研究分野として挙げている14の中から次の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、非常に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 と３Ｄゲーム開発用ライブラリ DirectX 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[1)]。

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)]。

今回、運転再現モード等を追加して、運転の評価・分析が行えるシステムに拡張した。これを「全日本学生フォーミュラ大会」に向けたドライバーの運転練習に活用し、総合成績アップにつながる成果が得られた[3)]。

２．使用したソフトウェアと問題点

２．１　開発環境

開発環境として Windows 用の C/C++コンパイラ Visual C++.NET 2005、３Ｄゲーム開発用ライブラリとして DirectX 9.0c、モデリングソフトとして Metasequoia 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[4)]。

２．２　Visual C++.NET や DirectX の問題点

Visual C++.NET と DirectX、特に後者の最大の問題点は２～３ヶ月ごとにバージョンが変わること、その度に一部の変数や関数の定義が変わって、前のバージョンでは動いていたプログラムがコンパイルエラーの塊になることである。数十ものエラーが現れてパニックになることも多い。DirectX を使ったゲーム作りに関する参考書は多いが、付録のサンプルプログラムがそのままでは動かないことが多い。本が書かれた時期と学生がそれを使って勉強する時期に１～２年の開きがあるのは普通だが、その間に Visual Studio や DirectX のバージョンアップがなされるため、コンパイルエラーが山のように出てしまうためである。初心者にはこのエラーの原因と対処法を見つけるのは不可能に近いので、多くの学生はこの時点で頓挫してしまう。ネットで調べても解決法はなかなか見つからないことが多い。

２．３　確実に動く基本プログラムを与えること

本学では、学生には最初、（１）作りたいゲームに適した DirectX のテキスト一冊[4)-6)]、（２）動くように直したその付録サンプルプログラム、及び（３）著者が作成した基本プログラムを与える。現時点では、Visual Studio.NET

2005 と、DirectX 9.0c(June 2007) の組合せを用いている。(3) の基本プログラムは次のようなものである。

(a) ポリゴンの組合せで立方体、球、平板、円錐台、ドーナツなどの３Ｄ基本図形を生成・描画するプログラム

・マウス、方向キー、Function キーなどの操作でそれらが順番に、あるいは逆順に現れるようにしたもの

・ライトやカメラの設定も含む

(b) この３Ｄ基本図形を組み合わせて作った車とＸファイル形式のキャラクタ(クマ)を描画し、それらが移動・回転するプログラム

・Ｘファイルはモデリングソフトで作ったオブジェクトをエクスポート機能によりファイル変換して作成

・音や音楽の設定も含む

助走期間にこれだけのものを与えて勉強させれば、あとは要所で手助けするだけで、やる気のある学生なら３Ｄアクションゲームを１年で作り上げることが出来る。

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良くすることができる。

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。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

モデリングソフト Metasequoia 2.4.0 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。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。

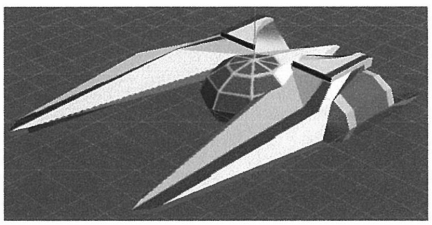

Fig. 3.1　完成した自機のモデル

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を計算してしまっては処理が重くなる。

これに対し、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。Bounding Sphere や Bounding Box [7]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。Bounding Sphere 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。Bounding Box 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。

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に飛行させることを実装したことである。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。

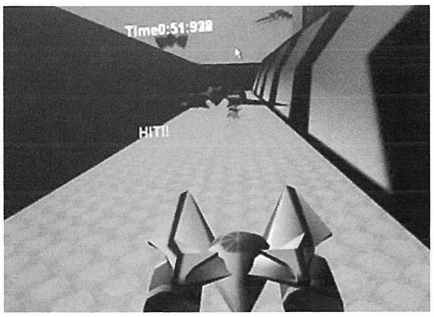

Fig. 3.2　ゲーム遊戯中の一画面

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 [8]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。

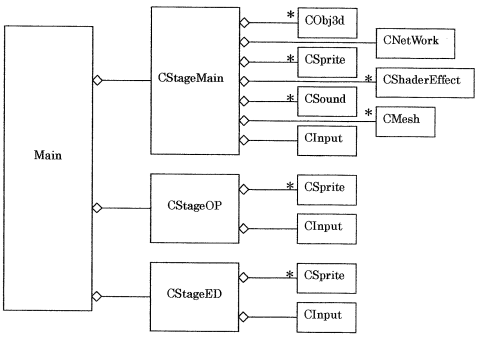

Fig. 4.1　作成したプログラムのクラス図

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。

４．３　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[8)]

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たりすることによって車の横滑りなどの表現が可能になる。

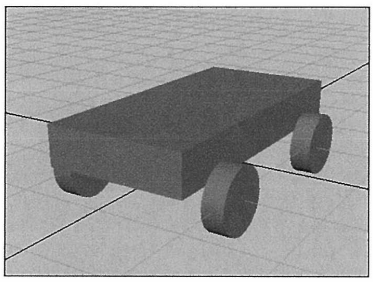

Fig. 4.2　車両型アクター

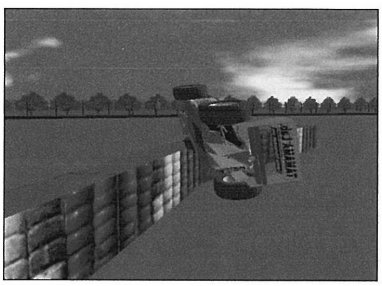

Fig. 4.3　ゲーム中で壁との衝突時の画像

４．４　評価

Fig. 4.5にゲーム操縦風景、Fig. 4.6にゲーム画面を示す。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。

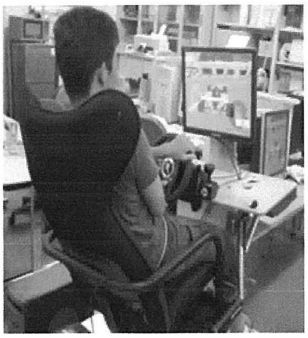

Fig. 4.5　ゲーム操縦風景

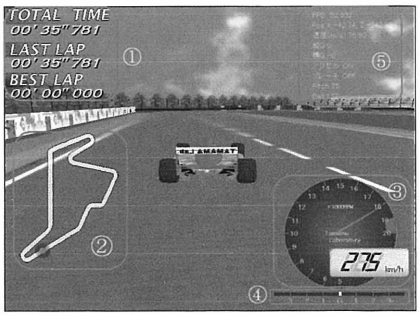

Fig. 4.6　ゲーム画面

５．全日本学生フォーミュラ大会用シミュレータ作成

「全日本学生フォーミュラ大会」[9]は社団法人「自動車技術会」の主催で2003年から静岡県袋井市の「エコパスタジアム」で毎年開催されている。学生の自主的なものづくりの総合能力を養成し、将来の日本の自動車産業を担う人材を育成する目的で設立された。学生のグループが自らフォーミュラスタイルの小型レーシングカーを企画・設計・製作する。この大会では車という「ものづくり」に関わる全ての活動を評価の対象とする。すなわち、車両性能だけでなく、コンセプトやデザイン、コストなど、車の持つ様々な要素を総合的に評価する。

とはいえ、本質的にカーレースである以上、走行性能により高い得点が与えられる。「動的審査」部門では、次の2つが最も重要である。

“オートクロス”：直線・ターン・スラロームを組み合わせた約900mのコースを1周する。2人のドライバーが2回ずつ、計4回のラップタイムを競う。

“エンデュランス”：22周のタイムレース。走行スピー

ドだけでなく、総合力や信頼性も評価の対象となる。

本学のチームを勝利させるために、Visual C++Ⓡ.NET と DirectXⓇ9.0 を使ってレーシングシミュレータを開発し、「全日本学生フォーミュラ大会 2007」で使われた公式コースをシミュレータのメインコースとして実装した[2]。今年の公式コースレイアウトは運営委員会規則によって大会直前まで発表されないことになっているが、発表されれば1日で実装することが可能である。

５．１　シミュレータの仕様と特徴

(1) 車種：F1のモデルを使用
(2) コース：「全日本学生フォーミュラ大会 2007」で使われた公式コースを実装
(3) 操作性：ステアリングコントローラに対応させ、実際の車の操作を体感可能
(4) レース画面：現在のポジション、総合タイムタイム、ラップタイム、速度の表示
(5) 物理モデル導入：衝突後の跳ね返りなど、F1 カー並みのリアルな挙動を実現
(6) 専用操縦席導入:プレーステーション用コックピットを採用
(7) 運転評価機能：運転記録のファイル保存、「再現モード」、「任意の運転記録との対戦モード」を実装

（７）は本シミュレータの最大の特徴である。「再現モード」では、良いラップタイムが出たときに、その過程を再現し、精査することで、なぜその記録が出たかを評価・分析することが出来る。「対戦モード」では、ベストラップタイムを出した記録と競争することでそれ以上のラップを出すヒントを見つけ、ドライバーの運転スキルの向上につなげることが出来る。また、過去の運転記録を選んで対戦することが出来るため、ドライバーのレベルに応じた「コンピュータ対戦」が出来る。このシミュレータをコンピュータゲームとして使うときに生きる機能である。

５．２　通常走行モード

学生フォーミュラ大会を想定して、実際に大会で使われるコースを再現したものを使用して運転練習を行うことができる。操作はステアリングコントローラなどの市販ゲームコントローラを使用する。画面には現在の速度やコース上の位置、ラップタイム、合計タイムなどの情報が表示される。本シミュレータはプレーヤの全運転過程（ログ）をファイルに保存する機能を有している。

５．３　再現モード

ファイルに保存した各運転ログは読み出して、コンピュータに再現させることが出来る。学生フォーミュラ大会に出場するほとんどの車は最高速度 100km/h 以上で走行することが出来る性能を有している。学生が作った車の大会とはいえ、まともなカーレース用のコースが用いられる。従って、S字やヘアピンカーブでのコースアウトによるタイムロスが勝敗を分ける。それを防ぐために、ドライバーの運転スキルやコースアタック戦略が要求される。本シミュレータを使って良いラップタイムが出たときは、その過程を再現し、精査することで、なぜその記録が出たかを評価・分析することが出来る。例えば、ヘアピンカーブの直前で何km/hまで減速し、どこで何km/hまで再加速したか、直線コースのどこで何 km/h まで加速したか、ハンドルをどの程度の角度で切ったか、アクセルやブレーキをどの程度踏み込んだか、などである。このとき、必要に応じて再生速度を調節することが可能である。また、カメラアングルを変更して様々な角度から走行時の様子を再現したり、車の軌道を視覚的に表示したりすることで、ライン取りの分析をすることができる。

５．４　コンピュータ対戦モード

過去の全てのプレーヤの運転ログがラップタイム毎にソートされてファイルに保存されている。その一つを選んでコンピュータ操縦の車に再現させ、それと対戦することが出来る(Fig. 5.1)。従って、プレーヤは、自分のレベルに応じて、過去のどの運転記録とも対戦することが出来る。この機能を「学生フォーミュラ用シミュレータ」として使うと、ベストラップタイムを出した記録と競争することでそれ以上のラップを出すヒントを見つけさせ、ドライバーの運転スキルの向上につなげることが出来る。

一方、この機能を「レーシングゲーム」として使うと、過去の任意の記録を選んで対戦することが出来るため、プレーヤのレベルに応じた「コンピュータ対戦」が出来る。

ゲームとしてのこの機能は計り知れない面白さにつながる。コンピュータに車を自走させるアルゴリズムの考案は簡単でないし、いろいろなスピードで、多様なコースアタックをする自走モードを多数実装することはほとんど不可能である。しかし、本シミュレータは、過去の運転ログのファイル保存とその再現機能を持たせることで、それを可能にした。

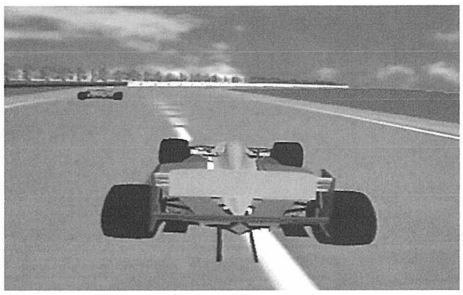

Fig. 5.1 「コンピュータ対戦モード」での走行シーン

6．まとめ

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 6.1 に示す。いずれも１万行以上の巨大プログラムである。

Table 6.1 開発したプログラム規模

プログラム名	Ｘファイル個数	ソースコード行数
シューティングゲーム	42	10,127
レーシングゲーム	20	12,938

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。

また、本学のチームのフォーミュラカーを「全日本学生フォーミュラ大会 2008」で優勝させるために、学生フォーミュラ用シミュレータを開発した。優勝こそ出来なかったが総合 12 位（昨年 33 位）と健闘した。本シミュレータはその一翼を担ったと自負している。Ｆ１でもドライバーはシミュレータも使って練習する。学生の大会にいち早くシミュレータを取り入れたのは参加 77 チーム中で静岡理工科大学のみであるので、大会関係者から評価された。

参考文献

1)玉真昭男，小松隆，青木悠：プログラミング教育と３Ｄコンピュータゲーム開発，静岡理工科大学紀要，第 15 巻，pp. 39-46(2007).

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.

3) 三浦義弘，鈴木絵美子，玉真昭男：物理モデルを使用したドライビングシミュレータ及び運転評価システムの開発，情報処理学会研究報告，2008-CG-133、pp. 55-59 (2008).

4) 大川善邦他: DirectX9 実践プログラミング，工学社，2003.

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

6) 秦森桂: 3D ゲーム制作入門，工学社，2004.

7) 鎌田茂雄：“DirectX 逆引き大全 500 の極意”，秀和システム，2006.

8)nVIDIA PhysX ホームページ，
http://jp.nvidia.com/object/nvidia_physx_jp.html，2008.

9)全日本学生フォーミュラ大会ホームページ，
http://www.jsae.or.jp/formula/jp/，2008.

※ Visual C++®, DirectX®は Microsoft 社、PhysX®は AGEIA 社、Havok®は Havok 社、Metasequoia®は O. Mizuno 氏、それぞれの登録商標です。