Vol.2011-CE-112 No.15
2011/12/18

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２

玉真昭男 †

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．コンピュータゲーム開発はそのための格好の課題となるので，本学ではプログラミング教育の最終課題として位置付けている．わが研究室では，Visual C++と DirectX を組み合わせて，更に高度な３Ｄゲーム開発に取り組ませ，これまでにレーシングゲーム，シューティングゲーム，格闘ゲーム，育成ゲームなど 10 種類以上の３Ｄゲームを作成してきた．今回は，開発希望者の多い縦スクロール・シューティングゲームを課題としたプログラミング教育について述べる．

Programming Education Based on Development of 3D-Computer Games – 2

Teruo TAMAMA †

In the programming education, what is important is not only mastering the programming language grammar and solving many exercises, but also having development experience of a program over 3000 lines. 3D-games are good targets for that purpose. In our research laboratory, students challenge to develop high-level 3D computer games using Visual C++ and DirectX. More than 10 excellent games, such as racing games, shooting games, fighting games, plant raising games, etc., have already been developed. In this article, programming education using vertical-scroll type shooting games, popular among students, as a target is described.

１．はじめに

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「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．

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，レーシング，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]．

最近は，単なる「遊戯」の域を越えて，ゲームの社会的・実用的応用を考えさせるため，学生に「シリアスゲーム」に取り組ませている．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した[2]．また，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 2008」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]．

今回は，開発希望者の多い縦スクロール・シューティングゲームを課題としたプログラミング教育について述べる．

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 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．

† 静岡理工科大学　総合情報学部コンピュータシステム学科
Shizuoka Institute of Science and Technology

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][5][6].

３.「シリアスゲーム」開発事例とその成果

３．１　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．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可能にした[2].

図１　ゲーム操縦風景
Figure 1 A Photo of Game Operation

３．２　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」[7]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．本学のチームを勝利させるために，Visual C++.NET とDirectX9.0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].その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．

結果は，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．実際の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，大会関係者の注目を集めた．

４．縦スクロール・シューティングゲーム

３年生後期に，本格的なゲーム作りに挑む最初の課題として「インベーダーゲーム」を取り上げている．1978年に発売され，以後歴史に残る爆発的ヒットとなった「スペースインベーダー」((株)タイトー製)と同じものを，どこまで作れるかに挑戦させる．「縦スクロール･シューティングゲーム」の走りとなった作品であるが，２Ｄなので，グラフィックライブラリとしてはＭＦＣ（Microsoft Foundation Class）を用いている．ＭＦＣは別にゲーム開発用ではなく，通常のビジネス用のライブラリである．

４．１　作成手順

懇切丁寧なマニュアルを用意し，以下の手順で製品と同等レベルの「インベーダーゲーム」が作れることをまず体験させる．

（１）MFC基本例題演習
（２）インベーダー，ＵＦＯ，自機などのビットマップ画像を作成
（３）インベーダー軍団（5×10）の動きを実装（図２）

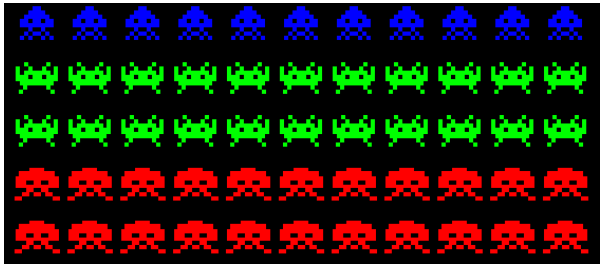

図２　インベーダー軍団
Figure 2　Invaders’Rank

（４）自機描画と方向キーによる移動操作の実装
（５）自弾，敵弾の動きと当たり判定の実装
（６）ＵＦＯの動作と出現タイミングの実装
（７）シェルターの組込みと敵弾による部分破壊の実装
（８）スタート／ゲームオーバー画面，スコア表示などを実装し，ゲームシステムの完成

（９）自機，敵機，背景の画像等を変え，動作モードも変えて，新ゲームの創造

（８）までで，本家『インベーダーゲーム』に迫るゲーム作りを体験した後，（９）で自分なりのオリジナルゲームに作り変えるように指導している．その一例を図３に示す．これは水中で海洋生物と戦うシューティングゲームである．

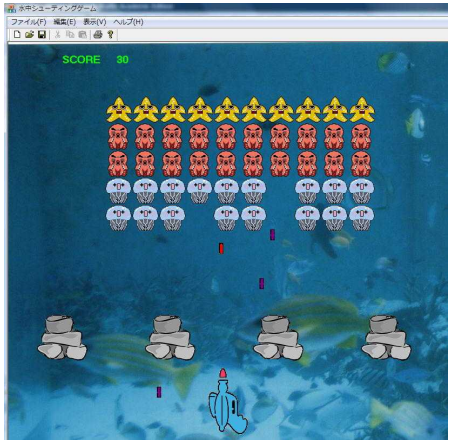

図３　水中シューティングゲーム
Figure 3　Under-Water Shooting Game

４．２　困難なポイント

「インベーダーゲーム」の作成では，以下の点がやや困難であるが，必要に応じて解説したり，部分マニュアルを追加したりしている．

（１）インベーダー軍団の動きの実装
（２）自弾，敵弾の動きと当たり判定
（３）画面のちらつき防止
（４）キャラクタの透明化
（５）シェルターの部分破壊

５．３Ｄ縦スクロール・シューティングゲームの開発

インベーダーゲームを完成できた学生は，次に，それと同じジャンルではあるが，本格的な３Ｄ縦スクロール・シューティングゲームに挑戦する．ここからは本番であり，自力で開発させるため，特に丁寧なマニュアルは用意していない．シューティングゲーム用の基本ゲーム要素クラスを組み上げた，ベースとなるゲームプログラムシステムを用意しておいて，与えるのみである．

５．１　プログラム構成

ある卒業生の作った，基本ゲーム要素をクラス化したプログラム（図４）をシューティングゲームのベースシステムとして使用している[8]．約 9000 行のソースコードからなる．

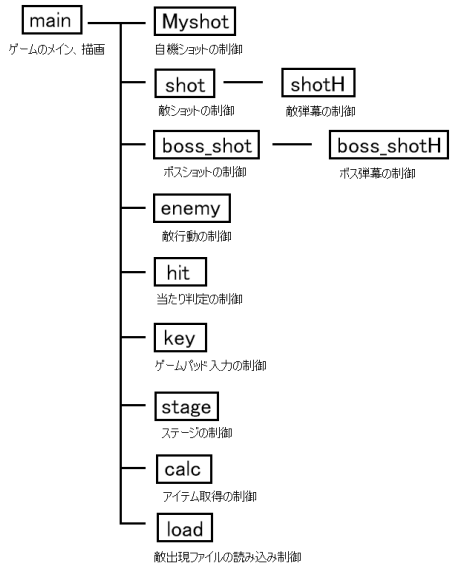

図４　基本プログラム構成
Figure 4 Basic Program Structure

これに，必要なら自分なりの機能を盛り込めるようプログラムを追加してカスタム化する．各自それを用いて，オリジナルなシューティングゲームを作成している．

５．２　作品例[8]

弾幕系のシューティングゲーム[9]であるので,敵機の動きと敵弾のパターンに特徴がある．90 種類もの弾幕を作った例を紹介する．

５．２．１　背景の動き

縦スクロールゲームにするため，背景の画像を一定のスピードで下にスクロールすることで，自機が上に向かって進んでいるように見せている．背景の模様を３パターン用意し，繰り返し表示させている．

５．２．２　敵の動き

ここではゲームに出てくる敵の動きについて説明する．敵データ処理の大きな流れは次のようになっている．

- ゲームカウンタがあるポイントに達した時，画面内に敵を出現させるフラグを立て，敵データを登録する．
- 登録された敵データがあれば，登録された移動パターンにそった敵の移動制御を行う．
- 敵が画面から外れると，表示フラグを消す．
-

まずは敵データの登録について説明する．ゲーム上には多くの敵が出てくる．その敵が持つべき情報は構造体リスト１として管理している．

リスト１　敵の構造体
List 1 Structure of the Enemy Variable

```
typedef struct{
      int flag, cnt, pattern, muki, knd, hp,
         hp_max, item_n[6];
//フラグ，カウンタ，移動パターン，向き，敵の種類，HP最大値，落とすアイテム
      float x,y,vx,vy,sp,ang;
//座標，速度x成分，速度y成分，スピード，角度
      int bltime, blknd, blknd2, col, state,
         wtime,wait;
//弾幕開始時間，弾幕の種類，弾の種類，色，状態，弾幕待機時間，停滞時間
} enemy_t;
```

敵を出現させるためには，いつどこで，どんな敵がどんな弾を撃ってくるのか，などの情報を指定してやらねばならない．これを全てプログラムで書いていたら手間が掛かるし，拡張性に欠ける．そこで，「敵＆敵弾出現管理表」を表計算ソフト Excel を使って作り，これを csv 形式で出力する（表１）．

表１　Excel で作成した「敵＆敵弾出現管理表」
Table 1 Enemy Motion & Enemy Barrages Management Table

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	/カウンタ	移動パター	敵の種類	x座標	y座標	スピード	発射時間	弾幕種類	弾の色	体力	弾種類	待機時間	アイテム1
2	/cnt	pattern	knd	x	y	sp	bltime	blknd	col	hp	blknd2	wait	item_n
3	90	0	0	200	-20	0	120	20	0	100	0	120	0
4	100	0	0	200	-20	0	120	20	0	100	0	120	0
5	110	0	0	200	-20	0	120	20	0	100	0	120	0
6	120	0	0	200	-20	0	120	20	0	100	0	120	0
7	130	0	0	200	-20	0	120	20	0	100	0	120	0
8													

一番左が敵の出現する時間になっている．60fps で動作するゲームなので 60 フレーム＝１秒になっている．上の５行は，ゲーム開始から 1.5 秒たつと最初の敵が出現し，以後 10 カウント（フレーム）＝1/6 秒ずつ遅れて４体の敵が出てくることを意味している．

このデータを敵の出現情報を入れる変数 enemy_order に読み込む．ステージごとに分けて，敵の情報もあるので enemy_order[ステージ数][敵データ数]という２次元配列になっている．

敵機の行動パターンは全部で 32 種類作成した．しかし，基本的には変更する値が違うだけで仕組みは同じなので，幾つかのパターンを上げるに留める．

（１）下がってきて停滞後上昇
（２）下がってきて左へ
（３）画面内をランダムにうろうろする
（４）左へ進み一回転
（５）左から右へギザギザ進む
（６）上から下へゆらゆら降りる

５．２．３　敵弾の動き

ここでは敵の撃ってくる弾の動きについて説明する．このゲームでは，敵１体が１発の弾を撃つのではなく，何発かの弾を規則的に出してくる．たくさんの弾が規則的に飛ぶ様子は，花火のような美しさがあり，このゲームの特徴になっている．

まず，弾幕の登録は敵が出現してからのカウント cnt と，設定した弾幕開始カウント bltime が一致したときに行う．弾幕のデータがどうなっているかをリスト２に示す．

リスト２　弾幕を定義する構造体
List 2　Structure for Enemy Barrage Patterns

```
//弾に関する構造体
typedef struct{
//フラグ，種類，カウンタ，色，状態，少なくとも消さない時間，エフェクトの種類
    int flag, knd, cnt, col, state, till, eff, eff_detail;
//座標，角度，速度，ベースの角度，一時記憶スピード float x, y, z, angle, spd,
     base_angle[2],
         rem_spd[1], vx, vy;
} bullet_t;

//ショットに関する構造体
typedef struct {
//フラグ，種類，カウンタ，どの敵から発射されたかの番号
     int flag,knd,cnt,num;
//ベース角度，ベーススピード
     float base_angle[2],base_spd[1];
         bullet_t bullet[SHOT_BULLET_MAX];
} shot_t;
```

弾一つ一つにスピードや角度，状態などがある．最短でも消さない弾のカウント till は，画面から消えたら消すようになっていると作れない弾幕があるためである．そこで最短でも設定した時間は弾が画面外に出ても消えないようにする必要がある．base_angle は弾の角度が変化するときに基準となる角度を設定する．

shot_t という構造体は弾幕データ一つ分を管理する構造体である．弾幕一つに各変数と SHOT_BULLET_MAX 個弾がある構造になっている．shot_t は敵の弾幕情報であり，敵を一度に表示する最大数と同じだけ用意することになる．

弾幕，弾は登録して使うので，登録されている情報を検索する必要がある．弾幕データの最大数までループして登録されているものを探し，登録されているデータがあれば計算させる．主なものは次の 10 パターンであるが，敵の動きと組み合わせると約 90 通りの弾幕を作成出来た．

（１）単発自機狙い
（２）３way 弾，４way 弾（図５）
（３）渦巻き弾（図６）
（４）蜘蛛の巣弾（図７）
（５）ブラックホール（自機吸い込み弾）
（６）混合弾幕　（図８）
（７）誘導弾　（図９）
（８）星弾
（９）花火弾　（図 10）
（10）強制回転弾　（図 11）

（10）の強制回転弾幕は敵から休みなく撃たれる弾によって，強制的に敵の周りを回転しなければならなくなる弾幕である．発射角度が少しずつ変わるので，それにあわせて自機を動かさなければならない．

図５　４way 弾
Figure 5　4-way Barrage

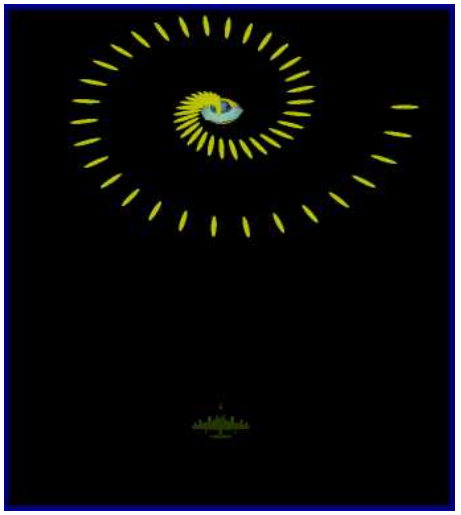

図６　渦巻き弾
Figure 6　Spiral Barrage

図 7　蜘蛛の巣弾
Figure 7　Spider-Web Barrage

図 8　混合弾幕
Figure 8　Mixed Barrages

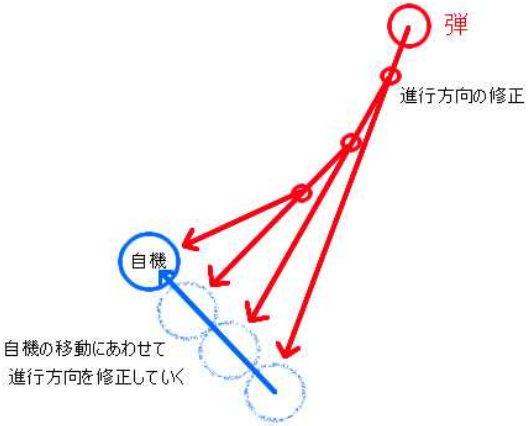

図 9　誘導弾（自機に向かって追いかけてくる弾）
Figure 9　A Guided Missile

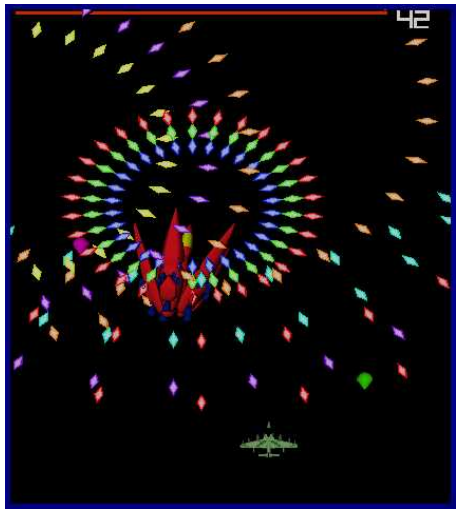

図 10　花火弾
Figure 10 Fireworks Barrage

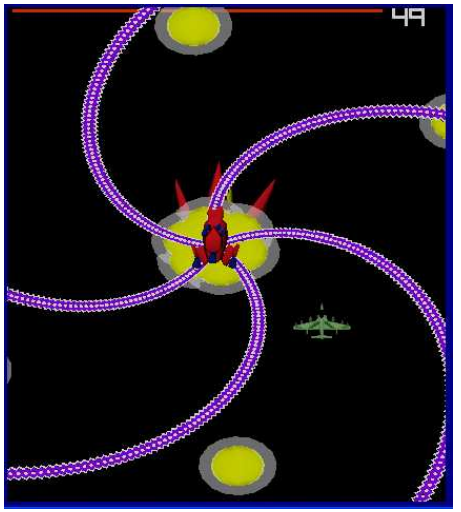

図 11　強制回転弾
Figure 11　Rotation-Forced Barrage

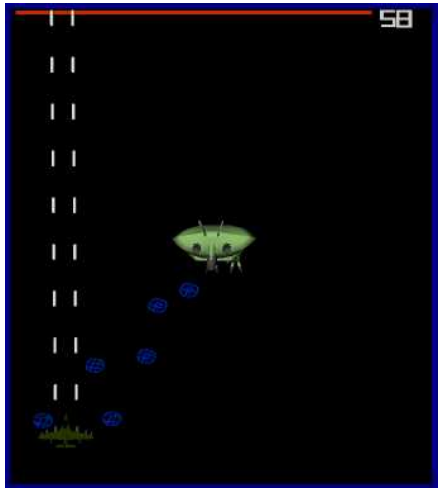

図 12　サブショット
Figure 12　Sub-shots

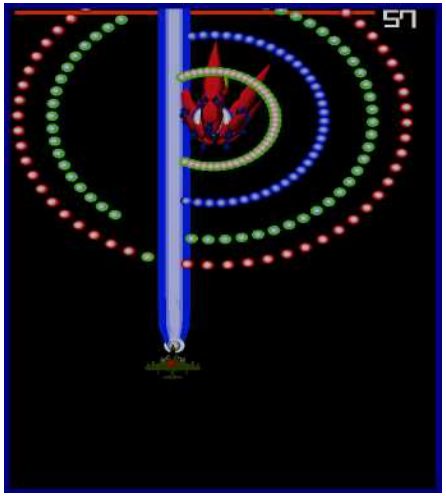

図 13　レーザービーム
Figure 13　Laser Beam

５．３　自機の攻撃パターン

このゲームでは自機からまっすぐ飛ぶメインショットの他にサブショットがある（図 12）．サブショットは敵を追うホーミング弾になっている．自機から一番近い敵を検索し，その敵の方向に飛んでいく弾を定期的に飛ばす．

２つ目は縦方向に伸びるレーザービームを出す攻撃である（図 13）．この攻撃はボタンを押したときの自機の位置に設置される．ビーム部分に敵弾が触れると敵弾を破壊し，敵自身が触れると敵にダメージを与えるようになっている．

５．４　ゲームの流れ

このゲームは４ステージで構成されており，各ステージ内は雑魚敵，中ボス，雑魚敵，ステージボスという流れになっている．雑魚敵を倒すとアイテムが出てきて，アイテムを取得すれば自機がパワーアップし，攻撃が強くなるようになっている．

５．５　操作説明

ゲームを起動するとタイトル画面が出てくる.タイトル画面では,ゲームスタート,練習モード，ゲームの終了を選ぶことが出来る．練習モードでは全４ステージの内,好きなステージを遊ぶことが出来る．

コントローラは Xbox 360 用のものを用いた．Microsoft 社の製品で，Windows PC と相性が良いためである．

ゲームを開始すると表示される右側のボードの表示について説明する（図 14）．上から順に，ハイスコア，現在のスコア，残りの自機数，メインショットのパワー，サブショットのパワーとなっている．

最終ステージ４のボス戦（図 14）に入ると，画面内から雑魚敵が消え，雑魚敵の撃っていた弾も消える．ボスの体力が表示され，制限時間も隣に表示される．体力を０にするか，制限時間が０になると弾幕が終了し，次の弾幕へと移る．すべての弾幕が終わるとボスを倒したことになり，ステージクリアとなる．ステージ４のボスを倒すことでゲームクリアとなる．

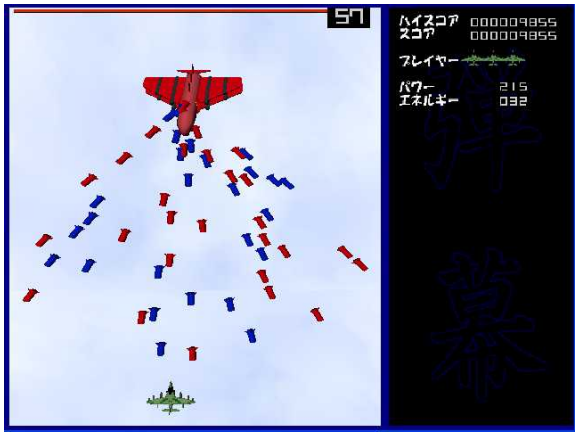

図 14　最終ステージ－ボス戦
Figure 14　Final Stage - Fight with the Boss

６．まとめ

大学祭やオープンキャンパスなどのイベントで展示しているが，参加したお客さんに十分に楽しんで頂けるレベルのゲームを開発出来ている．

一定のモノが出来ると,次々に機能を追加して難易度を上げたくなる.その過程で,また新しい技術を習得したくなる．数学に強くない学生が，改めて三次元幾何やベクトル解析などの教科書を学び直し,絵で確認することで初めて分かったと言って喜ぶ.ゲーム開発にはそのような相乗効果，正のスパイラル効果があるので，３Ｄゲーム開発は単にプログラミング教育に留まらず，いろいろな関連科目の総合学習教材になっていると確信している．

学生からは，「ゲーム制作を通して，ゲーム開発がいかに難しいか，どのような技術が必要であるのかが良く分かった．分からないところを調べ，試していくことでプログラミングの技術を向上させることができた．弾の動きや当たり判定などのシステムを考えながら実装していくことで，楽しみながら，多くの貴重な経験を積むことができた．」といった感想が述べられている．

謝辞

本原稿の作成に当たっては,本学 2009 年度卒業の瀧尾浩志氏の卒業論文を参考にしています．ここに謹んで感謝の意を表します．

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) 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.
8) 瀧尾浩志：縦スクロールシューティングゲーム“ＴＨＥ　ＢＡＲＲＡＧＥ”の制作，静岡理工科大学情報システム学科卒業論文，2009 年度.
9) 松浦健一郎/司ゆき：「シューティングゲームプログラミング」，ソフトバンククリエイティブ株式会社.