３Ｄゲーム開発のための オブジェクト指向型ゲームエンジンの構築

Development of an Object-Oriented Game Engine for Developing Specific 3D-Computer Games

玉真昭男†

Teruo TAMAMA†

Abstract: In our research laboratory, students challenge to develop high-level 3D computer games using Visual C++ and DirectX. More than 20 excellent games, such as racing games, shooting games, fighting games, plant raising games, etc., have already been developed. In this article, a "game engine" which we recently developed, powerful for developing vertical-scroll type shooting games, popular among students, 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20種類以上の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]．

2012年3月2日受理

†総合情報学部　コンピュータシステム学科

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]．また，自転車競争ゲームをスピードスケート選手用のトレーニングシステムに応用した例[4]もある．

さて，「ゲームエンジン」とは，様々なゲームを効率良く生成・開発出来るように設計された，ソフトウェアシステムのことである．例えば，レーシングゲームエンジンでは，３ＤＣＧモデリングソフトで車を作り，そのエンジンに入れれば，あらかじめ用意されたコース上をその車が走る．何もプログラムしなくとも，ゲームコントローラで操作でき，当たり判定も組み込まれているので，フェンスに当たれば跳ね返る．更に，モデリングソフトでコースや障害物を作りゲームエンジンに組み込めば，全くオリジナルのシーシングゲームを作ることも出来る．これがゲームエンジンである．

今回は，３Ｄゲームの効率的開発とチームで大規模なゲームを開発する場合に備えて，作成希望者の多い縦スクロール・シューティングゲームに用に，５年掛かりでゲームエンジンを開発したので報告する．

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．

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[5][6]．

３．自機や敵機のモデリング

モデリングに使用したMetasequoiaは３Ｄポリゴンモデラーと呼ばれる種類のソフトウェアである．ポリゴン単位で立体のオブジェクトを生成・

編集することができる.本研究において自機(Fig. 1)や敵,弾などはこのソフトウェアを使って作成した.

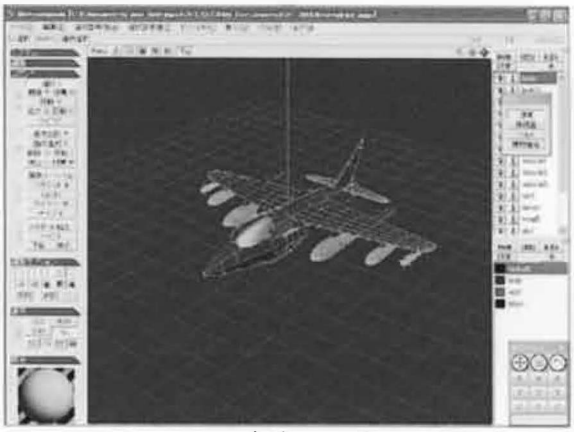

Fig. 1 自機のモデル

4．3D縦スクロール・シューティングゲームの開発

5年掛かりで,縦スクロール・シューティング用ゲームエンジンを開発してきた.その過程を振り返りつつ,構築したゲームエンジンの概要を説明する.

4．1 最初の作品[8]

真上から見た状況で自機を操り,迫り来る敵を倒し,最後に出てくるボスを撃破する,という流れのゲームである.また,敵を倒すことで得点が入り,獲得点数でクリアしたときにどの程度上手くプレイすることが出来たかが判る.プログラムの開発過程は自機のモデリング,敵機の行動パターンの作成,敵機,建物の配置,ゲームシステムの作成の4つに分けられる.

モデルの作成は 3Dモデラー Metasequoia を使ったためプログラムの過程には含まれず,短時間で高度なモデルの作成が可能になった.しかし,精巧なモデルが出来た反面,ポリゴン数が多く,容量も大きくなってしまい,精巧さと容量のバランスを取ることに苦労した.ゲームシステムは敵機・自機の判定,ボス撃破や逆に自機を撃墜されてしまった時の処理,白機のボタンによる攻撃の種類などを組み込んでいる.また,地上の敵も作成し,対地用の攻撃でなくては倒すことが出来ないようにしてゲームに幅を持たせた (Fig.2).

満足な点は,ボスの登場シーン,高低差を付けた敵の配置,自機の左右への動きにカメラも連動して動く効果などを上手くプログラミング出来た事である.

このプログラムでは,自機も敵機も,それぞれが発射する弾も全てプログラムで発生させ,制御した.従って,敵機や弾幕を増やそうとすると,その度にどのタイミングでどの敵機を出現させ,かつ,どのような動きをさせるかを一々プログラムで書かなくてはならない.これでは拡張性に限界があることは一目瞭然であった.しかし,この年は時間の関係で,それしか対応出来なかった。

今後の課題は,如何にしたら容易に敵機や敵の弾幕の数を増やし,また,パワーアップ,追尾攻撃,ロック機能など,ゲームとしての面白さを更に追及できるようなプログラム構成に出来るかであった。

Fig. 2 ゲーム画面

4．2 ゲームエンジンの開発

上の体験を基に,その後,ある卒業生の作った基本ゲーム要素[9]を全てクラス化し,シューティングゲームのベースシステムとした.後にこれを改良してゲームエンジンとした(Fig. 3).約 9000 行のソースコードからなる.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する[9][10].

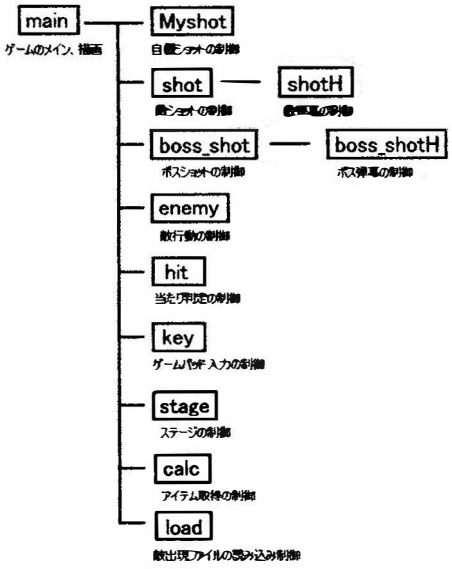

Fig. 3 ゲームエンジンのプログラム構成

４．３　作品例[9][10]

弾幕系のシューティングゲーム[9][10]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．

４．３．１　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．

４．３．２　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 List 1 として管理している．

List 1　敵の構造体

```
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（Table 1）．

Table 1　Excel で作成した「敵＆敵弾出現管理表」

	A	B	C	D	E	F	G	H
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
2	/cnt	pattern	knd	x	y	sp	bltime	blknd
3	90	0	0	200	-20	0	120	20
4	100	0	0	200	-20	0	120	20
5	110	0	0	200	-20	0	120	20
6	120	0	0	200	-20	0	120	20
7	130	0	0	200	-20	0	120	20
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．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

４．３．３　敵弾の動き[12]

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 List ２に示す．

List 2　弾幕を定義する構造体

```
//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，発射された敵の番号
    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（Fig. 4）
（3）渦巻き弾（Fig. 5）
（4）蜘蛛の巣弾（Fig. 6）
（5）ブラックホール（自機吸い込み弾）
（6）混合弾幕　（Fig. 7）
（7）誘導弾　（Fig. 8）
（8）星弾
（9）花火弾　（Fig. 9）
（10）強制回転弾　（Fig. 10）

（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．

Fig. 4　4way 弾

Fig. 5　渦巻き弾

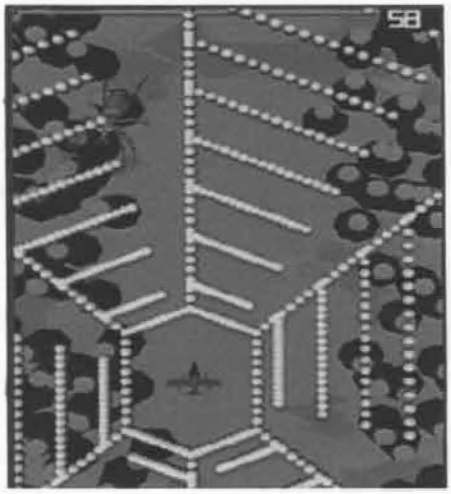

Fig. 6　蜘蛛の巣弾

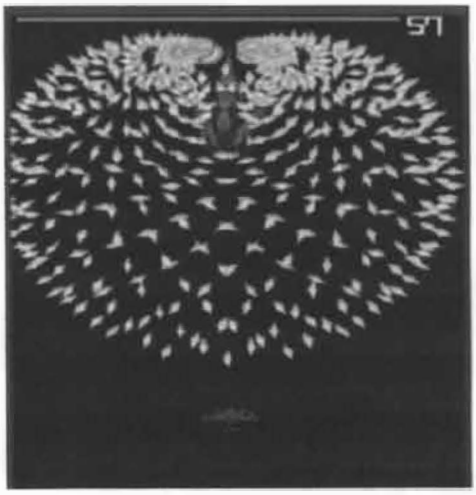

Fig. 7　混合弾幕

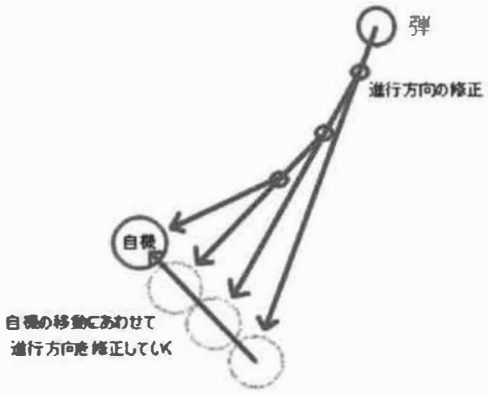

Fig. 8　誘導弾（自機を自動追尾する弾）

Fig. 9　花火弾

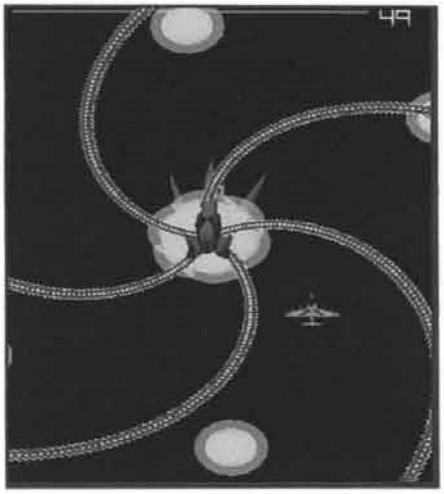

Fig. 10　強制回転弾

5．ボス

5．1　ボスの構成

ステージの最後にはボスが登場する．ボスは通常の敵と違い耐久値が高くて幾つかの攻撃パターンがある．本作品ではステージ4のボスまで作成した．ステージ1のボス(Fig. 11)は本体コア，本体の土台，色違いの小さなコア×3，小さなコアの土台×3の8つのキャラクターで構成している．これらはCBossクラスを基本クラスとして継承している．

5．2　ボスの移動

ステージ1のボスの本体は登場する時以外は動かない．代わりに，本体周りの小さなコア達が動く(Fig. 11)．小さなコアの動きは，一度本体から離れて一定時間が経つまで反時計回りに本体の周りを回転する．そして，本体へ合体する．この時，本体コアは自分の前にある小さなコアの色と同じ色に変わる．小さなコアが移動中の時はボスの状態は無敵である．本体コアは小さなコアが全て破壊されないと当たり判定の判定がされないためダメージを受けない．本体コアが破壊されたらステージクリアとなる．

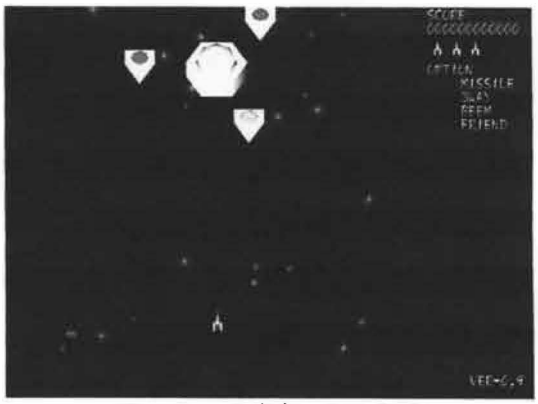

Fig. 11　小さなコアの動き

5．3　ボスの攻撃

本体コアの色によって攻撃パターンが変わる．本体コアの色が赤い時は，小さなコアからNWay弾が発射される．この時は本体前のコア以外は自機を狙って撃ってくる(Fig. 12)．青い時は，小さなコアからそれぞれ違う角度で直線に分裂弾を発射する．分裂した弾は進行方向に対して垂直に交わる方向へ移動する(Fig. 13)．分裂する時間は乱数によって決めている．黄色の時は，小さなコアから停止と移動を繰り返して自機を狙う弾が発射される．本体前のコアからはNWay弾も発射され，残り2つのコアからは一定時間ごとに停止と移動を繰り返して自機を狙う弾が追加される(Fig. 14)．

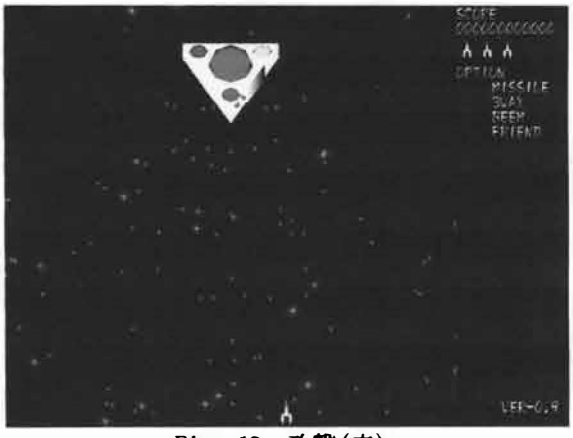

Fig. 12　攻撃(赤)

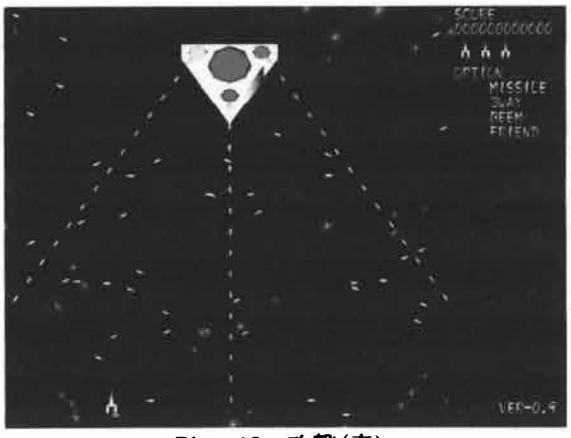

Fig. 13　攻撃(青)

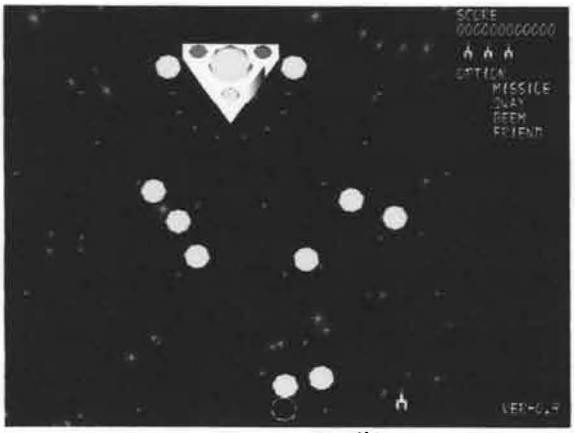

Fig. 14 攻撃(黄)

どこかのコアが破壊されている場合は，破壊された分だけ攻撃が緩くなるため安全地帯が出来てしまう．安全地帯を潰す為に破壊箇所からは NWay 弾を発射する(Fig. 15)．

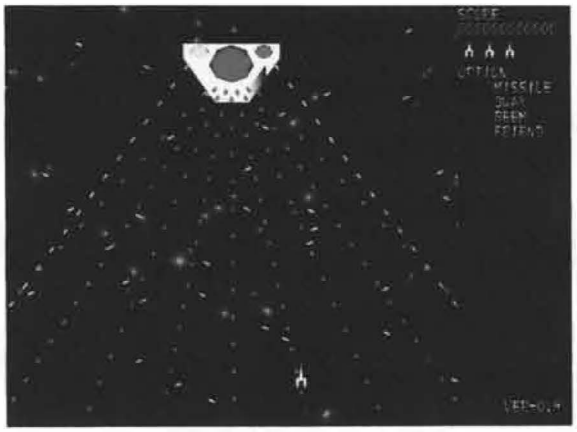

Fig. 15 破壊箇所からの攻撃

全ての小さなコアが破壊されると本体が攻撃を開始する．弾速の違う NWay 弾を時計回りに発射していく．一ヶ所だけでは避けやすいので四ヶ所からこの攻撃を行う(Fig. 16)．

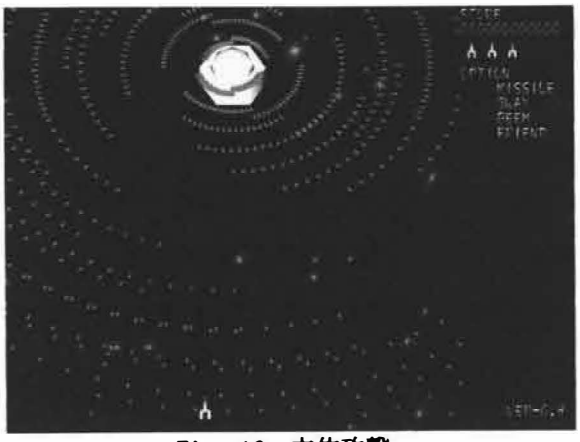

Fig. 16 本体攻撃

6．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 (Fig. 17)．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 (Fig. 18)．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．

Fig. 17 サブショット

Fig. 18 レーザービーム

7．ゲームの流れと操作説明

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．

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．ゲーム中で使うボタンの配置を Fig. 19 に示す．タイトル画面では十字キーで選択をして，メインショットのボタンで決定をする．

Fig. 19 Xbox 360 コントローラーのボタン配置

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（Fig. 20）．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．

Fig. 20 最終ステージ４のボス戦

８．ゲームエンジンの意義と功罪

ゲームエンジンを使うと，あまりプログラムを書かなくても一定の３Ｄゲームが作成できてしまう．「プログラミング教育」という観点からすると，これは好ましくないことかも知れない．

しかし，一方で，毎年ゼロから，個人単位でゲームプログラムを書かせていると，大規模なゲームはいつになっても作れない．時には，ゲームエンジンを使って効率を高め，しかも複数の学生からなるチームに共同作業で一つのゲームを作らせれば，研究室としては大規模で，迫力のあるゲームを開発することが出来る．大学祭やオープンキャンパス等で観客を驚かすゲームを作るには有効な手段である．

要は学生の能力に応じて，あるいは研究室としてどれだけの規模のゲームを開発したいかの状況に応じて，ゲームエンジンを使うか否かを判断すれば良いと考えている．

９．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 2005 年度卒業生藤田克英氏，2009 年度卒業生瀧尾浩志氏，2010 年度卒業生有ヶ谷直希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) 玉真昭男、富田寿人：シリアスゲーム開発を課題としたプログラミング教育，情報処理学会研究報告，2009-CE-102(20)，pp. 1-5(2009)．
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) 藤田克英：Visual C++ .NET と DirectX9.0 によるシューティングゲームの作成，静岡理工科大学情報システム学科卒業論文，2005 年度.
9) 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.
10) 有ヶ谷直希：3D 縦スクロール・シューティングゲーム” Space War”の制作，静岡理工科大学情報システム学科卒業論文，2010 年度.
11) 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.

注）Visual C++®, DirectX, Microsoft Excel, Xbox 360は米国 Microsoft Corporation の登録商標です.
Metasequoia は O. Mizno 氏作成のソフトウェアです．本文中では®および™は省略しています.