オブジェクト指向による３Ｄ格闘ゲームエンジンの構築

Development of an Object-Oriented Game Engine for 3D Fighting 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 3D fighting games, 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．

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][2][3][4]．

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，そのゲーム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，全くオリジナルのレ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のため，３Ｄゲームの中では最も難易度の高いスキンメッシュ・アニメーションを使った格闘ゲームエンジンを開発したので報告する．

2013 年 3 月 1 日受理

†総合情報学部　コンピュータシステム学科

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，キャラクタのモーション作成に Toy Studio 1.6.2.5 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．キャラクタのモデリングとモーション作成

３.１　キャラクタのモデリング

３Ｄモデリングソフト Metasequoia（メタセコイア）を用いて敵キャラクタ「夜叉」のモデリングを行った．その完成図を Fig. １に示す．

Fig. １　Metasequoia で作成した夜叉のモデル

３．２　夜叉のモーションの作成

１対１の格闘シーンを売り物にするためには，自キャラクタと敵キャラクタのリアルな動きが必要になる．そこで，格闘の技にはモーションキャプチャーデータを活用することにした．

３．２．１　モーションキャプチャ

モーションキャプチャとは，人物や物体の三次

元の動きをデジタル的に記録するシステムのことである．20～30 台もの沢山のカメラを立体的に並べた，Fig. 2 のような大掛かりなスタジオで，人体の各部に多くのマーカーを装着し，これを検出するトラッカーを使って三次元的な動きを３Ｄモーションファイルとして取り込む技術である[7]．モーションキャプチャは「モーキャップ(mocap)」と略称される．

Fig. 2　モーションキャプチャ用スタジオ[7]

このようなスタジオを借りて，自前で格闘技のモーションキャプチャを行ったら高額の費用が掛かるが，幸いネット上に無料のデータも公開されている．今回はこれを利用することにした．

例えば，Mocapdata.com[8]というサイトは，本来は有料でモーキャップ・データを作成，販売するサイトであるが，無料のデータも数多く公開していて有り難い．有料のデータパックは数千ドル（数十万円）もする．無料版でも，「歩く」モーションで 599 個，「空手」76 個，「テニス」73 個，などが公開されている．

３．２．２　モーションの組み込み

ToyStudio[9]というソフトを使い，夜叉に空手のモーションを組み込むことにした．今回は，ToyStudio のシェアウェア版を使用した．フリーウェア版もあるが，機能が限られるためである．

夜叉に，「strong punch」という技を組み込む方法を例として示す．サイト Mocapdata.com にユーザー登録をすると，モーキャップ・データをダウンロードすることが出来る．Biovision 社のファイル形式(.bvh)で，karate-09-punch strong-yokoyama.bvh というファイルをダウンロードした．

ToyStudio の「ファイル」メニュー >>「開く...」で，この bvh ファイルを開く．アニメーションの操作ボタンで，初期ポーズにしておく．

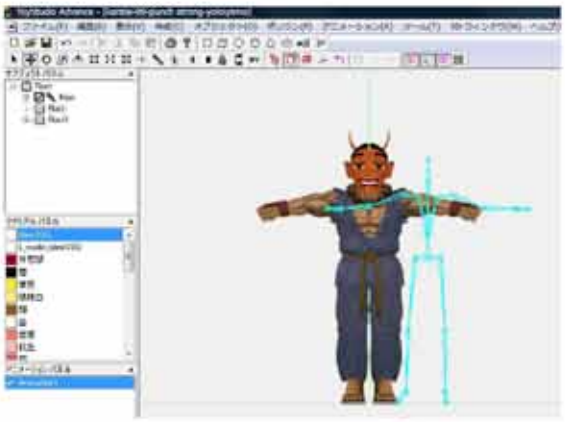


Fig. 3　ToyStudio の画面

次に，夜叉の位置を移動し，モーキャップ・データと位置合わせをする．この後，モーキャップ・データの各ボーン（骨）を動かして，夜叉の対応する部位と位置合わせする．その結果を Fig. 4 に示す．

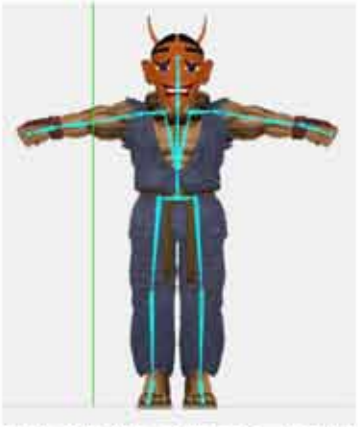


Fig. 4　夜叉の３Ｄモデルとボーンの位置合わせ

このあと「オブジェクト」メニュー >> 「スムーズスキニング」でスキンメッシュ化する．スキンメッシュとは３Ｄモデルのポリゴンデータを対応するボーンに貼り付ける操作である．これにより３Ｄモデルの各部位がボーンの動きに合わせて移動することになる．

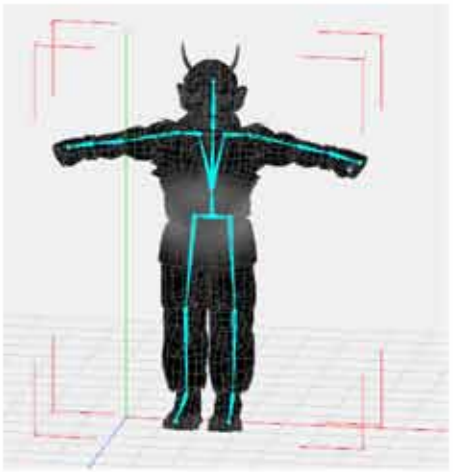
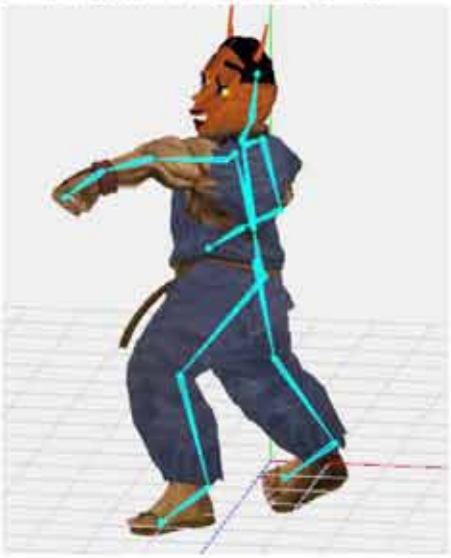


Fig. 5　３Ｄモデルのスキンメッシュ化

3．2．3　空手「技」の組み込み

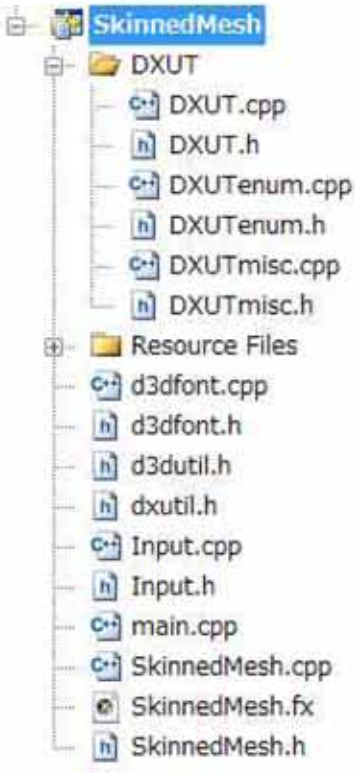
上のようにすると，夜叉にモーキャップの動きを組み込むことが出来る．アニメーションの操作ボタンを使って，動きを確認すると，以下のような動作が組み込まれていることが分かる．



Fig．6　「強いパンチ」のモーション

4．プログラム構成

プロジェクトは主に 10 個のソースファイルと DXUT ライブラリからなる．DXUT ライブラリは 6 個のソースファイルからなり，スキンメッシュ・アニメーションを実行するための DirectX 関連関数一式をまとめたファイル群である．



Fig．7　ゲームエンジンのプログラム構成

4．1　スキンメッシュ・アニメーション

モーション付きのオブジェクトを３Ｄ空間で動かすのがメッシュ・アニメーションだが，特に人間や生物などの軟質系の皮膚や衣服をまとったキャラクタのモーションをプログラムで動かすことはスキンメッシュ・アニメーションと呼ばれている．しかし，DirectX9 のマニュアルを見ても，そのチュートリアルや解説はほとんど無い．Microsoft 社関連では，わずかに Microsoft DirectX SDK¥Samples¥C++¥Direct3D フォルダ内の SkinnedMesh というサンプルプログラムと川西裕幸氏の解説[10]があるだけである．

それ以上に詳しいのが，ハンドルネーム IKD 氏の「ゲームつくろう！DirectX 技術編」[11]である．ほとんど独学で DirectX のスキンメッシュの仕組みを解読して，解説した労作である．本稿のゲームエンジンの開発でも大いに参考にさせて頂いた．

4．2　ボーン操作

ボーンとはスキンメッシュに埋め込まれた骨のことなので，この骨を動かすことにより，その回りにまとわれているポリゴンをスムーズに動かすことが出来る．Fig．5 はキャラクタ「夜叉」をスキンメッシュ化したものであるが，これにより３Ｄモデルの各部位がボーンの動きに合わせて移動することになる．

4．3　フレーム(Frame)とアニメーション

三角形ポリゴンが集まって形成されているのオブジェクトの最小単位を「メッシュ(Mesh)」という．オブジェクトを構成するメッシュは１つの場合もあれば，複数の場合もある．そして，オブジェクトは「フレーム(Frame)」という基本単位で管理されている．

X ファイル内で Frame は FrameTransformMatrix テンプレートと Mesh テンプレートの２つから構成されている（Fig．8）．

Fig．8　フレーム構造体のメンバ構成[11]

このように，フレームはメッシュとそれを座標変換させるローカル変換行列からなる．一方，アニメーションはフレームの中には無く，フレームとは独立に AnimationKey テンプレートの中で定義されている．

５．プログラミングのポイント

５．１　敵キャラクタの攻撃パターン

コンピュータとの対戦をゲームの遊び方の基本とすると，敵キャラクタをプログラムで自動的に動かす必要がある．そこで２つのモードを考えた．

（１）常に一定の間合いを取り，ヒットアンドアウェイの攻撃パターン

（２）ある範囲を勝手に動き回っては攻撃を仕掛けてくるランダム攻撃パターン(Fig. 9)

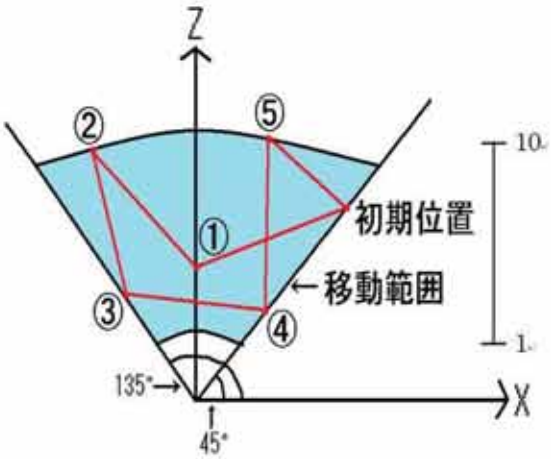


Fig. 9　敵キャラクタの移動範囲とパターン

ステージの中央を座標原点とし，そこを中心とした半径 10mの円形の仮想リンクを設定する．向かい合う２つのキャラクタは，この土俵のような円形のリンク内だけを移動できるとする．但し，このリンクは表示されない．

（１）の場合，プレイヤーの自キャラクタの操作に応じて，敵キャラクタをまずは一定の「間合い」（2mとした）を保つように動かす．自キャラクタが進めば引き，戻れば前に出て攻撃する．リンクの外周，いわば土俵の俵まで追い詰められたら，俵に沿って横に動く．左右どちらに動くかは乱数で決める．

プレイヤーから見た場合，敵が常に一定の間合いを取るように動いたら，攻撃が当たらないことになる．そこで，間合いを取るための敵キャラクタの反応には 0.1 秒の遅れを入れることにした．また，土俵際まで追い詰めれば，左右にしか動けなくなるので，ヒットする可能性が高くなる．

敵キャラクタは，逃げたりかわしたりするだけでは結局は間合いを詰められ，攻撃を受けることになる．そこで，プレイヤーの動きが止まれば，直ぐに逆襲する機能も追加した．これにより，緊張感と迫力が増すゲームとなった．

（２）は，プレイヤーに敵キャラクタの動きを予測させないように，ランダムな動きと攻撃をさせるものである．Fig. 9 のように，敵キャラクタは半径１～10m，角度 45～135° の扇型の領域内を動くとする．ある初期位置からスタートして，①，②，③，④，⑤，･･･の地点をめがけて直線的に移動する．各位置の座標，その間の移動スピード，繰り出す攻撃技の３つは乱数で決める．急に突進してきて強い技を繰り出すことがあるので，自キャラクタが攻撃を受ける可能性が高い．

（１）と（２）の動作・攻撃パターンは 5～10 秒間隔で変更される．この間隔も乱数で変えられるようにした．また，ＨＰ(Health Power)によっても変わる．当初，まだＨＰが高いときは（１）が多めとなり，攻撃が当たりＨＰが低くなると（２）が長めになる．

５．２　モーションの変更

自キャラクタ，敵キャラクタのモーションもＨＰによって変わるようにした．ＨＰが 50 以下になるとヨタヨタした動きのモーションになり，0 になると地面に倒れたモーションになって，一定時間後にゲームオーバー，またはゲームクリアになる．

各キャラクタについて約 20 種類のモーションを作成した．そのうち５種類はモーションキャプチャのデータを組み込み，残りの 15 種類は手作業で作成した．その中から，ガード，ジャンプ，弱攻撃，中攻撃，強攻撃，必殺技，グロッキー，転倒に相当するものを８モーション選び出し，自キャラクタの場合はゲームパッドの各キーに割り当てた．幾つかのモーション中の代表的なポーズを Fig. 10，Fig. 11 に示す．これらはいずれもモーションキャプチャのデータを実装したものである．

また，敵キャラクタの場合は，前節で述べた２つの攻撃パターンの途中で，どのモーションかを乱数で選び，自動的に繰り出されるようにした．ＨＰが 50 以下ではグロッキー，0 になると地面に転倒する．

(a)中攻撃－ストレート

(b)強攻撃－回し蹴り

Fig. 10　夜叉のモーション（１）

(a)必殺技－後回し蹴り

(b)ダメージ→グロッキー

(c)転倒

Fig. 11 夜叉のモーション（2）

5．3 当たり判定

当たり判定はゲームにつきものであるが，シューティングゲームなどではプログラミングの簡単化と処理の高速化のため，キャラクタを外接する直方体，円柱，球などに近似して判定することが多い．DirectX にはそのための関数も用意されている．例えば，D3DXComputeBoundingSphere 関数はメッシュの外接球の中心と半径を計算する関数である．この関数を使って，２つのオブジェクト（メッシュ構造）の中心座標と半径を計算し，両者の中心間の距離が２つの半径の和よりも小さければ「当たり」と判定する．

しかし，格闘ゲームではこの近似は使えない．突き技では手先が，蹴り技では足の先が，相手の体に当たったかどうかの判定をしないと不自然な判定になる．そこで，Fig. 5 のボーン構成のうち，手先のボーンとつま先のボーンの座標を常に計算し，それが相手のキャラクタの外接円柱の内側に届くか否かで，「当たり判定」を行うことにした．

より正確には，キャラクタを頭，胴体，腕，足などのパーツに分け，個々のパーツとの当たり判定を行う必要があるが，それは今後の課題とした．

6． ゲームの構成

6．1 ゲームの概要

本作品は対戦格闘を題材にした 3D 対戦格闘ゲームである．プレイヤーはフィールド上でキャラクタを操作して，対戦相手のキャラクタと勝負して，勝つことを目標とする．一人用のゲームで，対戦相手の行動はプログラムで決定される．

サンプル格闘ゲームでは，自キャラクタ１体と敵キャラクタ２体を用意した．

6．2 操作説明

ゲームパッドは Windows OS と相性の良い Microsoft Xbox 360 用のものを使用した．

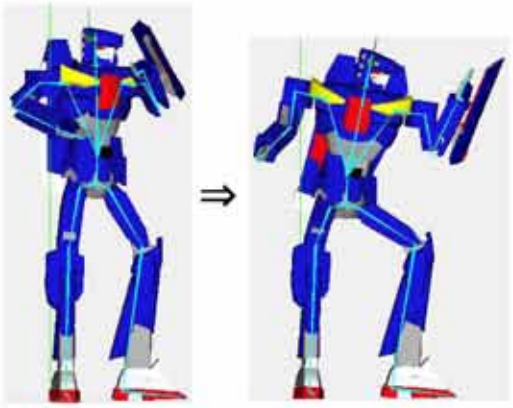
ゲームパッドのボタン配置(Fig. 12)

① 移動:方向キーを左又は右に動かすとキャラクタが入力した方向に動くことができる．
　ステップ：方向キーを相手方向又は反対方向に入力するとその方向にステップ
　しゃがみ：しゃがみ状態でも攻撃可能
② ジャンプ：ジャンプ中はガードができない．
③ 弱攻撃：中攻撃より威力はないが，攻撃の出のスピードが速いのが利点である．
④ 中攻撃：威力は普通であり，攻撃の出も普通である．
⑤ 強攻撃：中攻撃よりも威力があるが，攻撃の出が遅い．
⑥ ガード：どんな攻撃にも耐える．（立ち状態だと下からの攻撃には機能しない）
⑦ 必殺技：強攻撃よりも威力がある，各キャラクタ固有の攻撃である．

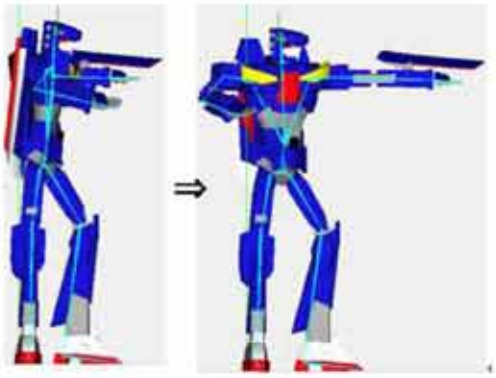
Fig. 12 ゲームパッドのボタン配置

６．３ 自キャラクタのアクション

自キャラクタのモーション例をFig. 13に示す．



(a) 基本動作ガード



(b)弱攻撃

Fig. 13 自キャラクタのモーション例

６．４ タイトル画面とゲームオーバー画面

タイトル画面には作成したゲームのタイトルとスタートボタンを表示する．スタートボタンを押すことでゲームをはじめることが出来る．自分が操作するキャラクタの体力(得点)HP が０になるとゲームオーバーとなる．

タイトル画面とゲームオーバー画面を Fig. 14 に示す．

６．５ ステージ画面

ステージの１例を Fig. 15 に示す．

画面の表示は次のようになっている．

①体力ゲージ：現在の残りの体力 HP である．このゲージがなくなると負けである．

②タイムカウンター：ゲームの残り時間である．残り時間が無くなった場合，残りの体力が多い方が勝ちとなる．

③勝利マーク：ラウンドに勝利すると，勝利マークが表示される．

④キャラクタ：現在使用しているキャラクタである．

⑤必殺技ゲージ：このゲージが満タンにたまると必殺技が使用できる．

⑥得点：現在の得点が表示される．得点は攻撃をヒットさせたら増やすことが出来る．

(a)スタート画面

(b) ゲームオーバー画面

Fig. 14 タイトル画面とゲームオーバー画面

Fig. 15 制作したステージとゲーム画面の例

７．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．

８．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．

また，ゲームエンジンを使うと，プログラムや数学に強くない学生にも迫力のある３Ｄゲーム開発体験を積ませることが出来，モノを作る楽しさや，満足感・達成感を与えることが出来る．

謝辞

４年生の加藤卓也氏には本格闘ゲームエンジンの最初のテストユーザーになって頂いた．キャラクタモデルやステージオブジェクトの提供，モーションの組み込み，敵キャラクタの動きの調整，サンプルゲームの評価など，多大な協力を得た．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] http://ja.wikipedia.org/wiki/モーションキャプチャ

[8] http://mocapdata.com/index.cgi?lang=jpn

[9] http://kotona.bona.jp/

[10] 川西裕幸：" DirectX - Cutting Edge DX9", http://msdn.microsoft.com/ja-jp/library/dd365147.aspx

[11]「ゲームつくろう！DirectX 技術編」，http://marupeke296.com/DXG_No23_AnimationFromXFile.html