３Ｄゲームソフトのための 仮想人間の動作制御ミドルウェアの開発

A Dynamic Motion Control Middleware for Computer Games

尾下 真樹 [1)]
Masaki Oshita

1) 九州大学 大学院 システム情報科学府 知能システム学専攻
（〒812-8581 福岡市東区箱崎 6-10-1 E-mail: moshita@db.is.kyushu-u.ac.jp）

ABSTRACT. A middleware for computer games has been developed to realize dynamic motion control of virtual characters in response to physical interaction between the characters and environments such as collision, gravity and external forces. Currently many computer games cannot generate realistic human motions in such situations because they generate character motions by replaying fixed motion data that are created and stored in advance. The middleware controls the joints of the character based on a motion data given by application program and realizes dynamic motion control such as reducing joint stresses and keeping its balance. We have developed a prototype of the proposed middleware and a demo software is going to be available on the web.

１．背景

近年、パソコンや家庭用ゲーム機の高性能化によって、ユーザの操作するキャラクタが計算機内の３次元仮想空間内を自在に動き回るような、仮想現実指向のゲームが実現されつつある。このようなゲームでは、ユーザの操作に応じて、キャラクタの動作をリアルタイムに生成することが要求される。

しかしながら、現在の技術では、人間が行うさまざまな動作を計算機によって動的に生成することは不可能である。そこで、現在の主なアプローチでは、アプリケーションで必要になる基本的な動作をあらかじめ作成しておき、プレイヤーの操作やゲームの進行状況に応じて連続的に再生することによってキャラクタの動作を生成している。しかし、この方法では、キャラクタは決められた一定の動作しか行うことができないという問題がある。そのため、特に、キャラクタ同士が衝突したり、重い物を持ったり、外部から力を受けたりといった、環境との相互作用が生じた時に、自然な動作を生成することが困難となる（図 1）。例えば、現在のゲームでは、キャラクタ同士が衝突した時に衝突の仕方に関わらず毎回同じ倒れ方で転倒したり、どんなに重い物を持っていても全く同じ動きをしたりといった、不自然な動作が生成されている。

コンピュータゲームにおいて、このようなキャラクタと環境の間の相互作用は非常に重要な要素である。それにも関わらず、このような力学的な影響に応じて、動的に動作を生成するための有効な手法はこれまでは存在していなかった。現在のコンピュータゲームでは、動作データの種類や再生のタイミングを工夫することでこのような不自然さが目立たないようにうまく工夫されてはいるものの、キャラクタの動きが固定されてしまうというのは非常に大きな制約である。今後より自由度の高いゲームを実現するためには、このような問題を解決することが不可欠である[7]。

図 1 力学的な影響に応じた動作生成の例。開発したミドルウェアによって生成された、キャラクタ同士の衝突に応じた動的な動作。

近年、コンピュータゲーム市場では、新たなビジネスモデルとして、ミドルウェアへの関心が高まっている。Play Station 2 や X Box のような高性能の家庭用ゲーム機の出現に伴い、これまでのゲーム機ではできなかった複雑な処理や高度な技術が実現可能となった。しかし、その反面、１つのゲームメーカだけで、高度な技術を駆使したゲームを開発することは非常に困難となっている。そこで、ゲームメーカの開発負荷を軽減するため、サードパーティがある特定の機能に特化したライブラリを開発しミドルウェアとして提供するビジネスモデルが提唱されている。ミドルウェアの具体例として、レンダリングエンジン、キャラクタアニメーション、物理シミュレーション、人工知能、群集モデル、立体サウンドといったものが挙げられる。現在、各種ベンダがこれらのミドルウェアの開発に取り組んでいる。このような

状況において、上記のような動作制御の問題を解決するミドルウェアを開発・提供できれば、コンピュータゲーム市場にとって非常に有益であると考えられる。

２．目的

本プロジェクトは、コンピュータゲームにおけるキャラクタの動作制御のためのミドルウェアの開発を目標とする。本ミドルウェアの特徴は、現在の技術では困難とされているような、衝突や外力などの力学的な影響がキャラクタに加わる状況において、キャラクタの自然な動作を動的に生成する機能を提供することにある。

本ミドルウェアでは、あらかじめ作成された動作データを単純に再生するのではなく、動作データに追従するようにキャラクタの関節角加速度を制御し、動力学シミュレーションによってキャラクタの動作を生成する。つまり、基本的には現在のアプリケーションと同様に、もとの動作データはほぼそのまま再生される。そして、環境からの力学的な影響が加えられた時にのみ、その影響に応じた自然な動作を生成する。その結果、既存のフレームワークや動作データを有効に活用しつつ、衝突や外力などの環境との相互作用に応じた自然な動作を生成することが可能となる。

本プロジェクトの最終的な目標は、開発したミドルウェアを商品としてゲーム開発会社に提供し、実際のゲーム開発に利用してもらうことである。本年度は、その最初のステップとして、ミドルウェアのプロトタイプと、ミドルウェアの機能を示すデモプログラムの開発を行った。

現在の一般的なゲームシステムにおける、キャラクタがあらかじめ作成された動作を再生することしかできないという制約は、自由度の高いゲームを開発する上で非常に大きな障害となっている。そのため、本プロジェクトの提案ミドルウェアは、現在のゲーム開発の状況に大きなインパクトを与えることができると予想される。提案ミドルウェアを早期に製品化することができれば、ゲーム市場におけるスタンダードを獲得することが期待できる。

３．仮想人間の動作制御手法

本節では、本ミドルウェアの基本技術となる動作制御手法について、その特徴や関連手法との比較について述べる。

（１） 背景技術

１節で述べたように、現在のコンピュータゲームでは、あらかじめ作成された動作データを再生することによってキャラクタの動作を生成している。しかしながら、近年では、動作データを単純に再生するだけではなく、インバースキネマティックスといった動作データを動的に変形するような技術も用いられるようになりつつある。このような技術を用いることで、地面の形状に応じて足の位置を動かしたり、拾い上げる物の場所に応じて手を伸ばす位置を動かしたりといった、動作データの動的な変形が可能となる。しかし、このような手法はキャラクタの姿勢にのみ注目して変形を行うものであるため、衝突や外力といった力学的な影響を考慮することは困難である。このような力学的な相互作用に応じた自然な動作を生成するためには、キャラクタの姿勢だけではなく、関節に加わる負荷やバランスなどの動力学的な要素を考慮して制御を行うことが必要になる。

一方、ロボット工学や生体力学の分野では、人間と同様の動作を行うようなロボットの制御手法や、人間の運動モデルに関する研究が進んでいる。しかし、現在の技術では、一般にゲームで必要とされるような多くの動作を自律的に行うようなコントローラを実現することはまだ不可能である。ロボットの制御手法は、人間らしい見ための動作よりも、安定した動作を実現することを目標としている。本プロジェクトのようにアニメーションへの応用を目的とする場合には、むしろ、人間らしい動きをうまくシミュレートしてやることが必要となる。

（２） 関連研究

コンピュータアニメーションの研究分野では、動力学を考慮して自然な動作を生成しようという試みはこれまでにも多く行われてきた。しかし、残念ながら、まだ実際のコンピュータゲームに応用できるような有効な手法というのは存在していないのが現状である。

これまでの主なアプローチとしては、上記のようなロボット工学の制御アルゴリズムをそのままアニメーションに適用した研究が主流であった[1][2]。これらのアルゴリズムでは、実際のロボットを制御する場合と同様に、キャラクタの関節トルクを制御することになる。しかしながら、関節トルクと、関節トルクによって生じる関節角加速度の間には複雑な関係があり、関節トルク空間で動作を制御することは非常に難しい。例えば、ある関節を動かそうとして関節トルクを加える、その関節のトルクは全身の関節に対して影響を及ぼしてしまうため、さらにそれを打ち消すような関節トルクを加える必要がある。ロボット工学の制御手法では、関節相互の影響は無視し、それぞれの関節に比較的強力な補償機能を持ったコントローラを設けることで、各関節を独立に制御するのが一般的となっている。そのため、このような制御手法をアニメーションに単純に適用しただけでは、もとの動作データを単純に追従するだけでも困難であり、衝撃や衝突が加えられた場合などに、人間が行っているような関節相互の影響を考慮した制御を行うことは困難である。

また、近年注目されている別のアプローチとして、時空間制約による動作最適化手法がある[5][6]。この手法では、関節負荷やバランスの誤差を小さくするような評価関数をあらかじめ定義しておき、その値をなるべく小さくするように動作データを関節角度空間で変形して最適化を行う手法である。この手法は、動作データを実行する前の動作プランニングには非常に適している。しかし、ダイナミックな運動中の関節負荷やバランスは主に関節角加速度に影響するため、関節角度軌道により間接的に関節負荷やバランスを制御することになってしまう。そのため、この手法は動作の途中で予定外の衝撃や外力が加えられた時に、それらに応じて能動的な制御を行うのには向いていない。

（３） 提案手法の特徴

このような従来の手法の問題点を解決するため、筆者らは、キャラクタの関節を角加速度空間で制御する新しい手法を開発した[3]。

本手法では、まずは入力された動作データに追従するように関節角加速度を計算する。この段階では、力学的な影響や関節相互の作用は全く考慮しない。そして、環境から力学的な影響（他のキャラクタとの衝突、荷物による重力、外部から押されたり引かれたりといった外力等）がキャラクタに加えられた時には、その影響に応じ

て自然な動作となるように関節角加速度に修正を加える。具体的には、ある関節に筋力の範囲を超えた大きな負荷が加わるような場合には、その負荷を減少するような制御を行う。また、キャラクタがバランスを崩しそうになった時には、全身のバランスを保つような制御を行う。その結果、例えば、全身のバランスを保つように腕や腰を動かしたり、腰にかかる負荷を減少するために腕を振ったり、といった人間が普段行っているような関節相互の影響を考慮した動作制御が実現される。本手法は、キャラクタを関節角加速度空間で制御するという新しいアプローチによって、このような制御を可能とした。さらに本手法では、関節相互の影響に応じた制御を効率的に計算するため、全ての関節を同じように制御するのではなく、人間の体を６つの部位に分類してそれぞれの部位ごとに最小限の自由度によって制御を行う。

４．開発ソフトウェア

本年度のプロジェクトでは、上記の制御手法を実際のコンピュータゲームに応用し、その有用性を確認するため、目標とするミドルウェアのプロトタイプの開発を行った。本節では、今回開発したソフトウェアの内容について述べる。

本プロジェクトで開発するミドルウェアは、最終的には実際の家庭用ゲーム機で動作することが期待される。しかしながら、Play Station 2 などのゲーム専用機で動作するソフトウェアの開発にはそのための特別の開発環境やノウハウが必要となる。そこで、今回のプロトタイプは、開発・実行環境としては Windows を使用し、C++ のクラスライブラリとして実装した。ただし、グラフィックス API に依存する描画関係のクラス以外は ANSI C++に準拠して実装してあるため、他の環境への移植は比較的容易であると考えられる。なお、グラフィックス API としては、OpenGL と Direct 3D の両方に対応している（現在は主に OpenGL を使用しており、Direct 3D はまだ一部の機能は実装されていない）。

（１） 動作制御ミドルウェア

ミドルウェア本体は、利用者のアプリケーションプログラム（ゲームソフト）とリンクされ、キャラクタの動作を制御する機能を提供する。

図 2 に、アプリケーションと提案ミドルウェアの関係を示す。キャラクタの骨格・幾何形状モデルやシーン情報、キャラクタの状態情報などは、アプリケーションとミドルウェアの間で共有される。

アプリケーションは、プレイヤーの操作やゲームの進行状況に応じて、キャラクタに行わせる動作をミドルウェアに入力する。ミドルウェアは、入力された動作をもとにキャラクタを制御する。ミドルウェアの内部は、コントローラとシミュレータの２つに分けられる。コントローラは、アプリケーションから入力された動作データをもとにキャラクタの関節角加速度を制御する。シミュレータは、コントローラから出力された関節角加速度をもとに運動シミュレーションやキャラクタ同士の衝突処理を行い、キャラクタの状態を更新する。

コントローラでは、前節で述べた動作制御アルゴリズムに従い、基本的にはアプリケーションから与えられた動作をそのまま追従するような関節角加速度を計算する。そして、衝突や外力などの環境からの突発的な力学的な影響が加えられた時には、それらの影響に応じた動的な制御を実行する。今回のミドルウェアでは、従来の手法をさらに拡張して、キャラクタが大きくバランスを崩したりしたような時にバランスを保つように足を踏み出したりといったより高度な制御を実現する機能を追加した。なお、コントローラの内部処理のアルゴリズムについては、現在特許出願準備中であるため、本稿では詳しい説明は省略する。

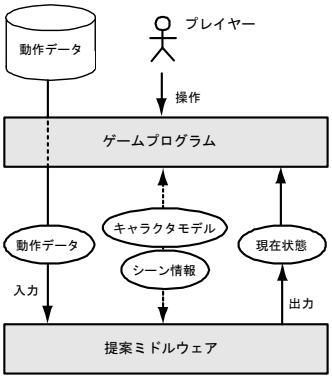

図 2 システム構成

（２） データ変換ツール

本ミドルウェアを実際に利用する際には、そのゲームアプリケーションで使用するキャラクタの骨格・形状モデルや動作データが必要となる。そこで、ミドルウェア本体と合わせて、市販のアニメーションツールを使用して作成したこれらのデータをミドルウェアが利用できる形式に変換するためのソフトウェアを開発した。

既存のアニメーションツールでは、モデル・モーションデータは独自のバイナリ形式で保存されることが多く、一般にファイルフォーマットは公開されていない。従って、自分のプログラムでこれらのデータを直接読み込むことは難しい。代わりに、これらのアニメーションツールでは、プラグインと呼ばれるプログラムをユーザが作成してアニメーションツールの中で実行させることで、任意の形式でデータをエクスポートできるような仕組みが提供されている。そこで、この仕組みを用いて変換処理を実現した。

キャラクタモデル・モーションデータを編集するためのメインのアニメーションツールとして、今回は、3ds max + character studio を採用することとした。3ds max + character studio は、3ds max という汎用的なアニメーションツールと、character studio という２本足のキャラクタの動作編集に特化したサブ機能の組み合わせである。実際にいくつか使ってみたアニメーションツールの中では高機能であり、また、映画やコンピュータゲームのためのキャラクタアニメーションの作成に広く使われているソフトウェアであるため、このソフトウェアを採用することにした。また、Poser や FiLMBOX といった

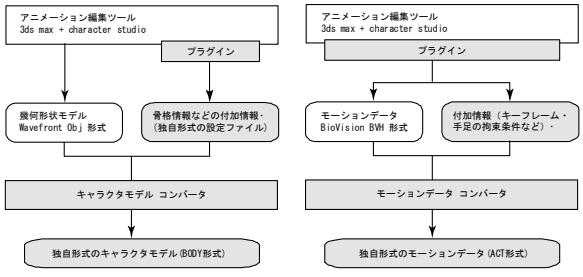

図 3 データ変換ツールの構成

アニメーションツールも補助として使用した。

キャラクタモデル・モーションデータの変換においては、複数のアニメーションツールを用意に利用可能とするため、全ての変換処理をプラグインとして実装するのではなく、プラグインでは一度中間形式に出力し、別に開発するコンバータによって最終的なキャラクタモデル・モーションデータに変換する２段階の方式を取ることとした（図 3）。図中でグレーの部分が今回開発したソフトウェアや独自のファイル形式である。この方式により、中間形式から最終的なフォーマットへの変換処理は共通化できるため、それぞれのアニメーションツールのためのプラグインは最小限の手間で作成できる。将来的には、他のアニメーションツールからのデータ変換にも対応することも予定している。また、本方式では、それぞれのプラグインは一般的な形式や単純な形式でデータを出力するため、他の用途にも再利用が期待できる。

また、ミドルウェアで使用する動作制御アルゴリズムでは、キャラクタの関節に加わる負荷の判定するための基準として、比較的単純な筋力モデルを使用する。その筋力モデルのパラメタを設定するため、市販のモーションデータ集に含まれるモーションデータを解析してパラメタを推定するようなプログラムを作成した。

（３） テストプログラム

先述のように、ミドルウェアはアプリケーションプログラムと協調して動作する。そこで、開発したミドルウェアの動作テストを行い、またその有効性を示すためのテストプログラムを開発した。本ミドルウェアの機能を示すためには、キャラクタが単体で動き回るだけのプログラムよりも、キャラクタ同士が激しく衝突したりすることで互いの運動に影響し合うようなゲームが望ましい。そこで今回は、プレイヤーがキャラクタを操作して敵キャラクタを倒すような簡単なアクションゲームを目指して開発を行った。

しかしながら、時間的な都合のため、プロジェクト期間内ではゲームと呼べるほどの完成度のソフトウェアを作成することはできなかった。今回作成したテストプログラムでは、プレイヤーはキャラクタを操作していくつかの簡単な動作を実行させることができる。また、マウス操作によってキャラクタに外部から衝撃を与えて動作の変化を確認できるような機能を実装した。これらの機能により、キャラクタ同士を衝突させたり、キャラクタの動作中に衝撃を与えたりすることで、ミドルウェアの機能によって動的に生成される動作を確認することができた。

５．アプリケーション

本節では、開発したテストプログラムによる、ミドルウェアの簡単な適用例を示す。図 4 は、テストプログラムの実行中の画面である。図 5 に、テストプログラムで生成されたキャラクタの動的な動作制御の例を示す。それぞれの動作例では、衝撃が加えられた時のキャラクタの姿勢や、衝撃の位置・大きさに応じて、さまざまなリアクションが生成されている。また、図 1 はキャラクタ同士の衝突に応じて生成された動的な動作の例である。

従来のゲームでは、基本的にキャラクタは一定の動きを繰り返すことしかできなかった。そのため、あるキャラクタが別のキャラクタを殴ったりした時には、毎回同じ反応が生成されていた。一方、本ミドルウェアを利用することで、従来の同種のアクションゲームゲームと比較して、例えばキャラクタ同士が衝突した時などに、衝突の仕方などに応じた自然な動作が実現されていることが分かる。

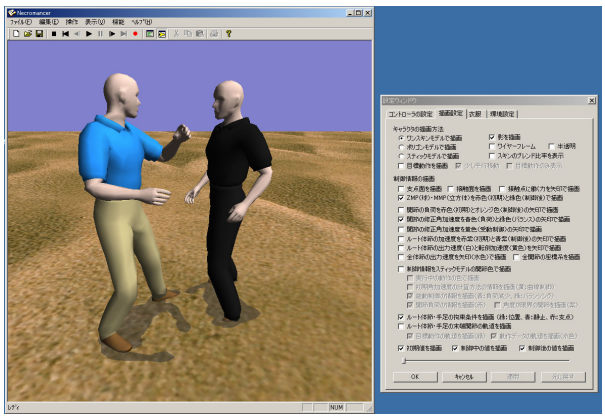

図 4 テストプログラムの実行画面

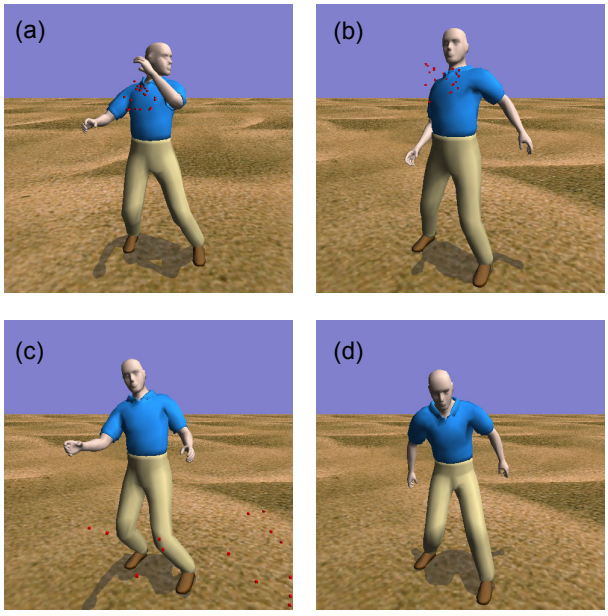

図 5 衝撃に応じた動的な動作制御の例。(a)(b)衝撃に応じた制御。(b)バランスを崩した時に左足を後ろに踏み出してバランスをとるような動作。(c)バランスを大きく崩し尻から着地するような動作。

今回開発したテストプログラムは、まだデモゲームとしては十分な完成度には達していないいものの、ミドルウェアの機能や有効性を確認するという当初の目的は十分に達成できたと考えられる。

６．今後の予定

本節では、プロジェクトの今後の発展、及び、社会への成果の還元のための計画について述べる。

（１） デモアプリケーションの公開

今後は、今回開発したテストプログラムを発展させ、デモアプリケーションとしてウェブ上で公開することを予定している。これは本プロジェクトの開発成果を広く知ってもらうことが目的であるため、無償で公開する。

現在のテストプログラムでもミドルウェアの最低限の機能は可能であるが、開発成果としてアピールするためには、単にミドルウェアの機能を示すだけではなく、

デモアプリケーションを見た人に面白いと感じてもらえるものにする必要がある。そこで、現在、テストプログラムを発展させ、実際に簡単なゲームとして遊べるような内容にするための改良（ゲームシステムの追加、キャラクタ・動作データの作成、ゲームバランスの調整など）を行っている。

なお、開発ソフトウェアや技術情報の公開は、下記のWWW ページを通じて行う予定である。

成果公開用 WWW ページ：http://www.e-motion.ne.jp

（２） ミドルウェアの提供

本プロジェクトの最終目的としては、開発したミドルウェアをゲーム製作会社に提供し、ゲーム開発に使用してもらうことが目標である。

しかしながら、実際問題として、個人レベルで製品開発やサポート、商品化のための作業の全てを行うことは難しい。そこで、本ミドルウェアを実用化するため、提案ミドルウェアの製品化に協力してくれるパートナー企業、提案ミドルウェアを利用して実際にゲーム開発を行ってくれるゲームソフト会社を探したいと考えている。これから公開予定のデモアプリケーションに興味を持ってくれるゲームソフト会社が見つかれば、詳しい内容説明や、実際に使用する側の要望のヒアリングなどを行った上で、ミドルウェアの本格的な提供のための準備を進めていきたい。

今回開発したミドルウェアは、どちらかといえばまだ研究用のプロトタイプであり、制御アルゴリズムやシミュレーション処理などにもまだ不確定な処理が残っている。実際にゲーム開発に使用してもらうためには、インターフェースや中身を整理し、ゲームに必要な機能（例えば高速な描画機能など）も追加していく必要があると考えられる。また、ドキュメントの整備も必要となる。今回開発したプロトタイプを実際のゲーム開発に利用するためには、最短でも半年～１年は必要になると考えている。そこで、開発成果の公開や宣伝と平行して、ミドルウェアの整理・開発作業も継続していく予定である。

（３） ライブラリの一部・プラグインの公開

今回のプロジェクトで開発したプラグインや一部のクラスライブラリは、ミドルウェアの一部としてだけではなく、それぞれ単体でも有用なツール・ライブラリとして利用可能であると考えられる。例えば、今回開発した 3ds max のモーションデータを BVH 形式で出力するためのプラグインは、3ds max や他のアニメーションツールを利用しているユーザにとってはモーションデータのコンバータとして活用できる。また、キャラクタの基本的な骨格モデル関連のクラスや、動作データ関連のクラスなどは、キャラクタアニメーション関係のプログラムを作成する上で必要となる機能であるため、これらを公開することは多くのプログラマにとって有用であると思われる。そこで、最終的なミドルウェアに先行する形で、これらのプラグイン・クラスライブラリを公開することを考えている。具体的な内容・公開方法（有償 or 無償）・ライセンス形式などについては現在検討中である。

（４） 論文発表

提案ミドルウェアの実現のために開発した技術は、コンピュータグラフィックス関連の学術論文誌・国際会議などで、研究論文として発表していく予定である。

コンピュータグラフィックスの応用分野では、映画業界に代わる新たな市場として、コンピュータゲームに代表されるリアルタイムアニメーションの分野が非常に注目されつつある。特に、コンピュータゲームについては、最近は国際会議のメインテーマにも取り上げられたり、専門の国際会議が開催されたりと盛んに研究が行われている。

本ミドルウェアで扱っているような動力学を考慮した動作制御手法に関しては、その必要性については認識されていながらも、これまで決定的な手法というのはなかった。今回開発した技術は、研究としても十分新規性があるため、研究論文の発表などを行っていきたいと考えている[4]。

７．まとめ

本プロジェクトでは、コンピュータゲームにおいて、キャラクタの自然な動作を生成する機能を提供するためのミドルウェアのプロトタイプを開発した。また、簡単なデモプログラムを開発にすることにより、その機能と有用性を示した。今後は、今回開発したテストプログラムをゲームソフトとして発展させて、WWW を通じて無償で公開する予定である。デモアプリケーションを多くの人に見てもらうことによって、今後の改良のための意見を収集しつつ、ミドルウェアの実用化に協力してくれるゲームソフト会社などを探していきたいと考えている。

参考文献

[1] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos, "Composable Controllers for Physics-Based Character Animation", SIGGRAPH 2001, pp. 251-260, 2001.

[2] Jessica K. Hodgins, Wayne L. Wooten. David. C. Brogan, and James F. O'Brien, "Animating Human Athletes", SIGGRAPH '95 Proceedings, pp. 71--78, 1995.

[3] Masaki Oshita and Akifumi Makinouchi, "A Dynamic Motion Control Technique for Human-like Articulated Figures", Computer Graphics Forum (EUROGRAPHICS 2001), Vol. 20, No. 3, pp. 192-202, September 2001.

[4] Masaki Oshita and Akifumi Makinouchi, "A Dynamic Motion Control Middleware for Computer Games", SIGGRAPH 2002 Sketches and Applications, Conference Abstracts and Applications, San Antonio, Texus U.S.A., July 2002 (to appear).

[5] Zoran Popovic, and Andrew Witkin, "Physically Based Motion Transformation", SIGGRAPH '99 Proceedings, pp. 11-20, 1999.

[6] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. "Efficient Generation of Motion Transitions using Spacetime Constraints", SIGGRAPH '95 Proceedings, pp. 147-154, 1995.

[7] 中野修一, “PlayStation2 はビジュアルを革命するか – ミドルウェアの展開”, pp. 12-16, Oh!X 1999 年春号, 1999 年 5 月.