芸術科学会論文誌 Vol. 1 No. 1 pp.22–29

モーショングラフ: 自律型キャラクタの動作生成手法

齋藤 豪　†井元 崇之　†中嶋 正之

東京工業大学 精密工学研究所
†東京工業大学 情報理工学研究科

Motiongraph: a Technique of Action Generation for an Autonomous Character

Suguru SAITO　†Takayuki IMOTO　†Masayuki NAKAJIMA

Precision and Intelligence Laboratory, Tokyo Institute of Technology
†Graduate School of Information Science and Engineering,
Tokyo Institute of Technology

仮想世界内で自律的に行動できるキャラクタのアニメーションを生成するために、一連の動作すべてを事前に決定しておくことはできない。そこで、キャラクタの様々な状態をノードとし、動作の断片をリンクとするモーショングラフを定義し、任意の次の状態が指示された場合に、自動的にパスを決定して一連の動作を生成する手法を提案する。 さらに重み付きパス探索の手法を用いることにより柔軟なアニメーション生成を可能にする。

1　はじめに

自律的なキャラクタは仮想世界中に活気を生み出し、ゲームの中では主役以外の重要な役割を演じる。それらキャラクタが自律的に様々なアクションを起こすにはどのような仕組みが必要であろうか。映画のシーンを作るためのアクションであれば、レンダリング前に全ての動作を定義しておけば良い。しかし、自律的なキャラクタではソフトウェアの実行時になって初めて動作が決定されるので、一連の動作を一繋がりにして事前に用意しておくことは不可能である。そこで本稿では、少ない動作の断片を用意し、状況に応じた自然な動作になるように繋ぎ合わせる手法を検討する。以下、第二節では動作生成に関する従来研究を述べる。第三節ではモーショングラフの定義、作成方法について述べ、第四節ではモーショングラフの一部を改良するために急ぎ値を定義する。第五節ではモーショングラフの動作生成結果とリンクパス探索経路を述べ、第六節ではモーショングラフの意味付けを議論し、第七節ではまとめを行う。

2　従来研究

キャラクタに動作生成を行わせるには、動作を時間の関数とみなし、基本動作を用意した後、係数を変化させることによりバリエーションを与える手法や、ダイナミックスを利用する手法、様々な一連の動作を保存しておき、それを再生する手法がある。

関数的手法 [1][2][3] は、動作に微妙な変化を与えたい場合に、係数の値を調整することでそれが可能な一方で、不自然でない動作を再現するための関数定義が困難であったり、状況に応じて自律的に動作決定を行う際の選択が課題となる。

ダイナミックスを利用した動作生成手法では、状

態遷移図を用いた動作の概略に基づき、物理法則に従い適応的に姿勢を決定する手法がとられる。しかしながら自然な姿勢を生成するには、キャラクタの物理モデルの定義やモデルと状態遷移との調整等が難しく、多くが移動動作の生成の研究に留まっている [4][5]。従って現状では自律的なキャラクタに対する動作生成には課題が多い。

動作を保存・再生する手法としては、モーションキャプチャ装置で取り込んだ動作をつなぎ合わせる方法、セルアニメーションの製作で用いられるキーフレーム法が考えられる。これらの手法では不自然な動作が生成された場合、動作の時系列データの編集や新たなキーフレームの挿入をすれば良いという柔軟性がある。本稿では、キーフレーム手法の柔軟性に着目し、キーフレームを多量に用意し、それらを必要に応じて呼び出して使うことにより多様で柔軟な動作生成を可能とするモーショングラフを提案する。

モーショングラフは状態遷移図の一種であるので、どの程度の時間的粒度を一つの状態として用いるかは動作制御の上で重要な問題である。文献 [6][7][8] の提案手法では、一つの動作を一状態として定義している。この場合、一つの動作に微妙な変化を与えるような動作の中間の細かい指定が難しいと考える。Perlin ら [9] は動作を動作名称のレベルから関節角の変化量のレベルまで階層化して定義しているので、動作を細かく設定できる反面、動作生成時における動作指定の仕方が複雑になってしまう。本稿の提案手法では、姿勢を表すキーフレームを基本の状態として用い、動作指示の簡便性と動作の修飾指定のバランスを取る。

3　モーショングラフ

3.1　モーショングラフの定義

今回は動作の保存形式として、三次元キャラクタのポーズを離散的に用意し、それらをキーフレームとするキーフレーム法をベースとする。従来のキーフレーム法では、一つの動作に対して連続キーフレームが一つ保存されており、そのキーフレーム間の結合は静的であった。このデータ形式では、ある動作

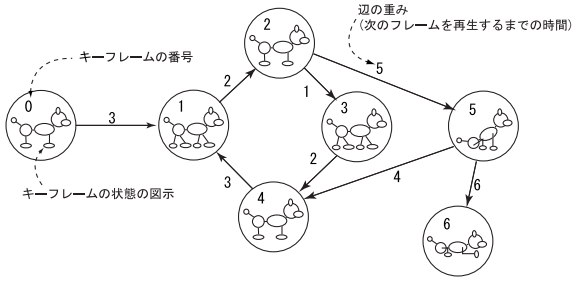


図 1: モーショングラフの概念図

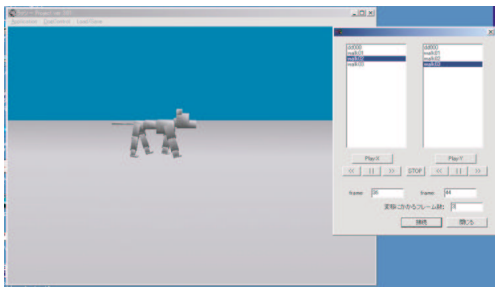


図 2: モーショングラフ作成支援ソフトウェア

から別の動作へ移るには、一つの連続キーフレームの再生を中断するか、再生終了後に次の連続キーフレームを先頭から再生する以外に動作を結合する方法はない。そこで、キーフレームの結合を有向グラフとしてあらわし、連続キーフレームの途中から別の連続キーフレームの途中へでも、滑らかに接続可能な場合には、リンクを張れるデータ構造を作り、これをモーショングラフと呼ぶ。データ構造の概念図を図 1にあらわす。モーショングラフのノードはキーフレーム番号とデフォルトで選択されるリンクを、リンクは次のキーフレームまでの再生時間の情報を持つ。従来のキーフレーム法をラインであるとすると、モーショングラフはネットであると言える。

3.2　モーショングラフの作成法

まず、キャラクタ骨格の三次元リンク構造体を用意する。次にモーショングラフの作成を以下の手順で行う。

1. 市販のアニメーションツールを用いて連続キー

フレームのデータを作成する。この状態ではまだデータはライン結合である。

2. 今回作成したモーショングラフ作成支援ソフトを用いて、接続しても不自然でないノード間にリンクを張り、変移時間を入力する。この作業でデータがネット結合になる。
   図 2にそのソフトのインタフェースを示す。右側のダイアログは連続キーフレームデータを二つ選択するもので、左のウィンドウは再生確認用である。このツールを用いて、不自然でないキーフレーム同士を選び、リンクを新規に作成する。

3. テスト再生をしてみて、リンク、変移時間の調整を行う。

3.3 モーショングラフによる動作生成

動作生成の前に、まずグラフの最短経路長計算アルゴリズムであるウォーシャル- フロイド法 [10] を用いてモーショングラフの経路長マトリックスをあらかじめ用意しておく。

アニメーション生成は、二つのモードから成っている。

一つはデフォルトの動作を続けるモードである。歩いている場合に歩き続ける、立っている場合は立ち続けるなど、新たな動作の指令が来ない場合にこのモードが使われる。このモードではモーショングラフ上をデフォルトのリンクを選択して状態遷移が行われる。

もう一つのモードは新たな動作の指令が来た場合である。新たな動作指令はモーショングラフ中のノードを一つ指示されることで行われる。このような命令は行動を決定するエージェントプランナが発行することを想定している。指令を受けると、現在の動作状態と指定されたノード間最短パスを求める。パスに含まれるノードに付随したキーフレームとリンクに付随した所要時間を使って三次元モデルの姿勢や関節角を次々に線形補間することにより、一連の動作を生成する。

以上のようにモーショングラフではキャラクタに対する目的の動作命令を動作による状態変化後のキーフレームを指示するという単純な方法で実現している。

4 急ぎ値

4.1 急ぎ値の定義

前節で定義されたモーショングラフでは、指令された状態、すなわち目標ノードが与えられたときに最短時間でたどり着ける経路を求めた。よって、現在の状態と目的の状態が同じであれば、常に一つの動作しか選択されることはない。しかしながら、最短時間で目的の状態へ移ることがキャラクタに常に求められているとは限らない。例えば、図 3のように立ち止まり方にも緊急な場合とそうでない場合があり、また図 4のように伏せであってもキャラクタの気分によって変わることがある。従って、目標の状態への経路をなんらかの形で選択できることが望ましい。そこで状態の変化の仕方に注目して “ 急ぎ値” という新しい値を特徴的なリンク “特徴辺” に付加する。急ぎ値の定義は次のとおりである。

急ぎ値 キャラクタがどれだけ目的の動作をはやく実行したいかを 1〜100 までの整数値で表す。急ぎ値の値は相対的変数で、数値が大きい方がよりはやく目的の動作を実行したいと考える動作をする。

図 3、4の例では、「より早く止まりたい」「より早く伏せたい」場合に急ぎ値が大きくなる。ゆっくり歩いていて、急ではなく徐々に速く走る状態に変移する場合には、急ぎ値に小さな値を設定する。

4.2 急ぎ値の設定

図 4の伏せの例では二つの伏せ方があった。この場合、急ぎ値はモーショングラフの分岐後のリンク部に付加される。概念的には図 5であらわされる。

4.3 急ぎ値付きの経路探索法

急ぎ値付きのモーショングラフで動作の指令を与えるには、前節での指令方法と違い、目標ノードに加えて急ぎ度 C(0〜100) を必要とする。

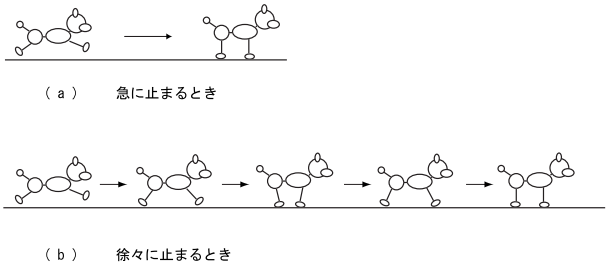


図 3: 走る動作から、立ち止まる動作への変移

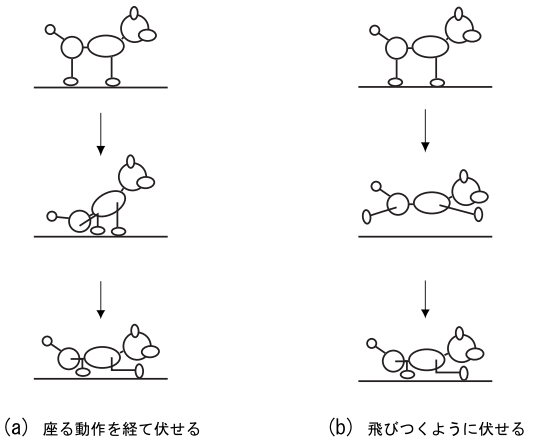


図 4: 立ち動作から、伏せる動作への変移

そして、急ぎ値付きモーションググラフの経路探索では、以下のように動的にリンクの重み aC_i を変化させた後、重み付き最短経路探索を実行する。

今、リンク i に対して c_i を次のように定義する。

$$c_i = \begin{cases} c & :\text{ノード x と y 間が特徴辺で} \\ & \text{あるとき、その急ぎ値 } c \\ 0 & :\text{辺 xy が特徴辺でない場合} \end{cases} \quad (1)$$

ここで、c_i とリンクを通過するためにかかる再生時間 a_i を使い、リンク i の重み aC_i を次のように定義する。この重みを用いて最短経路を探索する。

$$aC_i = \begin{cases} (\ C - c_i\ )^2 a_i & :c_i \neq 0 \wedge a_i \neq 0 \\ (\ C - c_i\ )^2 & :c_i \neq 0 \wedge a_i = 0 \\ a_i & :c_i = 0 \end{cases} \quad (2)$$

急ぎ値による動的な重み変更は、指令が与えられる毎に重みの再計算を必要とし、最短経路長計算アルゴリズムを実行しなければならないことを意味するが、ある目標ノードへの経路を一つの変数の導入だけで変化させることが可能となる。

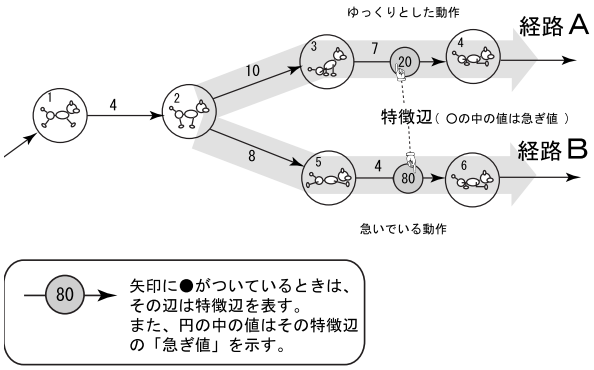


図 5: 急ぎ値付きモーショングラフ

5　再生実験

ここでは急ぎ値付きモーショングラフを用いて実際に二つの動作の再生実験を行う。

はじめにアニメーションツールで図 6で示される 4 つの移動動作のキーフレーム連続データを用意する。次に 3.2節で示したツールを使い、ポーズが似ているキーフレーム間を結合する。複数のリンクを持ったノードには、指令がない場合に移るノードへのリンクを一つデフォルトリンクとして設定する。グラフ中で複数のパスへ枝分かれする部分では、部分的なアニメーション再生を行い、それを見て動作の変わり方の急ぎ具合を主観的に判断し、リンクに急ぎ値を設定する。こうして作られたモーショングラフを図 7に示す。異なった速度での移動動作は個々にループを形成している。

現在のキーフレーム番号が 2 のとき、新しい指令が無ければ、デフォルトリンクをたどり続け、**walk01** の歩行ループを繰り返してキャラクタは歩き続ける。現在のキーフレーム番号が 0 で、走れという指令、すなわちノード番号 16 から 20 へ移動せよと命ぜられた場合はどうなるか。3.3節で説明を行った単純モーショングラフでは最短時間経路を通って目標ノードへ遷移する。急ぎ度付きの動作指令を受けると動的

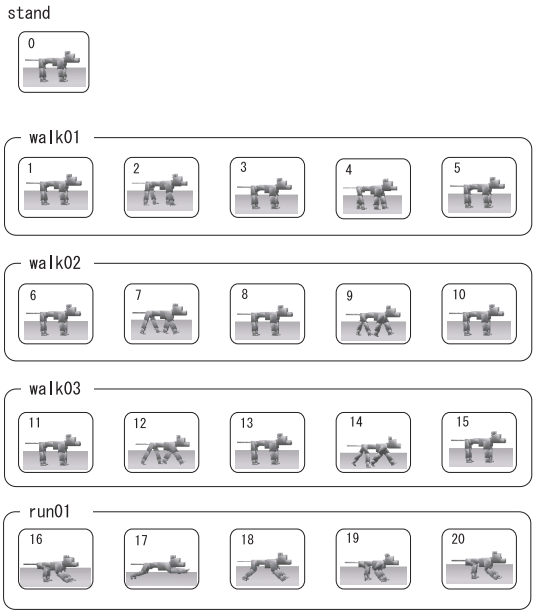


図 6: 歩行動作および走行動作のキーフレーム

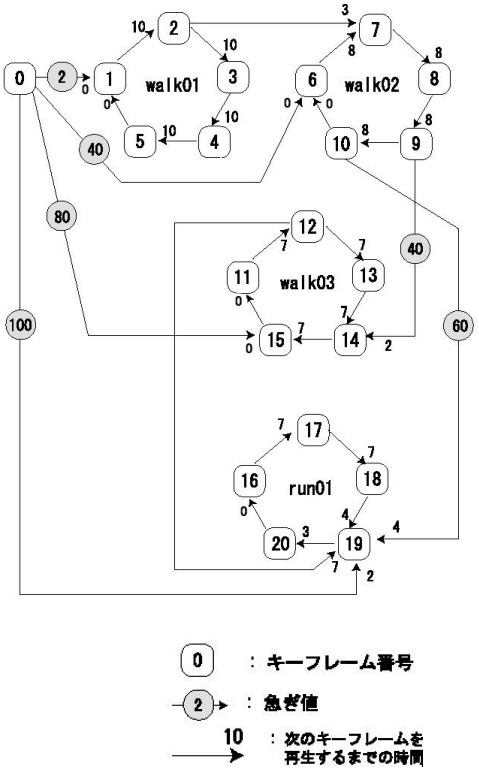
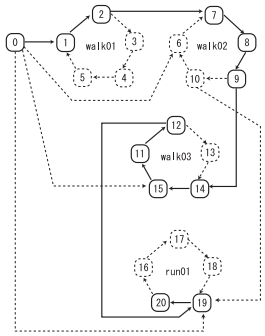
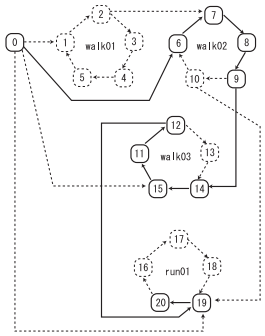


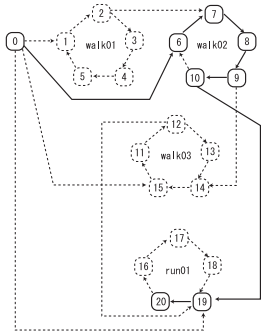
図 7: 移動速度変化用のモーショングラフ



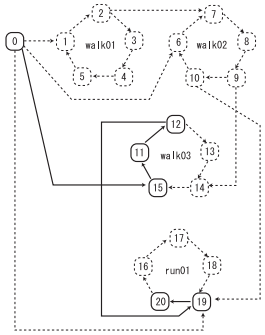
経路 (a) (急ぎ度 1〜20)



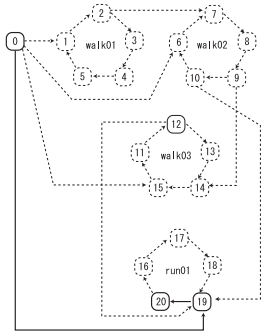
経路 (b) (急ぎ度 21〜51)



経路 (c) (急ぎ度 52〜59)



経路 (d) (急ぎ度 60〜91)



経路 (e) (急ぎ度 92〜100)

図 8: 急ぎ度による経路の変化

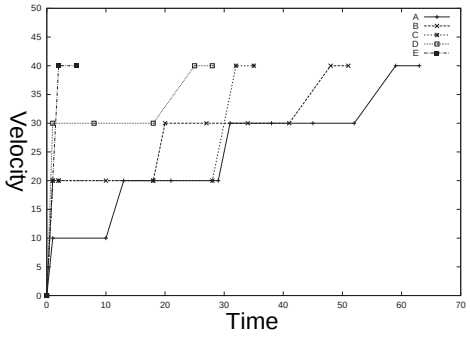


図 9: 急ぎ度による経路変化と速度変化の関係

に変化する重みに依存した最短経路を通って遷移するので、図 8で示されるように急ぎ値に応じて経路が変化する。急ぎ度を大きくして走れの指示を出すと急に全速力で走りだし、小さくして指示を出すとゆっくりと走り出すように経路が選択されていることがわかる。

図 9に経路変化と速度変化の関係を示す。速度はキーフレーム間のコマ数とキャラクタの水平移動量から割り出している。最終速度までの速度の変化は図 7 のモーショングラフが十分に規模が大きくないため滑らかではないが、最終速度になるまでにかかる時間は急ぎ度に対して単調な変化であることがわかる。従って、急ぎ度により希望する状態になるまでの時間の制御に成功しているといえる。

実際のアニメーションの違いを図 10と図 11に示す。縦軸は時間を表す。

二つ目の再生例として、4.2節の説明で挙げた立った状態から伏せに移るアニメーションの例を示す。用いるモーショングラフは図 12である。アニメーション結果は図 13のようになる。急ぎ具合に応じて飛ぶように伏せる場合と、後ろ足から伏せる場合を選択することができている。早送りのようなに再生速度を単純に変えるだけでは実現できない動作の選択性を実現できていることがわかる。

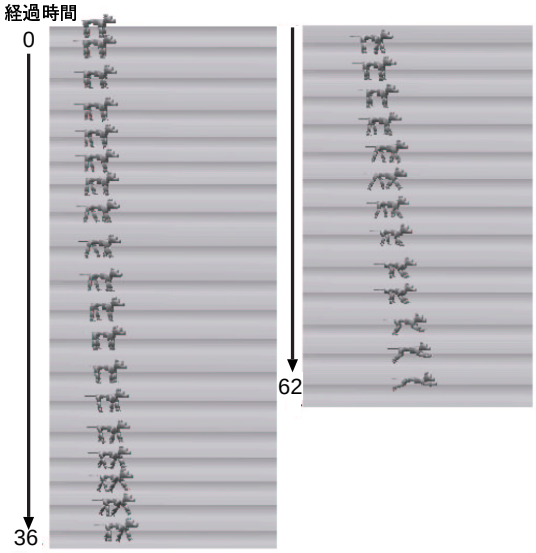


図 10: 経路 (a) の生成アニメーション

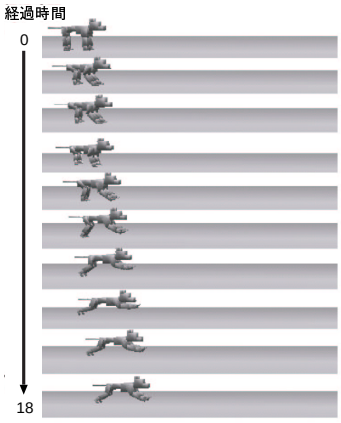


図 11: 経路 (e) の生成アニメーション

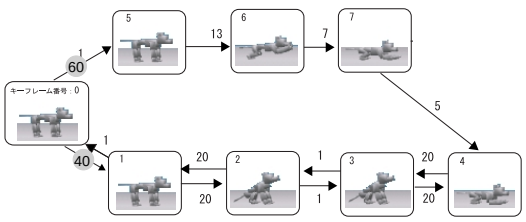
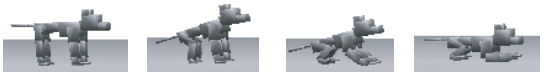


図 12: 伏せる動作用モーショングラフ

6 議論

本節では、モーショングラフとは自律的キャラクタにとってどのような位置づけなのか、二つの観点



ゆっくり伏せる場合 (急ぎ度 1〜49)

勢い良く伏せる場合 (急ぎ度 50〜100)

図 13: 伏せる動作アニメーション

から考える。

6.1 動作に対する修飾の仕方

一つの例として、止まれに対する指示の修飾の仕方を考えてみる。止まれと指示されるとき、「急いで止まれ」のように、加速度を指定される言い方と「1 メートル先で止まれ」のように停止状態を指示されるのとではどちらが動作を行うための基本的な命令なのだろうか。停止状態を指定された場合にはそれを実現するための加速度を逆計算しなければならない。従ってこの例から動作に修飾を与えるには完了時の条件を追加するのではなくて、動作の仕方の変化具合を追加することがより基礎的であると考えられる。このことから前節での「急ぎ値」は動作の変化具合を与えるものであるので動作に修飾を与える方法としては基礎的な手法の実現方法であることがわかる。

では、「1 メートル先で止まれ」のような完了時の条件付きの命令を動作の仕方の変化具合の指示で実現するにはどうしたらよいであろうか。「止まれ」の場合には、(1) 何時から制動動作に入るか、(2) どれだけ急いで止まるか、が分かれば条件を満たすように止まることができる。モーショングラフの状態遷移を管理しているエンジンに対して適したタイミングで急ぎ度付きの停止状態のノードを指示すれば、原理上望んだ位置で止まることが可能である。

実際には適したタイミング、急ぎ度の決定は容易ではないであろうが、初めて運転する車の場合に人間も制動距離の把握は経験して上達することから、そこには学習プロセスの導入が有効ではないかと考えられる。

6.2 モーショングラフと物理シミュレーション

一般にキーフレーム法では、動的な物理シミュレーションは必要でない。連続キーフレーム自体が不自然でない時系列を構成しているためである。キーフレーム法を拡張した現在のモーショングラフでも物理シミュレーションの利用は行っていない。

ここで例えば、滑べりやすいところをヨタヨタと歩く動作の生成について考えると、その実現にはモーショングラフを使った二つの手法が考えられる。一つ目は単純なヨタヨタ歩くキーフレームを準備する手法である。二つ目は、まず普通に歩くモーショングラフを準備し、状態遷移させようとする。次に物理シミュレーションを行って、希望する姿勢が取れるか判定を行い、無理な場合には予定したノードとは異なった姿勢のノードへ遷移先をシフトさせてしまう。予定したノードが変化したので新しいパスの探索計算を行い、次の状態遷移をさせようとする。以下は繰り返し、といった手法である。

一つ目の手法に比べ二つ目の手法では、モーショングラフの数の増大を抑えられる利点が得られそうである。しかし最も重要なのは、二番目の手法ではモーショングラフの状態遷移エンジンへの指示は、あくまで希望する動作だけを与えれば良い点である。キャラクタに動作指令を出す側とすれば、外的要因まで考慮してモーショングラフの経路選択に指示を出すことは実際には非常に困難である。

従って外的要因に依存した動作の変化は物理シミュレーションを加えることで実現することが望ましいと考えられる。

また、このように外的要因による動作変化をモーショングラフに入れないとするならば、モーショングラフを主体的に動かす動作の記憶の仕方であると位置付けることができる。この点は伊藤らの研究 [11] においても近い考察が行われている。

7 終わりに

本稿ではエージェントキャラクタ用の動作生成手法としてモーショングラフの提案を行った。モーショングラフは、単純で機械的な命令発行で柔軟な動作の指示を与えることができる手法である。自律的キャラクタの動作命令に関してや、アニメーションの振付けの指示に関しては従来から研究がなされていたが、動作の詳細の指示まで考慮しつつ簡便な動作指示が行える実際的な実装に関する提案は筆者らの知る限りなされてこなかった。今後は、Funge[12] らの示すようなエージェントの階層モデル、およびその階層間の API についてより検討を行う予定である。

また、ノード間の類似度の自動判定を検討し、リンクの自動生成を行う手法、動作をキャラクタの身体全体でなく、部分に分割してモーショングラフを作成する手法の研究を行っていきたいと考えている。

参考文献

[1] Munetoshi Unuma et al. “Fourier Principles for Emotion- based Human Figure Animation”, Proc. SIGGRAPH95, pp. 91 – 96, 1995.

[2] Armin Bruderlin, Lance Williams “Motion Sigranl Processing”, Proc. SIGGRAPH95, pp.97–104, 1995

[3] 服部元史, 西澤 俊, 田所論, 高森年, 山田和人 “文楽人形のモーション・キャプチャ・データと舞踊譜との相互変換におけるコンピュータ内での身体運動の記述形式”, IPSJ Symposium Series, vol.2000, no.17, pp.9–15, 2000.

[4] Jessica K.Hodgins et.al “Animating Human Athletics”,proc. SIGGRAPH95, pp. 71–78, 1995.

[5] Joseph Laszlo, Michiel van de Panne, Eugene Fiume “Limit Cycle Control And Its Application To The Animation Of Balancing And Waling”, proc. SIGGRAPH96, pp. 155–162, 1996.

[6] Justine Cassell et. al.“ANIMATED CONVERSATION: Rule-based Generation of Facial Expression, Gesture & Spoken Intonation for Multiple Conversation Agents”,Proc. SIGGRAPH94, pp.413–420,1994.

[7] Tsukasa Noma, Liwei Zhao, Norman I. Badler. “Design of a Virtual Human Presenter”, IEEE Computer Graphics & Applications, Vol. 20 No.4, 2000.

[8] Christian Babski, Daniel Thalmann “Real-Time Animation And Motion Capture in Web Human Director(WHD)”, Web3D & VRML 2000 Symposium, 2000.

[9] Ken Perlin, Athomas Goldberg “Improv: A System for Scripting Interactive Actors in Virtual Worlds”, Proc. SIGGRAPH96, pp.205–216, 1996.

[10] 酒井善則, 植松友彦 “情報通信ネットワーク”, 昭晃堂, 1999.

[11] 伊藤淳, 吉田典正, 北嶋克寛 “キーフレーム及びイベント駆動を用いたバーチャルヒューマンの歩行”, 情報処理学会研究報告 99-CG-95, pp.7 – 12.

[12] John Funge, Xiaoyuan Tu, Demetri Terzopoulos “Cognitive Modeling: knowledge, Reasoning and Planning for Intelligent Characters”, Proc. SIGGRAPH99 pp.29–38, 1999.