グラフィクスとCAD 91－8
（1998. 8. 28）

VRMLヒューマノイドアニメーション用動作ストリーム符号化の一手法

樋尻 利紀　西谷 和博　望月 義幸
中 俊弥　浅原 重夫

松下電器産業株式会社 マルチメディア開発センター
〒571-8501 大阪府門真市大字門真1006

インターネット上で3次元CGを扱うためのモデリング言語であるVRMLにおいて、人間のような骨格構造を持つ3次元CGキャラクタを定義するための仕様が、VRMLコンソーシアムにて、ヒューマノイドアニメーション Ver.1.0(H-Anim)として、既に標準化されている。この仕様に加え、我々は3次元CGキャラクタの動作ストリームデータを符号化するための一手法を提案する。これによって、3次元CGキャラクタの動作データを、電話回線のような狭帯域のネットワーク上でもリアルタイムで送受信が可能となる。また、動作データをストリームデータとして送受信することにより、動作再生までの待ち時間をなくすことができる。

A Motion Data Stream Compressing Method for VRML Humanoid Animation

Toshiki HIJIRI, Kazuhiro NISHITANI, Yoshiyuki MOCHIZUKI,
Toshiya NAKA and Shigeo ASAHARA

Multimedia Development Center, Matsushita Electric Industrial Co., Ltd.
1006, Kadoma, Kadoma City, Osaka, 571-8501, Japan

In VRML, a modeling language for 3D CG on the internet, the specification to realize lifelike movement of a 3D CG character with skeletal structure (such as a human) has been standardized as the VRML Humanoid Animation Ver. 1.0 in the H-Anim WG of the VRML Consortium. To extend to this specification, we suggest a method of compressing the motion data stream for a 3D CG character. By this method, it is possible to send/receive motion data in real time on a network with narrow band width such a telephone line. Moreover, by sending the motion data with the streaming data from server to client, the time required before playback can be greatly reduced.

1．はじめに

インターネット上で 3 次元 CG(Computer Graphics）を扱うためのモデリング言語である VRML(Virtual Reality Modeling Language)[1][2]において、人間のような複雑な骨格構造を持つ形状を定義するための仕様が、“ヒューマノイドアニメーション Version 1.0” として既に標準化されている。ヒューマノイドアニメーションでは、人間の骨格構造における各関節データを階層的に持つ構造になっており、これを用いることで人間などの複雑な多関節構造を持つ形状をリアルに動かすことができる[3][4]。

この仕様に加え、我々は 3 次元 CG キャラクタの動作ストリームデータを、電話回線などの狭帯域のネットワーク上でもリアルタイムに伝送するための、『動作ストリームデータ符号化の一手法』を提案する。これにより、一般に用いられているパソコンなどのコンピュータ同士を電話回線などで接続するクライアント･サーバーシステムにおいて、3 次元 CG キャラクタの動作データをリアルタイムに送受信することが可能となる。また、従来は、動作再生に至るまでに、データ量の大きい動作データファイルをすべて受信する必要があったが、動作データを時間軸に沿って、ストリームデータとして送受信することにより、動作再生までの待ち時間をなくすことができる。

この技術は、CG キャラクタを用いた放送系コンテンツを可能とする基礎技術となる。

2．ヒューマノイドアニメーション

ヒューマノイドアニメーションにおいて、3 次元 CG キャラクタを定義する一般的な表現方法としての一例を挙げ、ヒューマノイドアニメーションについて簡単に説明する。

人間のような骨格構造を持つ形状のキャラクタを、図 1 に示すように定義した場合について考える。そして、図 1 の骨格構造を階層的に表現したものを図 2 に示す。なお本稿では、断りのない限り、図 1 のように定義した 3 次元 CG キャラクタを用いており、すでにクライアント側にその形状データが存在しているものとして説明をする。

このように定義した各関節に対して、ルート(体の基準位置、図 1 の 1 参照)の位置ベクトルの情報や、ルートの方向ベクトルと回転角度の情報、さらにルート以外の各関節(図 1 の 2～22 参照)に対して回転軸となる 3 次元ベクトル(回転軸ベクトル)とその周りの回転角度の情報等を与えることによって、3 次元 CG キャラクタをリアルに動かすことができる。

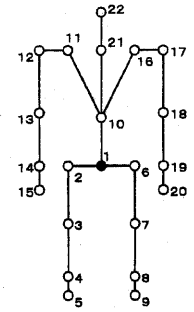

図 1　3 次元 CG キャラクタの骨格構造例

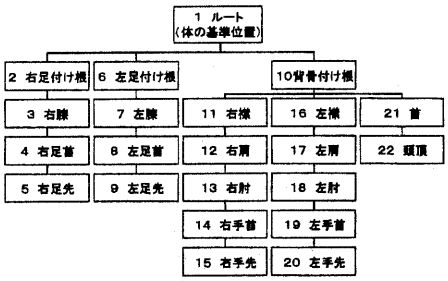

図 2　3 次元 CG キャラクタの骨格階層構造例

3．動作ストリームデータとその送受信方法

3 次元 CG キャラクタに対する位置ベクトルの情報や回転軸ベクトル、回転角度の情報等を、動作ストリームデータとして定義し、その動作ストリームデータをクライアント･サーバー間でリアルタイムに送受信する方法について説明する。

まずサーバー側で、3 次元 CG キャラクタの動作を表現する入力データ(一般的に用いられているモーションキャプチャ等で得られたデータ)を解析し、図 3 に示すように、①ルートの位置ベクトルの情報、②ルートの方向ベクトルと回転角度の情報、③ルート以外の関節での回転軸を表す角度ベクトルとその周りの回転角度の情報、の 3 つに分類する。この理由は、ルートの位置ベクトルと方向ベクトルには、精度が必要であっ

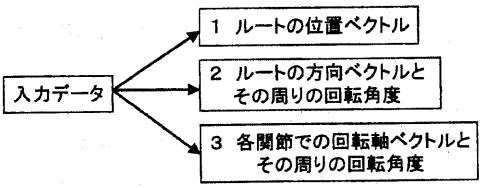

図3 動作データの分類

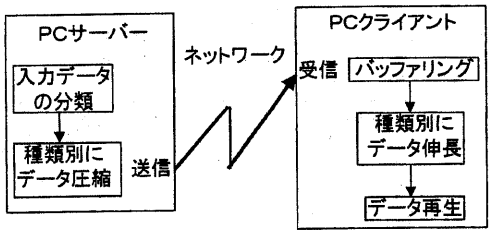

図4 システムの全体構成例

てあまり圧縮することはできないが、その他の関節では、かなりデータを圧縮することができる(第6章参照)。よって、それぞれのデータに対して、異なる圧縮方法を適用するために、動作データを分類しておく。

そして、これらルートの位置や方向を表すベクトル及び各関節の回転角度を表すデータそれぞれを、チャネルと呼ぶことにする。たとえば、図1のキャラクタの場合について考えてみると、ルートの位置ベクトルを表すチャネル、ルートの方向ベクトルと回転角度を表すチャネル、ルート以外の21関節分の回転軸ベクトルと回転角度データを表す21チャネルがあり、合計23チャネルを持つことになる。また各チャネルデータの詳細について、①ルートの位置ベクトルは、x、y、zの3成分を持つ3次元ベクトル(SFVec3f)であり、②ルートの方向ベクトル及び③ルート以外の関節でのベクトルデータは、回転軸となるx、y、zの3成分を持つ3次元ベクトルと回転角度を表す1成分を持った、合計4成分を持つデータ(SFRotation)である。

図4に、本システムの全体構成例を示す。図4に示すように、サーバーにおいて、まず動作の入力データを解析し、①~③のデータに分類する。その後、チャネル総数と、各チャネル毎の情報として、チャネル識別子、チャネルデータサイズ、チャネル名、チャネルタイプ等の情報をサーバーからクライアントに対して、前もって必ず一度は送信しておく必要がある。これらの情報を含んだパケットを、"チャネル定義パケット"と呼ぶ。チャネル定義を変更したい場合には、その時点で、新たなチャネル定義パケットを送信すればよい。

クライアントでは、このチャネル定義パケットを受信、解析して、チャネル識別子とそのチャネルの動作データを持った、"データパケット"の受信に備える。

ここで、このデータパケットについて、もう少し詳しく説明する。各チャネルデータを任意に定められた時間で1ブロックとするような、時間軸方向での"ブロック化"を行う。たとえば、1秒単位でブロック化する場合、1秒分の動作データが1ブロックとなり、その1ブロックが1データパケットとなる。サーバーでは、このデータパケット毎にデータを圧縮し、送信する。圧縮方法の詳細は、第4章で述べる。

このブロック化されたデータパケットには、ヘッダ情報として、ある基準時からの時刻を表すタイムスタンプを付加する。このタイムスタンプをサーバーとクライアントが正確に管理することによって、サーバーとクライアント間で同期をとることができ、リアルタイムにキャラクタの動きを再生することが可能となる。なお本手法では、入力データは前もってサーバーに入力される場合と、モーションキャプチャ等でリアルタイムに入力される場合の、どちらにも対応できる。

4. 動作ストリームデータ圧縮/伸長方法

第3章で述べたように、動作ストリームデータのデータパケットを定義する。ここでは、このデータパケットに対してのデータ圧縮/伸長方法について説明する。データ圧縮は、電話回線などの狭帯域のネットワーク上を想定して行うものである。

データ圧縮方法は、図3に示すような3種類の動作データそれぞれに対し、個別の圧縮方法を適用する。以下、それぞれの圧縮方法について詳しく説明する。

まず、①ルートの位置ベクトルに対する圧縮方法について説明する。データ成分の詳細は第3章でも述べたように、3次元空間内での位置情報x、y、zの3成分である。このデータに対し、時間軸方向の圧縮を行う。一例として、図5に、3次スプライン補間を行う場合の、時間軸方向のノード点抽出の様子を30frame/blockと15frame/blockについて示す。図5において、30frame/blockの場合、ブロック内のノード

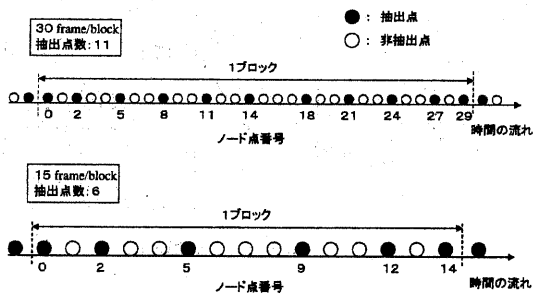

図5　3次スプライン補間のノード点抽出

点に時間順に0～29の番号を付けた時には、0、2、5、8、11、14、18、21、24、27、29といった11点に当たるノードを抽出し、抽出されたノードのデータのみ送信し、それ以外のノード点のデータは送信しない。この場合のノード点の選び方は、不等間隔にしてある。これは、3次スプラインの特徴として、境界付近では微係数を0と仮定するため、これに実際のデータが引っ張られるのを防ぐためである。

また、15frame/blockの場合も同様であり、ノード点番号で、0、2、5、9、12、14といった6点に当たるノード点を抽出する。さらには、6frame/blockの場合も同様に考えられる。また、精度を重視しない場合には、線形補間などを用いても良い。ただし、時間軸方向の圧縮をする場合、補間方法を識別するための情報をチャネル定義パケットに、"圧縮率識別子"として付加する必要がある。

次に、②ルートの方向ベクトルとその周りの回転角度に対する圧縮方法について説明する。データ成分の詳細は、3次元方向ベクトルのx、y、zの3成分と回転角度の合計4成分である。このうち回転軸を表す3次元ベクトルを正規化し(ベクトルの大きさを1とする)、極座標表現に変換することにより、2成分(γ, θ)で表現することができる。すなわち、全体で4成分から3成分への空間軸方向の圧縮となる。また、これら3成分はすべて角度データあり、通常は4バイトの浮動小数値で与えられており、0～2πの値をとる。さらに、量子化を行い、4バイトの各成分をそれぞれ1バイトで表す。

この②ルートの方向ベクトルとその周りの回転角度のデータに対しては、時間軸方向の圧縮は行わない。これは、体の姿勢を表現するための角度の自由度が大きいため、2πの不定性による姿勢運動の不連続性が生じるのを防ぐためである。ルート以外の他の関節では、人間の動作として骨格構造上、ある程度制限が加わるため、急激に回転軸ベクトルの方向が変化することは少ないが、ルートの方向ベクトルは、急激に変わり得る。そのため、時間軸方向に対してスプライン補間等を適用した場合、スプラインの性質上、意図する補間ができないことが多い。よって、②には時間軸方向の圧縮はせず、空間軸方向の圧縮と量子化のみとする。

圧縮方法として最後に、③各関節での回転軸ベクトルとその周りの回転角度について説明する。データ成分の詳細は、②ルートの方向ベクトルとその周りの回転角度と同じで、3次元方向ベクトルのx、y、zの3成分と回転角度の合計4成分である。このデータに対しては、前記①と②に適用した圧縮方法を共に適用する。すなわち、時間軸方向の圧縮により、まずノード点抽出を行うことで、送信するデータ量を減少させ、次に空間軸方向の圧縮により、4成分から3成分に圧縮する。さらに、各成分に対して、4バイトから1バイトへの量子化を行う。

続いて、クライアント側でのデータ伸長方法について説明する。

まず、チャネル定義パケットの圧縮率識別子により、①～③のデータそれぞれがどの補間方法でデータ伸長すればよいかを決定する。たとえば、3次スプライン補間で、データが30frame/blockの場合、11ノード分のデータを基に、30ノード分のデータに伸長しなければならない。このため、クライアント側で、このデータ伸長にかかる計算時間を考慮して、データパケットのバッファリングを行う必要がある。具体的には、データ再生プロセスと、データの受信･バッファリング･解析･伸長プロセスの2プロセスを、クライアント側で同期をとりながら実行する。したがって、リアルタイムで動作データを再生するためには、図6に示すように、実時間のデータ再生中に、別のプロセスで次のブロックのデータパケットを受信し、データ伸長を行い、次のブロックの再生準備をしておく必要がある。

また当然、空間軸方向の圧縮、すなわち3次元ベクトルの3成分から極座標表現の2成分に圧縮したベクトルデータ②及び③も、極座標変換の逆変換により3成分に伸長し、さらに角度データの成分として、4バイトから1バイトに量子化されているデータも、4バ

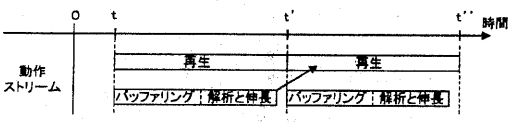

図6　データの再生プロセスと伸長プロセス

イトにデータ伸長を行った後、再生プロセスに動作データとして、データを渡す。

5．動作データ圧縮率の評価

本章で、動作ストリームのデータパケットのデータサイズを、非圧縮の場合と第 4 章で述べた方法による圧縮の場合について、比較してみる。ここでは、動作入力データについて、位置ベクトルデータの 3 成分、及び回転軸ベクトルとその周りの回転角度データの 4 成分は、各 4 バイトの浮動小数値として表されているものとする。また、ブロック化は 1 秒単位で行っている。この時、単位としては、[byte/block]= [byte/sec]である。

30frame/block、非圧縮のデータパケットのデータサイズを UC_{30}[byte/sec]、圧縮したデータパケットのデータサイズを C_{30}[byte/sec]とすると、

(1)非圧縮の場合:

(位置ベクトル) = 4[byte]×3[成分] = 12[byte]

(角度データ) = 4[byte]×4[成分] = 16[byte]

(2)圧縮の場合:

(位置ベクトル) = 4[byte]×3[成分] = 12[byte]

(角度データ) = 1[byte]×3[成分] = 3[byte]

$$UC_{30} = \{(\text{位置ベクトル})\times 1 + (\text{角度データ})\times(\text{関節数})\}\times 30 = \{12\times 1+16\times 22\}\times 30 = 10{,}920[\text{byte/sec}]$$

$$C_{30} = \{(\text{位置ベクトル})\times 1 + (\text{角度データ})\times(\text{ルート以外の関節数})\}\times 11 + (\text{角度データ})_{ルート}\times 1\times 30 = \{12\times 1+3\times 21\}\times 11+3\times 30 = 915[\text{byte/sec}]$$

同様に、15frame/block、非圧縮のデータパケットのデータサイズを UC_{15}[byte/sec]、圧縮したデータパケットのデータサイズを C_{15}[byte/sec]とすると、

表1　動作データ圧縮率

	30frame/block	15frame/block
非圧縮 [byte/sec]	10,920	5,460
圧縮 [byte/sec]	915	495
圧縮率[%] (圧縮/非圧縮)	8.4	9.1

$$UC_{15} = \{(\text{位置ベクトル})\times 1 + (\text{角度データ})\times(\text{関節数})\}\times 15 = \{12\times 1+16\times 22\}\times 15 = 5{,}460[\text{byte/sec}]$$

$$C_{15} = \{(\text{位置ベクトル})\times 1 + (\text{角度データ})\times(\text{ルート以外の関節数})\}\times 6 + (\text{角度データ})_{ルート}\times 1\times 15 = \{12\times 1+3\times 21\}\times 6+3\times 15 = 495[\text{byte/sec}]$$

表 1 に、動作データの圧縮率をまとめる。表 1 からわかるように、第 4 章で述べた方法で動作データを圧縮すると、非圧縮のデータを 1/10 以下に圧縮することができる。

6．考察

6.1　圧縮によるデータの誤差

まず、角度データに対する、4 バイトから 1 バイトへの量子化によって生じる誤差について考える。1 バイトは、256 段階を表現でき、角度は 360 度であることを考えると、量子化による誤差は、360/256 で、約 1.4 度ということになる。ヒューマノイドアニメーションのキャラクタ定義では、図 2 に示すように、ルートからの階層構造をとっているため、量子化誤差が累積することになるが、図 2 のようなキャラクタでは最大 6 階層となっている。したがって、この場合、関節の先端部で最大約 8.4 度の誤差が生じることになる。しかし、これは先端部までのすべての関節に誤差を含むという最悪の条件が揃った時のみ、起こることであり、通常は8.4度の誤差として現れることはほとんどない。パソコンなどのコンピュータの画面上にある、人間のようなキャラクタの手先や足先の 5 度程度のずれは、

精度を要求される場合以外、通常のキャラクタアニメーションでは、さほど問題にならない。

また、スプライン補間等の時間軸方向の誤差については、やはりデータが急激に変化する部分に対しては、その変化の度合いに応じて誤差を含んでしまう。しかし、動作自体は連続であるため、動作に不自然さを感じることはない。また、人間のようなキャラクタということに限定すれば、各関節の動きはある程度制限されるため、第 4 章で述べたように、ルートの方向ベクトルに対してのみ、時間軸方向の圧縮を避けることで、人間の動きとして、問題なく滑らかな動きを再生することが可能である。

人間の骨格構造を持つ 3 次元 CG キャラクタに対して、動作再生を実行している例を図 7 に示す。

6.2 回線の状態やシステムの性能による動作データの適応的選択

本システムの特徴として、データの伝送回線や、クライアントのコンピュータの性能により、圧縮率の異なる動作データを適応的に選択するということが可能である。

表 1 に示したように、表 1 だけでも圧縮／非圧縮、30frame/block と 15frame/block のそれぞれの組合せで、4 種類のデータが存在する。これらのデータ以外にも、用途に応じて、たとえば量子化はせずに時間軸方向のみの圧縮といったデータも用意できる。伝送回線やクライアントのマシン性能に余裕がある場合は、非圧縮のデータを送受信すれば良い。そうでない場合には、システムの性能に見合ったデータを適応的に選択することによって、動的な変化に対応した、動作データのリアルタイム再生を実現できる。

また、表 1 の圧縮したデータの場合、現状一般的に用いられている 28.8kbps といった転送速度のモデムでも、十分に転送可能である。

7. まとめ

本稿では、VRML を中心とするインターネット上での 3 次元 CG モデリング言語において、人間のような骨格構造を持つ CG キャラクタをリアルに動かすための仕様“ヒューマノイドアニメーション”を拡張し、その動作データを圧縮し、ストリームデータとして送受信する手法について述べた。

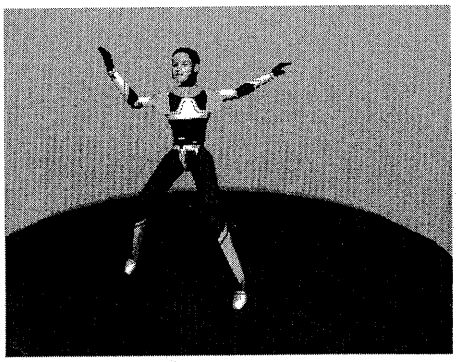

図 7　3 次元 CG キャラクタの動作再生実行例

ネットワークを介して、人間のような骨格構造を持つ 3 次元 CG キャラクタの動作データをリアルタイムに伝送、再生する場合、その動作データに対して、時間軸方向、空間軸方向、量子化といった圧縮方法を、システムの性能に応じて適用することによって、専用回線から電話回線のような狭帯域のネットワークに至るまで、幅広く対応できる。

なお、本稿では、ヒューマノイドアニメーション用として述べてきたが、骨格階層構造を持つ形状であれば、人間のような形状でなくても、本手法を適用できる。

今後は、本方式を既存の音、画像ストリームと併用することで、VRML ストリーミングの応用を広める必要がある。

参考文献

[1] VRML 2.0 Specification (http://www.vrml.org).

[2] Jed Hartman, Josie Wernecke, “The VRML 2.0 Handbook”, Addison Wesley Developers Press, 1996.

[3] Toshiya Naka, Yoshiyuki Mochizuki, “WonderSpace : Interactive 3D Animation Browser”, SIGGRAPH'97 Visual Proceedings, p111, August 1997.

[4] Bernie Roehl, “Specification for a Standard VRML Humanoid Version 1.0”, Humanoid Animation WG, August 1997, (http://ece.uwaterloo.ca:80/~h-anim/).