7P−1

# 対話的なアニメーション・デザイン・ツールの構成

李 明苑 杉野隆司 国井利泰†

クボタコンピュータ（株）応用技術部
†東京大学理学部情報科学科

## 1 はじめに

現在、多数のアニメーションがアプリケーションごとに生成されている。したがって、生成された物体に関するデータは各アプリケーションに依存し、他のアプリケーションがそれを利用するのは別の処理を伴うことになる。また、同じ物体に対してもアプリケーションに依存するためにデータが重複され、システムの利用が非効率的である。その上、エンド・ユーザがアニメーションを生成するためには、プログラム言語とグラフィックスシステムの知識を要求する。このような従来のアニメーションの方法を改善し、ユーザ・フレンドリーなシステムを提供するために本論文ではユーザが容易にアニメーションが生成できるアニメーション・デザイン・ツールの構成方法とそのデータ構造について論じる。

## 2 動きを伴う物体のデータ構造

アニメーション・デザイン・ツールは次の機能を含まなければならない。

- 物体の位相と幾何に関するデータの生成
- 物体の動作の定義
- 物体の位相と幾何を変更する機能
- 動作を変更する機能
- 物体とその物体の動作の検索（表示）

上記の機能を備えるためには、各物体に対して二種類の関係を示すデータ構造が必要である。一つは物体を構成する幾何要素間の関係で、例えば、物体とセグメント間の関係、セグメントとループ間の関係、ループとエッジ間の関係、エッジと頂点間の関係等が含まれる。もう一つは関節を持っている物体の場合、セグメント間の関係である。このような関係を含むデータ構造を次の関係式に基づいて構成する。

$$Motion = \sum_t Object_t \quad (1)$$

$$Object_t = \sum_i Segment_i \quad (2)$$

$$Segment_i = \sum_j Loop_j \quad (3)$$

$$Loop_j = \sum_k Edge_k \quad (4)$$

$$Edge_k = \sum_l Vertex_l \quad (5)$$

式 2にはセグメント間の関係が含まれている。セグメントのデータは、物体の論理的構造、即ち、物体のセグメント間の関係を示す木構造にしたがって格納される。すべてのフレームにおける物体の位相と幾何に関する情報は次のようなデータ構造に基づいて保管される。

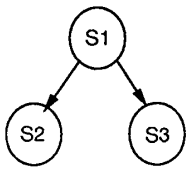
```
typedef struct   _Object {
        int        object_id;
        Segment **segment;
} Object;

typedef struct   _Segment {
        int    segment_id;
        Loop **loop;
} Segment;

typedef struct   _Loop {
        int    loop_id;
        Edge **edge;
} Loop;

typedef struct   _Edge {
        int      edge_id;
        Vertex **vertex
} Edge;

typedef struct   _Vertex {
        int    vertex_id;
        float x;
        float y;
        float z;
} Vertex;
```

図−1は三つのセグメントから構成される物体のセグメント間の関係を表している。各ノードはセグメントを、

---

Interactive Animation Design Tool
*Myeong Won Lee*[1], *Takashi Sugino*[1] and *Tosiyasu L. Kunii*[2]
[1]*Kubota Computer Inc.*, [2]*The University of Tokyo*

各アークはセグメント間の関係を示す。上記のデータ構造に基づいて図−１の物体を構成すると、セグメントＳ１、Ｓ２、Ｓ３の順番で物体の位相と幾何に関するデータが構築される。これによってアニメーション中のセグメント間の関係が維持できる。




（図−１）セグメント間の関係の例

## 3　アニメーション・デザイン・ツールの構成

アニメーション・デザイン・ツールは次のサブシステムから構成されている。

- 物体ジェネレータ：物体の位相と幾何データを求める。ユーザによって対話的な方法 [3] で入力された物体に関する情報は、アニメーション・デザイン・ツールによって処理できるように第 2章のデータ構造に自動的に変換される。このサブシステムによって物体の論理的構造と物体のセグメントの処理の順番が表現できる。

- キーフレーム・ジェネレータ：これは物体の動作の一部分を生成するサブシステムである。ユーザによって指定された時間での物体の動作を定義し、その時間におけるキーフレームを生成する。これは、物体の位相と幾何を変換できるパラメータを対話的に設定することによって行なわれる。

- 内挿フレーム・ジェネレータ：キーフレーム間のすべてのフレームを自動的に生成するサブシステムである。すべてのフレームに表示される物体の位相と幾何変換のために、パラメータ補間関数 [2] が含まれている。

- 動作コントローラ：動作の表示と制御を行なうサブシステムである。ユーザによって与えられた時間間隔での物体の動作を表示し、物体の動作の中で一部分の動作を検索する。また、物体の動作を制御するための様々な操作を加えることができる。

## 4　動作のためのパラメータの設定

本章では動作を定義するためのパラメータを対話的に設定する方法について記述する。

- 回転パラメータの設定：物体の回転の動作を表すためには、まず物体の各関節での自由度に従う回転軸を決定することが必要である。回転軸は各関節におけるローカル座標系を基にして設定する。各フレームにおける回転パラメータは次の構造によって与えられる。

```
typedef struct _Axis {
        int   kind;
        int   type;
        float degree;
} Axis;
```

上記のデータ構造に基づいてユーザによる複数の回転軸の設定が可能である。

- 位置パラメータの設定：ユーザによって各フレームにおける物体の三次元空間での位置を示す座標値を設定する。これはグローバル座標系における物体の位置データである。各座標値は次のデータ構造によって指定される。

```
typedef struct _Pos {
        float x;
        float y;
        float z;
} Pos;
```

- タイミング・パラメータの設定：物体の動作を表示する時、実物のように見えるためには動作の速度を制御する機能は重要な要因である。各フレームにおけるタイミング・パラメータを設定するためにはキーフレームの選択の方法、また内挿フレームの数を決めることが必要である。そして、回転と平行移動のパラメータはタイミング・パラメータの関数として表す。

## 5　おわりに

本研究ではアニメーション生成のためのユーザ・インタフェースの設計方法を提案した。これは、クボタコンピュータ（株）のＡＶＳ [1] を利用して開発している。ユーザは回転、平行移動、タイミング・パラメータ等を指定することによって容易にアニメーションが生成できる。今後の課題としては動作の制御のための様々なアルゴリズムを開発してアニメーション・デザイン・ツールに結び付けることが期待される。

## 参考文献

[1] *AVS(Application Visualization System) User's Guide & Developer's Guide,* Stardent Computer Inc., 1989.

[2] Scott N. Steketee and Norman I. Badler, *Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control,* Proceedings of SIGGRAPH, vol. 19, no. 3, pp. 255-262, 1985.

[3] 李 明苑、国井利泰、稲葉朋生、四次元ジオメトリックモデラによる対話的アニメーション生成に関する研究、第６回ＮＩＣＯＧＲＡＰＨ論文コンテスト論文集、pp. 171-180, 1990.