3N−6

時間評価アクターを用いた宣言型アニメーション言語

馬場博巳　藤原俊行　乃万司　岡田直之
九州工業大学 情報工学部

1 はじめに

アニメーション言語の研究は，記述の抽象度を高めアニメーション制作者の負担を軽減する方向へ進んでおり，アクターがタスクレベルの指示に応じて適当なプランニングを行ない動作する (Task Planning)，アクターが周囲の環境を認知し自律的に動作する (Behavioral Animation) 等の手法が考えられている [1]．このようなアクターを用いることにより，アニメーション制作者は，例えば，イベント駆動の形式で必要最小限のイベント (動作) 間の連鎖を指定することにより，ストーリーを持つアニメーションを制作することができる [2]．

しかしながら，ストーリーには「太郎と花子が橋で出会った」など偶発的な状況も含まれており，このような場合，イベント駆動の形式のみでアニメーションのストーリーを記述することは困難である．

そこで，我々は動作時間の評価可能なアクターを用い，スケジューリングの機能をとりいれたアニメーション言語とそのシステムを提案する．本言語においては，イベント間の時間関係は宣言的に記述される．なお，本言語は，アニメーション制作者が直接用いる場合のほか，我々がすすめているマルチメディアの理解，生成，ならびに変換に関するプロジェクト MULTRAN において，自然言語文章の物語から動画像への変換の中間言語として用いる予定である [3]．

2 システムの特徴

2.1 スケジューリングに基づくイベント駆動

コンピュータ・アニメーションに必要な情報は，アクターの動作内容に関する情報と，それらの動作間の時間関係の情報とに分類される．その中には，先に述べた偶発的な出来事の記述も含まれる．この偶発的な出来事の記述は，アクターの行動の結果に注目した記述であり，システムは，この結果の原因となるアクターの行動の初期設定を計算し，それに沿ってアニメーションを生成する必要がある．

そこで，我々のアニメーション言語は，スケジューリングに基づくイベント駆動方式をとる．アニメーション言語による記述からコンピュータ・アニメーションを生成する際に，一旦各アクターの行動のスケジューリングを行ない，それを基に偶発的な出来事に必要なアクターの行動をシステム側が強制的に開始させ，アニメーション全体をコントロールする．

2.2 時間評価機能を備えたアクター

アクターの動作に必要な時間を求めるために，アクター自身に動作時間の評価機能を持たせる．スケジューラはアクターに対して一種の予行演習を行なわせることで，必要な時間情報を得る．これは，アニメーション制作者にとって各アクターの動作に要する時間を把握するのが困難であるため，アクター側に時間評価を行なわせるわけである．

2.3 その他の特徴

イベントは，原則としてアクターの状態変化としてとらえる．ここでいう状態変化とは，例えばアクターが「停止状態」から「歩行状態」に変化することを指す．これにより，動作の詳細な指定が可能になる．しかも，自然言語における「A 地点から (B 地点に向かって) 歩き始める」のような文の記述と対応可能なため，言語・動画像間の変換においても利用しやすい．しかし，実際にはアニメーション制作者が直接アニメーション言語を利用する場合を考え，「A 地点から B 地点まで歩く」といった時区間をもつ動作も直接扱うことができる．

出発地点や移動先などのイベント内の情報が省略された時，スケジューラはアクターの状態を考慮して適切な情報の補完を試みる．情報補完が不可能な時は，アニメーション制作者に入力を促す．

イベント情報，および時間情報は宣言的に記述する．従って，情報の追加，変更が比較的容易である．

3 アニメーションの制作方法

アニメーションの制作には，アクターを定義し，(1) イベント情報と (2) 時間情報を宣言的に記述する．以下にその記述方法とスケジューラの働きを述べる．ただし，アクターの定義は本稿ではふれない．

3.1 イベント情報の記述

動作を記述する時，瞬間的な状態の変化に注目する場合と時区間をもつ動作を考慮する場合とがある．本アニメーション言語では，そのどちらもイベントとして取り扱うことができる．

イベントの書式は，以下のような形で宣言的に記述する．

```
event( イベント名, [ … イベント内容 … ]).
```

3.2 時間情報の記述

二つのイベントが発生する時間差を記述する．この情報がない場合，スケジューラは，記述順にイベントを発生させる．

時間関係を表す基本述語として，次の三種類が用意されている．但し，A，B はイベント，ΔT は時間を表す．

delay(A, B, ΔT): A が始まり ΔT 遅れて B が始まる．

A Declarative Animation Language Using Actors with Time Estimators
Hiromi BABA, Toshiyuki FUJIWARA, Tsukasa NOMA, Naoyuki OKADA
Kyushu Institute of Technology

before(A, B, ΔT): A が終わり ΔT 遅れて B が始まる.

end(A, B, ΔT): A が終わり ΔT 遅れて B が終わる.

上記の述語ですべての時間関係を表現するが, ΔT が 0 の場合は, 次の述語も使用できる.

start(A , B): A と B が同時に始まる.
(delay(A, B, 0))

meet(A , B): A が終わると同時に B が始まる.
(before(A, B, 0))

finish(A , B): A と B が同時に終わる.
(end (A, B, 0))

equal(A , B): A と B が同時に始まり同時に終わる.
(delay(A, B, 0) かつ end(A, B, 0))

3.3 スケジューリング

3.3.1 スケジューラの動作

各動作のスケジュール決定には, あらかじめシステムが持っている動作の優先度の情報を基に, 高い優先度の動作から動作時間を決定する.

アクターが動作を実行する時間がイベント中に陽に記述されていれば, その情報を利用する. 省略されている場合にはアクターに時間評価をさせて情報の補完を行なう. 動作間の時間関係の記述がなければ, 記述された動作の順番にそって実行する.

こうして生成したスケジュールには, 時間的矛盾が生じることもある. その場合は, アニメーション制作者との対話で矛盾解消を試みる.

3.3.2 アクターの時間評価

アクターの時間評価方法は, 動作内容をシミュレートして求める方法と, デフォルト値を用いる方法とがある. 移動などの動作では移動開始と終了の地点, さらにアクター特有の移動手段から移動に要する時間が求められる. しかし, 動作の性質によっては上の方法では求まらないものがある. その場合はデフォルト値を用いる.

3.3.3 スケジューリングの例

例として以下のようなシナリオをとりあげる.

1. ウサギが山のふもとから木まで走る.
2. ウサギは休み始める.
3. カメが山のふもとから頂上まで歩く.
4. ウサギは休むのをやめる.
5. ウサギは走りだす.
6. ウサギは頂上につく.

これをアニメーション言語で表現すると以下のようになる.

```
event(e1, [rel:run, subj:rabbit1,
    startpoint:foot1, endpoint:tree1]).
event(e2, [rel:rest, asp:start,
    subj:rabbit1]).
event(e3, [rel:walk, subj:tortoise1,
    startpoint:foot1, endpoinnt:top1]).
event(e4, [rel:rest, asp:stop,
    subj:rabbit1]).
event(e5, [rel:run, asp:start,
    subj:rabbit1, startpoint:tree1]).
event(e6, [rel:run, asp:stop,
    subj:rabbit1, endpoint:top1]).
```

上記のように, イベントのみを記述し時間関係を省略した場合, 生成されるアニメーションは, 以下のようになる.

> ウサギが山のふもとから木のところまで走った後休憩を始める. するとカメが山のふもとから頂上まで歩く. その後, ウサギは休憩をやめて再び走りだし, 頂上につく.

そこで, イベント間の時間情報を例えば次のように追加する.

```
start(e1, e3).
meet(e4, e5).
finish(e3, e6).
```

この条件により, シナリオは以下のように変更される.

> ウサギとカメは同時に山のふもとを出発する. カメが歩き続けている間に, ウサギは木のところに着いて休憩し, 再び走り出す. そしてウサギとカメは同時に山の頂上に着く.

このシナリオに出てくる“走る”, “歩く”のような動作の所要時間を決定する優先順位は, “休む”よりも高い. 後の例では, 時間関係が存在するため, “休む”以外の他の動作の所要時間が決定した後に, “休む”時間の長さが割り付けられる.

4 まとめ

我々は, 時間評価アクターを用いた宣言型アニメーション言語を提案した. 従来のアニメーション言語では, ストーリー中の偶発的なイベントは取り扱いが困難であったが, 時間評価アクターを用いてスケジューリングを行なうことによって, それらを容易に実現できる. また, イベントやイベント間の時間関係を宣言的に記述するため, シナリオの追加や変更が容易である.

さらに, 本言語はイベント間の関係に関する制約プログラミング言語であるともいえる. 自然言語文章をイベント間の制約集合としてとらえると, 自然言語と本言語の親和性の高さは明らかであろう.

参考文献

[1] Magnenat-Thalmann, Thalmann: *Computer Animation*, 2nd revised ed., Springer-Verlag, 1990.

[2] 宮本, 花田, 吉川: 動画生成のための並行動作モデル, 情報処理学会ソフトウェア工学研究会資料, 90-SE-73, 1990.

[3] 乃万, 中村, 甲斐, 岡田: 日本語文章のアニメーション言語への変換に向けて, 「自然言語処理の新しい応用」シンポジウム, 1992.