アニメーション作成ツール CBAD における例題による action の作成

1S−1

笹倉万里子　岩間憲三　満田成紀
(財) 京都高度技術研究所

1 はじめに

われわれが研究開発中の CBAD はアニメーションに登場する対象 (actor) とその動作をあらかじめ定義しておき，それらの対象に対する動作要求を継ぐことでアニメーションを作成するシステムである．CBAD の特徴はものとものとの関係を constraint で表すことができるということである．これにより，ユーザはプログラミングよりも一歩抽象度の高いレベルでアニメーションを記述することができる．CBAD に関しては第 43 回大会で発表した [1].

今回は，CBAD における actor の定義，特にその中の action の定義について述べる．アニメーションを作る時にユーザが独自の actor を作成する必要に迫られることがある．独自の actor には表示される actor のイメージを新たに登録したりすることも必要であるし，また，独自の action を定義する必要があることもある．action の定義は actor の動きを定義することであり，一般的にはプログラミングが必要である．しかし，われわれはプログラマでないユーザにも使えるようにするために動き方を示す何枚かの図をユーザに入力してもらうことによって action を定義する方法を提案する．

例題によりものの指定を行うことは特にノンプログラマにとって有効であると考えられている．ユーザインタフェース関係で例題による指定を実現したものとしては ユーザインタフェース構築に例題による入力を用いた peridot [2] と グラフィックエディタに例題による入力を用いた metamouse [3] が有名である．

2 例題から action を作成する時の基本方針

action を定義するためには，その action を実行した時，どのように変化が起こって結果としてどうなるか，という変化の過程と結果の両方が必要である．そこでユーザには例題として action が始まる時の最初の図，変化の過程を表す 0 枚以上の図，結果を表す図からなる複数枚の図を入力してもらう (図 1).

action はさまざまな場面で使われる一般的な動きの記述である．したがって，ユーザが例図によって示したいのは，actor の絶対的な位置ではなく，もっと抽象的な位置であると考えられる．そこで，われわれは例図に現れる actor 同士の位置関係を重視して action を作成する．

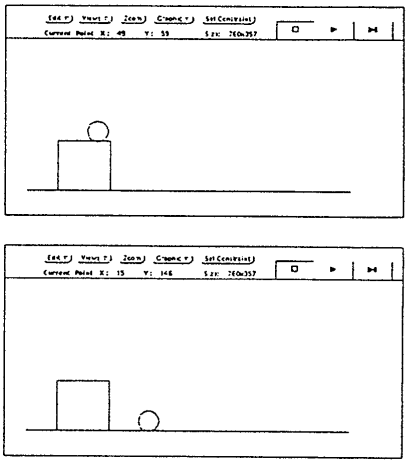

図 1: 例題として入力される図

actor 間の位置関係は CONNECT, OVERLAP, INNER, NEAR の 4 種類の関係と上下左右，layer からなる位置関係の表現を使う．この時，上下左右のところには，相手の actor との距離を入れておく．

ユーザが入力した複数枚の図はそこに登場する actor 間の位置関係で表現される．そして，連続する 2 枚の図の位置関係を比べて，その違いを調べる．違いがわかったら，その違いを生じさせることの出来る基本 action (CBAD があらかじめ用意している action) でその違いを表現する．つまり，新しくできる action は基本 action の組合せで実行されるわけである．

位置関係を表現できるような基本 action として move_to_object, move_along を用意する．これらは，ユーザが指定する時には移動先の目的地を actor 間の位置関係として表現し，実際にアニメーションを実行する時に具体的な座標を求める機能を持つ基本 action である．

3 図からコードを生成するまで

まず，与えられた図を位置関係で表現しておく．次に，図の変化を調べてそれに対応する基本 action を選ぶ．一般には図の変化を表すにはその 2 枚の図をどのように補間するかによって無数の可能性がある．われわれはそれをとくに一般的と思われる二つの動きに絞って変化を表している．一つは二つの図の間を直線的に補間する move_to_object であり，もう一つはものとものとが接している時に限り有効な基本 action で，相手に沿っ

Defining an action by examples in CBAD.
Mariko Sasakura, Kenzo Iwama and Naruki Mitsuda
Advanced Software Technology & Mechatronics Research Institute of Kyoto
この研究の一部はシャープ株式会社の援助のもとに行った．

て動くという move_along である．基本 action をこの二つに絞ったとしても，複数枚の図の変化を表す基本 action の組合せはたくさんある．そこで，それらの基本 action に評価値をつけ，評価値が高いものを選ぶようにする．評価の基準として次のことを考えている．

1. その基本 action を選んだ背景による評価
2. 例を正しく満たすかどうかの評価

1. は例えば2枚の図で二つの actor の相対的な位置関係は変わっていても CONNECT であるということが変わっていなければ，move_to_object と move_along の両方の可能性があるわけだが，このときには最初から move_along の方の評価値を高くする，というように使う．2. は例図が追加された時に今まで考えていた基本 action の組合せがそのまま使えるかどうかという評価である．

具体的に図の違いから action を作成する手順は次のようにする．最初に，最初と最後の図の間の位置関係の違いを調べてそれに対応する基本 action を生成する．次に途中の図を挿入しながら同じ作業を続ける．新しい図を挿入した時には以前に指定した基本 action の範囲に今追加した図が含まれているかどうかを調べる．もし，含まれていなければその基本 action の評価値を下げる．この過程で可能性のある全ての基本 action のリストが出来る．すべての図について変化を表す基本 action を選ぶことが出来たら，この一連の例を表す基本 action の流れを選ぶ．選び方は評価値の高いものを選ぶようにする．

4 move_to_object

move_to_object はアニメーションのストーリーを作成した時ではなく実行する時に具体的な行き先座標を特定する．これは CBAD の constraint solver の仕組みを使って行う．

CBAD の constraint の中には例えば二つの actor がくっついていることを示す glue のように，ユーザが指定した動きを指定した時，actor 間の関係を維持するために他の actor が動きを起こすものがある．そのために，CBAD はアニメーションの実行の前に一通り constraint の check を行う．この時に actor 間の関係を維持するために必要な動きをアニメーションに追加したり，矛盾する動きがある場合にはそれをユーザに知らせたりする．move_to_object はこの constraint の check のフェーズで具体的な行き先座標を決定する．

位置関係を表す constraint を positional constraint (p-const) と呼ぶことにしよう．すると，move_to_object は

1. p-const の設定
2. 最終目的地の決定
3. p-const の削除

の3つを行えばいい．ただし，複数の move_to_object が同時に同じ actor に対して指定されることもある．そのような場合にはそれぞれの p-const の間で相談して，そのどれもを満たすような場所を最終目的地としたい．そのためには

1. p-const1 の設定
2. p-const2 の設定
3. 最終目的地の決定
4. p-const1, p-const2 の削除

というように最初にすべての p-const を設定してから最終目的地の決定を行い，そのあと p-const を削除する．したがって一つの action の check は次のようになる．

1. 準備 最終目的地の決定.
2. cut, cut, cut... 1 cut 毎の check.
3. 後始末 必要な場合後始末を行う.

p-const は領域という自由度がある constraint である．最終目的地 の決定は実際には p-const の中でほとんど行われる．p-const は領域をパラメタとしてもらう．この領域は actor が自分が存在できる範囲として知っている領域である．p-const はそれに自分が知っている条件つまり目的地 actor との位置関係から導き出される条件を付加し，新しい領域を作る．そして，actor がその新しい領域の中に入っていなければ新しい領域の中に入るように actor を動かす．そして新しい領域をその actor が存在できる領域として登録する．

このようにすると，複数の move_to_pbject が一つの actor に対してある時にはだんだんと領域が狭まっていって最終的にどの move_to_object をも満たす領域ができる．conflict がおこるような場合は領域が存在しなくなることで判定できる．

5 おわりに

例題による action の入力の基本的な方針とその実現方法について述べた．図からコードを生成する部分はすでに実現されており，現在 constraint solver のインプリメントに取り組んでいる．今回の発表では actor の位置の変化だけに着目しているが，他にアニメーションに必要な回転，拡大縮小，イメージの切替えについても同じようにして例図から基本 action の組合せを決定し新しい action を定義することが出来る．次は，状況によって動作の異なる action の指定について考えたい．

参考文献

[1] 笹倉，Salpeter，岩間 “コンストレイントを用いたアニメーション作成ツール CBAD”，情報処理学会第 43 回大会第 5 分冊 pp.67-68, 1991

[2] Myers, B.A. “Creating User Interfaces Using Programming by Example, Visual Programming, and Constraints” ACM Trans. Prog. Lang. Sys. Vol.12, No.2(1990),pp.143-177

[3] Maulsby, D.L. and Witten, I.H. “Inducing Programs in a Direct-manipulation Environment” Human Factor in Computing Systems, 1989, pp.57-62