The Use of Relations for Motion Control in an Environment With Multiple Moving Objects

Hanqiu Sun
Business Computing Department
University of Winnipeg
Winnipeg, Canada R3B 2E9

Mark Green
Department of Computing Science
University of Alberta
Edmonton, Canada T6G 2H1

ABSTRACT

This paper considers the problem of motion specification for multiple moving objects in complicated environments. It addresses the problems of interactions between the moving objects and the environment, and the interactions between the moving objects themselves. This is considerably more difficult than motion specification for a single object due to the large number of interactions that cannot be predicted before the start of the motion. The solution proposed here is based on dividing the motion specification into a number of atomic units called relations. A relation is a mapping from the object that causes a movement to the object(s) that performs it. Each relation performs a simple dynamic behavior controlled by its own sensing ability and the state of the environment. A description of this approach is presented along with some of the techniques that we have developed for controlling relations. Also, several ballroom dancing examples are used to illustrate the power of this approach.

1. Introduction

Animating complex objects is difficult since they have a large number of parts, and thus a large number of degrees of freedom that must be controlled. Examples of such objects are trees, deformable materials, and human figures. Since these objects have a large number of moving parts, the task of selecting the proper motion for these parts requires considerable effort, skill, and time. To further illustrate this point, consider a human figure model as an example. A fully defined articulated figure can have over 200 degrees of freedom. Animating this figure requires specifying of the motion of each of these degrees of freedom for the entire period of the animation.

When an object is placed in an environment, additional constraints are introduced by the environment. These constraints come from both static and dynamic environmental influences. The static environmental influences are the boundaries of the environment, obstacles, and other static objects present in the environment, while other moving objects and scene events are the dynamic environmental influences. Consider a room environment with walls, chairs, and dancers. Each dancers' motion is constrained by the walls, chairs, and other dancers. Here, walls and chairs serve as the static obstacles and the other dancers serve as the dynamic obstacles. For natural movement in this environment, the dancers should automatically avoid potential collisions with these obstacles whenever necessary. How the obstacles are avoided largely depends on the object's goal, its internal state, the current environmental state, and stylized behavior of the object.

Dynamic environmental influences can lead to more complex and unpredictable motion patterns that are hard to plan for before the animation is produced. The complexity of specifying object motions that dynamically depend on each other increases with the number of objects in the environment, as well as the types of behaviors modeled among the objects. During an animation, one object's motion may restrict or stimulate another object's motion in many ways. One object's motion can cause another object to leave its current path or actively prevent the object from reaching its initial goal. A moving object may attempt to lead other moving objects; other moving objects may follow the lead or totally ignore such possibilities. At any time the leading role may be interrupted while the object turns its attention to a randomly occurring event. Since the motion of an object is influenced by other objects, the specification of its motion isn't independent of these other objects.

One effective approach to handling the large number of degrees of freedom in a single object's model is to explicitly model the structure of the object's subparts. Examples are the tree structures used for human figures and muscle grouping structures for facial animations. The structure for a single object remains constant through the whole animation. Its use greatly simplifies motion specification for a single object, by outlining the general rules for controlling the subpart's influences. Can the same idea be used to reduce the complexity of specifying a motion dynamically influenced by an environment? In comparison with the single object case, the environmental control structure can be more dynamic, individual, and dependent on each element of the environment and each behavior.

In this paper, we consider the problem of several objects moving in an environment, and their motion specification. The term environment refers to a collection of objects that form the background for the animation. That is, the objects that are not the main characters of interest. In this paper we assume that the environment is static, which means that the objects that form the environment don't move. The moving objects are the main characters in the animation, and we assume that these characters have sophisticated behavior and personality. From the previous discussion, it can be seen that any motion specification technique for moving objects in an environment must deal with the following three problems:

1) The large number of degrees of freedom in each object, along with their sophisticated motion vocabulary.
2) The interactions between moving objects and the environment.
3) The interactions between moving objects.

In this paper, we present a new motion control technique based on the use of relations. A relation describes one dynamic control unit of an object's motion. It can be independently specified and individually used in the motion control process. An object has a collection of relations, not all of which are active at the same time, that define its reactions to the situations that occur in the animation. In this paper we present the basic ideas behind this approach to motion control and a significant application of this approach.

Previous approaches used for motion specification, especially in an environment context, are discussed in the next section. A discussion of the relation approach to animation is presented in section 3. One example based on ballroom dancing is used to illustrate the use of our approach.

2. Previous Research

Most of the current animation techniques deal with the motion of a single object. In other words, the motion of each object is specified in a separate control space. For scene motion, with several moving objects, existing techniques could be applied to coordinating the motion of several objects when each of them is in its own control space. This separation can be based on the use of predefined sequences. In this approach, the motion for each object is developed separately along its own predefined path. Along this path all the possible interactions with the other moving objects and the environment are planned in detail at each point in time. Based on each predefined path, some ad hoc technique is used to adjust for the desired behavior. This repeated test cycle is used to combine the objects' motions along the paths.

Applying the techniques developed for the motion of a single object to the case of multiple objects causes several problems with the quality of the motion and the efficiency of its specification. First, since the motion is developed for one object at a time, ad hoc techniques must be used to model the interaction between objects. In this way, it is not clear how the motion of one object is related to the motion of the other objects. Using blind guessing only makes the scene animation task more difficult, and extends the specification time. Second, since ad hoc approaches are used, modifying the motion for similar behaviors and environments can be difficult and usually requires a complete respecification of the motion. As a result, a simple change in the environment may require extensive changes in the motion specification.

In addition to the predefined path approach, several other techniques have been developed for specifying and controlling motion in static environments or for collections of moving objects. These techniques include the sensor-effector approach, behavior rule approach, and predefined environment approach. The sensor-effector approach simulates the neural network connections between the object's sensors and motor programs, that can lead to interesting behaviors in an environment. The behavior rule approach uses a set of behavior rules to control the way the sensor signals are mapped to the relevant motor programs. The predefined environment approach assumes a previously known static environment and based on that precomputes all the possible paths in the environment. The following subsections provide more details on these approaches.

2.1. Sensor-effector Approach

The sensor-effector approach bases the object's behavior on sensors, effectors, and a neural network connecting them. The way an object behaves in the environment depends on how the environment is sensed, and how this information is passed through the neural network to the effectors that produce the object's motion. The sensor-effector approach appears attractive due to the natural way it simulates the control process, starting with sensing, through the neural network, and to the object's response. However, the way that the sensors, effectors, and the neural network are actually connected in the real world, such as humans and animals, is currently still a research issue.

Motion specification in the sensor-effector approach consists of the definitions of the sensors, effectors, and the neural network that connects them. One significant work in this area is Braitenberg's book "Vehicles: experiments in synthetic psychology" [Braitenberg 84]. In this book, Braitenberg shows examples of motion behavior such as fear, aggression, love, selection, and foresight, which are produced in a simple environment with a light source and a few vehicles. It is interesting to observe the variety of motion behaviors produced by such a simple connective model, which is simulated by wires and mechanical parts.

BrainWorks [Travers 88], based on the ideas in Braitenberg's book, is an interactive graphical interface for constructing the nervous system of a simple animal. Starting with a brainless animal body, which is equipped with visual sensors and touch bumpers and simple motors to change the turtle's position or orientation, the user can select a piece of nerve of various types and use a tool to connect it to a part of the neural network. Once the network is built, the turtle can display reasonable behavior in response to an environment, such as seeking or avoidance.

Inspired by the ideas of Braitenberg, Wilhelms [Wilhelms 89] has proposed an interactive approach to behavior control in animation. This approach provides the user with an interactive environment for constructing the network between sensors and effectors. The nodes in the network can invert, emphasize, apply a threshold to the signal, or use more sophisticated procedural operations. This system allows the user to interactively set up the appropriate transfer network and visually experiment with the modeled behaviors.

2.2. Behavior Rule Approach

The behavior rule approach is another approach to behavior control that parallels the sensor-effector approach. Like the sensor-effector approach, it takes sensed information as its input and motor controls as its output. However, this approach uses behavior rules to control the object's behavior, instead of using a neural network as the sensor-effector approach does. A brief description of a simple animal brain model [Coderre 88], capable of several basic behavior patterns, is:

1. If both HUNGER and FEAR are high, effect a tradeoff between COMBAT and FORAGE.
2. If FEAR is high, COMBAT.
3. If HUNGER is high, FORAGE.
4. If FORAGE is recommending that there is a food element immediately available, then FORAGE.
5. If NEST has some non-trivial action to perform, then NEST.
6. Otherwise, EXPLORE.

More detailed behaviors can be produced starting with this simple brain model.

In this approach, behavior rules are used to determine the proper actions. The possible behaviors can be represented by a decision tree, with each branch of the tree representing one alternative behavior. At each node of the tree, one subbranch will be selected based on the behavior rules and a rating strategy, such as weighting or thresholding. One action satisfying the conditions in the current environment is selected by a tree traversal process.

One example of the behavior rule approach is Petworld [Coderre 88], which models a two dimensional world of pets, rocks, and trees. In this environment, pets have a body orientation and can move along their body orientation one unit at a time. Pets also have a field of view and can carry one rock, eat trees as food, use rocks to build nests, attack each other, and die from starvation or wounds. In Petworld, a decision tree which covers all possible behavior selections is built. The tree is traversed based on the current environmental conditions, such as a pet's internal state, HUNGER, FEAR, or INJURY, and the strategies rated for competition, compromise, and displacement behavior.

A model of flock-like behavior based on centralized, noncolliding aggregate motion has been produced by Reynolds [Reynolds 87]. This work models the behavior of each bird in a flock independently according to the bird's local perception of the dynamic environment. With the geometric form of birds and their ability to fly, the flocking behaviors such as avoiding collisions in between and the urge to join the flock are applied. The precedence order of these behaviors is: collision avoidance, velocity matching, and flock centering. The general flocking principles used in this model can be adopted for other kinds of aggregate motion, such as herding of land animals or schooling of fish.

2.3. Predefined Environment Approach

The third approach to the problem of scene motion is based on a static predefined environment. Since the environment is known before the motion is computed, the difficulty of specifying a particular motion is greatly simplified. All possible paths from the initial position to the goal position can be explored. With all the alternatives ennumerated, an optimal solution can be determined based on some selected behavior criteria, such as the shortest path or the path with the minimum energy use. Using this approach has the potential of modeling complex motion, since the environment is predefined and a large amount of precomputing can be used to select an optimal path. On the other hand, since the approach is based on a predefined environment, even small changes in the environment, such as removing or adding an object to the environment, will require a complete recomputation of the motion. Typical specification techniques used in this type of applications include visibility graph, path planning given behavior criteria, and passive motion in the environment.

Path planning [Latombe 89] is one of the techniques that has been used in this type of application. This technique searches the visibility graph precomputed for the obstacles for a collision-free minimum-cost path from a given initial position to the goal position. The initial path planning algorithms developed in robotics have also been used in animation applications as well.

Ridsdale [Ridsdale 88] has suggested using a knowledge-based system for planning the motion in a stage environment. His system can plan the motion from position A to position B based on the other characters present on the stage. If some obstacle appears in the path of a motion, a "good" path is selected to avoid the obstacle and at the same time match the predefined relationships with the other characters. The knowledge for the stage environment needs to be updated every time one of the characters on the stage moves. Stage update and reference at each control step may slow down the system feedback cycle. And, another concern is the ability to handle several character interactions during their motion.

One alternative to path planning is to model the physics of the passive collision process in an environment. Hahn [Hahn 88] has modeled a general class of three dimensional dynamic processes for arbitrary rigid bodies. The simulation of the dynamic interaction takes into account physical characteristics of elasticity, friction, mass, and moment of inertia to produce rolling and sliding contacts among rigid bodies. Moore and Wilhelms [Moore 88] also present a technique for collision detection and response. The collision is modeled by springs and an analytical collision response algorithm that conserves linear and angular momentum of articulated rigid bodies is used. Both of these techniques use a set of dynamic equations, plus the colliding constraints to guide the proper collision-free motions.

3. Relation Approach

3.1. Relation Concept and Definition

When an object moves through an environment, its motion is constrained by the environment. There are two types of environmental constraints: static and dynamic. Static constraints come from the static objects in the environment, and dynamic constraints come from the dynamic objects, such as the other moving objects and events occurring in the scene. For a natural behavior, a moving object should not pass through the boundary of the environment, hit any obstacles, or collide with other moving objects. In addition, the object may select a special path to a target object, back away from a disliked object, or respond to a scene event that may occur at any unpredictable time. The reason why an object moves fast, slow, or in a different style and pattern, depends on how the object is constrained or stimulated by the environment.

An environment may restrict and stimulate a scene motion in many different ways. These influences can be decomposed into units, based on the relationships between pairs of objects. For instance, while avoiding an obstacle, the object's motion is influenced by that obstacle. During the avoidance, the object may also be influenced by another obstacle located nearby, another moving object passing by, the occurrence of an event, or another object moving into sight. There are many environmental influences which must or may be considered in a scene. It is this collective effect of all the environmental influences that constructs a particular behavior. The collection of these influences in general determines the behaviors that can be explored in the environment.

A relation is used to specify how the environment affects the motion of an object. A relation is a mapping from the object that causes a movement to the object that performs it. A relation has three main parts: a source, a responder, and a response. The influences or constraints in an environment are based on one or more objects called sources, that could be the environment boundary, obstacles, other static and moving objects, and scene events. The source is the object(s) that causes the motion to occur. A source can be involved in more than one relation, and could control the motion of several objects. In each of these relations, the source may play a different role. For example a source object could attract one responder and at the same time repel other responders.

The responder object is the object that performs the motion. The same responder could be involved in several relations, each having a different source. Both source and responder objects can be an individual or a group of objects. When a relation is mapped to a group of sources, any member of the group can trigger the responder. When a relation has a group of responders, any or all the members of the group can actively respond to the source. The source object of a relation can be the same object as the responder. In that case, the object is influenced by itself.

There are four basic types of relations based on whether the source and responder are an individual or a group. These types are one-to-one, one-to-a-group, a-group-to-one, and a-group-to-a-group. The one-to-one type maps from an individual source to an individual responder. Examples of this are a person walking towards a door and a bird flying away from a pole. In these examples, the person and bird are the responders whose motions are enabled by the door and pole. Similarly, a-group-to-one type maps a group of sources to an individual responder. Examples of this mapping include a shark attacking a school of fish and a bird avoiding obstacles. The one-to-a-group type maps an individual source to a group of responders. Examples of this mapping are a school of fish chasing after a piece of food and running away from a shark. A-group-to-a-group type maps a group of sources to a group of responders. Examples of this mapping are the schooling behavior of fish and collisions amongst balloons.

The response specifies how the responder responds to the source. That is, it specifies the resulting movement. Any of the standard motion specification techniques, such as keyframe interpolation, kinematics, dynamics, or procedural descriptions, can be used here.

The basic relation framework specifies the object(s) that cause the motion, the object(s) that move and how they move. This basic framework must be extended in the following way. The response of a relation is only performed when a set of enabling conditions for the source object(s) becomes true. These conditions could include the distance to the source, whether it is friendly or hostile, its size, or its color. A relation can also have parameters that are used to vary the relation's response over the course of the animation.

At any point in time, a relation is in one of four control states, which are potential, active, suspended, and terminated. When a relation is in the potential state, it can potentially participate in the motion, but is currently idle. A relation in the active state is currently contributing to the motion of its responder object. If an active relation is temporarily blocked, for example by another relation, the relation enters the suspended state. In the terminated state, a relation cannot be considered as a candidate for motion control. This case occurs when either the source or responder object disappears from the environment.

Several control mechanisms, including environment control, interaction control, pattern control, and sequence control, can cause transitions between relation control states. Environment control changes the control state of a relation when the environment is redefined or changed in some other way. The control state of a relation can be changed by other active relations; this is called interaction control. In pattern control, the animator controls the state changes using two patterning structures: time patterning and relation patterning. Sequence Control combines the previously modeled behavior patterns into a sequential time ordering. More details on these control mechanisms are presented in section 3.3.

The software support for the relation control model presented in this section is divided into two parts, which are a relation description language and an interactive behavior editor. The relation description language is used to describe the objects in the animation and the relations that are used to control their motion. These descriptions form the input to the interactive behavior editor. In this editor, the animator can interactively select the objects in the animation, position them within the environment, select the relations to be used, and specify their parameter values. The animator can review the resulting motion, and then modify the objects or relations until the desired motion is achieved. The use of the relation description language and interactive behavior editor separates the specification of motion control into two interfaces: language and user-interaction. These two interfaces support both programmers and end-users (such as animators), who may not have sufficient programming knowledge, to use the programming system.

3.2. Relation Declarations

The declaration of a relation is divided into three sections which are: control header, action code, and finalization. The control header contains the parts of the relation that the animator can modify through the behavior editor, and the other two sections contain the specification of the responder's motion (Fig. 1).

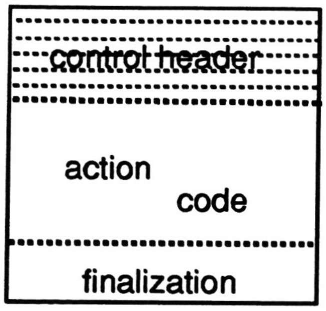


Fig. 1 General Structure of Relation Declaration

The control header section consists of a set of control slots, which are given initial values in the declarations that can later be modified through the behavior editor. The possible control slots are: the source and responder names, enabling condition, initial state, effective duration, strength parameters, switches, and motion aspect (Fig. 2). Each of these slots is described in more detail below.

```
/** control header of relation declaration
source name: string      /*either an individual or a group
responder name: string  /*either an individual or a group
channel name: Boolean expression /*enabling condition(s)
initial state: constant
effective duration: integer
strength parameters:
  /* parameters have one of the following forms
    string number          /*value kind
    string number1 -> number2  /*temporal kind
    string number1 <-> number2 /*stochastic kind
switches: string [ON | OFF]
motion aspect: constant    /*position, orientation, color, etc
```

Fig. 2 Slots Syntax in the Control Header Section

Names assigned to the source and responder slots indicate the object types that participate in the relation. These slot values are used by the system to select the necessary relations whenever a new environment is composed. These slots can be omitted from the relation declaration, to simplify the specification process. In this case, the values used in the previously specified relation are assumed.

The enabling condition slot describes how the source is detected. The slot name is the name of the channel used to sense the source, and the value is a conditional expression that is true when the relation should be active. A channel name is one of the four standard channels: visual, sound, smell, and tactile. An example of an enabling condition is: "visual channel: ifclosetoblock (distance)". This specifies that the source will be sensed through the visual channel and it must be a block located less than "distance" from the responder. Similarly, constraints on sound, smell, and touch can be specified as conditions on the appropriate sensing channel.

The initial state slot specifies the initial control state of the relation. The two initial values that are typically used are: potential and suspended. The default value for this slot is suspended. The effective duration slot specifies the length of time that the relation will remain active each time it is activated. An active relation lasts until either its enabling condition becomes false or its effective duration is over. The default value for this slot is an infinite duration indicating no time constraint.

The animator can use the strength parameters or switches (a special type of parameters) to control the motion specified in the action and finalization sections. A switch has a boolean value, while a strength parameter can have any type of value. There are three kinds of strength parameters: value, temporal, and stochastic. For value parameters, the value of the parameter is passed to the relation, the same as in a programming language. For temporal parameters, the value is computed as a function of time from the initial value number1 to the final value number2, within the given effective duration. Then the value computed at each control step is passed to the relation. In the case of stochastic parameters, a pseudo-random process is used to select the value of the parameter in the range (number1, number2), and then the value selected each time is passed to the relation. The use of switches adjusts a relation's behavior slightly in the presence of other influences. For instance, the presence of another passing-by object may determine the turning direction of an "avoiding-obstacle" relation. The use of parameters adjusts the strength of the relation's response, such as the speed of a turn.

The motion aspect slot specifies the part of the object that the relation controls, such as upper left arm or head. A set of system reserved constants are used to specify this slot, such as UPPER_LEFT_ARM or HEAD. The value of this slot is used in the conflict avoidance strategies outlined in the next section.

```
relation name {
  . . .  C statements -- required before conditional test
  channel name {
        . . .
         C procedure calls -- for state control
        . . .
         C statements -- for computing the excititory
                       responsive behavior
        switch: {
         . . .
         C statements -- for computing subtle response
                     differences based on switch settings
              }
         . . .
         library calls -- for common response control
         . . .
        !(channel name) {
         . . .
         C statements -- for computing the inhibitory
                        response behavior
         }
       }
  (channel name AND .. OR .. ) {
        . . .
         C statements -- for computing the response based
                     on the combined conditions sensed
                     from multiple channels
        }
        . . .
  }
```

Fig. 3 Syntax Rule in the Action Code Section

The action part of the relation declaration specifies the motion of the responder object (Fig. 3). The motion specified in this section depends upon how the source is detected. The name of a sensing channel can be used as a scale factor for the strength of the signal received on that channel. In addition, the names of the switches and strength parameters declared in the control header can be used in this section for the values of the corresponding parameters. Any motion specification technique, such as keyframing, kinematics, dynamics, or procedural control, can be used in this section of the relation declaration. One example is the use of kinematics to translate the responder to a new position. In our current implementation, the C programming language is used in the action part. However, this can easily be changed without affecting the model of motion control.

```
   /*  finalization of relation frame
 ==> {
        . . .
        . . .
         C procedure calls -- for state control
         C statements -- for finalization control
     }
```

Fig. 4 Syntax Rule in the Finalization Section

The finalization section of a relation declaration is used to specify the actions that occur at the end of the relation (Fig. 4). This section is mainly used when either the source or responder is a group. In this case, the action part of the relation is executed once for each member of the group. The finalization section is then executed once to perform any control that depends upon the group as a whole.

3.3. Structuring Mechanisms

When an object moves through an environment, its motion is generated by the collection of relations that are acting on it at each point in time. However, the relations don't act in isolation; there can be several relations acting on the same object at the same time. If more than one relation controls the same aspect of the object's motion, there is the possibility of conflicts. For example, if there are two attractive objects in the environment, two relations could cause the object to move towards each of the attractive objects. The net effect of these two relations is a motion that doesn't lead to either of the attractive objects. This is an example of the random structuring of relations; there is no guarantee that any of the relations will achieve their goal. This is not necessarily an undesirable situation, since interesting motion can be produced in this way. However, in a large number of cases, the relations must be structured to avoid these conflicts and to produce competitive, cooperative, and character-directed behaviors. This section presents four structuring mechanisms for achieving these goals.

One structuring mechanism, environment control, is based on the objects taking part in the relations. A relation is selected if both its source and responder objects are present in the current environment. When the environment is redefined or dynamically changed during a motion, the selected relations change to reflect the current state of the environment. A relation selected by the environment is in one of the two states: potential or suspended. A potential relation actively senses its enabling condition. When it becomes true, it automatically triggers its response and changes from the potential to the active state. At the end of its response, the relation returns to the potential state. A suspended relation waits to be switched to the potential or active state, by one of the other relations or a structuring mechanism. The dynamic changes in the environment during a motion, such as the end of an event or removing an object, determines whether a relation should be terminated.

The second structuring mechanism, interaction control, is based on the interactions between relations, where one active relation determines the state of other relations. A relation can activate other relations that will assist with its motion and suspend relations that cause conflicts in the motion computation. For example, when a relation detects an approaching obstacle, it suspends the relations currently controlling the object's motion and activates the relations that will avoid the obstacle. The particular set of relations used to avoid the obstacle could depend upon the state of the object. Another example of this type of interaction occurs when an object moves towards an attractive box. During this motion, the "attractive box" relation will be suspended when another moving object crosses its path, or if a large hole appears in the path. When these conditions arise, another set of relations are activated to avoid the problem.

The third structuring mechanism, pattern control, is based on collecting together all the relations that control a particular behavior pattern. Two pattern structures, time patterning and relation patterning, are used at this level. Time patterning collects together one or more relations that occur at a specified instant in time. This structure switches the relations to the potential state when the referenced time is reached. Similarly, relation patterning combines several relations that are used when another relation becomes active. These two patterning structures are used to add subtle behavior differences at either an absolute time, or a time that is relative to the action of another relation. Both of these structures can be assigned names and used as the basic elements in the next higher modeling level.

The fourth structuring mechanism, sequence control, is based on the time ordering of behavior patterns. If more than one behavior pattern is modeled, their ordering introduces different sequential behaviors. Each pattern in an ordered list can be stretched or compressed in time. A list of patterns can call another list to form a branched control structure.

The strategy for handling conflicts between relations is based on the motion aspect specified in the declaration of a relation. A priority is assigned to each of the relations in each set with the same motion aspect, and the active relation with the highest priority (in each set) is selected for execution. This priority scheme need not be applied to all aspects of the object's motion (this is under the control of the animator). The priority of a relation can be preassigned in the relation definition, or it can be dynamically computed by the strength of the source, multipled by the level of the sensing channel used to detect it. If this result exceeds a threshold, the relation is activated.

4. The Use of Relations for Dance Motion

A set of examples based on ballroom dancing have been developed to illustrate the use of the relation approach. These examples are based on the work of R. Lake [Lake 90] and use a library of standard dance steps and patterns that he has produced. In our examples, the dancer is modeled as a three-dimensional articulated figure that can perform the standard ballroom dance patterns. Besides that, the dancer can also step forward, backward, to the left or right, turn his body and head, and raise his arms to the initial dance position and put his arms down to the normal standing position. Each of the ballroom dances has a number of standard patterns, and each pattern has a number of steps represented by the positions of the figure's limbs. For instance, the steps in the box-step pattern of the waltz are leftfoot-forward, rightfoot-sidestep, leftfoot-close, rightfoot-backward, leftfoot-sidestep, and rightfoot-close, in that order.

The purpose of these examples is to select the patterns, given a specific dance, based on the ability of the dancer, his goals, and the current state of the environment. The selected pattern is performed in a room environment bounded by walls, with blocks randomly placed within the room, and multiple dancers. In this environment, each dancer performs his favorite dance patterns, and varies his heading, speed, dance steps, and head and arm motions based on the relations in the environment. The behavior of a dancer is dynamically influenced by the other dancers, blocks in the room, and the room's walls.

This section describes a detailed example involving two dancers, and several blocks within a room bounded by walls. The behavior that we are interested in modeling in this example is:

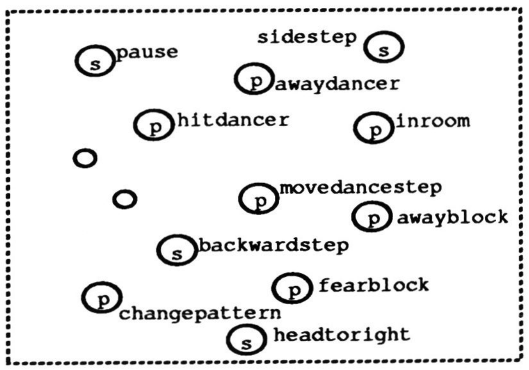
1) The waltz and foxtrot are initially selected by the two dancers, respectively.
2) The dancers switch their dance pattern after repeating it a few times.
3) While dancing, each dancer tries to avoid any potential collisions with the other dancer, any of the blocks, and the room boundary (walls).
4) However, collisions between the two dancers may occur when they are facing away from each other.
5) Also, the style of avoiding static and dynamic obstacles in the room depends on the dancer's character and the obstacles' properties. For instance, the dancer may show a like or dislike behavior towards a particular block, while avoiding it.

The relations that are used in this example are:

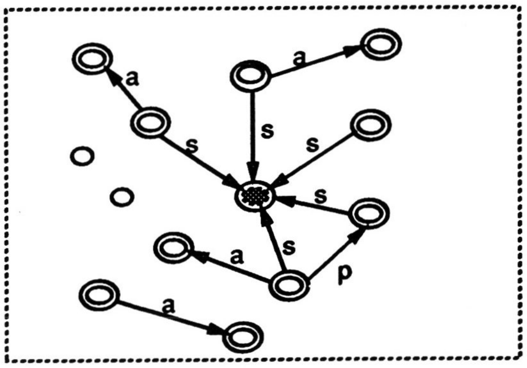
relation	(source, responder): enabling condition?	response behavior
inroom	(walls, dancers): too close?	turn away from the wall
awayblock	(blocks, dancers): too close?	turn away from the block
dislikeblock	(blocks, dancer_1): color red?	quick reverse turn
fearblock	(block_i, dancer_j): is block_i?	back a few steps
movedancestep	(dancers, dancers): motivated?	move to the next pattern step
awaydancer	(dancer_i, dancer_j): too close?	take a few side steps
hitdancer	(dancer_i, dancer_j): hit?	close the current step & restart
changepattern	(dancers, dancers): three times?	change to a new pattern
headtoright	(dancers, dancers): called for?	turn the head to right
pause	(dancers, dancers): called for?	pause
sidestep	(dancers, dancers): called for?	take a few side steps
forwardstep	(dancers, dancers): called for?	take a few forward steps
backwardstep	(dancers, dancers): called for?	take a few backward steps

Here, the relation "inroom" is used between two groups, the group of walls as the source and the group of dancers as the responder. The other group relations are described in a similar way. A group relation can also be applied to a specific group member, such as dancer_1 (the 1st member in the dance group), or any group member indexed by a variable, such as dancer_i. A relation addressing a group member is similar to the case of a relation used for an individual. Each relation performs a primitive behavior that can be repeated during its effective duration, while its enabling condition remains true. The response behaviors used by the above relations are dance step, walking step, side step, forward step, backward step, pause, head turn, and body turn. Among these behaviors, changing to a new pattern only takes one control step to complete, while the others take several steps depending on their response durations. Notice the condition used for the last few relations: called for? This condition is used for relations that are called by other relations in different situations. Since keyframing is used in this example, each relation produces one key position or orientation in a dancer's motion at each keyed time step.

Fig. 5 shows the initial state control diagram for the relations, where each circle represents a relation, and its initial state is indicated inside the circle. The circles without an initial state are the ones not being selected by the environment. In this figure the letter p denotes the potential state and s the suspended state. Also shown is the dynamic state control diagram for the relations, where the possible state controls amongst the relations are outlined. These controls are produced by one of the mechanisms discussed in section 3.3, using the interactive behavior editor. In the dynamic diagram, the double circle represents an active relation. When a relation becomes active, it can issue a state control to another relation. The type of each state control is indicated along the arrow pointing from the calling relation to the called relation. The type is determined by the final state to be changed to, for example the letter p is used for the potential state and so on. A state control is issued whenever the calling relation becomes active. More detailed explanations of the dynamic state controls are given below.

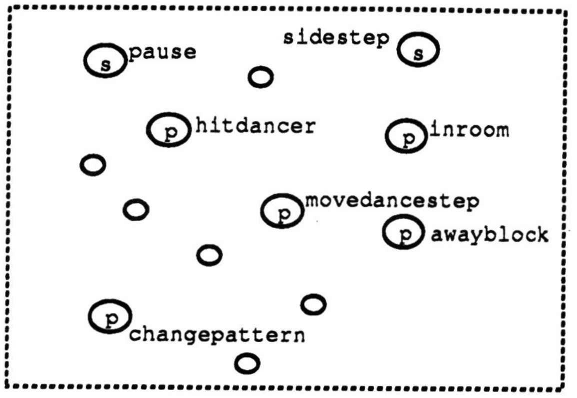


initial state control diagram

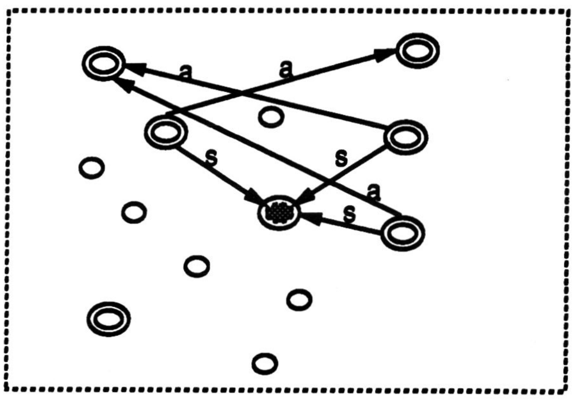


dynamic state control diagram

Fig. 5 State Control Diagram of Timid Dancing Behavior



initial state control diagram



dynamic state control diagram

Fig. 6 State Control Diagram of Patient Dancing Behavior

Initially, the relation "movedancestep" is active and moves each dancer to the next pattern step. This active relation can be interrupted by the relations for avoiding blocks, the other dancer, and the room boundary. To see how this is done, let's look at the case of avoiding blocks. At some point in time, one of the dancers comes close to one of the blocks that he is afraid of. These conditions trigger both the "awayblock" and "fearblock" relations at the same time. According to the state control structures among these relations, the "fearblock" relation issues a state control to the moving relation to change it to the suspended state, switches the avoiding relation to the potential state, and the backing relation to the active state, as shown in Fig. 5. This leads to a correct fearing behavior when the block is encountered. Similar explanations can be given for the other state controls. When the relation "changepattern" is active it does not block the moving relation, because its behavior only lasts for one time step. However, it calls the relation "headtoright" every time a new pattern is used.

These relations produce a "timid" dancing behavior in the room environment. This timid behavior includes the dancers pausing when hit, stepping back at a special block, frequently switching to new patterns, and looking around while changing to a new pattern. Now if we change the state control structures of the relations to those shown in Fig. 6, a different dancing behavior is produced. This behavior shows that the dancers pauses when avoiding a block or a wall, change patterns without any additional response, ignore other dancers until being hit, and when hit take a few side steps to make more space between them. This behavior can be called "patient". If we record these two behaviors as two patterns, P1 and P2, four behavior sequences can be produced at the sequence level, such as P1->, P2->, P1->P2->, P2->P1->. Similarly, other alternative dancing behaviors can be modeled by simply selecting different relations and revising the state control structures among the relations. The potential of behavior editing can be even wider if a variable relation set is used.

5. Conclusions

When the motion of a single object is extended to an environment with multiple moving objects, the task of motion specification becomes quite difficult. This difficulty grows with the number of objects involved in the motion and their range of behaviors. This paper proposes a new motion specification technique, which decomposes the control of a motion into a number of small units called relations. Each relation addresses one environmental influence (static or dynamic) in a scene motion. These relations can be coordinated using a range of control structures, and an interactive behavior editor can be used to compose motion sequences.

The relation approach provides viable solutions to the three problems outlined in the first section of this paper, as follows:

1) Relations allow the animator to decompose complex motions into smaller units, instead of specifying their motion as one large unit. In this way, the animator can use hierarchical and sequential structuring techniques to construct complex motions from a set of simple motions called relations.

2) Relations allow the animator to explicitly state relationships between objects in the environment and the motion of the objects they affect. A relation can be constructed for each of the interactions between the moving objects and the environment. Since each reaction is specified separately, it is easy to fine-tune and modify the actions.

3) Each type of interaction between the moving objects can be described by a separate relation, which simplifies tuning and modifying the motion. The dynamic state control structures provide natural mechanisms for modeling the interactions between the moving objects, which may not be predictable prior to the motion, and determining the important influences on the object's motion.

The two major differences between the relation control approach and behavior rule approach are: the approach to motion specification and the interface provided to the animator. The relation control approach is based on the view of an environment. It concerns how the moving objects are related to each other and to the environment, and how their motions are dynamically influenced by their relations. The behavior rule approach directly specifies how the proper behavior should be performed. The relation control approach supports both programming and interactive control interfaces. Relations, as the unit of behavior, can be interactively structured at different control levels. These interactive control levels make the task for modeling and modifying a set of possible scene behaviors easier. The behavior rule approach codes the high-level description of behavior in detailed programs. No intermediate behavior control structure is proposed for modifying the behavior.

References

[Braitenberg 84] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, *The MIT Press,* Cambridge, Massachusetts, 1984.

[Coderre 88] B. Coderre, Modeling Behavior in Petworld, *artificial life,* Addison-Wesley Publishing Company, 1988.

[Hahn 88] J. Hahn, Realistic Animation of Rigid Bodies, *Computer Graphics,* 22(4), PP. 299-308, 1988.

[Lake 90] R. Lake, Dynamic Motion Control of Articulated Figures, *M.Sc. Thesis,* Department of Computing Science, the University of Alberta, 1990.

[Latombe 89] Jean-Claude Latombe, Introduction to Robot Motion planning, *IEEE Videoconference on Robotics,* April, 1989.

[Moore 88] M. Moore and J. Wilhelms, Collision Detection and Response for Computer Animation, *Computer Graphics,* 22(4), PP. 289-298, 1988.

[Reynolds 87] C. Reynolds, Flocks, Herds and Schools: A Distributed Behavioral Model, *Computer Graphics,* 21(4), PP. 25-34, July 1987.

[Ridsdale 88] G. Ridsdale, The Director's Apprentice: Animating Figures in a Constrained Environment, *Ph.D Thesis,* The School of Computing Science, Simon Fraser University, 1988.

[Travers 88] M. Travers, Animal Construction Kits, *Artificial life,* Addison-Wesley Publishing Company, 1988.

[Wilhelms 89] J. Wilhelms and R. Skinner, An Interactive Approach to Behavioral Control, *Proceedings of Graphics Interface' 89,* PP. 1-8, 1989.