EFFECTIVE CONTROL OF HUMAN MOTION ANIMATION

Diana T. Dadamo

MS-CIS-88-52
GRAPHICS LAB 22

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

July 1988

Acknowledgements: This research was supported in part by NASA Grant NGT-50066, DARPA grant NOOO14-85-K-0018, NSF grants MCS-8219196-CER, IRI84-10413-AO2 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

Effective Control of Human Motion Animation

Diana T. Dadamo

July 7, 1988

Abstract

In this paper, we describe TAKE_ONE, a parallel method of specifying human motion animation by a controlled mixture of values from three kinds of simulation: kinematic, dynamic, and constraint. In addition, tools to assist an animator to define qualities such as realism, individuality and expressiveness are developed.

The issues in comprehensive animation methods are explosion of complexity, difficulty in determining values of input parameters, and lack of ease in fine-tuning an animation. We discuss the advantages and issues involved in structuring the specification of an animation. We provide a structured method to use or to convert to, a parameterized motion definition. We introduce a method of specification to allow the development of the essential and minimal definitional qualities of an action. The result is that a reduction in run-time complexity and user-specification is effected and groundwork for an action database is done.

We develop techniques to refine kinematic animation specification so that it is more representative of actual positional goals and so that it is compatible with the use of the other methods. We provide a structure to systematically merge animations from the three methods, through user or program control, and provide an interface to an iterative method of definition and fine-tuning. Examples are provided to show the power of the TAKE_ONE method, including: an object placement example whose implementation is explained in detail, a wheel-turning task, and finally, a classical ballet pirouette which will serve as a goal example for our completed work.

Chapter 1

Introduction

1.1 The Goal

Our research is a result of the need for realistic, expressive human motion interacting with an environment. The current systems available give us the ability to position a figure, provide full kinematics and dynamics simulation, satisfy geometric constraints and use graphical and language interfaces. Thus, we now have the ability to create a methodology that uses these tools in a structured, high-level manner to achieve natural human motion.

1.2 Human Motion Animation

Human motion animation has received considerable attention over the past decade. The first approach was to define a kinematic specification of the path for each joint of the body. Interpolation methods to provide sufficient smoothness to this path were developed and these methods produced reasonably acceptable animations. The problems with kinematic methods, however, were many. First, the motion could not be realistic in the sense that the body would react to internal or external forces. Secondly, the complexity of the specification was intractable and so, only simplistic and brief animations were attempted. Third, there was no way to control the specification so that refinements were systematic or that previous defined animations could be modified to fit a new situation. A minor improvement came with the general use of inverse kinematics so that the user might specify a global position of a terminal joint of the body (e.g. hand, foot) and allow the angles of the connecting joints to be automatically determined. This technique was the first attempt at a higher-level specification but did little to address any of the objections noted above. Track systems which allowed the grouping of parameters so that they might be separately controlled also were developed. These allowed finer control over the interaction of the animation variables, but again did not provide a major solution to the method's problems.

Dynamics systems were developed to address the problem of realism and to have the figure realistically interact with its environment. Spectacular animations have been produced by such methods, but dynamics systems have substantial problems. First, as in keyframe systems, we still have the explosion of complexity, but it is more difficult issue since the user now has to specify forces instead of positions. This has been shown to be a serious limitation. Secondly, dynamics systems suffer from a time complexity which limits the availability of their use. Third, dynamics

systems have no convenient means to achieve positional goals.

Animations by constraints has also been attempted. These systems start with a set of global conditions and iteratively find a solution. Constraint systems have excelled in solving specifications that have no closed-form solution (such as simultaneous satisfaction of multiple goals in a close loop). Path control of the solution, however, is very difficult since by definition, the final solution is unknown.

Recently, attempts at hybrid systems have been presented. These systems merge specifications from one of the methods in various ways. One method is to provide a kinematic specification for certain animation variables and to provide dynamics for others. Another method is kinematically driven so that the dynamics forces are 'seeded' by a predefined path. Other methods use kinematic contraints to reduce the degrees of freedom for a dynamic simulation or to provide constraints on the magnitudes of the forces in a dynamics simulation. These systems are the closest to our approach in that they recognize the need for specification from each of the simulation methods, but their handling of the merging of these methods is too limited for our purposes.

Lastly, there have been the Artifical Intelligence systems which approach the problem of specification at the task level or natural language level. These systems comprise a rapidly evolving research area, but have limitations in our application. First, the models in these systems are quite complex and the set of possible animations are small. Secondly, while these systems address the problem of decomposition of a specification into smaller units, they do not provide a means to generate how the smaller units should be animated.

1.3 Our Approach

Our approach to the problem of human motion specification, is to construct TAKE_ONE, a methodology of the process of providing specification of an animation and of its implementation. TAKE_ONE will include the existing methods of kinematic, dynamic and constraint specification ("KDC methods"), along with our implementation of Effort-Shape methodology, a way to represent *expressiveness*. We will show improvements on the KDC methods by providing a significantly improved keyframe animation system, and introduce a precise system for merging kinematic, dynamic and constraint simulations, which substantially differs from any hybrid method currently available. This merging is done by a parameter to the system, called FLOW which was suggested by the Effort-Shape literature.

An important aspect of the TAKE_ONE system is the controlled method of iterative refinement of an animation. Animation design presently suffers not from the power of the specification but from the lack of control of the specification. Certain approaches have been taken to address this problem (mainly the Artifical Intelligence language-oriented interfaces, task-level specifications, etc.). [1] However, this work is at a *process definition* level, which is at too high a level for many applications. Also, the AI methods do not yet address the refinement of the animation in any structured way. In the TAKE_ONE system, we provide a method to locally, iteratively refine an animation for general user applications. As an adjunct to this, the development of a minimal set of TAKE_ONE parameters that provide definition of a database of primitive action definitions is necessary.

[1] For an example, refer to [19]

We view an animation as a *limit* of a process of refinements of initial data (usually taken from a data base) augmented by additional data (e.g. more keyframes) and by TAKE_ONE parameters.

We will implement the method of adding specification by the use of well-defined modifiers and through language-semantics (e.g. "gravity", "resisting") so that the TAKE_ONE system will be available to a novice user. A user might also begin with with a human-language expression of a TAKE_ONE parameters instead of numerical values.

The set of TAKE_ONE parameters specifying an animation will be hierarchically grouped to avoid exponential explosion of the specification. We hope to avoid much of the complexity of this problem by the techniques of minimization and structured control that are intrinsic to TAKE_ONE. Also, we will have access to AI inheritance schemes for structured handling of the parameters.

Current work on a Natural Language interface and a high-level architecture for task animation control that could function as a program interface to TAKE_ONE will also address this problem.

1.3.1 Criteria for *Effective Control of Human Motion Animation*

We require an effective human motion animation system to be computationally efficient and usable by a non-expert. To achieve this, we must allow for a data base of actions that are usable in multiple situations. Also, we must allow for structured refinement procedures of new or archived actions so that they may be modified to fit the current application and these procedures must be understandable by the non-expert user. Finally, we hope that the structured development of the specification of actions might provide a tool to test hypotheses of the factors that constitute the identity of an action and also, lead to insights into the factors or behaviors that constitute human actions.

We categorize these goals in four criteria of *specification*, *refinement*, *adaptability* and *analysis*, whose meaning we define as follows:

- Specification
  - *to isolate necessary definitional qualities of a motion*
  - *to include expressiveness, individualistic qualities of an actor*
  - *to be understandable by a non-expert user*
  - *to offer reasonable complexity of specification and computation*
  - *to handle non-orthogonal (overlapping) specifications*
- Refinement
  - *to include a continuous (limit-directed) method of refinement*
  - *to include clearly defined methods of control*
  - *to permit locality of modification*
  - *to detect non-limiting (conflicting) criteria*
- Adaptability
  - *to create a data base of essential definitional qualities of an action*
  - *to allow instantiation of archived actions to particular agent, attitute, intent*

- *to permit degrees of freedom in specification to be expressed as ranges of the parameters of definition, not absolute values*

- Analysis
  - *to allow qualitative/quantitative analysis of motion*
  - *to increase knowledge about motion*

1.4 Contribution of this Thesis

1.4.1 What we plan to do

Our work is directed towards the development of a more complete specification of motion semantics. The work in this thesis is intended to develop and test interaction methodology that uses some of the established techniques in human motion animation to execute those semantics. We intend to use the present computer animation systems with modifications essential to our work. In particular, we will initially concentrate on the development and verification of a parallel control structure which forms one of the parameters to TAKE_ONE, called FLOW.

The FLOW method is an outgrowth of an attempt to implement the Effort-Shape methodology [5] [6] [2] a categorization of the factors inherent in human and animal motion. Our model quantifies (with computationally essential modifications) the Effort-Shape model in terms of computer animation techniques. We also draw on the Effort-Shape model for its natural language vocabulary of the motion factors. Essential to the system is a verification and development of our model. The decomposition and structured control of animation specification in itself will comprise much of our work. The final test of an implementation of Effort-Shape will provide a means to measure the completeness of our system.

We intend to develop a hierarchical concept and the implementation of *reduction* and *generalization* of specification. This will mean, respectively, the distillation of a specification of an action into essential definitional TAKE_ONE parameters, and secondly, the generation of a minimal definition of an action so that the additional degrees of freedom may used to control the individual qualities of a range of instantiations.

This is necessary to show the control of the power of the parameters of TAKE_ONE, and also, to prove the essential property of iterative refinement.

We will examine language interface methods to FLOW, and the perceived change in the actor's intent, personality or talent as we vary the animation parameters. We develop examples that illustrate the proper use of the FLOW mixture, criteria for how and when to break up an animation into smaller units, and simple ways of specifying FLOW and the parameters to merge the units.

1.4.2 Limits of Our Planned Work

We will consider a verification of the TAKE_ONE system to be established when the following capabilities may be demonstrated:

- *An action database is established*

[2] Effort-Shape is described in Chapter 2.

- *A non-expert user will be able to construct a basic animation using this database*
- *The basic animation may be iteratively modified by a non-expert to produce a set of animations for different criteria of agent individuality, agent expressiveness and animator intent*

1.4.3 What we plan to not do

Our work is comptatible with ongoing research in general process control and a natural language interface to animation planning. We do not intend to implement interfaces to these, except as is necessary or convenient to demonstrate our model. We will, however, direct our work within the content of this compatible research and show methods for merging the methods when appropriate.

We intend to integrate our system with an extensible model of articulated figures and their environment [5] whose interface is current in development. Joint work on the development of criteria for a generalized motion data base will contribute to the organizational placement of TAKE_ONE in task animation applications and will be presented as it becomes available.

1.5 Test Cases

In Chapter 5, we present a detailed example that has been implemented in TAKE_ONE. This example is a simple case of the use of FLOW to avoid the use of inverse dynamics to achieve positional control. We will refer to two additional animations in this paper, whose implementation is part of the planned work.

The first is an animation of a human figure turning a wheel. This example has simple enough specification to serve as a general test case of the power of the TAKE_ONE parameters, most importantly, FLOW. Unspecified degrees of freedom (such as turn rate, initial end ending velocities) will be examined for how they affect the identity of the action (e.g. heavy or light wheel) and the characteristics of the agent.

We will also refer to an additional example, PIROUETTE, through this paper, since PIROUETTE provides a very challenging animation goal. PIROUETTE is a sequence of simple classical ballet turns (composed of *en dehors* turns from fifth position) Classical ballet offers idealized positional goals as a definition of a *en dehors* turn. We will augment this with a specification of the turn from a dynamics point of view, and then examine the FLOW mixture of the combination. Issues to settle are:

- *1) Can the "dynamic" quality of the turn be obtained through a minimum of specification (keyframes, forces, constraints)?*
- *2) How can the positional goals be relaxed from the ideal?*
- *1) Can the animation be controlled through TAKE_ONE parameters to provide different interpretations of the action or actor?*

Other issues will be posed as the animation is tested through TAKE_ONE.

The kinematic specification of a single pirouette turn involves an initial placement of the feet (in fifth position, with feet turned oppositely outward) and the arms in first position (rounded and central to the body, at sternum-level). The legs are initially straight.

To start the turn, the actor moves one arm to the side (opposite direction to which she will turn) and bends the knees.

The actor then 'pushes off' floor with both feet, raising the front leg by sliding it up the other in a formalized bent position, closing the arms, and rising to stand on the toes of the supporting leg. This push off is both in the direction of an upwards push (to effect the rising onto the toes of the supporting leg) and the two pushes sideways by both of the legs that produces a torque about the central axis of the body. The movement of the closing of the arm, (though it does affect the moment of inertia and therefore, the angular velocity), is known to contribute little to the magnitude of the torque about this axis.

The front leg is ideally at the knee of the other leg when the turn is 1/4 completed. The actor has turned while keeping the body as straight as possible and still has the head directed back to the original focus.

The next action to take place is the maintenance of the head directed to the orginal front, until this is impossible to accomplish. At that time, the head is quickly turned around the body until it can again focus on the orginal focus. This action is called 'spotting' and while it does have some dynamics effect, it is not of much interest to our simulation, since it provides little driving kinematics or dynamics. However, ideally, the head is again focussed towards front when 3/4 of the turn is completed. If a second turn is desired, the turn proceeds with only the head providing any additional movement. The movement of the head is known not to contribute to the torque about the body.

There are dynamics criteria of magnitude on the original torque caused by the sideways push of the feet, and the degree off true vertical of the central axis of the body that will determine the number of possible repetitions of the turn.

When the actor is ready to stop, this takes place at the 3/4 position of the turn. The actor is facing 270 degrees from the initial starting position and the head has been turned around to again focus towards the front. The actor then begins to open up both arms and to slide the front leg down the supporting leg while bending the supporting leg. Both legs arrive at the position on the ground as initially (fifth position) but with the legs bent. The arms are fully open and the legs back in position and the head is focussed forward at the end of the turn.

Some driving dynamics are provided in [14], the positional goals by [11] [12]; and the constraints by the joint constraints of the body. Additional constraints are on the magnitude of the torques that can be produced by the feet and certainly, a non-self intersecting constraint of the actor. The definitional degrees of freedom include

- *1) the flexibility in timing of the positions*
- *2) a range of error allowed in achieving the positioning goals*
- *3) the addition of dynamics forces to achieve expressive qualities*

The second item is the most interesting since failure to achieve some of the positional goals produces side-effects. As an example, if the body is not rotating at a true vertical, the angular acceleration of the turn will cause a horizontal acceleration thereby 'tilting' the turn. After some time, this horizontal force will cause a large enough deviation off true vertical to allow the gravitation force on the body to cause the turner to fall.

This example will be referred to throughout this paper and will be presented as work to be done in the Research Plan.

1.6 Overview of the Chapters

Chapter 2 reviews related work is several relevant areas of human motion. We present the problems that are inherent in each of the methods if used as the sole means of specification. We present the attempts to combine the specifications from each of the methods and conclude that these hybrid methods are not sufficient. We draw on previous work to establish what attributes are necessary in effective human motion control, and establish concrete criteria to use in evaluation of our system.

Chapter 3 gives an overview of the TAKE_ONE system and its methodology. We discuss the related work in formal movement disciplines to provide a set of motion attributes that will form the parameters of the TAKE_ONE system. We introduce one of these parameters, FLOW, and give the details of its use. We introduce the semantics of the TAKE_ONE system, that is, how the parameters of TAKE_ONE can be interpreted by a non-expert user and how the parameters can be similiarly refined. We briefly discuss the possible application of a hierarchy for inheritance of TAKE_ONE parameters to control the definitional explosion of complexity. Finally, we evaluate TAKE_ONE in terms of our criteria of Chapter 1.

Chapter 4 continues with the presentation of TAKE_ONE by detailing a second parameter, ADM. ADM will be used to control kinematic specification both by allowing automatic generation and iterative refinement of a kinematic simulation. Additionally, ADM will allow a flexible means to adjoin a kinematics simulation to a dynamics simulation.

Chapter 5 gives a detailed example that has been produced through the TAKE_ONE system.

Chapter 6 presents the current status of the TAKE_ONE system and presents the Research plan.

Chapter 2

Related Research

2.1 Representative Existing Models

2.1.1 Parametric Interpolation

Early approaches to animation were purely kinematic. The user would specify the positions of an animation variable and the time at which these positions should occur. These position-time pairs were referred to as *key positions*, from the terminology of Classical 2-D animation. To provide enough information for animation (24-30 frames per second), i.e. values for the *in-betweens*, (the times between key positions), interpolation methods would have to be used. The most satisfactory of these interpolation methods was the cubic B-Spline which allowed second-order continuity (visually: "no jerk") and which allowed the most independence of control over the interpolation of the different sets of in-betweens ("local control").

The number of animation variables to control a human figure, however, was inconvenient to handle. To animate a human figure in an environment produced additional concerns of relative timing. And finally, in a jointed figure, motion of one joint affected another.

The parameterization of an animation addressed the concerns. Keyframes which included key positions for a group of animation variables at one time, were the first parameterization. In this method, each animation variable in a keyframe would be separately interpolated, but the user would need to only specify a keyframe-time sequence. Track systems gave the user finer control over the interaction of the keyframes. Inverse kinematics (e.g. in the use of *reaches* where the position of the finger could be specified, and the angles of the shoulder and elbow automatically determined) were added as a standard procedure in kinematic simulations to address the interaction problem. Finally, the attention was focussed on greater control of the dynamics of the path traversed by an animation variable over the course of the animation. The interpolation methods yielded very similiar dynamics for any input, i.e. a smooth changes in acceleration by virtue of the *cubic* spline. Much work was done to address this problem through the method of interpolation, which while it produced many sophisticated methods of intrinsic interest, it was not successful as a solution to the timing of human motion animation. The introduction of dynamics systems became the alternative approach.

Gomez: TWIXT

Gomez [10] developed a pure kinematic system, called TWIXT, that formalized the interaction between dependence and independence of the control needed for the animation variables by the concept of the *track*. In track systems, animation variables are partitioned into groups; each of these groups are separately animated. The final specification was a juxtapositioning of the animations from each track. Thus, the user could independently and iteratively determine the relative timings between each of the tracks. In this system, the tracks were the parameters to be controlled.

Steketee-Badler: Double-Interpolant

Steketee and Badler [16] increased the control over the interpolation of an animation variable by separating the specification of the path and the speed of traversal of that path. Their method was a double (B-spline) interpolant mapping keyframe times to keyframe index ("kinetic spline") and keyframe index to position ("motion spline"). The resultant motion was a composite of these two interpolations. The double interpolant method also allowed for acceleration control across the boundaries of separately interpolated animations ("joining"). Thus, specification of the kinematic path and the speed of the traversal were othogonalized. Many other variations on the method of interpolation for kinematic simulations have been developed and provide similiar capabilities, especially when coupled by a kinematic *reach* capability, described above. The double interpolant method, though, provides what we believe is the cleanest semantics and the most intuitive user interface, and thus, formed the basis for TAKE_ONE's kinematic simulation ability.

2.1.2 Kinematic Simulations

Zelter: Motion Control Program

Zelter [20] developed a kinematic system for human gait based on state changes of the variables ("Motion Control Program"). The motor control program would decompose a high-level specification of an action (e.g."walk") into subactions, and a low-level processor would subdivide these into atomic actions. (e.g. "heel-off"). This work is representative of the last major efforts at pure kinematic control, for while it significantly addressed the particular problems in the target domain ("walking") and drew support for its methodology from biomechanical studies, it still produced uniform dynamic and unexpressive animations.

Badler et al: TEMPUS/JACK

Badler [4] developed an interactive system, TEMPUS, that allowed definition of a keyframe to be any sequence of control commands. Thus, a keyframe might have an instantiation of a new body, positioning of another body, change of color of an object and a change in the view orientation, all within a single keyframe. However, the control commands were mainly limited to static properties of a variable so that varying constraints conditions (i.e. move the hand with the glass) were difficult. (Our test case, GLASS, was produced through this system.) A unifying model [5] of environment and human body model has recently appeared to address these problems. [1]

[1]The interface JACK [15] is currently under development and is expected to be available for use in our planned test cases.

2.1.3 Dynamics Simulations

As we saw, improvements in kinematic simulation methods could not adequately address the problem of imparting dynamics to a kinematic simulation. Attempts at dynamics systems then appeared, and while, theoretically, dynamics would produce the most accurate and powerful form of specification, it was quickly found that the method was almost completely intractable. Specifying forces was an impossible task for the average user; sample values had to be supplied. The methods to resolve these forces to obtain a sequence of positions for animation required non-linear methods (such as numerical approximations to integration) and so the time complexity of the simulation was a limiting factor. Finally, the power of dynamics was so great, that unless a suitable set of opposing forces could be determined to control their extent, the simulations had to be artifically clamped. Thus, for human motion animation, dynamics simulation systems very quickly developed into the current hybrid models, attempting to merge kinematic or constraint specification to provide that control and dynamics simulations to provide the driving forces.

Armstrong et al: Dynamics simulations

Amstrong, Green and Lake [1] developed a dynamics animation system which allows iterative user control of the dynamics parameters. The effects of dynamics conditions on a single joint is automatically propagated through the body, thereby relieving the user of specification for these dependent variables. Path specification is artifically handled, even for the case when the motion is driven by pure dynamics. If a torque is applied to a free-swinging limb, the path is controlled by a constraint on the torque and its rate of change. Thus, the realism sought by the use of dynamics is reduced since dynamics only provides the criteria to initiate a motion, not to control it. [2]

Girard: Dynamic-Kinematic Simulation

Girard [9] used a combination of dynamics and kinematics to produce articulated figure locomotion. His method separated the locomotion task into a dynamically determined path and kinematically determined leg motions. Hybridization therefore occurred across degrees of freedom. Interactions between the methods were limited to the dependency of kinematics on an adjustment to the animation figure's banking angle during movement. Girard established a database of postures which could be instantiated and modified (though mostly, by conditions on the dynamics) to fit the current application. The user had access to a interactive method of refinement of path control. Interestingly, this system was designed to perform a type of iteration of global specification (i.e. applied to the entire system, not to just a single force) to achieve a final goal animation. However, this iteration was under program control and was directed to the resolution of the dynamics forces. The user, though, had the ability to view the incremental effect of adding dynamics specification.

2.1.4 Constraint Simulations

Pure constraint systems do not yet produce results sufficient for human motion animation. General forms of constraints might be expressed as set of logical conditions over time, and their satisfaction ordered in some manner. For human motion, these conditions may may be expressed as positional constraints ("kinematic constraints") or as constraints on forces ("dynamic constraints"). For these

[2] In TAKE_ONE's terminology, this is equivalent to FLOW of all-dynamics at start. See Chapter 3.

applications, logical conditions (e.g. A is NOT B) must be expressed over a continuous domain (e.g. A is 6.5 inches away from B). The current model of human motion constraints is of a *spring and damper* system, so that the constraints may have *weights.* Thus, a constraint for A to be 6.5 inches away from B might be weighted higher than a constraint for B to be 4 inches away from A. This weighting is necessary to resolve multiple constraints over a continuous domain, to avoid *contradiction.* (In a pure logical formation, the two constraints above are contradictory). Satisfaction of human motion constraints can provide a goal position, but is not convenient at solving the rate of this satisfaction. Thus, in the above example, resolution occurs when A is 6.5 inches away from B; if initially B is not at that position, the control of the movement of B to A ("satisfaction of the goal") is unspecified. The complexity of logical conditions necessary to express a realistic environment and the temporal nature of human motion constraints leads to a system of multiple constraints overlapping in time. The method of resolution currently in use is to solve these conditions with an iteration through time, thereby preventing global control of the solutions (i.e. path of movement caused by multiple satisfaction of the system of constraints). Existing models of human motion animation "solve" this problem by some form of path control, usually by a limited implementation of kinematic specification. We will see that the existing approaches to this task are not satisfactory. An example of the use of constraints to position a figure at an instant in time is given in [5].

Barr: Dynamic Constraints

Barzel and Barr [REF] developed a system based on dynamic constraints. The user has a database of driving forces (e.g. gravity, torque) and a set of intuitively expressed constraints (e.g. "point to nail" which requires a point of an object to remain at a global position in space). The system resolves the set of simultaneous constraints into a set of forces necessary to achieve the constraint goals, through a method of inverse dynamics. Path specification is modeled by a sequence of point-to-nail constraints. Thus the system allows the hybrid control of kinematically specified paths and user-specified forces to drive the motion. The addition of dynamic constraints is handled artifically, however, since the complexity of their specification includes a user-designated function to designate the speed of the satisfaction of the constraint. Also, the use of constraints to solve a path specification is costly. The most serious objections is that the mixture of specification is handled in an inherently conflicting manner. The kinematic specification includes the velocity of the path traversal which can not be effected by any additional constraints or forces. Thus kinematics and dynamics are viewed as mutually exclusive methods of specification.

Wilhelms: Kinematic-Dynamic simulation with Dynamic Constraints

Wilhelms [18] has developed a system that provides a hybrid K-D control, with constraints modeled by springs and dampers. As in [7], a kinematic specification is reduced to the set of forces necessary to achieve the path. However, in this work, since an animation is considered as transitions through a set of states, the problem of rate of satisfaction of positional goals is avoided. The hybridization is achieved by allowing an animation variable to be specified either by kinematics or by dynamics at any point in time, but not by both methods simultaneously. The hybridization, however, only intensifies the effect of an inaccurate kinematic path, and control of the composite set of forces along a path is very difficult. Thus, the most important property of kinematic path specification, i.e. the control over the smoothness (differentially continuity) of the animation over that path,

is sacrificed. Combining this with the loss of speed control of traversal over the path (as in [7]), kinematic specifications are reduced to point-wise positional constraints.

2.2 Problems with Existing Models

Problems with the existing systems arise because each of the three KDC methods are not clearly understood for the power of their specification. This is a opposite issue to one of *implementing* the specification. One of the present systems, [7], elegantly handles simultaneous specifications from a set of forces to which a KDC specification has been diluted. However, this system actually limits the power of a kinematic specification. Kinematic specification, historically, was used to specify path definition, speed and smoothness of path traversal, absolute durations, and discrete positional goals. This overloading was clearly evident in the ease of use of the method (since path control, dynamics, and constraints were specified through the same two parameters, i.e. a set of keypositions and a set of keytimes. But conversely, this overloading made the determination of the values of these parameters very difficult. Dynamics systems have only transferred the overloading to a less intuitive, more computationally complex method.

Certainly, dynamics and kinematics are dual to each other in that $F = ma$, but that also implies that the two methods can not be orthogonal. Thus, the problem of overspecification in a K-D method must be systematically handled.

On the other side of the spectrum, constraint animation systems are naturally underdefined. The addition of kinematics or dynamics is clearly seen as necessary but the problem of how to incorporate them into constraints without inheriting their overspecification has not been resolved.

Thus, any attempt at the development of a methodology would ideally start by defining the purposes and extent of the tools to be used. For human motion animation, these tools are already existent, though not clearly understood. In this thesis, and specifically, in the test cases in our planned work, we hope to make the functionality of each tool, more precisely defined.

2.2.1 Problems with Keyframe systems

Keyframe animation, since it was the simplest and most commonly used, was our first focus in our attempt for clarification of each of the KDC methods.

There are some intrinsic problems with keyframe animation. First, the specification of exact times and positions of the keyframes require special training or expertise, and many everyday actions, such as touching one's nose, do not have a fixed set of essential keyframe position/times that define the action. Further, in a keyframe system, even the absolute duration of a keyframed animation cannot be changed, yet we know as many actions which have flexible durations (stirring; brushing hair) as those which are interpreted differently depending on their duration (touching a table versus smacking a table). Thus, keyframe animation can provide little insight to what constitutes an *essential* definition of an action. Keyframe animation simply provides a *specification*, at best, of a *single instantiation* of a movement while keyparameter systems have a limited application of their parameters. We seek to determine methods of parameterizing a *general definition of a movement*, whose parameters may be fixed or flexible.

We summarize, below:

- Keyframe Animations were Overdefined

- *pure keyframe systems need many keyframes to precisely define the action; but few keyframes are usually known or represent positional goals*
- *dynamics can only be imparted through specification of timing of a keyframe so keyframes existed simply to fix the dynamics*
- *iterative refinement was done by ad hoc methods: i.e. additional keyframes, manual editing, guessing at changes in the key times*
- *there was no way to specify ranges of acceptability or "Don't Care" for the values of the animation variables*

- Keyframes at non-key times can introduce errors or redefine the animation
  - *additional keyframes, occurring at non-key times, are more likely to include arbitrary values; imprecise/ vaguely-defined animations could afford to have a substantial range of error in specification or interpolation as long as the resultant animation "looked good"*
  - *keyframes at non-keytimes usually forced an arbitrary parameter association (i.e. interaction between animation variables by co-definition in a keyframe); this interaction, since it is implicit, is not easily discernible*

- Keyframe methods lack ways to specify dynamics and phrasing (joining) of animation segments
  - *many keyframes were used simply to fix the dynamics*
  - *most systems required long animation durations to avoid the problems of phrasing (smoothly joined segments were done as a single animation; discontinuous segments could be done separately)*

We will see in Chapter 4 how the ADM parameter of the TAKE_ONE system addresses these concerns.

2.3 Summary

None of the existing systems provides a structured way to refine an animation, nor do they address the problem of essential definition. Armstrong's work recognizes the importance of refinement and definition, but only as a way to address the non-intuitive and intractable control of dynamics systems.

Girard's work, the constraint systems and the AI approaches allow modification of existing specification due to changes in the environment or due to the individuality of the actor. These systems, however, do not provide a means to define that individuality.

Further, most of the systems assume that the user has predefined the specifications of the animation. This uncertainty, however, is one of the qualities that distinguishes human motion from robotics. A user may have a general idea of the motion initially, so that the values of the parameters might be bounded, but these values will not be accurately known. Clearly, this has been the case in dynamics systems and keyframe systems intrinsically violate this uncertainty criteria. For human

motion animation, unlike robotics motion, there is a need to provide the user with a structured method of iterative refinement. None of the present systems can provide this capability.

We summarize the most serious objections to any other of existing methods of human motion animation:

- Specification methods are conflicting, produce overdefined systems, are not-well understood conceptually.
- Attempts at hybridization of the current KDC methods cause a decrease of functionality in at least one of the methods.
- Input parameters may be too limiting (keyframe systems for absolute positional values; dynamic systems for complexity of specification, especially in controlling positional goals).
- There is no way to adapt present animations to other jointed figures, i.e. essential definitional qualities of the motion are not understood.
- It is hard to include human qualities such as individuality and expressiveness within same action.
- Current models are insufficient to express gradations in the motion due to level of experience, talent, foci.

Thus, the present models are insufficient to satisfy our criteria of specification, refinement, adaptability and analysis. In the next chapter, we present our approach to these criteria.

Chapter 3

Our Approach To Human Motion Animation

3.1 Our Methodology

3.1.1 Expressiveness versus Forces

The term dynamics is commonly used to express both the actual forces that are being simulated and to represent an expressive quality of an action. While expressiveness might be reflected in changes in acceleration during a path traversal (which necessarily are caused by internal or external forces), this model of expressiveness is not powerful enough for TAKE_ONE's purposes.

Hereafter, we will confine the term *dynamics* to mean the computation for and with forces. [1]

We define *expressiveness* as the quality of a body that relates to its intent, talent or personality. A model which we can use as a basis for a specification of expressiveness is Effort-Shape. The Effort-Shape methodology and notation was developed by Laban [6] in an attempt to classify motion characteristics. We will describe some of the elements of Effort-Shape and show how we incorporate our interpretation of this model into TAKE_ONE.

3.1.2 Effort-Shape

During a study of Effort-Shape notation [6], [8] it became clear that one of the most important ideas behind observations of human movement was that there are a small number of dimensions along which qualities of a motion might be described. It was observed that actions had certain inherent ranges and combinations of Effort-Shape dimensions which were essential to the interpretation of the action (e.g., a slap as opposed to a punch) or to the intent or expression of the actor (e.g. urgency, nervousness).

One of the chief proponents of Effort-Shape motion analysis, Irmgard Bartenieff says:

"In all activity, there is a constant play for proportionality among the Effort elements to balance exertions and recuperations for most effective function and expression. Furthermore, the same activity, done by two individuals, may be organized with somewhat different Effort punctuations – Effort rhythms – even though the same elements are being used. When two people do the same

[1] If necessary to apply the term *dynamics* to represent *expressiveness*, we will use the term in quotes.

thing, it is not the same [2]."
In Badler [2], the Effort-Shape notation was related to a computational model for the use in human motion animation. In this work, the Effort dimension was abstracted as:

- the relative attack and decay accelerations of the motion
- whether the motion is characterized by end effector goals or just joint angle changes
- whether the motion is characterized by muscular control or freedom

We examine each point in turn.

The first point is rather well-established in the computer graphics literature as the notion of "easing." To implement this, we have developed a method of kinematic specification: ADM, the attack/decay method of TAKE_ONE. ADM will allow us to obtain a finer degree of control over acceleration than is usually offered by traditional key parameter systems. An additional advantage of ADM is that it can be used for direct modification of keyframe times to achieve continuity of acceleration across a Dynamic-Kinematic simulation boundary. We will return to ADM in the next chapter.

The second point implies that the animator should have a *continuous* choice between totally indirect actions (specifying joint motions individually) and totally directed actions (relying on inverse kinematic procedures to achieve goals for articulated chains of joints). Previous methods have only provided the cases at the both extremes. The implementation of this point is part of our planned work.

The third point might be taken to imply that a muscle model is required for natural animation, but we desire a simpler and less expensive computational model. Thus, we model this dimension by describing how a joint (or degree of freedom) *responds to or initiates* force. If the joint is able to resist forces, the muscles are in control and the motion is said to be *Bound.* If the muscles are not resisting the force, then they are reacting to it in a physical way and this is said to be *Free.* As in the previous point, we wish a continuum between these two extremes. This continuum from Free to Bound is called FLOW.

FLOW can be viewed as a global modifier of the other Effort dimensions. In the Effort-Shape notation, changes in FLOW are one of the most important characteristics of natural human motion. As Bartenieff continues:

"Flow is the initiator of action. Although it is not necessarily dominant, and may not appear identifiable as Bound or Free, its neutral continuity as flux will still underlie all other Effort elements [3]."

The individualization of motion and the combinational role of the FLOW concept has been established from extensive observations of actual movements. We will provide experimentation to test whether our implementation of FLOW has similiar properties.

3.2 FLOW parameter of TAKE_ONE

The FLOW parameter is used to reflect the user's determination of the relative *weights* or dependency of any animation variable's position and orientation on the results from each of the simulation

[2] Bartenieff [6], page 53
[3] Bartenieff [6], page 55

methods. Thus, a user may designate that an animation may start off being 100% dependent on the kinematic position/orientation, but may be forced to be 100% dependent on dynamics at the end. (Examples of this would be a ball-toss or a free-fall in our GLASS example). Between the two extremes, (and associated with the times between starting and ending of the animation), the dependency may vary smoothly or discontinuously.

The FLOW parameter of TAKE_ONE is based on the key observation that the state of the world (e.g. a body's configuration (position and orientation) and their derivatives) is defined by each of the simulation methods. We may obtain these values directly from a dynamic simulation, from a kinematic simulation via approximation or ADM and from a constraint simulation via approximation. Further, the necessary information to instantiate any of the methods can be obtained from this state information without consideration of which method or methods contributed to the definition. (see Figure 3.1)

FLOW is represented by sequence of (time, value) pairs, where each value is a triple (k, d, c), for the associated instant in time. Each component of the value is a *weight* to be applied to the corresponding data from a (Kinematic, Dynamic, Constraint) triple-simulation. Thus, at any time t_i, if $position_j, orientation_j$ represents the position, and orientation of any animation variable from simulation method j,

$FLOW value = (k, d, c) \quad where \quad k + d + c = 1$
and,

$$resultant\_position = k \times position_k + d \times position_d + c \times position_c$$

$$resultant\_orientation = k \times orientation_k + d \times orientation_d + c \times orientation_c$$

An important property of this formulation is that incremental changes in any of the FLOW components are easily reflected in the others. Triangular Interpolation method gives weights to each of the kinematic, dynamic, and constraint resultant values, where the weights are interpolated between specified values to provide a smooth continuum of values. The final position at the designated time is a sum of the weighed values.

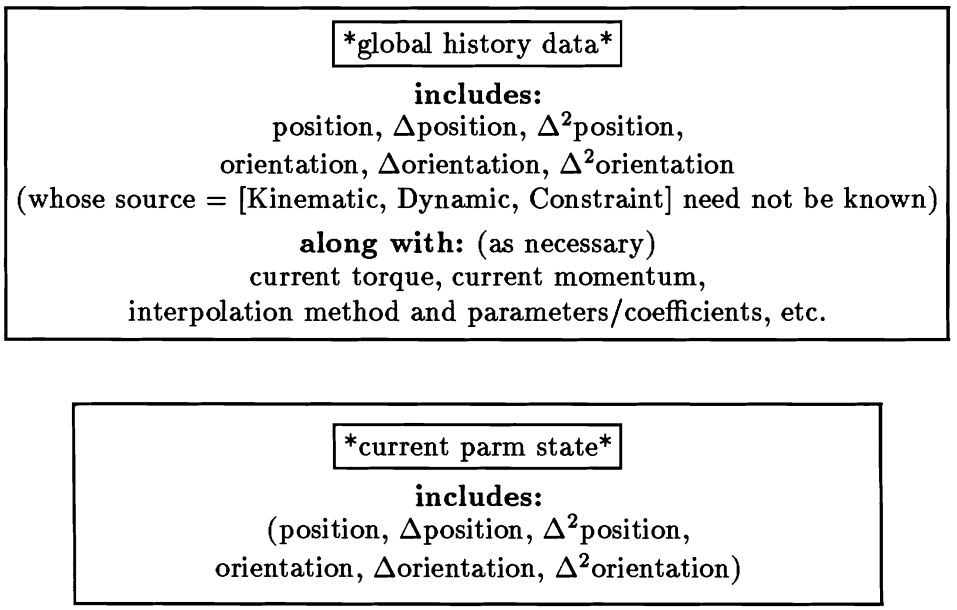
3.2.1 An example of FLOW

In this section, we refer to Figure 3.2.

FLOW need not be specified for all frame times. Instead, we input a FLOW specification of (time, value) pairs, (t_i, f_i), $i = 0, n-1$. To get the value of FLOW at the frame times, we use one of the interpolating methods. Presently, we simply linearly interpolate, but we will examine other methods of interpolation in our later work.

We break *Flow* into separate segments for the interpolation. In this case, we break the specification into two branches, one from t_0 to t_i and another from t_{i+1} to t_{n-1}, and interpolate from f_0 to f_i and from f_{i+1} to f_{n-1}. (We also allow a discontinuous FLOW to be specified by $t_i = t_{i+1}$ and $f_i \neq f_{i+1}$.)

Figure 3.1: Common DataBase per Animation Variable



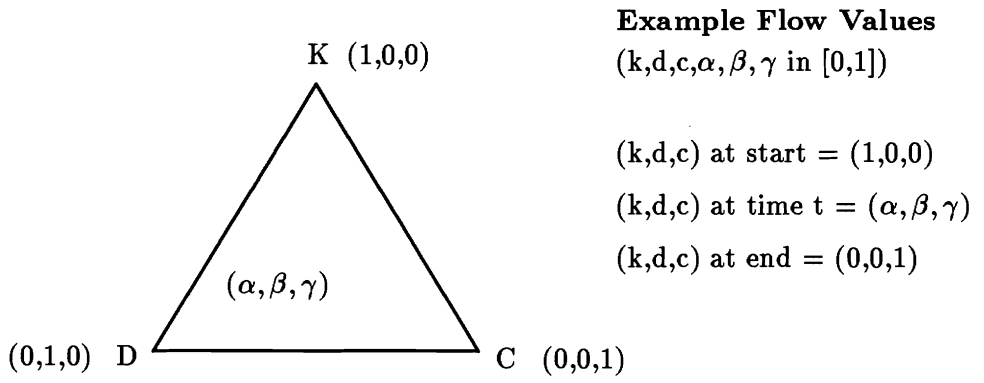


Figure 3.2: Flow Diagram: an example

The user can input a minimum of two FLOW (time, value) pairs for any animation. Suppose the user specifies a FLOW of $(t_0, f_0) = (0, (1,0,0))$; $(t_i, f_i) = (t, (0,1,0))$; $(t_{n-1}, f_{n-1}) = (0, (1,0,0))$ where t is some point in the (normalized) time range $(0,1)$. This FLOW signifies an animation that starts and ends with total kinematic control but achieves total dynamic control at time t. This FLOW is typical for motions that have determined starting and ending positions but may be driven by dynamics within the interior of the animation. Chapter 5 gives an example of FLOW in an animation that was produced by the TAKE_ONE system.

The determination of the value t may be obvious, as in the case of regular circular motion (i.e. $t = 1/2$). Alternately, it may have to iteratively determined, *determined as a constraint* or determined by other parameters in our implementation of the Effort-Shape formalism.

3.2.2 Use of FLOW to direct simulations

In the simplest case, the animation must have a full simulation for each of the methods to be merged via FLOW. A minor improvement is obtained by *directing* the invocation of a simulation method as needed by FLOW. If, for some method, for some interval, the designated weight from FLOW is zero, then TAKE_ONE should prevent the running of that method of simulation. [4] Conversely, if at any time, FLOW requires a value from a non-current simulation, a default value or an instantiation (possibly, with additional user input) should be done.

FLOW can invoke dynamics simulation by animation variable, avoiding a full dynamics simulation. Animation variables can be organized (see Research plan) into hierarchical groups with variables *inheriting* the dynamics specifications of other parameters. Since FLOW allows the *parallel* power of both kinematic and constraint specifications, in some cases, the dynamics simulation may only need to include certain forces (e.g. torque or gravity) as in the case of PIROUETTE. This allows the use of closed form equations for the dynamics simulation, greatly decreasing computational complexity.

Similiarly, since constraints and dynamics are integrated into FLOW, only the kinematic specification of **actual** goal positions is required. This reduces the complexity of the kinematic specification.

Further, the inclusion of a kinematics simulation avoids the necessity of determining the forces for inverse dynamics to achieve a goal position at some goal time. FLOW allows the switch from forward dynamics *driving* the action to inverse dynamics *controlling* the action to be specified in the Flow array.

3.3 Motion Semantics of the TAKE_ONE SYSTEM

3.3.1 Categorizing FLOW

Providing precise, expressive and easy interfaces to TAKE_ONE is a major goal of our work. While the power of FLOW can be easily demonstrated, it still needs an method of specification that can be easily defined and interpreted by a non-expert user. We suspect that only certain combinations of FLOW will be seen in natural motion. A known useful FLOW is: *all kinematics* at the end-points of an animation, *all dynamics* by the mid-point. This is the FLOW used in initial runs of the GLASS example. (The final version has 50% at the midpoint, since at 100%, the effect of gravity

[4] For a kinematic simulation, we must either run the entire simulation or else entirely exclude it.

would cause the glass to contact the table almost immediately after the start of the animation. This actually is a more interesting animation; for illustrative purposes, we choose a slower rate of descent).

Other possibilities are dual of the above (*all dynamics at the ends, all kinematics* at the mid-point) or (*all kinematics* at each key-time of a kinematics simulation, *all dynamics* at the mid-point of each interval). We will examine the possible categorizations of FLOW in our planned work.

3.3.2 Modification by differentials "more" or "less"

Well defined semantics for "more" and "less" can be offered for both the ADM parameter described in the next chapter and FLOW, since they are continuous dimensionless quantities. In FLOW, for example, we can specify:

$$\textbf{for "more" kinematics: } \delta k = (1 - k)/2$$

$$\textbf{for "less" kinematics: } \delta k = k/2$$

where the other δd and δc change accordingly. If repeated, these form a natural progression of refinements.

It is interesting that a user, once given the ability of parallel specification, can easily adopt corrections to an animation within this framework. Thus, in a test run for TAKE_ONE, a previously kinematically defined (i.e. overly smooth) motion of a leg kick was immediately critisized for 'not having enough dynamics' driving the motion at the highest part of the kick. [5] Thus, there is some evidence that this method will provide a partial answer to our goal of refinement.

3.3.3 Adding "Intelligence" to TAKE_ONE

The addition of FLOW and ADM parameters, to the specification of possibly three different simulations, would accentuate the user-specification task, were there not any orderly method to provide defaults for at least some of these values. Associated work by other researchers is being done to address this issue from a process-control level, though we must will provide a basic ability for generating default value for the parameters of TAKE_ONE.

Another need is an intelligent database so that typical animation variables (e.g. "right shoulder") might be known to the system. We can immediately recognize the necessity of this database since the inclusion of dynamics forces requires that the 6-degrees of freedom for a single joint be identifiable, so that directional directives of the joint's motion might be queried.

3.4 Summary

The motion semantics of TAKE_ONE will be developed to provide the availability of the system to a non-expert user, control the explosion of complexity and allow expressiveness to be included in an animation. Thus, this will satisfy all but one of our criteria for *specification* developed in Chapter 1.

[5] Informal Testing, Graphics Lab, 1988

The remaining problems of our criteria demand closer examination. We need to test whether the FLOW and ADM parameters of TAKE_ONE are sufficient to achieve individuality of an actor or of a particular instantiation of an action. We still have at our disposal the other dimensions of Effort-Shape whose implementation could provide additional specification.

The question of non-orthogonality (overlapping) specifications is partially resolved by the ability to control the merging via FLOW and also, by the ADM kinematics-dynamics join application (an example of the latter is in Chapter 4.) Since we have only addressed the question of how to resolve non-orthogonal specifications and not how to prevent or minimize it, our model is clearly insufficient. (We might again remark that in an interactive, iterative system, that minimization of non-orthogonality is equivalent to providing the user with clearly definable tools of specification.) This insufficiency shows that at the present time, for this criteria, our system has done no better than the other hybrid models. However, there is no loss of functionality of any of the KDC methods in TAKE_ONE, and it achieves a truly parallel method of specification. The concept of *generalization*, since it allows the minimization of specification (and therefore, minimal opportunity for non-orthogonality) will allow us to test the degree to which this non-orthogonality can be controlled.

Finally, in order to allow iterative refinement, we must carefully handle the addition of specification at each iteration. This procedure will be examined in the implementation of *generalization* in TAKE_ONE. Important issues that must be addressed are continuity of specification (to yield *a limit* and methods to recognize when that limit is not possible (i.e. further refinement is constrained along some dimension).

An additional result of this work will be a method to *construct* an essential definition of a motion, thereby satisfy our remaining criteria of *specification* and *analysis*. PIROUETTE and also, the wheel turning example, will provide our test cases for evaluation of *generalization*.

Chapter 4

ADM parameter of TAKE_ONE

In this chapter, we introduce a method of animation control by Attack/Decay Map of Keyframe Times ("ADM"). We discuss, in the TAKE_ONE system, how this parameter can be used to provide a *minimal* definition for a kinematic specification. We discuss the modification of an existing kinematic specification via the ADM map to achieve acceleration control at the end-points. We show how this can be be used in an application at a Kinematic-Dynamic simulation boundary. We next discuss generation of a kinematic specification via the ADM map to *structurally and iteratively* define the specification. This application will provide a solution to the problems inherent in traditional keyframe animation outlined in Chapter 2.

4.1 Introduction

As we presented in the previous chapter, the Effort dimension was abstracted as *the relative attack and decay acceleration* of a motion. This is the ADM parameter of TAKE_ONE, which we will examine in detail in this chapter.

The ADM has as a basis the interpolant method Stekette [85], which allows temporal control to be separated from spatial control by separately interpolating the keyframe time to keyframe index function, and the keyframe index to motion variable value. This was extended by adding a third interpolant - a preprocessing Hermite function - mapping key times onto key times to effect acceleration control at the end points. In this application, the Hermite function is constrained to be monotonic to preserve the order of the key events.

An immediate advantage of the triple interpolant method is that it can be used for direct modification of keyframe times to achieve continuity of acceleration across a Dynamic-Kinematic simulation boundary. Further, the juxtaposition of a dynamics simulation can use (or define) these accelerations, for user control over the degrees of continuity between the two different simulations. That is, an ending acceleration found from a dynamics simulation, can be translated to a starting acceleration input to the ADM method, to provide second-order smoothness across the simulation bounds. Similiarly, an ending acceleration can be used as input into a dynamics simulation, that begins at the end of a kinematic positioning. We discuss both these points in the detailed example presented in Section 2.

Internal keyframes are less affected by the Hermite (ADM) mapping than keyframes occurring at the end-points so that this modification is appropriately localized. However, there is a trade-off of low-level control of the keyframe times to achieve the higher level goal of acceleration control.

The ADM, however, is sufficient to specify smooth sequences of key-times for any sequence of key events. We will briefly describe in section 3 how TAKE_ONE might iteratively define a kinematic simulation through the use of a small number of basic keyframes and ADM mapping parameters (i.e. kinematic *generalization*). In this manner, the user would achieve the greatest control over the specification.

The ADM method, by the use of starting and ending accelerations, can only minimally affect interior keyframe times. This would prevent our attempts for *reduction* since the space of Hermite-interpolated functions is smaller than the space of continuous functions, which is the family to which monotonic sequences of keyframes times belong. However, a little reflection shows that the number of *true* keyframes, i.e. actual kinematic goals, for most actions, are few if the ability to specify constraints, and dynamic criteria is available.

We will return to *generalization* and *reduction*, in our planned work.

4.1.1 Associated Questions

Bound on ADM

The requirement that keyframe times, under a *reduction* transformation, maintain their temporal order, places bounds on the magnitude of the ADM starting and ending inputs. [1]

Further, the ADM bounds in conjuction with dynamics specification leads to a relationship between the allowable bounds and the *absolute* duration of an action. Thus, the differences that we see everyday, of modification of the *absolute* times of essentially similiar action, depending on context of the surrounding movements, may be affected by this requirement for monotonicity. We hope to examine this possibility.

Time as a dependent variable

An associated concept is to generally remove **absolute time** from the specification of an animation, and instead replace temporal specifications by range constraints of TAKE_ONE parameters. We will investigate the extent of this possibility in our planned work.

4.2 A Detailed Example: ADM use in Interaction of Methods

In this section, we refer to the animation specification that is described in Figure 4.2. We will use the notation "KP" to represent the term *Kinematic Positioning* which is TAKE_ONE's terminology for a keyframed animation (kinematic simulation).

Between t_0 and t_i, the user has provided a kinematic positioning KP#1. FLOW requires a dynamics simulation starting from t_i. At t_i, the kinematic simulation defines a $current\_parm\_state_k$. Using this value, a dynamics simulation can be initiated and FLOW used to merge the two methods between t_i and t_p. The relative weights of this merging would be found from interpolating FLOW from (t_i, (1,0,0)) and (t_p, (0,1,0)) to produce a FLOW value (α, β, γ) , $0 < \alpha < 1$, $0 < \beta < 1$, for any time t, $t_i < t < t_m$.

At t_j, FLOW requires a value from a kinematic simulation, but KP#1 has ended. By default, we can assume a holding value from KP#1 and continue uninterrupted until t_m.

[1] This work has been completed and will be presented in the thesis.

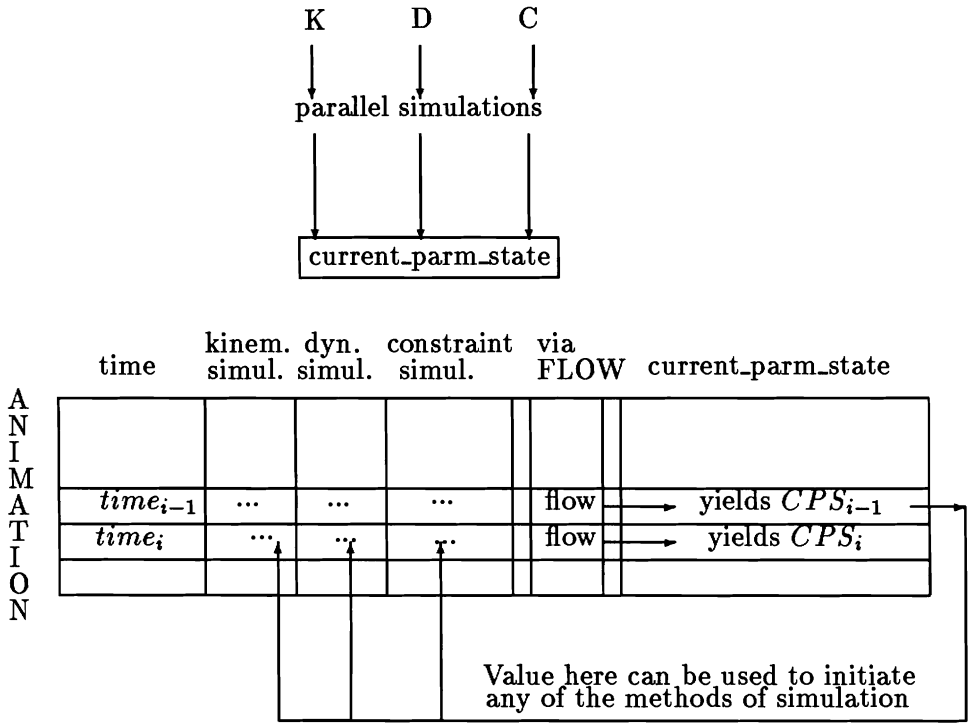


Figure 4.1: Interaction of Methods

At t_m, another kinematic simulation is specified by the user (see below for an alternative, at this point). The user may wish to simply insert the new KP#2 and continue in a standard way, merging via the FLOW or can optionally require the KP#2 be *smoothly merged* by the current_parm_state at t_m. To do the latter, TAKE_ONE would use the current_parm_state to obtain ADM parameters to transform the keyframe times of KP#2 so that the initial accelerations match the current values. This would produce a kinematic positioning which is closely related to the current dynamics simulation. The effect is a *capture* of the dynamics simulation by KP#2.

A more interesting case arises if KP#2 is not fully specified.

In the extreme example, only the terminal keyframe (at t_{n-1}) and an ending ADM parameter (ending acceleration) are specified. KP#2 keyframe at t_m would be the default value held from the keyframe at t_j, and the initial ADM parameters would be obtained from the current_parm_state at t_m. The ADM map is then used to generate KP#2.

In either case, FLOW between t_m and t_k now expresses the rate of increasing influence of kinematics in the animation. That is, FLOW between t_m and t_p specifies the rate at which the inverse dynamics would have needed to be specified, in a pure dynamics system.

At t_p, since the FLOW value is not at $(1,0,0)$, the dynamics simulation between t_p and t_q is not needed and TAKE_ONE would not continue the dynamics simulation part t_p.

Comment: If, instead of a FLOW value at $t_m = (0,1,0)$ and at $t_p = (1,0,0)$, we had FLOW values at $t_m = not - specified$ and at $t_p = (0,1,0)$, then between t_m and t_p, the FLOW value would range from $(\alpha', \beta', 0)$ to $(0,1,0)$. That is, between t_m and t_p, the dynamics effect would be increasing, not decreasing as in the above example.

The effect of KP#2 (under any option) is a *stabilizing* effect on the dynamics simulation. In this case, if KP#2 is not specified via ADM parameters determined by $current\_parm\_state_{t_m}$, a user might see a discontinuity in the animation at t_m. This is not a serious issue since TAKE_ONE has substantially provided alternatives to this situation. First, a KP simulation itself may provide approximations to the desired values through the common *current_parm_state*. Secondly, as we will see in this Chapter, since ADM addresses most of the problems in standard keyframe animation, a user is encouraged to use ADM for even *pure* kinematics simulations (see Section 4-6). The best solution is to encourage the user to use the ADM parameters at such transitions by providing an useful interpretation of the ADM parameter from the Effort-Shape formalism.

4.3 Iterative Specification

The minimization of the number of keyframes leads easily to the *generalization* of a kinematic positioning. Given at least two keyframes to specify starting and ending configurations and times, the ADM can be used to generate a kinematic simulation as follows. First, we might generate a small number of intermediate keys at proportional equal times,and at proportional equal changes arc changes of positional values. (Such decompositions are always possible along a continuous kinematic path). The ADM input parameters can then be used to specify a sequence of key times for the generated key values, that reflect the users' desired starting and ending accelerations for the term of the kinematic positioning. In this way, default kinematic values can be generated for the FLOW method, and also, the user is freed from defining keyframes simply to provide enough information to use a cubic spline. We will return to this issue in our planned work.

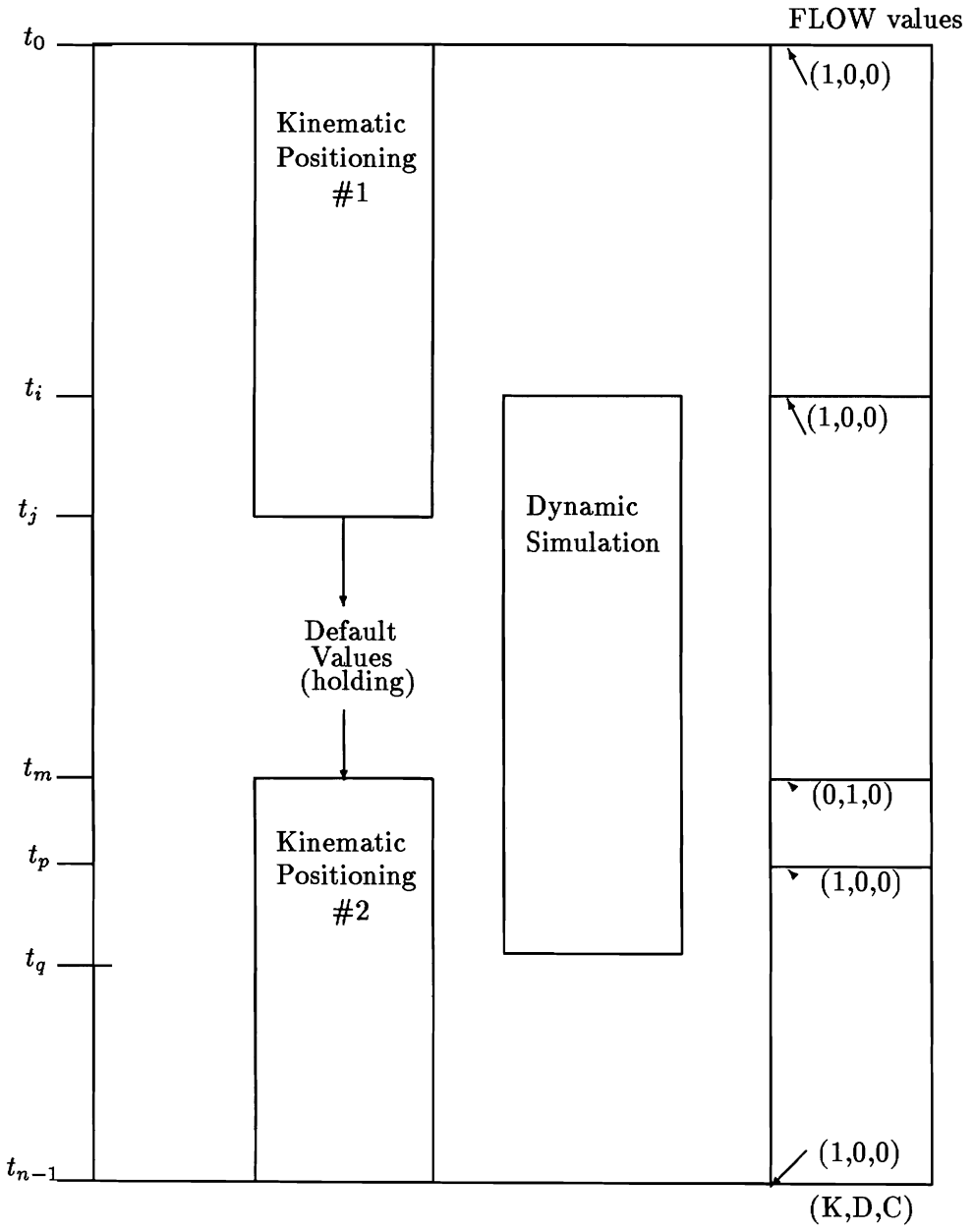


Figure 4.2: Animation Controlled via FLOW

4.4 Independent Use

The ADM method addresses the concerns in keyframe or key parameter animations outlined in Chapter 2 and may be viewed independently as an alternative method to traditional kinematic animations. As saw outline in Chapter 2, the problems with keyframe or key-parameter systems are:

- *Keyframe Animations were Overdefined*
- *Keyframes at non-key times can introduce errors or redefine the animation*
- *Keyframe methods lack ways to specify dynamics and phrasing (joining) of animation segments*

The ADM provides solutions to these since:

- *ADM needs a minimum of specification to initiate a kinematic simulation; actions can be decomposed as finely as possible and the ADM parameters used to smoothly merge the pieces.*
- *dynamics can be added to the end-points as a direct function of the ADM parameters.*
- *ADM needs only 2 keyframes and two parameter values to generate a basic simulation. Refinement of the animation can be assisted through the joint addition of automatically generated key positions and heuristic user modifications*
- *ADM has simple, structured iteration as a default*

The interaction of parameters may be simply addressed by any of the current kinematic simulation systems by the addition of polynomial and procedural forms.
The remaining problem which are not directly addressed by ADM are :

- *how to address range of acceptability or "Don't Care"*

This concern appears to not be able to be handled by ADM, and so must be addressed by other parameters or techniques of TAKE_ONE.

Chapter 5

Examples

5.1 Introduction

In this chapter, we present an example where taking a strict approach by any of the KDC techniques leads to expressive weakness or control difficulty. We then present our implementation of a simplified version of this example, below.

5.1.1 An Example Requiring Hybrid Control

There are numerous tasks whose specification requires a carefully choreographed interplay between kinematics, dynamics, and constraints. One such task is to place a heavy object on a cluttered surface. Ideally, kinematics would give the starting position; dynamics would cause the object to pull the arm down, and constraints would be used to cause the object to end up flat on the surface and to avoid obstacles on the way down. With kinematic control only, much interactive path planning would have to be accomplished along with careful spacing of the key times to get the proper dynamic effect. With dynamic control only, a complex set of time-varying forces would have to be determined to avoid the obstacles and stop exactly on the surface. With constraints only, the dynamics of the object could depend on the method of weighting the obstacles to be avoided; this together with a cluttered environment might result in an unusual path.

Through FLOW, the problem would be modeled with a changing dependency on each of the methods of specification. At the beginning of the motion, a kinematic specification, that is, potential key positioned targets at given key times, would dominate. At sometime during the movement, dynamics in the form of a gravity force would be applied (perhaps with joint stiffness forces to model some of the arm dynamic characteristics.) Constraints would be considered along the entire path to avoid obstacles and to control the final surface to object relationship.

In our example, as the object moves, kinematics becomes less important since, after the initiation of the action, the positional goals become increasingly better represented by dynamics and by constraints. The dynamics influences are relatively constant until the very end where they fade away to permit a graceful conclusion. Constraints modulate the dynamics path until the end when they dominate totally to fix the object neatly to the surface.

5.1.2 A Detailed Example

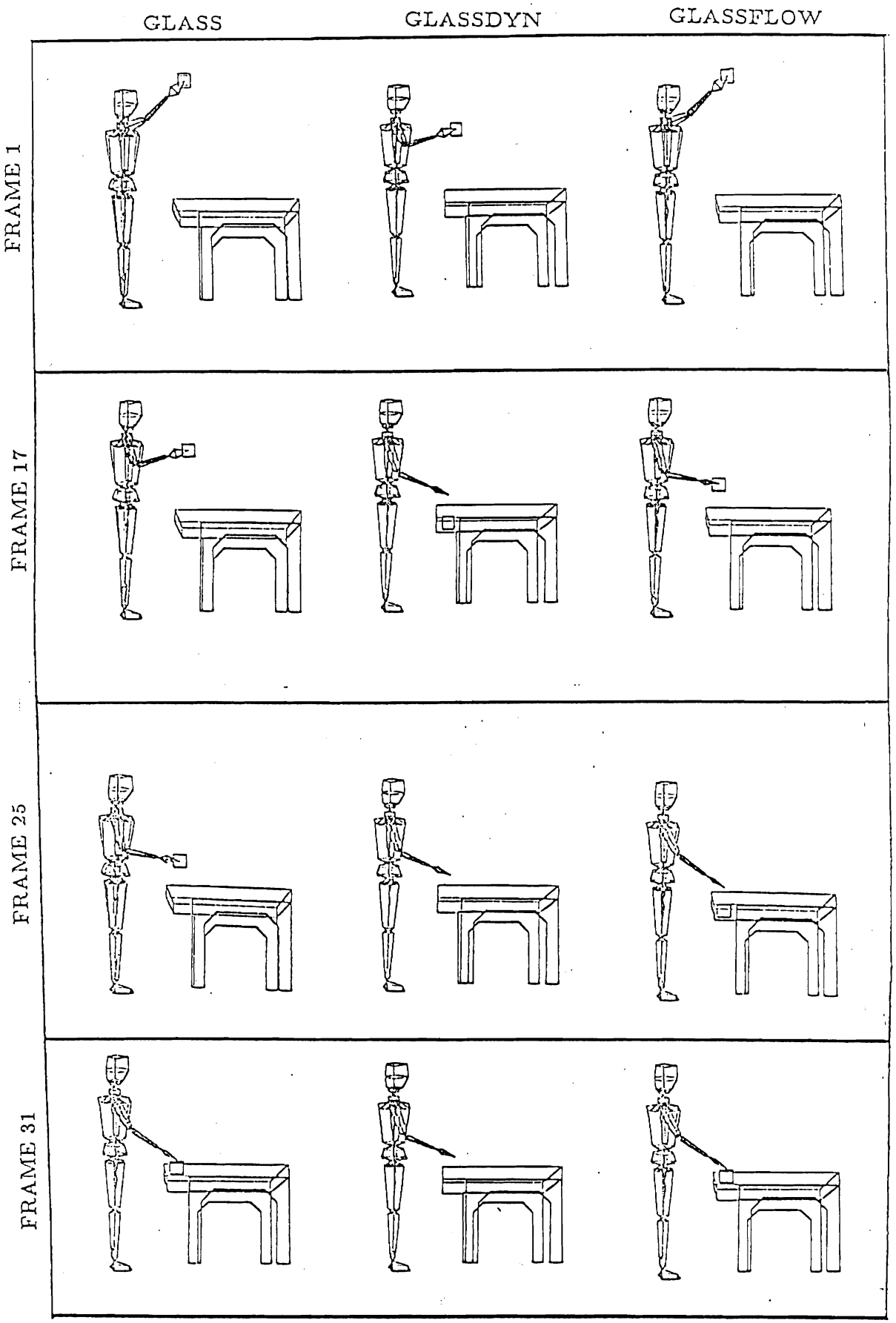
Our detailed example is a simplication of the object placement example described above. Also, our example does not include a constraint to prevent the glass from intersecting other objects in the environment.

In Figure 5.1, a series of frames are given for three versions of the animation. We include GLASS, which models the object placement purely kinematically. For this animation, we chose the least defined specification: i.e. the starting and ending keyframes and two in-betweens at proportionally even change in time and position.

The second animation is GLASSDYN, which is a purely dynamic model. In this animation, the only forces are gravity to effect a free fall of the glass and an upwards force imparted by the hand. The hand "reaches" for the glass at every frame corresponding to the kinematic keyframes. The figure does not show the motion of the hand or glass between FRAME 1 and FRAME 17 where initially the upwards force dominates. Very quickly, however, gravity overcomes this force, and the hand and glass fall rapidly. By FRAME 25, the hand has lost the ability to "reach" the glass and the motion of the hand ceases. The glass, however, continues to fall out of view. Since the user has access to kinematic control, the user can specify only those forces which are considered to *effect* the "dynamics" of the action, not a complete set of forces which are needed to *drive* the action.

The third animation is GLASSFLOW, which is an example of a pair-case in the FLOW system. It merges the first two animations with the FLOW values of (time, (k,d,c)) of (0.0, (1.0,0.0,0.0)), (0.5, (0.5, 0.5, 0.0)) and (1, (0.0,1.0,0.0)). That is, at time 0.0 (FRAME 1), the animation is controlled by *all kinematics*; at time 0.5, (between FRAME 15 and FRAME 16), the animation is equally effected by kinematics and dynamics; and at time 1.0 (FRAME 31), the animation is again controlled by *all kinematics*. FRAMES 17 and 25 show the effect of the dynamics specification to achieve a faster descent, while FRAME 31 shows the kinematic goal of ending the animation with the glass on the table. This example shows the trade-off between complexity of specification and the local use of the appropriate methods, since without the power of constraints, the glass falls through the table (FRAME 25) and then magically rises to rest on top of the table in FRAME 31. Varying the FLOW values to reach *all kinematics* at the first time when the glass hits the table surface would correct this problem, but then the user would have to know in advance when this condition would occur. The use of constraints would prevent the glass from falling through the table without any change in FLOW. The FLOW value of (0.5, 0.5, 0.0) at time 0.5 was arbitrarily specified; the user could, in a refinement of the animation, require "more kinematic control" at the midpoint of the animation and the FLOW value there would be correspondingly increased. The result would be a slower descent.

Figure 5.1: Glass Placement Example



Chapter 6

Research Plan

6.1 What has been done

6.1.1 Usage and Current Options

We have a working system of TAKE_ONE which includes an implementation of ADM, and FLOW. The methods of simulation allowed are:

- Tradition Kinematic Simulation
- ADM for Kinematic Simulation
- Closed-form Dynamics Simulation

The only grouping of parameters we support are determined by the kinematic simulation keyframes. We must apply a single ADM map to all variables within that group.

We allow kinematics and dynamics closed-forms to depend on the values from any kinematic animation variable. The converse, i.e. having a kinematic position depend on the dynamics, is not yet possible.

We do not allow FLOW to direct the durations of the simulations, as we presented in our discussion of Figure 4.2. Instead, we do full simulations for each of the methods, and then apply FLOW.

6.2 Proposed Extensions

TAKE_ONE must be extended to include constraints, which will be done with the transfer of the system to use JACK. We will include the method of *generalization* by specifically implementing kinematic specification definition through an iterative ADM approach. We will develop a user interface to allow *generalization* to be done for a dynamics simulation by defining a data base of forces (gravity, torque) that can be selected by the user. We will address the methods of constructing *generalization* for constraints.

We will extend the implementation of ADM to allow *generalization.* We will examine the extent to which an existing animation specification can be partially *reduced to* parameters of the ADM map for insertion into a database of defined actions.

We will examine our hypothesis of the montonicity bounds on ADM have any semantic interpretation and examine the degree to which time can be replaced by ranges of TAKE_ONE parameters.

We will decide if *range of acceptability or "Don't Care"* can be expressed within ADM.

We will implement a basic ability to provide default values for the parameters in TAKE_ONE.

We will refine the handling of FLOW and ADM to include intuitive handling and test the relationship between our implementation and the Effort-Shape methodology. We will implement the remaining dimensions of Effort-Shape.

We will develop a sample database of actions to test TAKE_ONE's reduction methodology and to establish its validity for the Future work.

We will produce our test cases to show the effectiveness of our methodology to capture individuality and expression.

6.3 Future tasks

The main future task is to show the effectiveness of TAKE_ONE as a comprehensive method of human motion animation. The power of TAKE_ONE as a tool to aid in the development of essential definitions for any class of motion must be examined. The flexibility of TAKE_ONE to handle archived definitions of this sort, from a database of actions, must be tested.

Bibliography

[1] Amstrong, William W., Mark Green and Robert Lake, "Near-Real Time Control of Human Figure Models" *IEEE Computer Graphics and Applications 7*, No. 6, June 1987, pages 52-61

[2] Badler, Norman I., "A representation for natural human movement," *Technical Report*, Dept. of Computer and Information Science, Univ. of Pennsylvania,Philadelphia, PA,1986

[3] Badler, Norman I. and Diana T. Dadamo, "The Flow approach to animation control," 1988 (submitted to Visual Computer)

[4] Badler, Norman I., Korein, Jonathan, Korein, James U., Radack, Gerald, and Brotman, Lynne S., "TEMPUS: A system for the design and simulation of human figures in a task-oriented environment," *First Annual Workshop on Robotics and Expert Systems*, Instrument Society of America, 1985, pages 251-257

[5] Badler, Norman I., Kamran Manoochehri, and Graham Walters, "Articulated figure positioning by multiple constraints," *IEEE Computer Graphics and Applications 7*, No. 6, June, 1987, pages 28-38

[6] Bartenieff, Irmgard and Dori Lewis, "Body Movement: Coping with the Environment", Gordon and Breach Science Publishers, New York 1980

[7] Barzel, Ronen and Alan H. Barr, "A Modeling System Based on Dynamic Constraints," (submitted to Computer Graphics: SIGGRAPH '88 Conference)

[8] Dell, Cecily, "A Primer for Movement Description," Dance Notation Bureau, Inc., New York, 1970

[9] Girard, Michael, "Interactive design of 3-D computer-animated legged animal motion," *IEEE Computer Graphics and Applications 7*, No. 6, June, 1987, pages 39-51

[10] Gomez, Julian E., "Twixt: A 3D animation system," *Proc. Eurographics '84*, pages 121-133, July, 1984, Elsevier Science Publishers B.V., New York

[11] Grant Gail, "Technical Manual and Dictionary of Classical Ballet", Dover Publications, Inc., New York 1967

[12] Hammond, Sandra Noll, "Ballet: Beyond the Basics", Mayfield Publishing Company, California 1982

[13] Isaacs, Paul M. and Michael F. Cohen, "Controlling dynamic simulation with kinematic constraints," *Computer Graphics 21*, No. 4, 1987, pages 215-224

[14] Laws, Kenneth, "The Physics of Dance", Schirmer Books, Macmillian, Inc. 1984

[15] Phillips, Cary B. and Norman I. Badler, "JACK: A Toolkit for Manipulating Articulated Figures," *Technical Report*, University of Pennsylvania, Philadelphia, 1988

[16] Steketee, Scott and Norman I. Badler, "Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control," *Computer Graphics 19*, No. 3, 1985, pages 255-262

[17] Witkin, Andrew, Kurt Fleisher and Alan Barr, "Energy constraints on parameterized models," *Computer Graphics 21* No. 3, 1987, pages 225-232

[18] Wilhelms, Jane, "Using dynamic analysis for realistic animation of articulated bodies," *IEEE Computer Graphics and Applications 7*, No. 6, June, 1987, pages 12-27

[19] Takashima, Yosuke, Hideo Shimazu, and Masahiro Tomono, "Story driven animation," *CHI + GI '87 Proceedings*, ACM SIGCHI, 1987, pages 149-153

[20] Zeltzer, David, "Motor control techniques for figure animation," *IEEE Computer Graphics and Applications 2*, No. 9, November, 1982, pages 53-59

List of Figures

Contents