Aesthetic Exploration and Refinement: A Computational Framework for Expressive Character Animation

by

Michael Paul Neff

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Department of Computer Science
University of Toronto

Abstract

Aesthetic Exploration and Refinement: A Computational Framework for Expressive Character Animation

Michael Paul Neff

Doctor of Philosophy

Department of Computer Science

University of Toronto

2005

Character animation remains a very challenging task despite many years of research on improved tools and algorithms. Production-quality animation is extremely slow to produce and only a small number of skilled animators are capable of creating it. This thesis explores methods to ease the creation of character animation, with a particular focus on expressive aspects of movement that convey personality and mood. The goal of the work is to identify which aspects of movement can be productively given computational representation and how this can best be done; its intent is not to develop a particular user interface. To reduce the scope of the problem, a restricted set of standing movements – including gestures, posture changes and weight shifts – is used as a test bed. An extensive review of the performing arts literature was conducted to understand which aspects of movement are most salient to expression. This study produced a list of key expressive aspects of movement. Computational representations for many of these *movement properties* have been developed, giving them a precise semantics. In addition, a software framework has been built that allows the various movement properties to be combined in a consistent and predictable manner by first mapping them to a low-level motion specification that is then simulated either kinematically or dynamically. A key feature of the system is a range of input modalities, including: character sketches that allow global changes to a motion sequence, high-level properties that adjust many aspects of a particular action, and low-level properties that precisely control particular attributes. These modalities support both rapid exploration of the aesthetic space and detailed refinement. Directly representing these resources in an easy to understand manner allows them to be customized and extended to meet the needs of a particular animator or a particular character. An implementation and various animations have been generated to illustrate the effectiveness of the approach. This work represents a step towards creating a general language for movement that can act as a bridge across artistic and technical communities to resolve ambiguities in the discussion of motion.

Dedicated with love to my mother, Diane Neff, my grandma and grampa, Ken and Eunice Whiteley, and my sister, Christine.

Acknowledgements

Eugene Fiume has served as my advisor throughout my time at the U of T. I doubt this work would have been possible working under someone else. Eugene gave me the freedom to work outside of the normal orthodoxy of computer science and ceaselessly encouraged my efforts. Rare was a time that I would leave his office and not be more excited about my work than when I had entered. Eugene also set very important standards for what constitutes quality research and encouraged me to constantly question the position and impact of my work. I've learnt a great deal from Eugene - many things having little to do with computer science.

As a young animation researcher, I had a Dream Team for a Ph.D. committee that along with Eugene included Demetri Terzopoulos, Michiel van de Panne and Craig Boutilier. It has been a privilege to hear their thoughts on my work. I would like to particularly thank Michiel for being consistently generous with his time, both when he was a stalwart of DGP in the early days of my thesis, and later when he had moved west.

Norman Badler served as the external appraiser for this thesis and was the consummate professional, all on a very short timeline. I appreciate his efforts.

Stephen Johnson - a professor at the Graduate Centre for Studies in Drama, a former teacher and a friend - provided important guidance to the arts literature during the early stages of this work.

DGP has been a rich environment in which to conduct my graduate work and there are a great number of people worth mentioning that have helped make it so. I'll only highlight a few who were particularly significant to my work, with apologies to the rest: Petros Faloutsos and Victor Ng-Thow-Hing (the fathers of DANCE and generous tutors); Joe Laszlo (paper collaborator and man of endless ideas); Sageev Oore (research collaborator and great for all things artistic, musical or zany); Dave Mould, Glenn Tsang and Chris Trendall (many discussions on all things computer science and not); Qinxin Yu (many discussions and an always positive disposition as my desk neighbour during the last few years). John Hancock and Alan Rosenthal have also provided excellent support as lab administrators.

Financial support which made this work possible was provided by NSERC and the Department of Computer Science.

It is hard to imagine a more rich and enjoyable community than is provided by Massey College. Many of my best memories from the U of T were there and I thank my college friends for discussions and distractions both. Hopefully we will share a beer again.

Gene was my roommate throughout my entire time at grad school and, incredibly, never complained about my changing moods or stressed out moments during the process of this work - a true friend indeed. A big cheers as well to the rest of the Perth crowd: Mark, Rosie, Janusc, Alex, Jenn, Pierre, Julian, Ula, Scott and Karyn (also a wonderful roommate and partner in math lessons). Rick has been my co-teacher the last three years and a great friend throughout. Lisa was an important source of support during this work.

And of course, to the Africa crowd (Joe, Lydia, Jessy, Mbogo, Gladwell, Greg, Linda, Dan, Jen and Denielle - honoris) for having nothing to do with this thesis! Haraka, haraka haina baraka. I believe it, now I just have to learn to live it!

Finally, an enormous thank you to my family who made it possible for me to get to this point and supported me every step of the way.

Am I really done?

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Motion is the spontaneous expression of life, its visible form. Everything alive chooses of itself its attitudes, orientations, gestures, and rhythms. There is, perhaps, nothing more personal to a living being–as far as the observer is concerned–than its movements. In reality there is no such thing as movement in general; there are only the movements of individual things.

Jacque Ellul, The Technological Society, p.330

1.1 Motivation

Compelling characters are at the heart of most animated productions, yet it remains a significant challenge for animators to create rich, nuanced characters with unique personalities that will engage audiences. There are few tools that directly support the creation of expressive character movement – those aspects of movement that convey personality and mood. This work seeks to fill that void.

Many animation tools are not well aligned with the true needs of an animator. Consider how an actor approaches a role. When working with a script, much of his focus is on the *subtext*, not the text itself. Actors will try to determine what a character's objective is for each line in a play. A simple line such as "You're home early." could be used to show affection or distaste depending on the character's motivations. The key creative question is “Which message do you deliver?”, and in the case of animation, “How do you use body language to make the message clear to the audience?”. The subtext thus illustrates the true nature of a character, illuminating his inner thoughts and feelings, and it is through the subtext that a character comes alive. Consider a simple movement example where a script calls on a character to grab an object. That action is given meaning by how the character grabs the object. Does he grab it in an angry swipe? Does he carefully lean in towards it to investigate it? Does he recoil away from it in distaste or fear? Does he show disdain for the object?

Tools that simply meet a constraint in a natural way, such as touching an object, provide little help to the true work of an animator. They achieve the text, but do not help with the subtext. To create animation that communicates effectively with an audience, an animator must be able to make the kind of variations described above. It is in these decisions that the real work of an animator lies, and it is control over these aspects of motion that we wish to provide.

Computer animation tools have traditionally focused on achieving natural, functional movement. The focus of our work is to move to the next level – to give an animator the tools necessary to efficiently explore the many expressive variations in how a character makes a movement. The hope is that such tools will better align with the animator’s true task, both leading to higher quality motion and allowing a greater variety of people to do character animation.

Producing animations is currently a very time consuming process. This makes the cost of experimenting with different approaches to a scene very high and limits the amount of exploration that can occur. In theatre, exploration is an important part of character development and many approaches to a character are often tried and discarded. The process of exploration can lead to unique and rich characterizations. It is desirable to better support this same process of exploration in animation and this is one of the objectives of this work.

It must be noted that performers and animators work hard to create *specific* characters that show emotion and personality in *specific* ways. If one wants to portray Rick stranded in Casablanca, it is not sufficient to generate a generic melancholy. The unique, situated, cool melancholy Humphrey Bogart created is what is required, and it likely emerged through refinement and rehearsal. Systems that create generic "sad" or "happy" movements are likely to be insufficient for production work. Movements must be customized for a particular character.

Expressivity is a key consideration in most production animation work. Until high level animation tools help animators to create the specific, nuanced characters they require, these tools will not find penetration in the artistic work flow.

1.2 Goals

Stated succinctly, the overall goal of this work is to make it easier to create expressive character animation. Based on the preceding discussion, three main subgoals can be identified:

- To allow an animator to both rapidly *explore* different approaches to a scene and to *refine* a movement sequence to achieve the exact animation desired.
- To provide an animator with direct control over expressively salient aspects of motion.
- To support the creation of *specific* characters.

1.3 Problem Statement

From the goals, we can formulate the following problem statement that guides this work:

> Use the arts literature to determine which aspects of motion are expressively salient. Build computational models that support direct control of some of

> these aspects of motion. Develop a framework that integrates these tools into a single workflow and can be extended to add additional movement properties. Ensure that the system both allows an animator to rapidly explore movement ideas and to refine a motion sequence so that the desired movements for specific characters can be generated.

This problem is difficult because there is not a good understanding of which aspects of movement have the most expressive significance, or even an authoritative cataloguing of what the various aspects of movement are. The arts literature provides valuable insights, but does not represent ideas in a precise and rigorous manner that would support direct computational representation. A mapping must therefore occur between the two domains. In addition, there are not absolute rules that govern human expressivity. A particular movement might accurately convey an idea for one character in one situation, but not work for a different character or in a different situation. The background of the audience viewing the animation is significant as well. The subjectivity inherent in the problem is one reason why our software design includes the human animator as an important part of the process. Finally, expressivity is generated by a disparate range of movement aspects. It is not sufficient to build a tool that only addresses one or two of these, but the framework must be able to combine a broad range of these aspects together into a cohesive system. Achieving such breadth is a significant challenge.

The problem is important because a good solution to it can both make it easier for animators to create quality motion and open up the medium of animation to a wider range of practitioners. Animation is a very difficult task. Frank Thomas, one of Disney's "Nine Old Men" who developed much of the studio's landmark work, claims that there were never as many as 20 animators at the studio that had mastered the craft, despite a staff of over a thousand [158]. It can be argued that the application of computer tools to animation has not made the task significantly easier. Better tools may increase the number of people who can generate quality movement. Animation is also very time consuming, with it often requiring months of animator time per minute of onscreen animation. This limits access to the medium and also limits the amount of exploration that professional animators can partake in. Tools that better support exploration will likely lead to more interesting characterizations as animators will be able to experiment with a wider range of approaches. Reducing the time required to generate a piece of animation will open up the medium to a wider spectrum of people who have stories they wish to tell.

1.4 Strategy

Modelling expressive human motion is a very large problem. In order to make progress on such a broad problem within the confines of a Ph.D. a careful strategy is required. The following approach has been taken with this work:

1. **Focus on a limited subset of movements.** Modelling the full range of human movements is too large a task. Concentrating on a subset of movements allows them to be explored in more depth. This work will focus on the creation of realistic, skeletal, humanoid character motion over a limited range of standing movements, including gestures, posture changes and balance adjustments. Although limited, this range of movement demonstrates considerable expressive variation while also avoiding dynamically challenging movements that would turn the focus more toward physical simulation or low-level control.

2. **Solve the problems that matter most.** In order to make progress, effort should be concentrated on those aspects of motion that are most expressively salient. We used the performing arts literature as a guide in this regard.

3. **Provide depth in select areas.** In order to demonstrate that the approach can lead to high-quality results, some sample areas must be explored in depth. This has

been done with tension modelling, shape modelling and through the implementation of a specific set of movement properties such as succession.

4. **Provide a broad architecture that is clearly extensible.** Due to the breadth of the problem, it will not be possible to explore all aspects in depth. A broad architecture must demonstrate an overall approach to the problem that can clearly accommodate the varied aspect of expressive motion, is effective on a number of test cases and is clearly extensible to a wider set of cases. We seek to validate our system by providing both vertical slices through it, and a broad architecture that is extensible to a range of movement properties.

1.5 Themes and Principles

In solving the problem set forth, a number of themes and principles have emerged that will be repeatedly reflected in design decisions.

1. **Put the animator in the loop.** We want to help animators tell their stories. Animators are in a unique position to make subjective aesthetic judgements. They should be in control.

2. **Customization.** Customization arises from the desire to create specific characters that move in a particular way. From a computer science perspective, we are faced with the problem of balancing the conflicting needs for generic, reusable classes of characters and behaviours, with customization and specificity.

3. **Explicit representation.** If movement properties are explicitly represented in a comprehensible manner, their definition can be made precise, and they can be customized, edited and refined. Animation is clearly not solely a top-down or bottom-up process. Thus well designed explicit representations must also support animators working at various levels of abstraction. Building a high-level tool out

of well defined low-level building blocks allows both the high-level and low-level interface to be exposed appropriately to the animator. This allows her to make rapid, exploratory, high-level adjustments early in the process and then make low-level refinements as needed.

4. **Hierarchical control.** An animator will be able to interact with the system at different levels of abstraction. High level properties modify a number of aspects of a movement at once, supporting rapid, wholesale changes to a movement sequence and experimentation. Switching to lower level representations allows refinement. Explicit representations are used to facilitate these varied levels of control.

5. **Procedural representations.** Procedural representations of movement properties are used throughout. This provides an explicit representation and also facilitates customization.

1.6 Summary of Approach

The first step in solving the stated problem is to gain an understanding of which aspects of motion are the most significant to character expression. There needs to be a solid basis for this understanding in order to have confidence in it. It should thus be based on a rich tradition of study. For this we turned to the performing arts literature, the traditional field of expertise in expressive movement. For thousands of years, actors and dancers have needed to understand expressive movement in order to complete their work. For the better part of one hundred years, animators have struggled with the same issue. There is substantial literature in these fields that deals with the nature of movement and the relationship between movement and meaning. It is at once both a rich and a problematic source of information – rich because of the many insights into the nature of movement it contains; problematic because these insights are often not in an explicit form and are rarely represented in a rigorous mathematical way that would lead to easy computational implementation.

Most of this literature is intended for practising artists. It often includes physical and mental exercises through which a performer can induce an understanding of a particular aspect of movement. When particular analyses are proposed, they tend to be evocative rather than formally explicit, dealing with general areas of movement around the body, for example, rather than specific locations or joint angles which are the tools of computational representation. Performers need to "understand movement in their bodies", while we need to generate concrete, explicit models. Furthermore, different movement theorists have proposed different systems for talking about movement, each with their own terminology, and there is not always a shared language between them.

Despite these difficulties, the arts literature remains an invaluable resource. One quickly realizes that performance movement is different from daily movement [18], and we believe that the arts literature provides the greatest depth of information on the nature of expressive movement. We use the arts literature to guide our research, to tell us what aspects of movement are important and to make sure we are asking the right questions and solving the problems that matter. Due to its evocative nature, there is a mapping that must occur between the ideas contained in the arts literature and the final implementation of those ideas in an animation tool. The Arts tells us the properties that we need to model. It is then up to us to determine how best to model them.

Movement properties, or *properties* for short, are the building blocks of our system. Each property is designed to encapsulate an aesthetically relevant aspect of movement. A list of key movement properties was distilled from the arts literature. Computational implementations of many of these properties have been developed, along with a software architecture that allows these properties to be combined and applied to movements. Together, the properties and architecture form the backbone of our system and provide an appealing workflow.

Animators create animation sequences by applying and then adjusting properties attached to actions in a script. Because properties directly vary aesthetically important aspects of movement, they are more expedient to use than a keyframe system, while still providing fine control at an appropriate level. Properties also range from high-level effects that vary a large set of motion parameters, such as *CreateRecoil*, to specific low-level controls, like *SetDOFValue*. High-level properties derive from low-level properties, and all properties in our system are procedural.

The animator's workflow supports both exploration and refinement. An artist starts by listing a set of actions in a script which define an initial animation sequence. She can make global changes to the animation by applying different *character sketches* to the script. Sketches, written using properties, outline basic character features such as a slow tempo, hunched back and high tension in the shoulders. By switching sketches, an animator can quickly experiment with various broad characterizations. She can then refine the sequence by adding more properties and adjusting the action descriptions, character sketch and existing properties as needed.

Such a workflow needs to be contrasted with existing approaches. While the psychology and feelings of a character are often meticulously worked out in current workflow through the storyboard, dialogue and soundtrack, for example, there is little direct connection between such in-the-large views of a character and the lower-level motions that ultimately realize them. Our system promotes the early synthesis of character traits and behaviours, which are an executable and embedded part of the animation being constructed. It certainly does not obviate the need for low-level motion control systems, but it does provide a parametric wrapper for them.

The system generates both kinematic and dynamic motion based on forward simulation. Kinematic motion is faster to generate and will precisely satisfy the control input. Dynamic motion will not provide the same precision, as momentum effects can cause small errors in timing and extreme movements can even cause a character to unintentionally fall over. Dynamics, however, provides many benefits. It allows a character to automatically react to external forces. It creates subtle effects caused by momentum and gravity, such as overshoot and pendular motion as a character loosely brings his arm to his side. Tension control also provides an intuitive method for warping the motion envelope. Including both kinematic and dynamic motion generation in the same workflow provides animators with an easy migration path from traditional kinematic animation to dynamic animation.

A skeleton is used for our sample animations as this focuses the viewer's attention on what can be achieved by simply varying skeletal motion - what we control - and avoids the distractions of varying body types and clothing. The latter are important in creating a character, but they would be handled by other tools in the production pipeline and can obscure the role of skeletal animation in the final result.

A prototype system has been built to demonstrate the efficacy of the ideas developed in this thesis. Numerous animations have been created with this system and they are available online at: http://www.dgp.toronto.edu/people/neff .

1.6.1 System Overview

A high level overview of the system is shown in Figure 1.1, which illustrates the data path from human input at the top to animation output at the bottom. Input occurs through the four channels discussed above. The *script* contains a time ordered list of actions, defining the framework for what the character will do. *Actions* are the basic unit of movement in the system. A simple example script might have two possibly overlapping actions specifying that the character should wave at a certain time and adjust his posture. The *character sketch* makes global changes to the entire script. It is used to explore various broad character types, such as an old man or someone who is young and energetic. *Animator edits* are used to refine a motion sequence. Generally they modify *how* a movement is performed, but they can also be used to generate new actions.

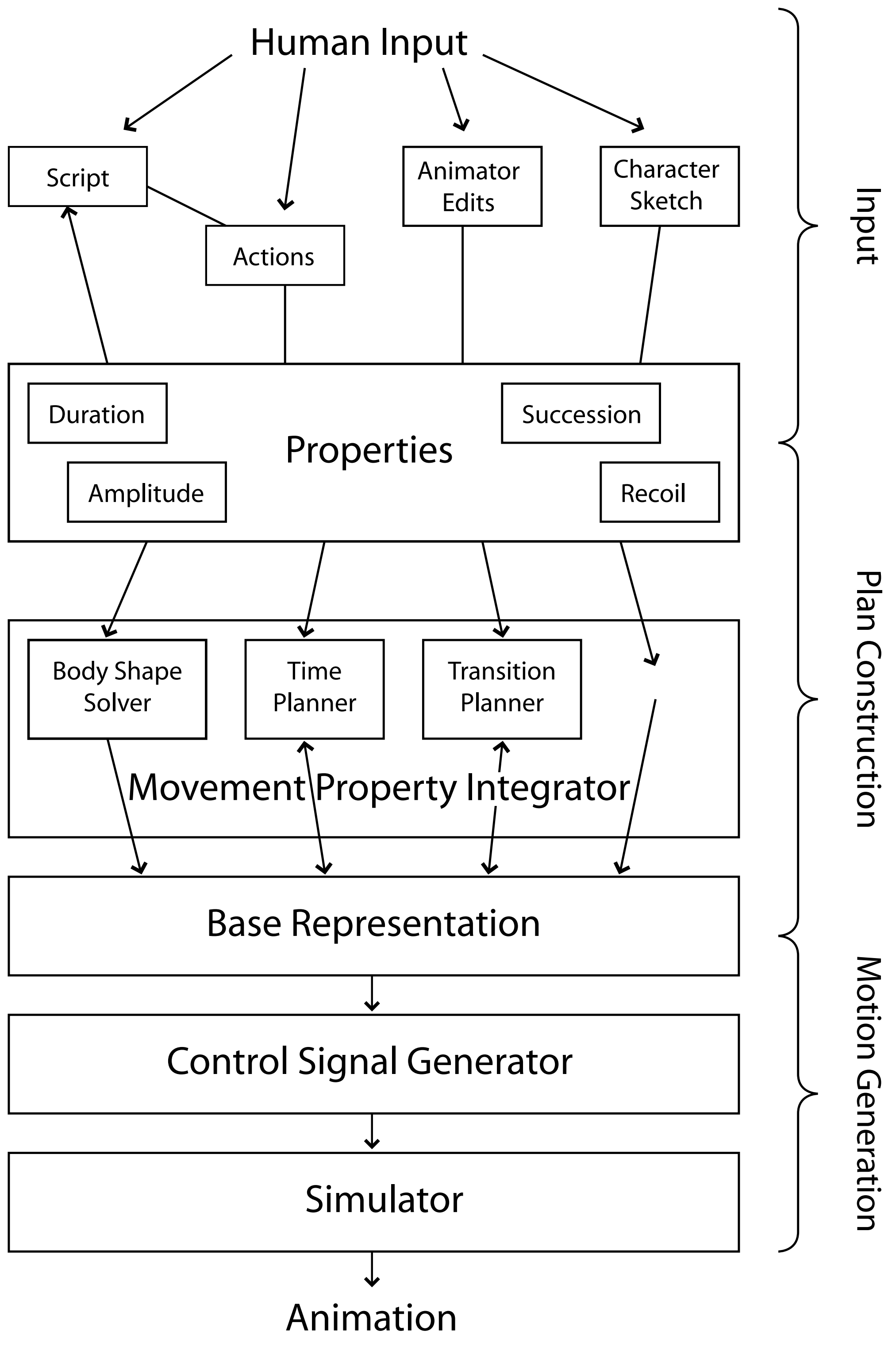


Figure 1.1: Basic system architecture showing the three main stages of animator input, assembling the motion plan and movement generation.

Once again, *Properties* define the movement language for the system. Actions are flushed out through the application of properties, and the character sketch and animator edits are defined using properties.

After the user has specified their instructions for the animation sequence, the various input channels are combined in order to generate a motion plan – the *base representation* – which can be executed to produce an animation. This process is coordinated by the *Movement Property Integrator* (MPI) which controls how properties are mapped into the base representation. Some properties use *planners* or *solvers* to achieve their desired effect. The system thus operates at two levels: a higher level abstraction that is convenient for the animator to work with and an underlying representation that can be simulated.

The *control signal generator* runs off the base representation and at each time step, provides the simulator with all needed control information such as computed joint torques. The *simulator* then updates the state of the system and exports animation frames. The system can produce either kinematic or dynamic motion by selecting the appropriate simulator.

Our software is built on top of the DANCE framework. The integration of our software into DANCE is discussed in Section 7.7, after the details of the system have been described.

1.6.2 Skeleton

Two skeletal models are currently employed in the system. The first is a 47 DOF full body skeleton that can freely move in world space. It is shown in Figure 1.2 and the degrees of freedom at each joint are indicated. The second model is a 28 DOF torso model. It has the same degrees of freedom as the full body model, but the hip is fixed and there are no joints below the hip. Both models include two DOF clavicles. Many previous physics-based animation models have not included these joints to reduce computational costs [173, 45, 103, 181]. In our work, the clavicles have been found to make a significant

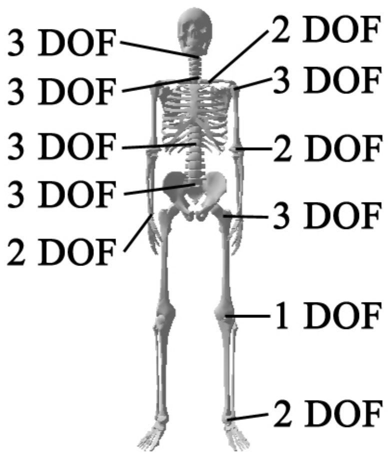


Figure 1.2: The degrees of freedom for the full skeleton.

contribution to the expressive impact of a character's movement.

Simulation details for the skeleton are contained in Chapter 6. Kinematic simulation provides real-time feedback, which is useful when iteratively refining a motion. Dynamic simulation is slower, running at about one tenth real-time for the torso model on an already quite old Pentium III 800MHz machine.

The focus of this work is not to improve the speed of physical simulation, but to explore how physical simulation can be used to improve the expressivity of animation. Moore's law and the work of researchers focused on accelerating physical simulation will bring about interactive rates for skeletons on the order of the complexity used here.

1.7 Summary of Contributions

The problem this thesis seeks to solve is how to create animation tools that: will let animators work directly with aesthetically meaningful constructs so that their work is more expedient and easier to control; will allow animators to work at different levels of abstraction, supporting both rapid exploration and fine tuning; and will provide animators excellent control over the final animation.

The contributions of this thesis fall into three major categories in support of this problem. First, based on the arts literature, we provide an analysis and categorization of the properties that make movements evidently expressive. This analysis is based on a synthesis of a wide range of works on expressive movement. As well as guiding our work, we hope it will provide important guidance to the field on what problems are important to solve in order to improve character animation tools. Some of this analysis was presented in [115]

Second, we have developed representations for a number of low-level, aesthetically important movement properties. These allow direct control of such properties as the amount of tension in a character's body, balance, timing and warping of the motion envelope. Our work on tension modelling was presented in [113] and three other movement properties – succession, extent and amplitude – were presented in [114].

One important movement property is the *shape* a character assumes with his body while executing an action. Chapter 5 will present tools that support both high and low-level control over a character's shape. These have also been discussed in [116] and [118] An application of the tools can be seen in Figure 1.3. The character needs to reach for an object. The important decision for the animator is *how* the character should reach for the object, as this will illustrate to the audience what the character thinks of the object. The figure shows four different poses produced by our system. The left most is the default solution and the right three each have a different *shape set* applied – the high level interface we use for varying pose. This is a good example of giving an animator control over the subtext – how the character reacts to the object – as well as the text – that the character touches the object. The shape control system also illustrates exploration and refinement. Shape sets take a very small parameter set that facilitates rapid exploration: the name of the shape set, an optional world space constraint and an intensity value. If the animator is not satisfied with the solution, he can load it into the low-level interface that has a larger set of controls and further refine the pose. This is facilitated by the high-level shape sets being built out of the controls that are available

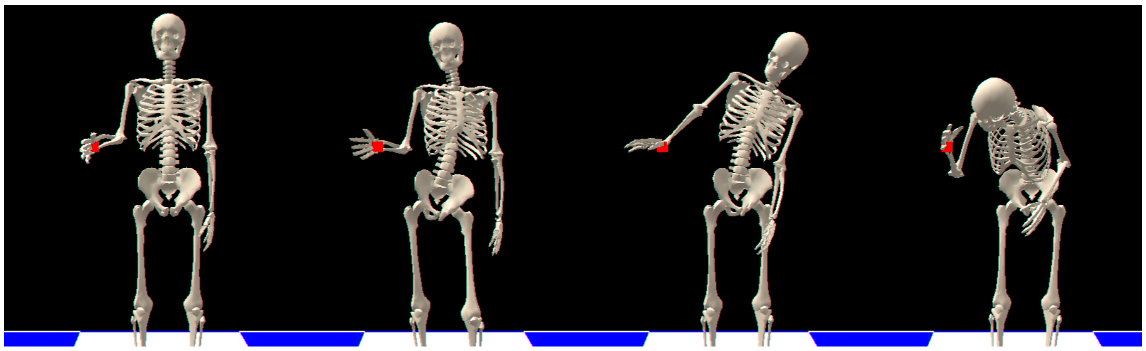


Figure 1.3: Four different reach postures satisfying the same constraint. The last three use shape sets to modify the expressiveness of the stance.

in the low-level interface.

The third contribution is the development of a software architecture that supports the integration of the varied movement properties into a final animation sequence. This was presented in [115] and [117]. The system allows higher level movement properties to be procedurally defined using low-level properties, which in turn allows animators to work at different levels of abstraction, first roughing out a motion and then fine tuning it as needed. The full set of movement properties are mapped to a single base representation that can be executed to create an animation. Both movement properties from different input sources and movement properties that control different aspects of a character's movement are integrated into this final representation. Several different input channels are supported, including: action descriptions, a character sketch and animator edits. These support both broad exploration and detailed refinement.

Figure 1.4 shows the same frame from three different versions of an animation where different edits and character sketches have been applied to each version. In this frame, the character is gesturing towards the audience to show them an object. The first image is from the base sequence. In the second frame, an "old man" character sketch has been applied to the base sequence. In the third frame, "beauty-line" edits have been applied that cause the character to strike a more sensual pose and an "energetic" character sketch has been layered on top. Although not visible in the figure, the timing and motion

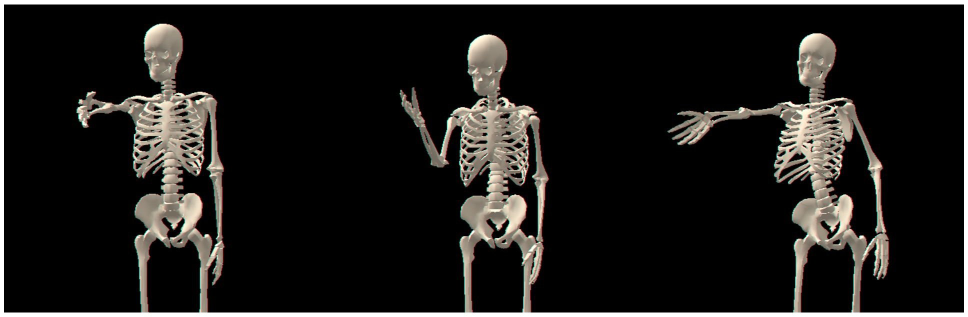


Figure 1.4: Three frames from an animation in which a character shows an object to the audience. The left frame is from the base animation and the right two frames are from sequences which character sketches and animator edits have been applied.

envelope of each animation sequence has also been changed. The architecture allows these adjustments from different sources and affecting different motion aspects to be seamlessly combined in the final animation sequence. Again, this example illustrates the principle of exploration and refinement as changing character sketches allows an animator to very quickly try different approaches to the sequence and animator edits can then be used to fine tune the motion.

Chapter 2

Scientific Background

By the work one knows the workmen.

Jean De La Fontaine, French poet (1621 - 1695)

It is fascinating to watch a person move and observe the interplay of limb movements, deformations of clothing, hair, skin and muscles, and changing facial expressions. All of these factors must come together to generate convincing, realistic character animation. Accordingly, significant research effort has been dedicated to each of these areas. The focus of this thesis is articulated skeleton motion, so only research directly related to that will be summarized here. An extensive body of work remains, much of it related to functional problems such as how a world space constraint is met or how physically accurate motion can be generated. A smaller body of work has focused on expressive aspects of motion and that will be given emphasis in this review. Other work will be summarized to better place this thesis in context.

In order for coherent movement to be created, there must be some notion of what makes a natural, coordinated, human movement. The discussion of previous work is categorized based on where this knowledge resides: *Data driven* approaches do not explicitly represent movement properties, but work by reconstructing motion from samples of high quality motion. *Data derived models* build a representation of a movement property from input movement samples. *Explicit models* also contain a representation of movement properties, but this is based on a body of theory, such as the arts, or biomechanics, or on animator knowledge. Both model based approaches normally provide users with broader control over the movement. *Physics-based animation* uses the laws of physics to generate physically plausible movement. *Direct control* approaches have a limited model of movement, but rely on the input of the animator to create reasonable movements. This category can be subdivided into *interactive* techniques that rely on real-time performance and *keyframe* techniques that rely on low level input from the artist. This chapter concludes with a discussion of several system level approaches to creating animation.

2.1 Data Driven Approaches

Data driven approaches rely on the reconstruction of captured motion with little explicit representation of movement properties. Motion capture has been an important animation technique since the early days of 2D animation when rotoscoping was first used. Rotoscoping is a process whereby filmed motion is traced onto cells frame by frame to create an animation [105]. Today, 3D magnetic, optical or mechanical capture systems are used to generate densely sampled joint motion data and the relevant issues are how to edit and reuse this data and how to create seamless, convincing transitions between sampled pieces.

Motion capture data provides very high quality motion, but also has several disadvantages. First, it is difficult to edit the motion, hence if a new motion is required, it will often need to be captured. Second, large motion libraries are difficult to search. Third, smooth transitions are required to transition between motion clips. Fourth, motion captured from one figure cannot be directly applied to a geometrically different figure without violating constraints, such as foot placements. Finally, animator control over expressive aspects of the motion is quite limited as this information is embedded in the captured clips and is difficult to edit. Recent research has tackled several of these problems.

Several techniques have been developed for editing captured data. Bruderlin and Williams apply multi-resolution filtering to motion capture data to break the signal into frequency bands [30]. Animators can adjust the gains of the various frequency bands in a manner analogous to how people use the equalizer on their stereo. They suggest different bands capture different aesthetic qualities of the motion, but do not provide a mapping between particular expressive properties and how to achieve them by modifying frequency bands. Witkin and Popovic edit motion by applying smooth deformations [172]. Animators can adjust poses at various points in the animation sequence. These new poses become constraints for a deformation curve that is added to the original motion. The method preserves high frequency behaviour and can also be used for forming transitions. Rose et al. apply spacetime constraints (explained in Section 2.5.1) to the transition problem [141].

Gleicher applies spacetime constraints to retarget motion captured on one character to another character with different limb lengths [54]. He minimizes changes from the original motion and maintains the high frequency attributes of the motion. The final motion is the original motion summed with a displacement curve. In similar work, Popovic and Witkin use spacetime constraints to edit motion capture data [131]. They extend the work by using a dynamics formulation that employs a simplified physical model and preserves the physical accuracy of the motion.

Work by Wiley and Hahn [170] and Rose et al. [140] blends between several sampled motions to create a continuous range of movements. Wiley and Hahn use linear interpolation while Rose et al. use radial basis functions. Rose et al. [140] focus on expressive variations in motion by interpolating within a large captured motion set. Actions, such as walking, are called *verbs* in their system. They capture the motion of each action of interest being performed in a large number of ways. These variations then define an *adverb* space. During playback, the user can specify an action and a location within the adverb space. The final motion is generated by interpolating between the captured motions. By modifying the location in adverb space, the user can change the style of the motion.

2.1.1 Generating New Motion Sequences

A significant focus of recent research has been on how to create a new motion sequence to satisfy a set of animator constraints from previously captured motion data. An early system was proposed by Lamouret and van de Panne [87] that involved sampling a large set of motions for a given activity and then storing very short duration samples (e.g. a single step from a walk). This would provide a motion database. Movements can be reconstructed by selecting the most appropriate sample for a given situation and then customizing it to exactly meet the given conditions. An animation would be constructed by stringing these short samples together.

Motion graphs have received much recent attention [8, 84, 94, 55, 9]. The basic idea of these techniques is to build a graph out of short clips from a captured movement sequence and then generate a motion sequence to satisfy user constraints by traversing this graph. Transitions between clips are normally generated automatically or semi-automatically based on the similarity between poses, velocity, orientation etc. either for a specific clip or a short sequence of clips. Lee et al. [94] accept a variety of forms of user input, including selection of clips, specifying a path for the character to follow or filming the user acting out motions. Kovar et al. [84] can generate different styles of locomotion along an arbitrary path. Arikan and Forsyth [8] use a hierarchical representation of their graph and refine an initial solution to obtain the final path. In later work, Arikan et al. [9] allow the user to paint the timeline with different verbs that are then used to decide which motions will be played. They use support vector machines to help an animator pre-tag the motion library with labels corresponding to the different verbs that will be available in the system. Gleicher et al. [55] have the user help design hub poses that form nodes in the graph. This provides a better graph design. The motions coming into and out of the hub are warped so that they can be played concurrently (*snapped together*) with no need to adjust them at run time.

In a related approach, Li et al. [97] build a two level model of motion. Individual clips are stochastically modeled as *textons* and transitions between clips are modeled in a transition matrix. When motion is generated for each clip, it will be statistically similar to the input motion, but visually different.

Pullen and Bregler use statistical approaches on motion capture data to generate animations with the natural variations present in human motion [134, 133]. They perform a wavelet decomposition on the motion capture data to divide it into frequency bands. For these bands they define various correlations between the different joint angles and sequential time samples. These are represented with Gaussian functions and can be resampled to generate new, but statistically similar, examples of the same movement sequence. In this way they can create a set of consistent yet unique animations from the training data. Their examples are of locomotion.

2.1.2 Summary

In summary, data-driven approaches work well when high quality motion is required and the animator needs only limited control over the motion. They are appropriate for applications such as games. When an animator needs to introduce significant "acting" into the sequence for high quality character work, as is required for movie production, data-driven approaches are less appropriate due to the difficulty in editing the motion. Acting can be introduced by motion capture performer, but it is difficult for an animator to add this in to a sampled clip. Also, high quality character animation will often be customized to be appealing when viewed from a particular camera, even if this causes the motion to be unrealistic from other view points. Generic motion capture data is difficult to customize in this way. Data-driven approaches give good looking motion at a relatively low-cost, again at the expense of a loss of control, so they may be appropriate when animator time is a scarce resource.

2.2 Data Derived Models

Data derived models develop a representation of some property of movement by analyzing motion capture data.

An intriguing approach for creating expressive animation is based on calculating "expressive transforms". In this work, motion is captured of a subject performing an action in a neutral way and in a way that corresponds to a particular emotion (e.g. anger). The difference between the two motions is then used to define an "emotion transform" that can be applied to other motions. Interpolation and extrapolation allow control over the emotion's intensity. A system based on a Fourier series expansion of the motion signal is presented by Unuma et al. [162], while one that uses a non-periodic basis is presented by Amaya et al. [4]. These works separate the style with which the motion is done from the physical task the motion achieves. While interesting, it is unlikely that this approach will generalize well for creating a range of characters. A captured "angry" transform is specific to the person who generated the motion and does not necessarily relate to how a different character would express that emotion, or even the same character in a different scene. Furthermore, when an expressive transform is applied to a motion, the resulting motion does not necessarily maintain the physical constraints present in the original clip and it is unclear how to re-establish these constraints while maintaining the effect of the transform.

A related approach was presented by Brand and Hertzmann [24]. They take movements performed in various styles as input and use a Hidden Markov Model to learn the stylistic variations of the different movements as well as key states in the movement. A style vector v can be extracted that allows different styles to be specified. They can use the Style Machine they create to either change the style of a structurally similar animation or to generate new animations using elements from the captured set. For one set of simple examples, the style vector had dimension three, but for a set of longer training examples the vector grew to 3500 dimensions. At this point, direct adjustment of the vector is unlikely to provide intuitive control and even the meaning of the three vectors in the first case is unclear. The system does support interpolation between specific sampled styles, such as modern dance or ballet. One drawback of this approach is that the animator does not have control over the nuances of the style, thus providing limited ability to refine the motion in arbitrary ways. Approaches based on learning or parameter extraction cannot guarantee that learned animation will remain physically consistent.

In a related approach, Vasilescu uses multilinear analysis to extract the motion signature from captured motion clips [168]. She views motion as the combination of the action performed and the movement signature of the person performing the motion. Her system works on walking motions and includes three components: the system can extract motion signatures, motion signatures can be used in the synthesis of new movements, and a subject can be recognized from their movements based on the model of their motion signature.

In a promising approach, Pullen and Bregler [135] present a system for texturing and synthesizing motion based on a statistical analysis of motion captured data. A user can provide a roughly keyframed piece of motion. The high frequency behaviour from the captured clip is then automatically applied to the keyframed sequence. In addition, they can determine values for degrees of freedom (DOFs) that are not modified in the keyframed sequence based on joint correlations in the motion capture data.

Data-derived models share many of the strengths and weaknesses of data driven approaches. They can give quite good quality results with limited animator input, but control remains difficult. It is often not possible for an animator to generate an exact feature that they require. Such tools should be effective in applications where the required quality of the motion is not too high. These include cases where the amount of attention paid to the motion is low, for instance for background characters. For characters that will be carefully scrutinized by audiences, such as the lead character in a movie, more precise control is likely needed. Data-derived models also offer the potential for people with very limited animation skill to create relatively good motion – certainly much better motion than they could using traditional tools without training. They also reduce the amount of time required to generate reasonable motion, although not necessarily the amount of time required to generate excellent motion as there is not a clear process for refining the motion generated by these tools. If the final output of these tools could be easily edited by animators, supporting a refinement phase, they might ease the animation task in even the most challenging case of lead characters. Research on data derived models has so far focused on the entire motion signal. Expanding these techniques to attempt to learn particular features of a movement sequence may allow these aspects to be represented and modified independently. This would significantly increase the utility of these models and make them more complimentary with the explicit models discussed below.

2.3 Explicit Models

Explicit model systems build a representation of the aspects of movement they are trying to represent, but unlike data derived approaches, the representation is based on movement theory, biomechanics, or the knowledge of the animator or system designer. The work of this thesis lies in the explicit model category.

The EMOTE system developed by Chi et al. [36] is the published work that has the most in common with our own. EMOTE is a kinematic system based on the Effort-Shape portion of Laban Movement Analysis that modifies the nature of arm and torso movement. *Effort* notation is used to model how a movement is performed. Chi et al.

[35, 11, 36] use Effort as the basis for four sliders that control the transition envelope of arm movements going from a rest position to touching a point in world space. The sliders correspond to Laban's four effort components: *Space*, *Weight*, *Time* and *Flow*. A detailed discussion of the meaning of these components is provided in Section 4.3.4. The user can also adjust *space* parameters that control the extent of the movement and the degree of torso involvement. This is one of the first systems to successfully use an analysis system from the performance literature to aid in the creation of animation.

Our work shares two key properties with the EMOTE system: both base their understanding of movement properties on the arts literature and both manually create explicit models of these movement properties. Our work differs from their's in that while they focus solely on the Effort-Shape portion of Laban's work, we attempt to synthesize a set of movement properties from a much wider sampling of the relevant arts literature. In the end, we incorporate a wider range of movement properties and create a system that is extensible to include additional movement properties as needed. As well, our property definitions are procedural and customizable, so they can be adjusted to meet a particular animator's needs. The EMOTE system does not allow the property definitions to be varied. Our system also supports the dynamic simulation of movement.

Perlin describes an interactive puppet system where multiple puppets can be controlled by selecting their action states from a button panel [125]. The system plays back predefined animations that are interactively selected at run time and controls transitions between them. Animations are defined by start and stop joint angles and interpolation functions, which are either sine or noise based. This work is significant as it introduced the application of structured noise to motion synthesis. Structured noise can be used to give sway and randomness to movement, helping to create a natural appearance. It can suggest physical effects without needing direct physical modelling.

Our work shares many of the underlying goals of this work and both systems provide an explicit representation of movement. Perlin's work uses noise to create continuous movement and suggest some physical effects. Our system can explicitly model physical effects. The action descriptions used in Perlin's system are low-level, providing transition functions for each DOF. Our system introduces higher level movement properties and allows an animator to define movement at different levels of abstraction.

Lee et al. look at the task of having a modelled human automatically move an object in a natural looking manner [96]. Their approach considers the strength of the model human and generates variations in the motion for different weight movement targets.

In their early work, Bruderlin and Calvert present a model for goal- directed human walking that combines kinematics and dynamic simulation [27]. Knowledge of a walk cycle is used to drive a simplified dynamic simulation that has a more complicated kinematic model fit to it to produce the final animation. They allow the animator to control velocity, stride length and/or stride frequency, as well as 28 secondary parameters that adjust the style of the walk. In later work, they replace the dynamic simulation with cubic spline and linear interpolation, claiming the interactivity this affords outweighs the small loss in realism [29]. They also introduce a kinematic grasping model. Biomechanical knowledge of the activity is combined with interactive control in their human running animation system [28]. This system also allows the user to adjust physical parameters, such as velocity, as well as expressive parameters such as the amount of arm swing or torso tilt.

Explicit models offer roughly opposite strengths and weaknesses to data based approaches. They are generally constructive in nature, starting from nothing and building a representation of motion, so more work is normally required to generate a quality motion. Conversely, data driven approaches often begin with a motion and then the user works to modify it, so the cost of generating reasonable motion is low, but the difficulty in editing remains. The advantage offered by explicit models is better control. They aim to offer the same advantage of direct control approaches like key-framing – excellent control over the final motion – while also making it easier for an animator to generate the desired motion. Properly designed explicit approaches offer the potential of allowing an animator to get precisely the movement features he wants, while also allowing him to work more expediently than with more low-level direct control approaches like keyframing. Explicit models also arguably have more potential for allowing a character to react to unexpected circumstances/interactions in a virtual environment or other online setting, as they can generate motion based on the current situation, rather than relying on motion that was generated in another situation and trying to remap it to the current circumstances. A final advantage of explicit models is that in general the effect of a given parameter in an explicit model is well understood because the model is hand crafted. This is not always the case in statistical or machine learning approaches, where parameters are automatically derived and their meaning may be unclear. Explicit models thus offer the potential for more precise control

2.4 Direct Control

Direct control systems provide very limited representation of movement properties, focusing instead on giving the user either interactive control or full offline control over the final movement. While not supporting real-time interactivity, our system does try to provide an animator with full control over the user's final motion.

2.4.1 Interactive Systems

Interactivity is very important in the creation of expressive animation. Reeves et al. argue that "...animation is most often successful because of interactive control and human input. A person must make perceptual and emotional judgements." [137, p.33] Interactive systems seek to provide the animator with real-time control over the generation of the animation. This allows him to use his subjective judgement to shape the development of the motion. Examples of this approach are the computer puppetry of Oore [122], the interactive dynamic control of Laszlo [90] and the interactive animation of system of Dontcheva et al. [39]. In all of these works, the animator uses an input device to interactively vary the motion of a character and gets real time feedback on a visual display. The animator has complete control over the final product. This is also an important design goal in our work. While these approaches provide direct, interactive feedback, our work operates on the director-actor metaphor. Here, an instruction is given to an actor, the actor performs a piece of motion taking that instruction into account and then the director can review the result and issue subsequent instructions. This approach allows us to dynamically simulate more complex models as it removes the real-time constraint and also allows the animator to work at a higher level of abstraction by introducing *movement properties* which form the language by which the animator instructs the character.

Thorne et al. [161] have introduced a direct control system called *Motion Doodles*. This system does not provide complete control over the final animation, but does allow a user to interactively specify the animation. They have developed a cursive symbol language for specifying motion which can be input using a stylus.

Other work has begun to explore the use of natural language input to control animated characters. An overview of the work Badler's group has done in this area is given in [11] and the work of the Thalmanns' groups is discussed in [34].

2.4.2 Keyframe Systems

Production keyframe animation systems such as Maya [2] offer an animator excellent control over the final motion. The keyframe approach continues to see extensive use in high end production, because while it is very painstaking, it does provide animators with the level of control they demand. Most keyframe systems provide largely low-level tools, including some of the inverse kinematics tools discussed below to help with character positioning. One of the goals of our work is to provide comparable control to keyframe systems while also providing animators with the option of working at a higher level.

2.4.3 Forward and Inverse Kinematics

Kinematics refers to the study of motion without reference to the underlying forces that generate the motion. In forward kinematics, the character's DOFs are interactively updated to change the character's pose. A hierarchy of transformation matrices is constructed to represent a character's skeleton. In this way, changes to joints closer to the root of the skeleton will cascade to limbs further from the root. For instance, changing the shoulder angle will move the entire arm, but have no effect on the torso. Interpolation in joint space normally leads to more pleasing motion than interpolation of world-space constraints the character must maintain, and hence is the basis of most keyframe animation systems. It is difficult, however, to satisfy world space constraints, such as touching an object, using forward kinematics. Inverse kinematics is often used to deal with such problems.

The *IK problem* is to solve for a set of joint angles that will allow a kinematic chain to reach a given point in world space. Research on inverse kinematics (IK) has been a significant focus in character animation for at least twenty years. A brief summary of this literature will be provided. Three main approaches have emerged to solve the IK problem. The first approach relates an instantaneous change in joint angles to an instantaneous change in the position of the end effector: $\dot{x} = J(\theta)\dot{\theta}$ using the Jacobian J. This relationship can be inverted by calculating the pseudo-inverse of J. By locally linearizing this equation and taking a series of small steps, the joint angles can be solved for. An example of this approach is [53]. The second approach to IK uses optimization to solve for the joint angles in order to satisfy the active constraints in the system (e.g., [179]). The third approach uses a dense sampling of motions and interpolation to achieve desired constraints (e.g., [142] and with reaching motions [170]). A variation on this approach uses learning techniques to generate poses close to those in a sample set [57].

Our *body shape solver* used a combination of analytic and optimization based IK. The analytic techniques are fast to compute and give direct control over properties of interest. Optimization is used with the torso and allows the system to trade off between various world space and aesthetic constraints. Lee and Shin [95] also use a hybrid IK solver, but their solver is not designed to meet the same aesthetic constraints.

While inverse kinematics tools are useful for positioning a character to achieve a world space constraint, they do not effectively facilitate the exploration of the widely varied set of expressively interesting postures available while meeting that constraint. In production systems, a character will often be rigged with many local IK solutions that only involve a few joints, allowing an animator to position various parts of the skeleton. This does allow the animator control over the character's posture, but in a way that is time consuming. Many full body IK systems support secondary objectives, but these are often simple objectives, aimed at ensuring that the motion appears 'natural'. One common secondary objective is to maintain joint angles near a desired value, often their centre [53, 174]. Interpolation methods will by definition generate the same posture as is in the sample data and it is difficult to vary substantially from this. Our work encourages an animator to actively explore the expressive range of posture.

Boulic et al. present a system that combines inverse kinetics for balance control and inverse kinematics for reach constraints [23]. Their solutions take the form of the pseudo-inverse strategy for inverse kinematics as discussed above. They cascade the solutions, using the IK solution as a constraint in the null space of the inverse kinetics solution. A similar approach is taken by Aydin and Nakajima [10], who extend the work with a force distribution scheme to deal with different centres of support. Baerlocher and Boulic present an IK system that supports constraints at an arbitrary number of priority levels [14]. Position and orientation constraints are supported for arbitrary end effectors, including the centre of mass. A look-at orientation constraint is also supported.

2.4.4 Balance

A number of systems have explored balance control. Phillips and Badler use an optimization based IK system to allow animators to interactively control the balance point of a character [129]. Wooten uses a feedback method to control dynamic balance which is inspirational in our work [173]. He computes an error between the desired projection of the centre of mass on the ground plane and the actual centre of mass projection. The X and Z errors are then used to update the corresponding ankle and hip angles to reduce the error.

Recent balance work has used the *Zero Moment Point* (ZMP), a measure of dynamic balance, to correct balance errors in motion sequences, ensuring that balance is always maintained [153, 123].

2.4.5 Summary

The strengths and weaknesses of direct control approaches are similar to explicit models and indeed the boundary between the two categories is blurred. Direct control approaches emphasize control, but this often comes at the price of increased animator work. Interactive approaches combine control with real-time authoring, but the quality of the final motion may not be as high. Offline systems like keyframing allow very high levels of control and can be used by a skilled animator to create excellent motion, but are also very laborious to use. Explicit model systems attempt to provide most of the control available in the low-level direct control systems while also allowing an animator to work more expediently.

2.5 Physics-Based Animation

A major thrust in computer animation research has been to use physical simulation to improve visual realism. The application of dynamics to character animation is part of this movement. Dynamic systems look at motion by modelling the underlying forces which create the motion. The fundamental problem becomes one of control: How can these forces be applied over time to create the desired motion?

Two main approaches have been used for generating physically based character animation. Optimization approaches, often called *spacetime constraints*, take the laws of Newtonian mechanics as constraints and solve for motion using an optimization process. Forward dynamic simulation generates forces at each time step, integrating Newtonian laws of motion to update the character state. Our work takes the latter approach, building on previous efforts in hand tuned control for torque regulation.

2.5.1 Optimization Approaches

Witkin and Kass formulated the dynamics problem as a trajectory optimization through time in introducing “Spacetime Constraints” [171]. They specify what a character has to do, how it has to do it, the physical structure or the character and a list of physical resources. The problem is formulated as a constrained optimization and solved offline, yielding a force trajectory through time. The equations of motion are used as constraints on the optimization process. The approach has been used effectively for low DOF models, but the computational cost remains prohibitive for complex models. This is due to the equations of motion which govern complicated articulated skeletons being highly non-linear, leading to a very difficult optimization problem.

Brotman and Netravali introduced to computer graphics a related approach called *optimal control* [26]. Traditional keyframe systems use interpolating splines to calculate the inbetween frames of an animation, but these do not preserve the underlying physics of the system. Brotman and Netravali use keyframes as their constraints and seek an optimal control path that will interpolate the keyframes in a physical simulation. Their approach solves a series of boundary problems between each pair of keyframes, minimizing control energy and maximizing the smoothness of the trajectory.

Cohen builds on both of these approaches in his spacetime formulation [37]. He uses a cubic B-spline to represent each DOF. The optimization problem is then to solve for the values of the control points that solve the dynamics problem and minimize control energy. Other extensions include the introduction of spacetime windows, which allow for sub-problems to be defined which can be solved more quickly. This also allows spacetime to be used in more interactive situations as shorter motion sequences are solved for. Cohen also allows the user to interact with the optimization process in order to guide it to a desirable solution. Keyframes can be used to help guide the process to certain local minima and to avoid others. Liu et al. extend this work by replacing the cubic B-spline basis with a hierarchical wavelet basis [99]. This allows detail to be added locally, only where it is needed. Reducing control points in smooth areas of the curves and allowing local refinement leads to a better conditioned and more efficient optimization procedure. As discussed above, spacetime constraints have also been used more recently in order to edit captured data.

Optimization approaches are problematic because it is difficult to generate objective functions that represent desired expressive ends. The physics based constraints related to the motion of complex articulated figures are highly non-linear, so the technique may not converge for higher DOF models. This limits its application for realistic human animation. Optimization approaches might yield an initial starting point for a motion sequence that could then be further refined from 'vanilla' motion into expressive motion, but this would require the output of the optimization solution to be in a form that supports easy editing. Motion that does not contain well chosen key poses is difficult to animate [80, 124]. Optimization output does not contain these keys.

Recent work has looked at simplifying the optimization problem so that the technique can be applied to realistic human character models. Liu and Zoran Popović [98] do not generate muscle torques, but simply enforce a small number of linear and angular momentum constraints. They focus on highly dynamic motions such as jumping, running and gymnastics that are largely defined by their dynamic attributes. A motion capture database is used to find initial frames for jumps and other motions and optimization is used to generate the rest of the frames. Safonova et al. [144] also employ the strategy of simplifying the optimization task. They use PCA to build a lower dimensional space for a particular behaviour and then perform optimization on this space. Fang and Pollard [47] develop a set of constraints that can be differentiated in linear time rather than quadratic time, making the optimization process more efficient.

2.5.2 Controller Approahes

Another major approach to the use of dynamics has been to create controllers that regulate the application of force or torque over time. Two strategies have been used to create controllers: they have been automatically generated for simple creatures or hand-crafted for more complex characters.

Automatically Generated Controllers

Various approaches have been taken to generate motion controllers automatically. Van de Panne et al. describe a system that, for a desired goal state, discretizes state space into a series of hypercubes [166]. Torque values are specified at the corner of each hypercube which should be applied to move the system to the desired state. These state space controllers can be defined automatically using an optimization method and spiralling outwards from the goal state.

Van de Panne and Fiume describe a system where creatures are modelled as networks of sensor and actuator nodes [165]. Controllers are synthesized in a two stage process that first randomly searches for potential locomotion controllers and then uses optimization to fine tune the best ones found. A controller's fidelity is measured by testing a creature's performance using the controller. A similar generate and test strategy is proposed by Ngo and Marks [120]. Their system is based upon stimulus-response pairs which cause their creatures to perform a certain action based upon a sensed stimulus. They use a genetic algorithm implemented on a massively parallel computer to search the parameter space for potential controllers. Sims uses a similar approach based on a genetic algorithm search, but includes the creatures morphology in the genotype [147]. Here, both the control strategy and the creature itself evolve during the search process. Grzeszczuk and Terzopoulos use machine learning to develop locomotion controllers for highly flexible, many degree of freedom animals like snakes and fish [65]. In later work, Grzeszczuk et al, use a trained neural network to actually replace the physical simulator [66]. This can provide a speed up of one to two orders of magnitude. The approach also leads to a fast method for learning controllers.

The controllers just mentioned all rely on sensory input. Van de Panne et al. have investigated controllers that require no sensory input, but act like virtual wind-up toys [163]. These controllers are built using cyclic pose control graphs. Each node in the graph contains a desired pose and a duration this pose should be held for. PD controllers are used to transition to the pose, it is held for the set duration and then the next pose is transitioned to. The search problem is to find the best set of poses to achieve a given locomotion goal, such as walking a maximum distance. A stochastic generate and test strategy is employed, followed by an optimization stage for the best controllers found. A number of controllers can be generated and these can then be interpolated between to generate a range of motions [164]. Pose control graphs can also be unwound and an optimization can be performed on a series of nodes to insert a different motion into a sequence [164]. For instance, a hop could be inserted into the middle of a walking sequence. Van de Panne and Lamouret propose aiding the controller search process by adding a guiding force that will help a creature stay balanced while it is learning to locomote [167]. This is analogous to a parent helping to support a baby as it first learns to walk. As the controller improves, the amount of guiding force can be reduced and eventually removed.

Hand-tuned Controllers

In an early paper, McKenna and Zeltzer present a controller for a six-legged insect [106]. Their system employs a gait controller that generates the correct stepping pattern by adjusting oscillators connected to each leg. The actual movement is obtained by invoking a motor program for either the stance or step phase of the motion. Forces are applied through exponential springs that have their rest position varied over time. The compliance of the springs allows the creature to navigate rough terrains.

Raibert and Hodgins adapt controllers built for robots to the animation domain [136]. They present hopping motions for a biped, quadruped and two dimensional kangaroo. All their creatures have large springs in their legs which store energy. Additional energy is provided by adjusting the rest length of these springs. These creatures can balance, adjust their speed and display different gaits.

Control systems often make use of either Proportional Derivative (PD) controllers or Proportional Integral Derivative (PID) controllers [70], the former being more common in computer animation work. Both controllers work on a single control variable, such as a joint angle, and attempt to drive it towards a user defined setpoint. This introduces the idea of an error term e that measures the distance the between the actual and desired value of the control variable. Formally,

$$e = \theta_d - \theta \tag{2.1}$$

where θ_d is the desired value of a DOF and θ is the actual value. The PID controller can then be formulated using this error term as:

$$\tau = k_p e + k_i \int e dt + k_d \dot{\theta} \tag{2.2}$$

where τ is the torque generated by the controller. The first term is the proportional term, representing an ideal spring, and k_p is the proportional gain. The second term is the integral term which eliminates steady state error and k_i represents the integral gain. The final term generates damping and k_d is the damping gain. PD controllers have the same formulation, but do not contain the middle integral term.

Hodgins et al. use PD controllers to adjust joint forces in human figures in order to animate running, vaulting and cycling [72]. Their controllers are hand crafted. Finite state machines are used to achieve transitions between the various activity states. Faloutsos et al. [46] use a similar approach for creating the movements of a virtual stuntman and Wooten [72, 173] for gymnastics. Faloutsos et al. [45] use support vector machines to categorize the domain over which their hand tuned controllers can operate. Hodgins and Pollard present a method for adapting a controller from one character to another character with different body proportions [71]. The control values are first scaled based on the body differences and then an optimization process is used to fine tune the controllers. Laszlo et al. use a similar PD control technique for walking, but they address the stability problem by including feedback in the control scheme [91]. Perturbations to the control parameters are applied based on feedback to keep certain key system parameters on their desired limit cycle.

Mataric and collaborators have presented a variety of control strategies [103, 102]. For gestures, they use PD control to achieve specified joint angles. End effector positioning is achieved using impedance control where the mechanical impedance of the end effector is modulated by varying the torques at each of the joints. The mechanical impedance of a spring is its stiffness and in this approach, the entire arm is made to perform like a virtual spring and damper. They also present a joint space force field, which is a weighted combination of step and pulse functions multiplied with individual force functions that are similar to PD control for small gain, but drop off for larger errors.

Kokkevis et al. take a similar approach to animation as ours, tracking desired kinematic trajectories with a control system and using a dynamic simulator to generate the final motion [83]. They divide DOFs into primary and secondary groups and accurately track kinematic trajectories for the primary DOFs, allowing the secondary DOFs to move passively. Our system takes a continuum approach, providing control information for all DOFs, but allowing users to specify how closely the trajectories are to be followed. This can continuously range from precisely tracking the trajectory to allowing fully passive motion.

The work of Lamouret and Cani is similar to our work in that they feature rough kinematic trajectories that are loosely followed by a control system [86]. The goals of their work, however, are synchronization of movements, collision handling and object interaction, rather than expressive control of an animation.

One contribution of our work is to show that expressive motion can be achieved, and indeed enhanced, by the generation of controller based movements for gestures and other low-dynamic motions.

2.5.3 Hybrid Kinematic and Dynamic Approaches

A number of projects, especially early ones, augment kinematic systems with simplified dynamics. The locomotion systems of Bruderlin and Calvert [27] and the PODA system of Girard and Maciejewski [53] are examples of this. Isaacs and Cohen present a system that allows a keyframe animation system to be embedded within the dynamic analysis [77]. Some parts of the body can be specified kinematically, while other parts react to the dynamics of the system. Behaviour functions are used to apply forces to achieve certain goals and inverse dynamics is used to solve for the force required for a certain motion, as needed. More recently, Ko and Badler use inverse dynamics to enhance their previously kinematic walking model [81]. The model is based on a generalization of captured biomechanical data and which they augment to allow their figure to carry varying weight loads. Inverse dynamics is used to calculate the impact of these loads on the figure's balance and required joint torques. Adjustments can then be made to keep the motion balanced and to keep joint torques within the figures strength limits.

2.5.4 Combinations of Dynamics and Motion Capture

Some work has looked at combining motion capture and dynamics. As mentioned, Popovic and Witkin [131] include dynamics in their optimization method to create physically valid modifications to motion capture data. Conversely, researchers are beginning to use motion capture data to drive the creation of control strategies. Zordan and Hodgins use a tracking controller to track motion capture data for the upper half of a character [181]. They scale the gains based on the inertia of the body part attached by the joint and use an optimization method to determine the best gains to track the input data. Once set, the gains are locked for the animation. This is the simplest method for creating a controller from motion capture (mocap) data, but it will not work for lower body motion where balance is an issue. They use a separate controller to regulate the lower body of their character. Playter uses mocap data and feedback control strategies to create a running controller [130]. The feedback control keeps the running motion within its dynamic range and allows for user adjustments whereas the mocap data provides the basic shape of the running sequence.

2.5.5 Summary

Employing the physical equations of motion allows computational models to include an important aspect of real motion: physical correctness. Highly dynamic motions, such as tumbling or falling, are largely determined by the physical constraints on the motion. In such cases, much of the work required to create convincing motion is taken care of by ensuring that the motion is physically correct. For less dynamic motions, such as gestures, the physics of the situation places only loose constraints on the motion. There are many ways that the motions can be completed and other information is required to produce natural motion. Even in these cases, though, physics can help generate important nuances of character motion, such as force transference between limbs and momentum effects like overshoot.

Using physics greatly increases the computational cost of generating motion and also makes the task more difficult as stability issues need to be addressed. In physical simulation, a momentarily off-balance character will fall over, even if this is not intended. Such problems do not occur with kinematic approaches to motion generation. Despite the increased difficulty, modeling physics allows the realism of generated motion to be greatly increased. Dynamic control of motion thus remains an important, if challenging, goal.

2.6 System Approaches

A number of system level approaches have been proposed for generating character motion.

2.6.1 Artificial Life

The field of behavioural animation was ushered in largely by Reynold's pioneering work on flock modelling [139]. This model operated by giving each "boid" its own behavioural rules and local sensing. Group behaviour emerged out of individual behaviour. Terzopoulos, Tu and Grzeszczuk extended this work to create a virtual aquarium in which fish move by contracting their muscles and participate in more advanced behaviours such as mating and predator-prey chases [156]. In Europe, Nadia Magnenat-Thalmann and Daniel Thalmann have applied this approach to animating virtual humans [157, 79, 21]. These systems fall under the title Artificial Life or *ALife* for short. As our focus is not on autonomous creatures, these works will only be dealt with briefly.

One of the earliest animation systems to support goal-directed movement was developed by Morawetz and Calvert [110]. Their system, GESTURE, is driven by an animator written, high-level specification script from which a low-level animation script is generated. The final animations are achieved by consulting gesture specification functions that use specific knowledge about the action to create an appropriate animation. Movements are encoded in a graph that stores joint angles at nodes and timing information and labels on the arcs connecting them. Movements are interruptible.

Becheiraz and Thalmann describe a system in which characters interact non-verbally in a virtual garden by changing their posture and the distance they stand from each other [21]. The gesture set they use was developed by psychologists who tested users' impressions of the postures on a number of bipolar scales such as rejecting/receiving, other-concerned/self-concerned and cold/warm. Becheriaz and Thalmann use four of these scales and give each character a model of their attitude, character, emotional state, relationships with other actors and desire to communicate. Characters wander in the environment stopping to talk to other characters who interest them. They adopt the posture that is closest to the state of their internal model. As they perceive the other character's posture, their state values are updated and they adopt new postures.

2.6.2 Character Systems

Character systems are related to ALife systems, but generally they are designed to create characters that will interact with human users. These systems operate at a higher level than our work, dealing with issues of action selection, but generally not providing original methods for modifying the style of movement. For this reason, they will be dealt with only briefly.

Perlin and Goldberg introduce a script based system for designing interactive characters called Improv [127, 126, 56]. Their system architecture is based on a layering scheme that ranges from high level, long-term goal behaviour at the top to low level actions like a wrist flick at the bottom. The top set of layers correspond to the behavioural engine and the bottom layers to the animation engine. The animation system is based on their earlier real-time puppet system described in Section 2.3 which makes extensive use of structured noise, sine and cosine based interpolation functions. They also support action compositing and action transitions.

Animations and behaviours are organized into groups at each level of abstraction. Only actions from different groups can be defined, which allows the system to avoid conflicts. The characters are brought to life through the behavioural scripts which are written for them. Characters are given number based personalities. Scripts decide on actions in a probabilistic manner.

The Synthetic Character group at MIT Media Lab was founded by Bruce Blumberg. They study behavioural models for interactive virtual characters, making use of research in ethology. Blumberg and Galyean emphasize the importance of going beyond completely autonomous characters to characters that accept direction [22]. They allow this direction at three levels: the motivational level (e.g. you are hungry), the task level (e.g. go to that tree) and the direct level (e.g. wag your tail). Animation is achieved by using animator specified motor skills. Behaviours are contained in a loosely hierarchical behaviour network and compete for control of the system with a winner take all strategy. Losing behaviours can still suggest secondary commands, but the winning behaviour has ultimate control.

Working from an artificial intelligence rather than graphics perspective, Joseph Bates' OZ group formerly at CMU[1], has been studying believable agents and interactive story [104, 20]. Their agent research focuses on believability, seeking to create agents that have broad, but shallow competence. Broad agents are more capable of maintaining a user's willing suspension of disbelief. Their work differs from traditional AI in that they are trying to create specific characters that are subjectively believable, rather than general solutions that are objectively measurable. Indeed, they argue against creating characters by setting parameters in some general personality model, suggesting that one would never get Bugs Bunny or Hamlet this way [104]. How many parameters would one need?

[1]Now the company Zoesis.

Their architecture includes a programming language for agent actions, plans and goals.

Barbara Hayes-Roth, who leads the “Virtual Theatre” group at Stanford, takes what Petta and Trappl [128] refer to as a “drama-theoretic” approach to the autonomous agent problem. Her work is strongly influenced by Keith Johnstone’s [78] writings on improvisational theatre [69]. Her research aims to create characters that can work together to create improvisational stories based on the constructs of plot (temporally constrained sequence of events), role (class of individuals whose prototypical behaviours, relationships and interactions are known) and character (personality defined by a coherent configuration of personality traits). Each agent forms an abstract control plan to describe its intended behaviour [68] which can be modified during execution in response to the actions of other agents.

Norman Badler’s group at the University of Pennsylvania has done extensive work on modelling real-time virtual humans (see [11] for an overview). They have developed a two layer architecture based on Parallel Transition Networks (PaT-Nets) at the high level and Sense-Control-Act (SCA) loops at the low level [13]. Networks of S, C and A nodes define behaviours. They control low level phenomena such as controlling foot placement so the agent does not trip. The high level PaT-Nets are parallel finite state machines that can be used to describe various activities the character can undertake. They use them to represent patterns of activity, such as those used in a hide and seek game. Their architecture also includes Parameterized Action Representations (PARs), which can be used to link spoken instructions to character actions.

One system they have developed for controlling locomotion includes a third layer, the agent model, at the top of the architecture [13]. The agent model contains a list of personality traits that map down to the locomotion parameters in the system. Given a high level goal, the system will generate the correct locomotion to meet it based on the personality and state of the character.

2.6.3 Gesture and Conversation Oriented Systems for Character Interaction

A few researchers have begun looking at automatically supporting the animation of both the verbal and non-verbal aspects of human conversation. Justine Cassel heads the Gesture and Narrative Language group at MIT Media Lab. Cassel et al. introduced a complete, but preliminary animation system for modelling conversations between two virtual characters [33]. The system models intonation, facial expressions, gaze direction and hand expressions. Cassell and Vilhjalmsson introduce a system for automatically adding appropriate facial expressions and head gestures to avatars based on text input [32]. In a related system, Zhao and Badler allow head and hand gestures and arm positions to be scripted to coincide with spoken text [180]. The work is interesting because it uses in an animation context the ideas of Delsarte, one of the movement researchers discussed in the following chapter, but it does so in a limited way.

Recently, Stone et al. [152] have presented a system that combines speech and gesture output based on captured movement and utterances. Given a script, it will match the correct utterances and motions to it from the captured data to produce seamless output.

2.6.4 Other Systems

Life Forms is an authoring tool that allows choreographers to prototype dance [31]. It tries to support the various phases of the artistic process. In the concept planning phase, choreographers can rough out arrangements of dancers on a stage. This process can be used to create a series of storyboards, giving a high level overview of the performance. The animator can then proceed to define stances and movement sequences. Dancer manipulations are done either using forward or inverse kinematics. Animations can be produced by interpolating through a series of stances. The system also supports procedural animation of walking, running and grasping and the incorporation of motion capture data. To support multiple dancers, the timing of each dancer can be modified and the path they take through space can also be edited.

The literature describes many previous animation systems, often with a procedural component. Early systems, such as *Scripts and Actors* [138], provided a language for defining modeling and animation. Independent *actors* control different visible elements in a scene and are invoked every time step. They can control low-level properties and synchronize with other actors. *MENV* [137], the animation system at Pixar, controls animation by changing the value of *avars*, articulated variables which control various aspects of the model. They can control a single aspect or be linked to several low-level variables. Our work is distinguished by its focus on direct control of aesthetically important qualities of animated characters, an iterative tool designed to support exploration and refinement, and the integration of dynamic simulation.

DANCE, the Dynamic Animation aNd Control Environment, is a freely available software framework for developing physics based animation systems [119]. The prototype system described in this thesis was built on top of this framework.

2.7 Analysis Frameworks

In [12], Badler analyzes the requirements for a movement language. He argues that movement primitives must have the following properties: descriptive significance, modifiability through generally accessible methods, independence of specific individuals and independence of specific motion characteristics. He further argues that Laban's Effort-Shape analysis meets these criteria and forms a suitable basis for modeling qualities of movement. This work also argues that kinetics and dynamics have an important role to play in building computational models of the Effort-Shape analysis.

In work that is complementary to this thesis, but operating in the opposite direction, Harrison presents detailed studies of users' perceptions of animated human locomotion [67]. The work seeks to understand the relationship between three representations: the mechanical or computational representation of the movement, the psychological representation of the movement that individual observers construct, and the linguistic representation observers use to discuss and categorize the movement. Part of their work examines how users will assign linguistic terms to various walking motions, employing a series of weighted bimodal scales that each contain two antonyms. This provides a methodology for ascribing linguistic terms to specific movement changes. Our work operates in the opposite direction, using terms taken from the arts literature to synthesize described changes in movement.

2.8 Conclusion

The previous work discussed above has inspired the shape of this thesis in numerous ways. The work of Faloutsos et al. [46] was significant for demonstrating on complex human skeletons that physics-based simulation could create important nuances of motion. It inspired the effort in this thesis to explore how physical simulation could be leveraged to improve the expressive quality of motion. The results of that exploration are discussed in Chapter 5.

Although their work was coming from an artificial intelligence perspective, the philosophical underpinnings of the OZ project were a significant influence. They emphasized the importance of creating *specific characters* and based their work in the character arts. A similar philosophy underlies this work. It reflects an appreciation of motion that was developed through first working in theatre and then trying to bring that understanding to bear on the problems of computer animation.

While interesting, it was felt that early work on the expressive style of animation, such as *emotion transforms*, did not sufficiently address an animator's need to create *specific* nuances for *specific* characters in *specific* settings. Animators demand precise and complete control over the motion they produce. For this reason, the strategy adopted in this thesis was to try to give animators handles that provide direct control over the aspects of motion they would want to change. These handles needed to be fine grained enough that they could adjust the various aspects of motion that they cared about without worrying about one handle interfering with another. At the same time, higher level handles make the authoring process more expedient, another goal of this work. This lead to the construction of hierarchical tools, that provide both broad control and refinement as necessary.

The work of Calvert [31] emphasized the importance of workflow, which became a major theme in this work and had not been given significant attention in other previous animation tool research. Animators, and actors, work progressively and iteratively. They need to be able to experiment to find the right approach to a scene. They also need to be able to refine motion in order to arrive at the animation sequence they require. The need to support both exploration and refinement became a significant theme within this thesis.

Chapter 3

Learning from the Arts: Surveying the Field

Science is one thing, wisdom is another. Science is an edged tool, with which men play like children, and cut off their own fingers.

Arthur Eddington, British Astrophysicist (1882-1944)

The notion of 'realistic' animation goes far beyond simply making the computer graphics look good. Computer animated characters should be believable. They should tell and be part of engaging stories; they should present rich, nuanced and unique personalities; their actions should communicate clearly with their audience. In short, they should have the ability to possess all the qualities seen in great characters from the theatre over the past many centuries.

When studying expressive movement, the arts literature from the fields of theatre, movement theory and animation serves as both a rich and a challenging resource. Expressive movement is the lifeblood of these fields. They have studied it longer and with greater intensity than any other discipline, trying to deduce its secrets. These fields, above all others, deal with how to communicate a particular character through movement. There is much to be learned from their years of study and the lessons they have distilled.

At the same time, the arts literature is a problematic source, particularly given the problem set out in the current research. We are trying to abstract general properties of expressive movement and represent them in inanimate computational form. Those working in the arts are dealing with a different problem: that of working with living performers, or with artists who create two-dimensional drawings. In each case, the knowledge of what makes movement expressive is embedded in the knowledge of the people creating the work. There is no need for external representation. An artist can interpret the lessons contained in the arts literature, combine them with her life experiences and build a new understanding of movement that she can use in her work. Often movement experts talk in analogies, trying to induce a certain understanding in an actor or a dancer. Still other times, the teachings are entirely in a physical form, providing exercise for an actor or dancer to complete in order to increase their understanding of movement. Even when attempts have been made to codify movement, there is still a far greater level of subjectivity and interpretation than would be desirable in a computational representation or in any more indirect human discussion where direct observation of movement is unavailable. The experts may not agree and may not even speak the same language.

When building computational models of expressive movement, we must be more rigorous and explicit. We need to be able to write down precise definitions of different movement properties and specify how these act on the body. It is neither the goal of the arts literature, nor necessarily of interest to those practitioners to express their ideas in this rigorous manner. In the end, we are left with the job of taking their lessons and mapping them to more concrete representations. There is inevitable interpretation that must occur in translating these lessons to computational form. That has been much of the work of this thesis.

Reviewing the theatre literature reveals an inescapable fact: much of what it means to be "human" is missing from the computational models developed within the computer graphics community. This is true when considering the physical properties of an actor, as well as their mental processes, experiences, and emotional memories. Only a real actor has the muscle memory of actions performed during an emotional crisis, or becomes nervous before walking out in front of an audience. Real actors may also be known to their audience outside of the context of their performance. Hence the problems encountered by a real actor differ from those encountered in animation. Similarly, the resources available to a real actor are different. Yet the goals of a real actor and an animator are precisely the same: How do you move an audience? How do you convey truth through movement? What makes a character alive and real for a spectator? How do you create a particular character and how do you use him to tell an engaging story? How do you make performance into art? It is the light the performance literature throws on these questions that will be our guide.

The performance literature defines what issues the computer animation research community must pay attention to if it wants to create effective tools for producing engaging, rich, nuanced expressive characters. Simply put, the performance literature defines the problem we as computer science researchers are trying to solve and provides part of the solution. It can thus provide a metric against which we can measure our systems to see if they are meeting the real requirements for generating expressive motion. The challenge remains to extract this information from the arts literature into a form that will be useful for computer animation.

The result of our investigation is contained in this and the next chapter. They are long chapters and not everyone will have the time or inclination to read them in their entirety. For this reason, a summary is presented next that captures some of the main ideas from the research. This summary can safely be skipped without loss of content, or the summary can be read first and then other sections read as needed.

3.1 Summary of Key Lessons

This section summarizes some of the key lessons discussed at length elsewhere in this chapter and the next. It can be skipped without loss of content, but will provide the busy reader with a flavour for the richer discussion that follows.

- There is a psycho-physical connection between the internal mental experiences of characters and their outward movements. Different acting methods explore this connection in different ways to help establish movement that conveys “emotional truth”. Actors can use their emotional memories to help determine correct actions.
- Actors develop much of their knowledge of movement, including what movements are appropriate for a given scene and character, through the use of their bodies.
- Movement is often a reaction to various stimuli: those that are internal, from another character and from the environment. Actors must be good at sensing and reacting to stimuli.
- Daily movement and performance movement differ. Performance movement is designed to communicate and is based on the twin principles of simplification and exaggeration.
- Movement properties have been grouped into three categories: *shape*, *transitions* and *timing*.

3.1.1 Shape

Shape refers to the poses the body assumes.

- The entire body should be completely engaged in a movement.
- *Posture* can be seen as a reflection of the degree of tension in a character’s body, ranging from a rigidly erect army officer at attention to a hunched old man.

- *Balance* is a fundamental movement property. Performers will often employ "precarious balance", working near the balance limit to heighten the excitement of their motions. Balance adjustments can act to intensify motions.

- *Extent* refers to how far an action or gesture takes place from a character's body. Movements can be grouped based on three zones: near, mid and far extent, each having different connotations.

- Various meanings have been associated with the use of different areas around the body and varied directions of movement. For instance, sad emotions generate movements that are downward and inward. See Section 4.3.2 for the detailed discussion.

- The varied use of different parts of the body is also significant and discussed at some length. For example, the spine can hold different curves, such as the long S curve known as the "beauty-line" that snakes through a character's body. Significant spine movements include pinching from the ends, shifting the central area of the torso out of its normal position and rotating around the central axis.

- How shape varies over time is also significant. For example, recoil can be used at the commencement of a movement, to first pull backwards from the end goal before moving forward to it. This backwards movement creates a negative space into which the movement can travel and acts to emphasize the movement. A different form of recoil can be used in reaction to a distasteful object, where a character pulls back or recoils from the object, but does not then move in forward direction.

3.1.2 Transitions

Transitions deal with how a character moves from one pose to the next, as well as other transient aspects of movement.

- Movement can flow continuously, be intermittently interrupted, or stopped to yield a pose.
- Motions that are interrupted before they reach their final goal are called *incomplete* and are common in daily activity.
- The *motion envelope* relates to the speed profile of a movement over its transition. Some movements will start slowly and end quickly. Other movements will do the opposite. The traditional ease-in, ease-out curve transitions most quickly during the middle of the movement and slows at the beginning and end. Laban's Effort notation captures the concept of motion envelopes well, along with other transition aspects such as the path a movement takes in space.
- The interplay of tension and relaxation in a character's body is one of the most significant transition properties. The expressive impact of tension will be discussed at some length in Chapter 6.
- Animated characters must have a sense of *weight* to appear realistic.

3.1.3 Timing

- Tempo and rhythm are the two most important components of timing. *Rhythm* refers to the cadence of a set of movements. For instance, the pattern could be long, long, short, repeat. *Tempo* refers to the speed at which motions are completed; the speed of the beat. Tempo is independent of rhythm.
- Each character will have an individual tempo-rhythm.
- *Successions* deal with how a movement passes through the body, and help to generate a sense of flow. In a normal or forward succession, a movement starts at the base of a character's torso and spreads out to the extremities. In a reverse

succession, the movement starts at the extremities and moves in towards the centre of the character.

The remaining material will be only briefly summarized. A discussion has been provided of different actions that characters can perform and their meaning, with a special emphasis on gestures and walking. Action selection – deciding what movements a character should make – is one of the main tasks of an actor or animator. This is discussed at some length, with an emphasis on traditional acting approaches to the problem. The actions chosen must reflect the inner life of the character. A summary is provided of the general considerations that come into play when constructing an extended movement piece. Interaction amongst characters often plays a central role here. It is important that characters actively react to what the other characters are doing in order to maintain believability. Also important is the physical spacing of characters and their orientation relative to each other. The chapter concludes with a discussion of how various movement properties can be put together to create specific characters. Factors that must be considered in creating a character's movements include his physical qualities, his role relative to other characters, the setting of the action, his psychological makeup and his emotional state. Several implications posed by the arts literature for the design of effective character animation tools are highlighted. These include the importance of context in determining the correct movement, the shortcomings of approaches that provide generic models for "happy" or "sad" movement, and the need to change both what actions are performed and the style in which they are performed when changing the character that performs a movement sequence.

3.2 Key Sources of Information

Given the long history of theatre, the diverse range of performance traditions in various cultures and the large number of people that practice the art of animation, it is problematic to put together a definitive list of key sources for insight into the nature of expressive movement. It is doubtless that individual critics will argue that a certain practitioner left off the list is worthy of inclusion and conversely claim someone who has been included made only a small contribution. We have attempted to survey what we consider to be the main sources from animation, acting and movement theory[1]. Many practitioners have developed their own, closed theory of movement. We do not present or side with any one theory of movement, but rather we attempt to synthesize principles that are shared by many practitioners, and in multiple art forms.

3.2.1 Traditional Animation

Traditional hand drawn, two dimensional animation emerged as an art form during the early days of Walt Disney Studios in the 1930s. This was a time of great experimentation, when the animators would take drawing classes, bring in live models to watch them move, and study film of actors, dancers and animals - all in an attempt to understand the nature of movement and how to represent it in animated form [160]. From this experimentation, a number of principles emerged that became the basis for the Disney style of animation. The lead animators in these early days, who discovered and codified these principles, were referred to as Disney's "Nine Old Men". Two of the nine, Frank Thomas and Ollie Johnston, provide a detailed account of the principles and their discovery in the beautiful book *The Illusion of Life* [160]. John Lasseter, a Disney trained animator that went on to have a lead role in the emergence of Pixar, is widely credited with introducing the Disney principles to computer animation through his 1987 SIGGRAPH paper [88].

[1]Prof. Stephen Johnson of the Graduate Centre for Studies in Drama provided important guidance on the theatre literature.

3.2.2 Theatre

An actor will normally create a character using one of two approaches, possibly blending the two. Either he will first attempt to get the movement correct, focusing on external representation in the belief that the correct internal emotional state will follow, or he will try to get the emotional state correct first, in the belief that the movement will follow [1]. The methodology of actor training will thus tend to follow one of these approaches. The famed Russian director Stanislavski [92], and the *Method Acting* approach which is based on his teachings, uses the latter approach, concentrating on the inner life of the character first and allowing the external movement to flow from this. Meyerhold, Eisenstein and, to a degree, Grotowski, among others, take the opposite approach, focusing first on an actor's movement and allowing the correct emotional state to follow [92]. Each of these theorists is introduced below.

Meyerhold and Eisenstein (early 20th century)

Vsevold Meyerhold was the foremost Soviet avant-garde theatre director and a former student of Stanislavski. Following the Russian revolution, he developed an exercise based form of actor training known as *biomechanics*, which he abandoned in the 1930's due to political pressure [92]. Sergei Eisenstein was his student during this time and developed a related method known as *Expressive Movement* [92]. They argued that what an actor felt inside was not important if it was not communicated to the audience and counter to Stanislavski's inward focus, sought to increase an actor's physical vocabulary [92]. They proposed that a highly trained actor could achieve emotional involvement through his movements [109]. Sonia Moore, a follower of Stanislavski, suggests that Meyerhold was ultimately disappointed in this belief (a contentious point), but his exercises remain a useful way to make an actor's body more responsive to his inner processes [109].

Grotowski suggests that the actor's creative process depends on discipline and spontaneity. He argues that Meyerhold focused on discipline, external formation, whereas

Stanislavski focused on the spontaneity of everyday life; two complementary aspects of the creative process. [58]

Grotowski (mid to late 20th century)

The famed Polish director Jerzy Grotowski formed the Theatre Laboratory in Poland with the goal of understanding the nature of acting. He drew widely on Soviet and Western theatre traditions as well as training methods used in the Orient [64]. They considered the actor's art and the actor-spectator relationship to be the core of theatre art [64]. Theatre Lab productions were often staged with minimal use of lighting, music, costume and set [64].

Grotowski developed a training regime which focused on physical and vocal exercises. The idea behind Grotowski's training is that actors approach movement ideas through their body [154]. By repeatedly participating in a set of exercises, normally in silence, the actor gains a greater understanding of the nature of movement by experiencing it through his or her body. The actor's work focused on "the composition of the role, on the construction of form, on the expression of signs - i.e. on artifice." [64, p.17] While it is often claimed that Stanislavski sought naturalism on stage (e.g. [109]), Grotowski sought truth in a different way:

> We find that artificial composition not only does not limit the spiritual but actually leads to it.. . . The forms of common "natural" behaviour obscure the truth; we compose a role as a system of signs which demonstrate what is behind the mask of common vision... [64, p.17]

He argued that man does not behave "naturally" in heightened emotional moments, so one must go beyond naturalism to reveal the deeper truth. [64]

The main principle of Grotowski's work is *via negativa*; a process of eliminating the blocks that are inhibiting the actor [15]. The goal was to have no time lapse between an inner impulse and an outer reaction [64]. An actor's body should be free and completely responsive to stimuli.

Another of the main tenets of Grotowski's theory of acting is that the whole body should be engaged in any movement. He writes:

> Our whole body must adapt to every movement, however small. Everybody must proceed in his own way. No stereotype exercises can be imposed. If we pick up a piece of ice from the ground, our whole body must react to this movement and to the cold. Not only the fingertips, not only the whole hand, but the whole body must reveal the coldness of this little piece of ice. [59, p.193]

Given that these types of actor training are situated in body experience, it is difficult to map the lessons contained to a concrete, computational form. We are talking here not about an individual merely feeling the experience, but rather gaining a subconscious understanding of movement through exercises. From this understanding, they can convey movement to others in a way that on the one hand everyone can see but on the other does not appear overly exaggerated or caricatured. Having personally experienced the Grotowski training exercises, it is clear that there are valuable lessons about movement contained in the work, particularly in terms of how movements flow together, the importance of a clear direction and focus in a movement, and the important role space and the environment play in defining a movement. These lessons will be incorporated in the analysis below.

Stanislavksi (early 20th century)

Probably no one has had a greater influence on the art of acting in the last hundred years than Constantin Stanislavski, director of the Moscow Art Theatre and inventor of the "Stanislavski System", popularized as "Method Acting". Stanislavski wrote a trilogy on acting technique: *An Actor Prepares* [148], *Building a Character* [149] and *Creating a Role* [150]. The first book focuses on the inner life of the character, the second deals with external representation (physical movement) and the third discusses how an actor can create a role. He was himself dismissive of the notion of a "Stanislavski system", arguing there are only the laws of nature [149, 109] which he had being trying to discover and suggesting that young actors should make up their own system that would work for them [109].

Fundamental to Stanislavski's teaching is the idea that actors must *live* on stage. They must not merely try to mechanically reproduce a set of actions, but there must instead be emotional truth to what they do [109]. He argued that an actor must learn to see, hear and talk anew on stage as his normal senses were prone to paralysis. Most importantly, an actor must actually *do* these three things while on stage, not merely pretend to do so. While he believed in the importance of technique and thought it was important for an actor to dedicate himself to physical training, he emphasized first focussing on methods to help an actor achieve psychological realism.

Moore suggests that Stanislavski's most fundamental teaching was the *Method of Physical Actions* [109]. Stanislavski believed that internal experiences and their physical expressions are united; there is a psycho-physical connection. He observed that an actor cannot easily control their heartbeat or the dilation of their blood vessels and argued that it is similarly difficult to feel joy, fear, grief etc. without a reason to do so. He argued that these things were part of the subconscious and sought a conscious process to get to the subconscious. This is what the *Method of Physical Actions* achieves [109].

The method has an actor "fulfill a simple, concrete, purposeful physical action which stirs the psychological side of the psycho-physical act" [109, p.19]. He thought that every nuance of emotion was connected with a particular physical action. The challenge is to find the right physical action to bring out the desired emotion. Unlike other theorists, Stanislavski did not promote a set lexicon of actions, where a given gesture would always generate a given emotion. Indeed, he thought the use of conventional gestures was a great fault of the acting profession [148]. Rather he thought the right gesture depends on the situation, individual traits, the tempo-rhythm the actor is working in etc. A great deal of preparatory work is required to find the right action, including analysis and exercises, such as mentally putting one's self in a similar scenario or recalling a situation where he felt an appropriate emotion. This will help him find the right gesture. The psycho-physical connection must not be broken. If the actor only performs the movement, the performance is mechanical and dead [109].

Stanislavski warned against any action designed to arouse a feeling for its own sake:

> *When you are choosing some bit of action leave feeling and spiritual content alone.* Never seek to be jealous, or to make love, or to suffer, for its own sake. *All such feelings are the result of something that has gone before. Of the thing that goes before you should think hard as you can. As for the result, it will produce itself.* [148, p.38, original emphasis]

Stanislavski argued instead that an actor's acting of passions "must grow out of [his] living in them." [148, p.38]

3.2.3 Training Texts

Our study is rounded out through the inclusion of texts on mime [40, 93] and physical aspects of acting [1]. These are training texts, aimed at improving the movement of practising artists. They contain lengthy examinations of the meanings associated with different movements and tend to provide more movement examples than the more theoretical texts.

3.2.4 Movement Theorists

Our next source of information is Francois Delsarte, Rudolph Laban, Eugenio Barba and Nicola Savarese - movement theorists that have studied expressive movement and attempted to extract the laws and principles that govern it.

Francois Delsarte lived in France during the nineteenth century and spent much of his life studying the nature of expressive movement. He conducted a wide range of observational studies, collecting data on a variety of people in many different circumstances and conditions. From this large collection of data, he postulated a set of laws that reflected his observations [146]. He used the trinity as an organizing principle, dividing a large number of movement properties into three categories. He gave these the generic terms: Normal, Excentric and Concentric (a centre, a going out from the centre and a going towards the centre) [146]. Believing his work still incomplete, he never published his ideas, but lectured on them from 1839 until his death in 1871. We know of Delsarte's work through his disciples.

Taylor argues that although his work seems to have a strong theological bent and is culturally constructed, it still shows remarkable insights into the nature of movement [155]. According to Ted Shawn, the ideas of Delsarte are the basis for American Modern Dance [146].

Rudolph Laban [85] is another very influential movement theorist. Born in Poland, he traveled much of the world studying movement before settling in Berlin, and eventually fleeing to London before the Second World War [154]. His work involved choreographing mass movement pieces, developing a notation for dance and founding a system of movement analysis. Late in his life, he studied movement in the work place, particularly movement associated with mass production [154].

Eugenio Barba and Nicola Savarese founded the field of Theatre Anthrolopology, motivated by the question: "Where can performers find out how to construct the material bases of their art?" [17, p.8]. They claim the field is not a science as they are not trying to uncover fundamental laws but instead study rules of behaviour and aim to discover 'good advice' on the nature of movement. Their focus is on human behaviour in a *performance situation.* They published the results of the first ten years of study in 1991.

3.3 Human Actors, Computer Models and Animators as Actors

Some discussion has already been given to the different capacity and challenges of human actors and computer animated characters. In particular, the psycho-physical connection that is present in real humans was highlighted. This section will explore the theme in more detail. It serves both to indicate the complexity of the actor's craft and to suggest possible considerations an animator or artificial intelligence system must take into account when generating a character. As well, this study may lead to new types of tools that better support the needs of animators based on an understanding of the artistic process of actors.

Even the most advanced computer based character systems have very few of the mental, emotional and physical resources a human can draw upon when creating a character. (See for example the impressive system developed by Zoesis Studios for an idea of the state of the art [100]. They support character authoring with an innovative programming language designed to preserve artistic intent in the final realization of the characters. They also include a model of character emotions and a motion synthesis system that employs hand generated motion clips with authored procedures for generalizing them.) Computer characters also have certain advantages: they will never have stage fright, properties such as blood vessel dilation can be directly controlled, provided the models are sufficiently complex, and they need not be constrained by physical laws and limits. Human animators have a rich combination of experiences, mental and physical resources to draw on, but their work is mediated by the tools they use. Better understanding the artistic process might lead to tools that are less restrictive in this mediation.

3.3.1 Mental and Emotional Exercises

Stanislavski argues that "Most frequently, especially among talented actors, the physical materialization of a character to be created emerges of its own accord once the right inner values have been established." [149] An actor does not necessarily need to consciously decide to behave, but it can flow naturally from his internal understanding of the character. Indeed, Grotowski emphasizes the importance of movements and reactions not coming from the brain [61], meaning that the person should not consciously think about their actions. He suggests that the body should react directly to stimulus. The character should hear and react; not hear, think about it, decide how to react and then make the motion. Avoiding conscious thought is not an option for an animator in most workflows. This suggests the unique opportunities performance animation might present.

Grotowski discusses how an actor must give himself, or honestly unveil himself, in a role [63, 61]. He uses his own inner truths to reveal the truths of the character honestly to the audience. This illustrates a fundamental difference between an actor and an animation. An actor exists as a person with rich life experiences outside of the role which he can tap on to communicate with the audience, whereas an animation is purely a creation, with no existence outside the screen[2]. The idea of an actor as a fully embodied human is fundamental to Grotowski's approach in which he suggests an actor must ask "What must I **not** do?", rather than "How can this be done?" [58]. While the actor needs to train, Grotowski's view is that the fundamental ability to express is already there in the actor and the work is to remove the blocks that are inhibiting it. He says:

> We must find out what it is that hinders him in the way of respiration, movement and - most important of all - human contact. What resistances are there? How can they be eliminated? I want to take away, steal from the actor all that disturbs him. That which is creative will remain within him. It is a liberation. If nothing remains, it means he is not creative. [58, p.209]

Conversely, animation is a constructive process. The models are quite impoverished and there is little there that an animator does not put there.[3]

There are a number of mental processes and exercises an actor can use to get at the right movements for a character. Much of Stanislavski's work focused on this. Stanislavski emphasized the importance of an actor developing their imagination. They need to fill in a character's past and future experiences so that they have more information

[2]Characters like Bugs Bunny may over time develop an existence for their audiences that extends beyond any particular role, and requires a maintenance of consistency, but they do not have daily life experiences. They do not *live* a life or create their own performances.

[3]Of course, once created, a character may live on in the minds of its audience well past the performance.

to draw on [109]. Imagination is useful in determining the subtext [109]. Stanislavski developed the *magic-if* as a tool for discovering a character's motivation: "*if* acts as a lever to lift us out of the world of actuality into the realm of imagination." [148, p.43] By asking if questions, like "what would I do if I was in situation x", the actor can understand the situation of the character and find motivation for his actions [109]. Finally, the actor must know the *given circumstances* of the character. This includes things like the plot, epoch, time and place of action. Reflecting on these will help the actor find correct actions. [109] All of these mental processes are also available to an animator. The issue then simply becomes how easily can an animator translate the insights garnered into movements of a character.

Actors also have emotional resources to draw upon. Stanislavski considered *emotional memory* to be a very important source for an actor [109]. He suggested that an actor must execute the correct physical action to relive the emotion. Grotowski also argued the importance of drawing on specific, real, intimate experiences [61]. He argues that drawing on intense physical experiences that an actor would not normally share can create intense moments on stage. It is important to note that Stanislavski argues that an actor can differentiate stage emotions from real emotions. "The actor must live true experiences, but *stage* experiences." [109, p.42]

Emotions can also drive physiological responses. They can create a visible physical response. For instance, an agitated character will exhibit a dilation or swelling of the blood vessels. Our ability to recall emotions and find through our body correct actions for them is likely one reason why Disney animators would often work out scenes in front of mirrors. They needed to be able to see what their body decided to do (or what movement they subconsciously triggered). In this manner, animators are likely to still learn through the use of their body. It would potentially be beneficial if tools can support this.

3.3.2 Physical Exercises

The importance of physical training for actors is routinely emphasized (e.g. [149, 109, 15]). Physical exercises can serve a range of purposes. They are a way for an actor to investigate movement [15]. Exercises can develop an actor's concentration, and ability to notice and react to external stimuli [15]. Some exercises are designed to develop a sense of rhythm. Grotowski insisted that the exercises be done in silence, claiming outward silence is necessary for achieving inward silence [62, 59]. His exercises are generally movements that would not appear on stage, but are part of an actor's research; both increasing their movement skill and improving their understanding of movement.

Some of the things Grotowski's exercises tried to build an awareness of include [15]:

1. One's centre of gravity and laws of balance.
2. The mechanism for contraction and relaxation of muscles.
3. The function of the spine in violent movements.
4. What complicated developments occur and how they relate to the repertory of every single joint and muscle.
5. Changes in respiration.
6. The rhythm of the head.
7. The relationship between position and movement.
8. How to combine vocal work and physical exercise.
9. The resonators located throughout the body that an actor can use to amplify and change her voice.

Finally, an actor must also breathe. There are a range of different respirations an actor can learn [63]. Animated characters need not breathe, but failing to do so detracts from their believability if human-like realism is the intent.

3.3.3 Animators as Actors

Animators behave as actors in two very concrete ways: First, both animators and actors are trying to express ideas by using the articulated skeleton that defines the human form. Second, animators will very often use their body to discover the correct movement for the character they are animating, physically acting out a scene many times before they animate it. Discussions with two professional animators revealed that whenever time allows it, they will shoot video reference of themselves performing a scene before they animate it [80, 124]. For them, the most challenging part of animation is determining the right actions and they find acting out the scene numerous times in different ways to be a very important technique to deal with this problem. Often they will shoot ten references for a scene, trying to find the best approach. Animation tools do little to recognize and support this aspect of an animator's work.

Animators are also visual artists. Unlike stage movement and generic motion capture, animation and film acting are 2D media. This allows an animator to customize motion in a scene so that it looks good while seen from a particular camera. This is important for creating visual appeal in animation [80, 124].

Chapter 4

Learning from the Arts: Lessons on Movement

The eternal mystery of the world is its comprehensibility...The fact that it is comprehensible is a miracle.

Albert Einstein

The previous chapter lays out the landscape of the performing arts literature and indicates some of its strengths and challenges. This chapter summarizes the lessons we have drawn from that literature on the nature of expressive movement. The goal of the analysis is a broad synthesis of principles, paying particular attention to ideas that were repeated across numerous sources. Principles have been extracted and classified into a number of categories: ideas that relate to the difference between daily and performance movement, fundamental aspects of movement that serve as building blocks, the types of actions characters perform, principles of effective movement pieces and finally, how movement can be varied to generate specific characters. This synthesis is described below with references indicating the sources of each idea.

The analysis is neither complete nor final. Rather, this thesis seeks to demonstrate the richness of the arts literature and the value it has to offer to computer animation:

that the language of the arts literature can be made sufficiently precise to be useful for computer animation, computational representations can be built of some of this language, and computer animation is better for it. The literature also highlights fundamental challenges that will elude computational solution for some time. Chief among these is the central role that context plays in determining how a specific character should move in a specific situation. Despite these difficulties, and the fact that the nature of movement may not be completely understood, there are partial understandings that can be made precise and are useful in their own right, as will hopefully be demonstrated through the remainder of this thesis. The task of making full use of the arts literature will not be completed in this thesis and could well be a generation's worth of work.

The arts literature allows us to tackle the problem of representing movement style and character properties in a direct and explicit way. We can move towards giving aspects of movement precise and intelligible definitions. Such approaches stand in contrast to statistical approaches that attempt to infer properties about movement but do so without an interpretative framework. It can be a challenge to apply meaning to the products of such analyses. The arts literature, by contrast, provides an interpretive framework for understanding movement. Both approaches are likely of value in the end and should be seen as complementary as one can inform the other. This thesis explores the benefits of explicit representation.

4.1 Performance vs. Daily Movement

As Barba argues, "[t]he way we use our bodies in daily life is substantially different from the way we use them in performance." [17, p.9]. Barba goes on to suggest that "[w]hile daily behaviour is based on functionality, on economy of power, on the relationship between the energy used and the result obtained, in the performer's extra-daily behaviour each action, no matter how small, is based on waste, on *excess*." [16, p.55]. The purpose of stage movement is to infect the audience with emotion [43] and it is generally when the attributes of an actor's movement are out of the ordinary, that they will have the greatest significance for the audience [1].

Because of the need to clearly communicate with an audience, performance movement is based on two basic principles: simplification [93, 160, 88, 17, 109] and exaggeration [160, 88, 16]. These two properties work together to clarify the meaning of a character's movement in the spectator's mind.

Simplification, also known as the "Virtue of Omission" [17], works to bring focus to certain elements of a character's movement by eliminating extraneous movements [17]. In mime, it is often necessary to slow down the pace of major actions to make them comprehensible [93]. Traditional animation preaches the importance of having a character only do one thing at a time [160, 88]. In her summary of Stanislavski's teachings, Moore suggests that "[e]very form of expression must be simple and clear", with an emphasis on precision [109, p.54]. All this lends clarity to performance motion that is often lacking in the movements of daily life.

Once a movement has been simplified, it is exaggerated to ensure that its meaning is conveyed to the audience. Frank Thomas nicely summarizes the interplay of simplification and exaggeration as follows:

> As artists, we need to find the essence of the emotion and the individual who is experiencing it. When these subtle differences have been found, we must emphasize them, build them up and at the same time, eliminate everything else that might appear contradictory or confusing. [159, p.6].

By so doing, an artist ensures that he/she is focusing the audience's attention on the key aspects of the movement and clear communication results.

4.2 On Manner and Action

For the purposes of discussion and analysis, we can divide movement into the actions a character chooses to perform and the manner in which he performs these actions.

This corresponds roughly to the *what* and the *how*. While these two categories may be interleaved in practice, their separation will give a clearer view of how expressive motion is accomplished. The next section will delve into the properties of a particular movement; the *how*. This will be followed by a discussion of *what* movements a character chooses to perform in the subsequent section.

4.3 Building Blocks of Movement

We divide the wide range of low-level movement properties into three main categories: *shape*, *transitions* and *timing*. Shape refers to both the poses that a character assumes at various instances of time and how those poses are ordered in time. Transitions refer to how a character moves from one shape to the next and include important properties like tension and relaxation. Timing refers to properties like the duration, tempo and rhythm of a movement.

To understand how these three categories combine in a movement, consider a character reaching for a glass of water to take a drink. *Shape* would include the character's original pose and his pose when he grabs the glass. Is he hunched over or erect? Did he lean towards or away from the glass? *Transition* properties relate to the movement of picking up the glass. Did the motion start quickly and end slowly or vice versa? What path in space did his arm take? Did he shake during the movement? *Timing* includes the tempo and rhythm of the motion.

4.3.1 Shape

One fundamental aspect of shape is *full body engagement*, which indicates that the whole body is used, even in small movements. This is a key aspect of the teachings of Grotowski [154] and Meyerhold [92]. Meyerhold apparently often observed "If the tip of the nose works, the whole body works" [92]. Our discussion of Grotowski above included his example of how the whole body should react to the fingers touching a piece of ice. Full body engagement serves a similar purpose to simplification and exaggeration, it acts to clarify and emphasize a movement. This is another example of how performance movement is taken to a heightened level beyond daily movement.

Poses can be either symmetric or asymmetric. Lasseter recommends avoiding perfect symmetry (twinning) as this generally looks unnatural [88]. Grotowski argues "if something is symmetric it is not organic!" [59, p.194] Asymmetric poses generate tension through opposition, whereas symmetric poses lack opposition and give balance [16].

Posture

Posture is one of the clearest indicators of both a character's overall personality and emotional state in a particular scene. Alberts suggests that posture is a combination of two components: the level of tension displayed in the body and the overall body position [1]. Body position includes standing, leaning, kneeling, sitting and lying down. Alberts proposes the following posture scale: hunched, stooped, slumped, drooped, slouched, sagging, tired, relaxed, straight, upright, uptight, erect and over-erect (at attention).

Alignment is related to posture. Generally, the spine will have a slight S-curve, but it can be more or less bent, or curved to one side [1]. The discussion of the torso below is also very closely related to posture.

Balance

Balance adjustments are fundamental to expressive movement. Indeed, Barba claims that the "dance of balance" is revealed in the fundamental principles of all performance forms [17]:

> The characteristic most common to actors and dancers from different cultures and times is the abandonment of daily balance in favour of a 'precarious' or extra-daily 'balance'. Extra-daily balance demands a greater physical effort – it is this extra effort which dilates the body's tensions in such a way that the performer seems to be alive even before he begins to express. [18, p.34]

Such balance adjustments can work to intensify a motion. “A change of balance results in a series of specific organic tensions which engage and emphasize the performer’s material presence, but at a stage which precedes intentional, individualised expression.” [18, p.35]

When we stand in daily life, we are never still, but rather are constantly making small adjustments, shifting our weight to the toes, heels, right side, left side, etc. These movements should be modeled and amplified by the performer [17]. In fact, being near the edge of balance or slightly off balance greatly heightens the excitement of a performer’s movements [85, 154]. Barba illustrates this by describing a mime who had to hold up a title card between two scenes. To make the pose interesting and alive, the mime had to find his “precarious balance”, stretching out and leaning well forward. [17] Being off balance also reflects on the action that has just been completed. [154]

Many performance traditions will impose physical constraints to make the balance task more difficult. Some Asian traditions require the dancer to walk on the edge of their feet; ballerinas will often work on toe. Precarious balance features a deformation of the position of the legs and a reduction in the base of support [18].

A proper balance adjustment can even give a static pose a sense of motion.

> The performer’s dynamic balance, based on the body’s tensions, is a *balance in action:* it generates the sensation of movement in the spectator even when there is only immobility. [19, p.40]

This is one of the secrets of great sculpture. There are other aesthetic reasons for doing this as well, such as creating a sense of greater elegance through elongation.

One of the fundamental roles of both Meyerhold’s Biomechanics exercises and Grotowski’s training exercises is to help develop an actor’s sense of balance and control over his body [92, 154]. Indeed, both Grotowski and Stanislavski emphasize the importance of actors learning their point of balance or centre of gravity [59, 148].

Delsarte suggests that balance can create a mood of security and control; imbalance suggests insecurity, indecision, fear and worry [146]. The pioneering mime Decroux argued that faking counterbalance, for example by extending the opposing arm when carrying a bucket, was a key element of mime [40].

Extent

Extent or extension refers to how far an action or gesture takes place from a character's body. It can be thought of as how much space a character is using while completing an action. If the arms are fully extended and straight out from the character's side, this would be maximal extension, whereas, if the hands were touching the torso and the elbows were held at the character's side, this would be minimal extension. Both Laban and Delsarte use the term *extension* [85, 154, 146]. Alberts refers to this as *range*.

Laban refers to the area around a person's body as the *kinesphere* and defines three regions within it [154]. The near region is anything within about ten inches of the character's body. This is the area for personal, intimate or perhaps nervous actions. The mid or middle area is about two feet from the person's body and this is where daily activities take place, such as shaking hands. The area of far extent has the person extended to full reach. It is used for dramatic, extreme movements and in general is used more on stage than in daily life. People learn the space around their bodies and determine the most comfortable way to perform an action [85]. This will vary from person to person. It can appear quite unnatural if the extension of various body parts differs [85].

Delsarte suggests that excitement, explosive anger, strong and violent emotions that are aggressive all act to expand action (or increase extent) [146]. Thought, meditation, concentration, fear, suspicion and repulsion contract a body's movements. Normal emotions and gestures are in-between. He also suggests that slow movements to emphasize vastness and grandeur are aided by full extension. On a related note, Stanislavski suggests that an actor must be decisive in his big movements [149].

Amplitude is a concept related to extent that can also be quite powerful. One of Grotowski's exercises asks an actor to take a large action and perform it repeatedly, each time reducing the amplitude while trying to maintain the same energy [154]. The actor goes to 50 percent, then 40 percent, eventually working down to 3 percent and 2 percent. The result is a movement that is very small and subtle, but very energized and alive.

4.3.2 Meanings Associated with the Parts of the Body and Space

Meaning of Space Around a Character's Body

Similar to the distance discussed above, the direction of an action relative to a person's body also modifies its meaning.

Shawn suggests that we can most clearly see the area in front of us, so things there are better understood and less feared. Forward gestures thus tend to be vital [146]. In mime, forward gestures would be to greet, to agree (ranging from a slight nod of the head to broad sweeping movement of the hand forward from the shoulder), to question (move head forward from shoulders) or to be surprised (moving forward to investigate) [93]. Laban suggests that the forward direction also corresponds to the future [154].

According to Shawn, Delsarte suggests that "gestures ending behind us can be of negation or of fear or may relate to the unconscious or subconscious." [146, p.33] The connotations in mime are similar, including to say no or refuse, to ward off a person, actions taken in abhorrence or hate, and to show fear. The degree of fear can be shown by first turning the chest away and ultimately burying the face in the shoulder for protection and bringing up the front arm as a protective barrier. [93] The area behind the body also references the character's past. [154].

The area above the character's head is associated with inspiration, aspiration, things that are heavenly and infinite [154, 146]. Delsarte suggests upward motions are good, positive, beautiful, etc. [146]. Upward motions can also convey pride [93].

Lawson suggests that all happy emotions move upward and outward [93]. The body seems light, arms open upward and outward and there is a general upward surge of movement. She suggests considering children being released from school to understand the movement. This matches well with Delsarte's Law of Altitude which suggests that the constructive, positive, good, true and beautiful move upward, forward and outward [146].

Laban suggests that the area from the waist down is grounded, firm and finite [154]. Motions that are only downward can also suggest that a character is tired or fed-up [93].

All sad emotions are downward and inward [93]. Lawson goes on to suggest that the body droops downward and inward toward itself when overcome with sad emotions: the head hangs toward chest, feet become heavy, the tempo slows and the rhythm becomes spasmodic. Ultimately the figure collapses, showing only a jerky heaving movement accompanied by sobs [93]. This again meshes well with Delsarte's Law of Altitude, which suggests that downward, backward and inward movements are destructive, negative, ugly and false [146].

Sideways movements can be either open or closed. In open moves, the hands are not allowed to narrow the chest, cover the face or cross the centre line of the character's vision. Such motions give an impression of goodness, bravery, honesty, truth and the like, as the subject is open and has nothing to hide, fear or be ashamed of [93]. Gestures outward from the side can also be used to suggest emotional magnitude [146]. In closed moves, the arms may cross over the centre line of the body, narrowing the chest. These may be accompanied by a furtive to-and-fro movement of the head. Such motions are generally perceived as unpleasant, crafty or cunning [93].

Finally, a turn is like the full stop at the end of a sentence, indicating a decision to act or a change of mind or mood [93].

It must be noted that these are general guidelines and should be considered points of departure, not viewed as a rule set that can be applied blindly to create contextually correct movement. Emotional impact can be generated by alternating between these extremes [93].

See also Section 4.5.6 for a discussion of the meaning of the amount of space between two people.

Direction of Movements in Space

An overall sense of direction is important for a movement. It helps give the movement a sense of focus. Describing one of his actors, Grotowski says "All his movements have a well-defined direction that is followed by all the extremities and, on closer observation, even by all the muscles." [59, p.186]. This leads to a decisive and clear movement piece.

See also Section 4.5.6 for a discussion of how people interact in space.

Body Self-Interaction

Touching one's own body is expressively meaningful. Grotowski writes: "Then there is the question of dialogue between the different parts of the body. When a hand touches a knee or when a foot touches another foot, all that is a search for security." [61, p.231] He argues that this dialogue must be concrete, but it must not come from the brain. An actor must not consciously think about it. In animation, this sort of movement must of course come from conscious intention or tools must be developed that allow an animator's instinctive impulses to affect the motion.

Use of Various Body Parts

Different parts of the body convey different meanings. In mime, the body is divided into four sections, the head, the torso, the arms and the legs [93]. We will follow that breakdown here. Delsarte used a similar division, dividing the body into the head, torso and limbs [146]. He also had a second division that split the body into the head, upper torso and lower torso. Worth noting is the emphasis he placed on the torso. This very expressive section of the body is often given little attention in computer animation.

Delsarte, not surprisingly, considers the head to be the mental zone [146]. He also subdivides the head twice more into mental, vital and emotional zones. He suggests the following action-meaning mappings for a list of basic head gestures: bowing downward suggests affirmation; hanging downwards suggests submission; lifting suggests assertion, inclination from side to side suggests carelessness, indifference or uncertainty; pivotal movement from side to side suggests negation; oblique movement upward toward an object suggests rejection; oblique movement from an object suggests secretive appellation; controlled movement up and down suggests a simple yes.

Alberts suggests that the face is the most important emotional signifier, followed by the body and third is verbal communication [1]. Often the face will give the emotion and the body only indicates the intensity of the emotion. Alberts suggests that the face is capable of approximately 20 000 expressions, 2-3000 of which carry emotional meaning. There are six basic expressions: happiness/enjoyment, sadness, fear, anger, surprise and disgust/contempt [1]. These are the expressions that psychologist Ekman found were recognized around the world [159] (Ekman considers disgust and contempt to be separate emotions, leading to seven universal emotions expressed on the face).

Grotowski argues that it is important to be able to control each facial muscle individually and to be able to set them at different rhythms. For example, an actor should be able to "make the eyebrows quiver very fast while the cheek muscles tremble slowly, or the left side of the face react vivaciously while the right side is sluggish." [15]

Eye movement is an important carrier of information [1, 93]. It matters whether a character makes eye contact, at what else a character looks, how quickly it moves its eyes, etc. Finally, it should be noted that no section of the body is independent of the rest. For instance, when a character fixes its gaze on something, this will modify the position of the spinal column as well [16].

The arms and hands are used to create a wide range of motions. Delsarte refers to the shoulder, elbow and wrist joints as thermometers because he feels they indicate *how* *much* rather than *what kind* of expression [146]. Raised shoulders act to strengthen any action, the more they are raised, the stronger the action. He argues no intense emotion is possible without the elevation of the shoulders, or forward contraction in the case of fear or backward pull to show aggression or defiance. “The elbow approaches the body by reason of humility, and moves outward, away from the body, to express pride, arrogance, assertion of the will.” [146, p.41] Elbows within the column of the body denote a lack of self-respect. Finally, a not rigid, but strong wrist indicates a strong, healthy condition. A limp wrist indicates weakness or a devitalized condition. A sharply bent wrist indicates a crippling influence. Wrist movement can also add definition to a gesture [93]. Stanislavski argues that the arms should neither hang in-front nor behind the body, but at the actor’s side. Elbows should turn outward, not inward, but this cannot be excessive [149].

There is a very large vocabulary of hand gestures, some that are culturally specific and some that are universal. Delsarte suggests the following general mappings: hands with palm down blesses, hands with palm up supports, hand with palm forward rejects, hand held vertically divides, clutching hands show greed or possessiveness, outward going hand is used for giving or receiving, forefinger pointed hand indicates or directs, and the fist threatens. Delsarte also provides a mapping that indicates the meaning associated with touching various areas on one’s face with one’s hand. Details are in [146]. Grotowski makes the more generic observation that stiff hands must be avoided. They should be kept supple due to their expressive importance [59].

Lawson suggests that women tend to touch those parts of the body most affected when she is overcome with emotion, such as the chest or head, whereas men clench their fists, put their hands behind their back or stuff them in their pockets [93].

When we are nervous, the rate of our basal hand movements (those small things we do with our hands all the time) increases. [1]

Shawn argues that one of the main contributions of Delsarte was realizing that the torso is the main instrument of emotional expression [146]. It was for this reason that

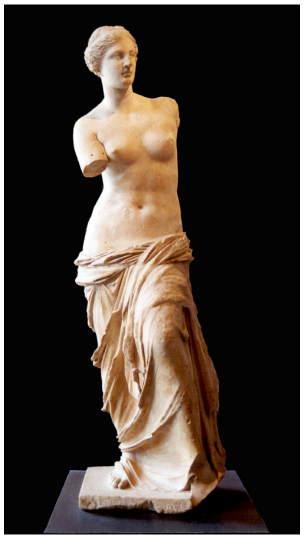


Figure 4.1: Venus de Milo.

modern dance moved away from the stiff and fixed spine that is traditional in ballet.

Delsarte suggested that the part of the torso that a person habitually holds forward is a strong indicator of what kind of person they are [146]. If they hold their chest high, this indicates self-respect and pride. If their abdomen is protruding, this indicates animality, sensuality and lack of bodily pride. A normal, balanced carriage will have the middle zone of the abdomen carried forward and the chest and abdomen withdrawn. This triad can be augmented by considering people who carry their head forward, normally indicating a mental or academic disposition.

The shape of the spine in the coronal plane is also important. The "S" curve or "Beauty Line" involves the legs, torso and neck in making a large S curve with the entire body. It is a key pose in Indian dance, where it is called *tribhangi*, meaning three arches. It was also prominent in ancient Greek sculpture, the Venus de Milo offers a clear example (Figure 4.1), and was taken up by Florentine sculptors in the 14th century. [19] This also serves to illustrate the importance of the interplay between the legs, spine in the torso and neck.

Laban suggests that there are three principal components of trunk movement: rotational movement about the length of the spine; "pincer-like" curling from one or both ends of the trunk and "bulge-like" shifting of the central area of the trunk out of its regular position [85].

Many exercises aim at improving the spine movement of actors, to make it more supple [59] or to individually control each vertebrae that makes up the spine [148]. Stanislavski argues that the actor must feel that the vertebrae are well placed and the spine is firmly attached to its base [149].

The torso can be expanded, contracted or relaxed. Expansion indicates different degrees of excitement, vehemence and power of the will. Contraction indicated different degrees of timidity, pain, effort or convulsion of the will. Relaxation indicates different degrees of surrender, indolence, intoxication, prostration and intensity of the will [146].

If a person leans his torso away from an object, this indicates repulsion. Leaning it towards an object indicates attraction. There is an exception to this in a certain form of sensual admiration that has a character lean away from the object of interest and cock his head slightly back towards it. If the character moves directly towards or away from the object, the action is vital [146].

Repeated rotational movements of the torso indicate childish impatience. Side to side movements indicate carelessness, vacillation of the will or indifference. Up and down movements indicate the despair of the weak [146].

The legs of a character have a functional role to play in supporting it, which limits their expressive range. Nonetheless, stance is very important [146]. Lawson suggests the rather stereotyped movement of trembling knees to show fear [93]. The main sources of variation in the legs are the width of the stance, the bend in either knee, whether the legs are turned out, in or straight and whether there is a twist in the pelvis. Stanislavski argues that the legs should be turned slightly out; rising on one's toes suggests flight; feet and toes modulate jerkiness and give a quality of smoothness and gracefulness to motion [149].

4.3.3 Shape in Time

Somewhere between pure shape properties and transition properties are properties that relate to how body shape changes over time. These include recoil, opposition and squash and stretch.

Recoil

Recoil is one of the most frequently cited movement properties [43, 16, 88, 160, 85, 42, 155, 146]. In its most basic form, recoil involves first making a movement in the opposite direction of the intended movement, followed by the intended movement itself [42]. Recoil serves to underscore and accentuate a movement [43, 42]. It is one aspect of the more general concept of *anticipation* important in traditional animation [160, 88] and is described by Barba as *The Negation Principle* because by moving in a direction opposite to the action one is effectively negating the action before it is performed [16]. This creates a void in which the action can exist. Recoil also allows the momentum of an action to be built, as with the backswing before a punch. Interesting to note, there are several analogies to the negation principle. For example, in painting, especially chiaroscuro, one often darkens an area close to an abrupt increase in intensity to enhance the perception of brightness. Some black and white printing papers will exhibit this same quality. Also, when you need to make a fast turn in one direction with your bike, you will often do a quick turn in the opposite direction.

Recoil is a standard device in oriental theatre, but is used less explicitly in the occidental tradition [16]. Indeed, Eisenstein argues that it needs to be veiled as it is not meant to enter into the consciousness of the spectator [42].

Delsarte's Law of Reaction [146, 155] uses the concept of recoil in a different way. It suggests that the body recoils, not in preparation for an action, but in reaction to either an emotional stimulus or the climax of an emotional state. Any object that surprises us will make the body react in recoil, the degree of recoil being proportional to the degree of emotion caused by the object [146]. A classic example of this is a person recoiling back from a snake, suddenly noticed in front of him. Many of the recoil examples developed later in this thesis will belong to this second type of recoil. Finally, when any emotional state has reached its climax, it provokes an opposing physical reaction – generally, a relaxation [146, 155].

Oppositions

Recoil is one form of *opposition*, which deals with the tension created by opposing forces [17]. As another example, Barba's second law on the opposite direction of impulses suggests that when one part of the body has an impulse, another part of the body has an impulse in the opposite direction [19]. In general, oppositions act to heighten the impact of a performer's movement. They act to create a sense of energy, or a differential between two levels of energy. Oppositions take multiple forms, including recoil, variations in tension and relaxation, soft and hard energy, opposite directions of movement, and changes in pace [17, 155]. Delsarte suggested that having two parts of the body move in opposite directions simultaneously expresses force, strength and physical or emotional power. [146] Conversely, *parallelisms*, or having two parts of the body move in the same direction at the same time, denote physical weakness and are also associated with mental, planned, thought-out activities [146].

Some of Grotowski's exercises would explore "opposite vectors", or opposition in movement [15]. For example, the hands might make a circular movement in one direction while the elbow works in the opposite direction. The creation of contrasting images was also practised, such as having the hands accept while the legs reject.

Gathering vs. Scattering Gestures

Imagine a character moves her arms out from her side, brings them together in front of her in a large loop and then pulls her hands in towards her middle. This is an example of a *gathering* gesture. On the other hand, if the character starts with her hands near her middle and thrusts them out and away from her body to the sides, this is a *scattering* gesture [154]. Delsarte calls scattering actions eccentric, gathering actions concentric and actions that are neither, normal. They are also called open, closed and neutral [155]. Generally speaking, scattering actions suggest openness, sharing and an external focus. Gathering actions suggest the character is closed, coveting or tormented and internally focused. Open and closed static postures have similar connotations.

Squash and Stretch

Disney animators considered *squash and stretch* to be the most important movement principle [160, 158]. Only the most rigid objects do not deform when they are moved [88] and as Thomas observes "all living flesh is subtle and stretches or bulges or sags or becomes taut in reaction to the forces working on it." [158, p.23]. Deforming their characters and objects as they animated became essential in order to give them a sense of life. Lasseter points out that hinged objects such as Luxo can squash and stretch without deforming [88]. This of course applies to the human skeleton as well, where bone deformation is normally not modelled for animation. Squash and stretch is also particularly important for facial animation [88].

4.3.4 Transitions

Transitions deal with how a character moves from shape to shape. They also deal with transient aspects of movement, such as the interplay of tension and relaxation.

Flow

Laban suggests that movement can flow continuously, be intermittently interrupted, yielding a trembling kind of movement, or stopped yielding a pose. [85] It is worth distinguishing between a motion that is paused [85, 154] or suspended [1] and one that is stopped. When a motion is paused, the actor maintains focus on the final destination and can continue the movement at any time. There is no perceptible loss of intensity nor break in intention. When a motion is stopped, the energy of that motion has been lost and the actor's focus is no longer on the completion of a motion. It cannot be seamlessly continued with the same intensity [154]. This may seem to be an esoteric point, but it is quite easy to distinguish between a paused and stopped motion when observing a performer, even if the pose is the same in both cases.

Stanislavski related flow to how energy moved through the body, "external plasticity is based on our inner sense of the movement of energy." [149, p.67] A smooth and regular flow of energy gives a smooth, measured and elastic step. Energy in jerks creates an uneven, choppy gait. Stanislavski told his actors that flow must be controlled and to think of a bead of mercury that they are consciously moving through their veins. Creating an endless, unbroken line with this bead will give smooth, flowing movement. [149]

According to Laban, motion can be either complete or incomplete. [154] Many actions in daily life are incomplete. They are interrupted before they reach their natural conclusion. Being able to stop an action midway can have a powerful effect as it can be a strong indicator of a character's internal mental process.

Disney animators found it effective to have the bulk of footage near extreme poses and less footage in between in order to emphasize these poses. [160, 88] They referred to this as *slow in, slow out*, indicating that when the animation is shown, most of the time is spent on the main poses. Interpolating splines are used to generate this result in computer animation [88], often going beyond simple ease-in, ease-out curves using tool such as tension, continuity and bias splines [82].

Arcs are another Disney animation principle [160, 88]. The arcs principle claims that natural movements are normally not straight, but follow some form of arc.

Finally, it should be noted that developing a sense of flow is very closely related to the use of *successions* discussed below.

Tension and Relaxation

The interplay of tension and relaxation is another widely cited movement property [40, 85, 146, 93, 16]. Tension and relaxation naturally interleave: there must first be relaxation in order for there to be tension and tension is followed again by relaxation. There is a consequent ebb and flow of energy that accompanies changes in tension [146]. This interplay between tension and relaxation is another way to create opposition and build interest in movement. [93, 146, 16].

Tension changes can take place through the entire body or a tiny part. They can occur suddenly or gradually and there can be spasmodic changes back and forth [93]. A rise in tension can serve to accent a movement [85]. For example, physical or emotional pain can be shown by spasmodic contractions of muscles, followed by relaxation as the pain eases [146].

Stanislavski stresses the importance of an actor being relaxed and avoiding tension in his body [148, 149, 109] arguing "*You cannot ... have any conception of the evil that results from muscular spasm and physical contraction.*" [148, p.91] An actor must learn to identify the sources of tension in his body and relax them [109]. Stiff arms and legs give the body a wooden quality, looking like a mannequin. "The resulting impression is that the actor's soul is likely to be as wooden as his arms. If you add to this a stiff back, which bends only at the waist and at right angles, you have a complete picture of a stick. What emotions can a stick reflect?" [148, p. 102]

Weight

It is critical that characters have a sense of weight [160, 158, 7, 85]. This is essential for creating a sense of realism. Thomas and Johnson suggest that it is an inability to correctly capture a sense of weight that makes cartoon characters lose credibility when viewed next to live action [160].

Laban relates weight to the use of muscular energy or force to either move a weight or to react to a resistive force [85]. Weight can come either from the weight of a body part to be moved or from an object being moved. Resistance can be internal, coming from the antagonistic actions of the character's own muscles and reflecting internal conflict, or external, coming from other objects or people. Resistance may involve strong, normal or weak muscular tension.

Weight is given two meanings, both the standard meaning related to the physical mass of the character, and a second subjective meaning that deals with the character's attitude towards weight. The latter deals with whether the character makes heavy or light motions, irregardless of the character's actual weight. Most of the above uses of the term refer to physical mass. Laban's effort notation below focuses on the subjective meaning.

Effort

Laban's Effort notation provides a useful way of talking about transitions. There are four components to effort – flow, weight, space and time – which can either be indulged in or resisted. [85, 154]

Flow ranges from bound to free. In the extreme, bound movement represents the inability to move. It is carefully controlled movement and is normally task specific, goal oriented and planned. Conversely, free movement is loose and undirected. In the extreme, it is impossible to accomplish any task in free movement [154].

Space varies from direct to indirect. Direct movements follow a straight line from A to B. They make minimal use of space. Indirect movements indulge in space, following curved paths and containing many rotations [154].

Time varies from sustained to quick. Sustained motions have a steady pace, like Tai Chi. Quick motions are often short or transient and will rapidly change directions [154].

Weight refers to how much a character indulges in a sense of weight, not how heavy a character is. Weight ranges from strong or heavy to light or weak. Weak is not necessarily

Descriptor	*Components*
dab	direct, light and quick
glide	direct, light and sustained
press	direct, strong and sustained
punch	direct, strong and quick
slash	flexible, strong and quick
float	flexible, light and sustained
flick	flexible, light and quick
wring	flexible, strong and sustained

Table 4.1: Eight composite descriptors of motion from Laban's Effort analysis [154].

feeble and strong is not necessarily oafish. A physically heavy character may move in a light way and vice versa. Weight relates to how much impact the character wants to have with his or her motions [154].

In general, movements will consist of combinations of several of these elements. Ignoring flow, there are eight combinations of the extremes of space, weight and time. These have been given names which are quite evocative of the types of movements that have these properties, as summarized in Table 4.3.4.

4.3.5 Timing

4.3.6 Tempo and Rhythm

The two main components of timing are *tempo* and *rhythm* [93, 85, 1, 109, 149]. Rhythm refers to the cadence of a set of movements. For instance, the pattern could be long, long, short, repeat. It deals with the patterning of full beats, eighth beats, etc. Tempo refers to the speed at which motions are completed; the speed of the beat. Tempo is independent of rhythm, in that a given rhythm could be performed with a fast or slow tempo. Tempo changes can hasten or draw out the action [149]. *Syncopation* can be used to put different accents on beats.

Stanislavski provides a particularly detailed analysis of tempo-rhythm for acting that will be relied on here [109, 149]. He argues that each character has their own tempo-rhythm and it is the work of the actor to find it. The psychology literature also suggests that people have a characteristic tempo [52]. A character's tempo-rhythm is normally different to that of the actor's, so the actor must be aware of his own tempo-rhythm as he works to find the characters. If a character is taking a strong decisive action, there will be only one tempo-rhythm. But a tortured soul like Hamlet will show several different tempo-rhythms at once. Tempo-rhythm has a strong effect on inner mood. Different tempo-rhythms can generate moods ranging from excitement to melancholy and despair [149]. Quiet, slow movements suggest a contented state.

Characters have both an inner and an outer tempo-rhythm. It is something they feel inside and something they express in their movements. A character might have a highly agitated inner tempo-rhythm, if for instance she had committed a crime, but have a slow, calm, cool outer tempo-rhythm. Many involuntary movements can result from this, such as averting eye gaze. Also, the inner rhythm will occasionally escape into her movements, giving brief bits of hurried motion before she can cover them up. These can give away the internal rhythm. An actor, by focusing on the internal and external rhythms, will automatically find these small adjustments that make the character work [149]. In animation, we must work to consciously show them.

Once an actor or animator feels the right tempo-rhythm for a character, the problem remains of how to align the character's movements with that tempo-rhythm. Stanislavski again uses the image of controlling the flow of a drop of mercury through the body [149]. The key movements are when the mercury passes through the shoulder, when the mercury passes through the elbow, from the upper arm to the forearm, etc. It is these split second moments that are aligned with the beat structure of the music or tempo-rhythm.

In any scene, different people should have different tempo-rhythms. Only in stylized scenes or when crowds are seized with a common purpose will people exhibit the same tempo-rhythm [149]. Entire plays or movement pieces also have a tempo-rhythm.

Other terms used to define timing include duration, which is the length of time an action takes [1] and speed, which is the rate at which we let movements follow one another [85]. Clearly both of these are related to tempo and rhythm.

Time Allotment

The timing or speed at which an object moves reflects the size and weight of the object. [88, 146] Larger objects have more momentum and thus accelerate and move more slowly than small, light objects. In his "Law of Velocity", Delsarte extends this idea to include the relationship between the inner gravity of feeling or meaning and the pace and size of its expression [155, 146]. Thus the deepest of feelings may be expressed with complete stillness [155]. According to Shawn, "[e]motions of profound and deeply serious import, then, require slow and large movement patterns; emotions that are petty, light, trivial, nervous etc. take on small and quick movement patterns." [146, p.64] Extending the idea to character types suggests that "[m]ajesty and nobility of emotion and character can only be conveyed by broad, slow movements – while the petty, small, nervous, fearful, irritated, shallow person and emotion is revealed by quick, small movements. [146, p.65] This same phenomenon is often at play in film making when superheroes gifted with great speed are shown not moving more rapidly than those around them, but shown moving in slow motion to lend gravity to their deeds.

It is important to spend the correct amount of time on anticipation for an action, the action itself and then the reaction to the action [88]. Timing can be used to indicate if a character is nervous, lethargic, excited or relaxed [88].

Successions or Activation Ordering of Body Parts

Successions deal with how a movement passes through the body. Rarely will every limb involved in a motion start and stop at the same time. Delsarte defined two types of successions: true or normal successions and reverse succession [146]. In a normal succession, a movement starts at the base of a character's torso and spreads out to the extremities. In a reverse succession, the movement starts at the extremities and moves in towards the centre of the character. Delsarte associated true successions with good, truth and beauty and reverse successions with evil, falsity and insincerity. Shawn claims that the conscious use of successions was fundamental to the development of modern dance [146].

Successions are part of the "Follow Through and Overlapping Action" principle of Disney animation [88]. Thomas and Johnson write:

> Our most startling observation from films of people in motion was that almost all actions start with the hips; and, ordinarily, there is a drop – as if gravity were being used to get things going. From this move, there is usually a turn or tilt or a wind up, followed by a whiplash type of action as the rest of the body starts to follow through....Any person starting to move from a still, standing position, whether to start walking or pick something up, always began the move with the hips. [160, p.72].

Such motion is consistent with a normal succession. It is worth noting that one of the purposes of Grotowski's training exercises is for actors to gain a better understanding of the base of the spine as the seat of their energy and to be able to initiate all movements from here [154]. Grotowski argues that "[t]he vertebral column is the centre of expression. The driving impulse, however, stems from the loins. Every live impulse begins in this region, even if invisible from outside." [59, p.191] This again reflects the outward flow of movement consistent with a normal succession. Stanislavski also gives examples of exercises designed to teach the use of successions in order to increase fluidity [149].

Grotowski extends the idea of succession in his dictum "first the body, then the voice" [59]. He suggests that body movement should always lead the voice. If there is no vocalization, than the hands act like the voice and should follow the core movement.

In Laban's work, successions are related to his flow parameter [85]. Freer flowing movements have a stronger normal succession and bound flow movements may have no succession.

4.4 Actions

Human movements are often broken down into different categories. Laban suggests three categories: steps, which include leaps turns and runs; gestures, using the extremities of the upper parts of the body; and facial expressions, which can include orienting the head. [85] Alberts divides actions into movements, which involve large areas of the body, and gestures, which are generally confined to the fingers, hands and head. [1] This definition of gesture seems somewhat limited, as often more of the body will be engaged in making a gesture, but normally the rest of the body is used merely to amplify the strength of the gesture.

As well as actions, it is important to remember that people also have *reactions* and these can often say a lot about a character. [1] For instance, how a character reacts to a gunshot might indicate whether or not he was expecting it.

4.4.1 Gestures

There are three types of gestures in mime: narrative gestures are used for verbal conversation and are limited in scope; descriptive gestures describe the look, feel, taste, smell and/or sound of some off stage event; emotional gestures describe the speakers or another person's emotions [93].

Gestures often accompany speech. These gestures include representational gestures, such as tracing a question mark in the air or pointing at an object; symbolic gestures, such as making an obscene gesture or waving a threatening fist at someone and verbal-mechanical gestures that have neither representational or symbolic meaning, such as chopping with the hand, jerking the head or rising on tip toe [42]. Speech gestures can repeat what is being said, for example when giving directions; they can illustrate what is being said, by say showing the size of an item; or they can accentuate, underscore or emphasize what is being said [1]. Gestures can also modify the verbal message, for example to lessen its impact, or occasionally gestures can contradict what is being said. This is often unintentional and would include cases such as a character saying he was open and standing stiffly with his arms crossed in front of his chest [1]. The use of gestures with speech has been extensively studied by Cassell [33] and others.

Alberts argues that generally people with higher social standing make fewer and less expansive gestures accompanying speech. Gestures are also often used to replace speech, such as indicating to someone to come here and look, or that you are cold [1]. Few gestures are universal as they are normally adapted to a particular culture. The frequency and extent of gestures also varies from culture to culture.

People also make a number of small, gesture like movements. These include rubbing one's nose, self-grooming, pulling on an ear, hand wringing, foot jiggling, stroking the chin, etc. [1]. Generally speaking, rubbing behaviour is reassuring; scratching and picking behaviour indicates anxiety, nervousness or hostility; behaviour that obscures the eyes indicates guilt, shame or deception; a fist symbolizes aggression or hostility; a hand to nose gesture frequently shows fear or uncertainty; fingers to mouth or lips show shame, embarrassment or self-consciousness; open hand gestures, particularly when accompanied with shoulder shrugs, indicate frustration, confusion or indecision [1]. People will also make small touching gestures to other people, such as picking lint off a shirt, or manipulate objects, such as tapping a pencil.

4.4.2 Walking

Walking is an incredibly important motion, both functionally and because of its wide expressive range. When developing a character, computer animators will often first develop a walk cycle, as all other motions play off this [25].

Shawn illustrates the expressive range of walking and its fundamental place within human movement:

> [The dancer] should study all types, kinds and conditions of human walk-

> ing – the strong, vital and purposeful walk, the indecisive, broken-rhythmed, changing in direction, changing in length of step, walk which indicates the worried, the fearful, the timid, the lost person. The sodden walk of the exhausted and defeated man; the light, on the balls of the feet walk of happiness and joy; the pert walk of the little shop girl on the make; the sensuous walk of Mae West; the slinky, catlike walk of menace; walking backwards, walking with sudden changes in direction, walking with bent knees and hanging torso; walking touching the ball of the foot first, the whole foot, the heel, the side of the foot: all of these and many more are part of the training discipline of the American dancer today. [146, p.72]

Lawson describes the following walks used in mime [93]. Notice the importance of weight shifts in varying the impression of the walk. Slow walks generally are calm and dignified. The weight is not transferred until the working foot rests solidly on the floor. A quick walk shows impatience, alertness or hurry. The weight is carried directly over the front foot and on the toes. The head should be leading the way and the heels touch the ground only momentarily. In a reluctant walk, the weight of the body is held over the back foot and the feet are dragged over the surface. The toe touches first and the weight of the body appears to be dragged backwards. In a tired walk, the feet can be dragged, but it is more normal for the feet to be picked up and dropped. Weight is thrust downwards from the shoulders, which may droop slightly forwards under the weight of the head. A frightened walk is characterized by erratic tempo and rhythm and normally moves in a zig-zag manner. Weight can be shifted: sometimes forwards, sometimes backwards and sometimes to the sides. There is tension in the body and the knees are unsteady. Finally, in a silent walk, the upper body is held up, each foot is lifted quite high and the toe is placed down first, rolling the foot to place the heel. Large steps are the norm.

Stanislavski provides a detailed analysis of the mechanics of walking [149]. In particular, he argues that someone can step heel first, whole sole first, or toe first and then roll down the whole sole. Of these, he argues the first is correct and the toes should bear weight at the end of the movement, completing the flow of the movement. For the movement to float, there should be a moment where the weight is just on the big toe before it is transferred to the other foot. The torso should remain largely motionless, with the spine absorbing shocks to keep the head free of jerkiness. Grotowski suggest that the centre of movement in gait is in different parts of the body as we age [15]. In infancy, it is in the legs; during adolescence, in the shoulders; for manhood, it is in the trunk; maturity in the head; and finally in old age it is in the legs again.

4.4.3 Action Selection

The most basic question an animator, dancer or actor must ask when creating any movement piece is "What move should I make?". Action selection is a critical task. Moore goes so far as to suggest "[t]he creative process of an actor's work is choice of actions" [109, p.56]

The rule of simplification applies to action selection. Restraint is important [149]. Everything that happens on stage must have a purpose [148]. Complex human psychological life should be expressed through simple gesture [109]. Stanislavski writes "Unrestrained movements, natural though they may be to the actor himself, only blur the design of his part, make his performance unclear, monotonous and uncontrolled." [149, p.69]

The actions performed must relate to the inner life of the character, or they will lack meaning [149]. Stanislavski's work allows an actor to develop the proper inner process, from which the outer expression will flow [109]. Outside of performance animation, an animator cannot directly use this approach. However, she may find it is a useful method to determine the right action in a given scene, which she can then generate through animation tools. It is important that every psychological aim is expressed physically and every action has a psychological aim [109]. Furthermore, gestures and movements must have concrete justifications. Otherwise, we are left with beautiful gestures with the "emotions of a fairy dance" [60].

"Frequently physical immobility is the direct result of inner intensity, and it is these inner activities that are far more important artistically." [148, p.34] Stillness can denote inner emotional weight. In computer animation research, there is a tendency to feel character related aspects of motion are constrained to the high frequencies, but the importance of immobility should be remembered.

Action choice determines the type of character which is created. Different characters might share an objective, but they will choose different actions to achieve it. There should be a continuous line through a character's actions and these should build towards the superobjective; the overall goal of the character [109].

One of Stanislavski's fundamental principles is that a performer should never try to act a feeling.

> Fix this for all times in your memories: *On the stage there cannot be, under any circumstances, action which is directed immediately at the arousing of a feeling for its own sake.* To ignore this rule results only in the most disgusting artificiality. *When you are choosing some bit of action leave feeling and spiritual content alone.* Never seek to be jealous, or to make love, or to suffer, for its own sake. *All such feelings are the result of something that has gone before. Of the thing that goes before you should think hard as you can. As for the result, it will produce itself.* The false acting of passions, or of types, or the mere use of conventional gestures, – these are all frequent faults in our profession. But you must keep away from these unrealities. You must not copy passions or copy types. You must live in the passions and in the types. Your acting of them must grow out of your living in them." [148, p.38, original emphasis]

Once more, we are left with the dilemma that animated characters cannot *live* in passions, the way actors – or animators – may. There remains an important lesson here, though. Actions must arise out of a context. Trying to act an emotion without connecting it to the context of the story will make for empty gestures.

Grotowski makes a similar argument, saying that actors should not illustrate words. A bad actor who is asked to act bored will try to show boredom. His actions and gestures illustrate the word. A man who is actually bored will be very active. Perhaps he will read a book, lose interest and put it down. Then he'll look for some food, but nothing tastes good today. Then maybe he'll try to have a nap, but be unsatisfied with this as well [61]. It is good to remember that behaviour is composed of small, logical, concrete actions [109].

Emotions should be made specific to a given character. Again, we can turn to Stanislavski:

> Most actors do not penetrate the nature of the feelings they portray. For them *love* is a big and generalized experience.
>
> They try immediately to 'embrace the unembraceable.' They forget that great experiences are made up of a number of separate episodes and moments. These must be known, studied, absorbed, fulfilled in their entirety. Unless an actor does this he is destined to become the victim of stereotype. [149, p.272]

This draws into question the trend in computer animation research to create systems that generically apply "happiness" or "anger" to any character. Following Stanislavski, these emotions are shown through a number of movements that are customized to be meaningful for a given character in a given situation. Emotional truth lies in finding these correct actions and performing them appropriately. Generic warpings ignore the role of context in defining meaning and the validity of such approaches is open to serious questioning.

Style has received little attention in the discussion so far. Grotowski is known for using a more stylized form of movement. He says that "[a] *sign*, not a common gesture is the elementary integer of expression for us." [64, p.18], arguing that in heightened emotional moments, a man does not behave "naturally". *Signs* are discovered through the exercises.

Disney animators often talk about the importance of making a character appear to think. Moore makes a related observation, stating that the amount of concentration a movement takes on stage should be the same as the amount it takes in real life [109]. There is a tendency in acting and animation not to show the amount of thought a real person would exhibit when acting out a task.

As a final note, creating a sense of continuity between actions is often the most challenging part of putting a movement sequence together. This is true with Grotowski's exercises and also one of the skills they seek to teach [59].

4.5 Building Effective Movement Pieces

4.5.1 Structure of a Piece

A great deal could be written about how to put together an overall movement piece, but we will simply highlight some of the main points. First and foremost, it is important to vary the texture of a piece. If the pacing is the same throughout or the piece consistently hits the same note, the audience will quickly get bored. But, according to Thomas and Johnston, "... if the overall pattern contains accents and surprises, contrasts of smooth-flowing actions with short, jerky moves, and unexpected timing, the whole thing becomes a delight to watch." [160, p.52]

The *super-objective* is the main idea of the play. Every aspect of the performance should relate to it. A character must have a *through line* of actions that runs through his entire role. Moore argues that an actor should work out the entire life of a character, past and future, so that she can situate the character in a grand arc. Characters also have their personal super-objective. [109]

Stanislavski argues for the importance of *perspective*. "Let us agree that the word 'perspective' means: the calculated, harmonious inter-relationship and distribution of the parts in a whole play or role." [149, p.169]. Perspective gives a view of how the entire story develops and allows the actions or movements early in the story to be adjusted so that they will be different to the actions later in the play. For example, in a thriller, the lead character may need to become increasingly more scared as the drama progresses. The early actions that show fear must be held back in order to leave room for the fear to grow. Every action must have perspective and must be related to the super-objective. This allows emotion and action to be appropriately varied over the course of a story.

Perspective highlights an interesting challenge for computer animation tools. Movements must adapt over the course of a play. If a character is sad at the beginning of the play and late in the play, the nature and intensity of the sadness will need to change. Otherwise, the story will be flat. Tools that do not support this variation are of limited use.

4.5.2 Interaction

It is vital that characters interact with the other characters and objects that share a scene with them. According to Grotowski "Something stimulates you and you react. That is the whole secret." [61, p.225]

Contact is Grotowski's term for an awareness of other people, their actions and their moods. He argues that theatre is composed of elements of human contact; "give and take". These are what define the score for the actor. "Take other people, confront them with oneself, one's own experiences and thoughts, and give a reply." [58, p.212] Contact between characters must be maintained during a performance. As one actor makes small changes to the set performance, the other actor should make adjustments as well and hence small changes to the performance should emerge from that contact [61]. Contact is important for maintaining believability. Stanislavski also emphasized the importance of an actor relating his behaviour to the other characters on stage. He must honestly absorb what they say and do and react to them [109]. The inner and outer adjustments people make to one another Stanislavski terms *adaptation* [148]. In the animation setting, it is similarly important that characters are sensitive to each other's behaviour and fit their actions to what the other characters are doing. They must have *contact* and *adaptation.*

Characters must also adapt to overcome physical obstacles [109]. A character must perceive and interact with objects. He must treat them as if they are what he wants the audience to perceive them to be. It is important to assign meanings to objects and then show those meanings in movement. For instance, a drink might be wine or poison. The character's movements should reflect which it is. [109]

4.5.3 Action Timing

Actions need time to breathe on stage if the audience is going to follow the inner thoughts of the character. Delsarte argued that there is a sequence of perception, recognition and action that underlies movements [155]. Allowing time for these different phases can make the communication with the audience more clear. This relates to the more general form of the *anticipation* principle in traditional animation. It is important to give the audience hints as to what is coming so that they are ready for it and will perceive it when it arrives [88, 160].

Secondary action can be added to a movement sequence in order to strengthen and clarify the meaning of a sequence. This can include small actions like wiping a tear, shaking his head or putting on glasses [160].

4.5.4 Storytelling and Audience

Interestingly, Thomas and Johnson argue that you should

> Tell your story through the broad cartoon characters rather than the "straight" ones. There is no way to animate strong-enough attitudes, feelings, or expressions on realistic characters to get the communication you should have. The more real, the less latitude for clear communication. [160, p.375]

As computer animation advances, we are able to capture more subtle expressions on more and more realistic characters. Making believable and communicative movement on a highly realistic model, however, remains exceedingly difficult.

Last but not least, we should not forget the role of the audience. The audience must be able to relate to the characters and feel an emotional reaction to them if audience is going to be engaged by the story. [160]

4.5.5 Aesthetic Considerations

Acting and animation are both media based on physicality. The role of movement is primary. Stanislavski argues for the importance of physical characterization: "without an external form neither your inner characterization nor the spirit of your image will reach the public. The external characterization explains and illustrates and thereby conveys to the spectators the inner pattern of your part." [149, p.3] Grotowski argues that our reactions must be physically shown as well: "The essential thing is that everything must come through the body. First and foremost, there must be a physical reaction to everything that affects us." [59, p. 204]

One should look for the unusual in what they choose to show in a movement piece. Moore argues that there are a variety of truths: the uninteresting ones and the interesting or unexpected ones. The unusual forms are more impressive on stage [109]. Grotowski argues that it is important to always show the unknown side of things to spectators [61].

Clichés should be avoided. Grotowski suggests that when a character says "What a beautiful day", he should not always say it with a happy intonation nor "Today I am a little sad" with a sad intonation. These are clichés. Strive instead for what the character's deeper intention is, what lies below. [61] This warning is particularly relevant for computer animation research. Too often, computer tools are designed to provide nothing but the clichés, such as a single "happy " or "sad" walk.

The subtext is a very important aspect of a play or animation and may be in direct conflict with the text. Much of the meaning comes from the subtext. It reveals the inner life of the character. Subtext *must* be revealed in visual signs [109].

4.5.6 Group Movement

As soon as two or more characters are placed together in a scene, a whole new level of communication opens up. How each character moves relative to the other character, what changes they make in stance and posture and how they position themselves in space not only speaks volumes about the relationship between the characters, but also comments on the personality and inner state of each character individually.

Following Alberts [1] there are several key properties that define an interaction:

1. What body positions do the two characters assume? Is one standing while the other remains seated? Is the one character leaning?
2. What posture do the two characters have and do they change their posture when they are together? It is illustrative if one character consciously adjusts his posture in the presence of another.
3. How are the two characters oriented relative to each other? Are they facing each other straight on or are they oblique? Do the feet and the face point in the same direction?
4. The distance between two characters is significant as will be discussed below.
5. Do the characters tend to mirror one another?
6. A particularly important and culturally relative property is eye behaviour. Where are the characters looking? How long to they maintain eye contact?
7. What gestures do the characters choose to make when interacting with each other?
8. And finally, do they touch each other? Having characters touch has a strong impact for the audience [160]. What is the frequency, nature, duration and location of the touching? Is there more dramatic physical contact, such as embracing, kissing, hitting or pushing?

Following to Alberts, the amount of space around the body that a person considers to be private or personal varies a great deal from culture to culture [1]. In North America, personal space extends outwards from the body roughly three feet in all directions, which is more than most other cultures. The intimate zone extends to about eighteen inches. The social zone is about four to ten feet. Four to six feet is safe for informal conversation, six to ten feet is more suited for formal interactions. The kind of greeting someone makes when passing a friend on the street is a function of the distance at which they pass. If they are more than about twelve feet a part, a wave is fine. Ten to twelve feet normally requires them to verbalize. Passing within six feet, they should generally enter into a conversation.

Power relationships are often a key part of interpersonal relationships and they can give a scene a great sense of drama. Delsarte's "Law of Force" states that "Conscious strength assumes weak attitudes; Conscious weakness assumes strong attitudes." [146, p.48] This idea is very similar to Johnstone's idea of status [78]. Only low status characters try to actively show off their power. Characters that are truly high status realize that everyone else is aware of this and need to make no show of their power. Status can manifest itself in many of a character's actions and status can also change during a scene. Status is discussed in more detail under the section on character.

Very often, physical interactions will change during a scene as two characters decide they like or dislike each other. Alberts suggests that courtship normally passes through the following ten types of physical interaction [1]: eye to body, eye to eye, voice to voice, hand to hand, arm to shoulder, arm to waist, hand to head, mouth to mouth, hand to body, mouth to body.

Eye gaze can be difficult to read in long shots or on stage, but it remains a very important clue for the audience. Alberts presents the chart shown in Table 4.5.6 of activities that increase and decrease eye contact between characters. A complicated set of factors is involved. The interpretation ultimately made of a character's gaze direction will depend very much on the overall context of the scene.

Regulating behaviour helps determine turn taking in conversations and is another important aspect of interpersonal interactions [1]. It is often dropped in stage movement as the actors are working from a script so there is no need for it. Adding regulating behaviour to a piece of performance motion, however, can help increase realism.

4.6 Characters

We have now described at some length the lower level properties that make up movement. The question remains, how does one select from these properties, customize them and put them together in such a way to create unique and specific characters? This section will explore what is meant by *character*, and how we can begin to construct unique movement patterns for a given character. Before proceeding with that, it is worth reflecting for a

Factors influencing increased gazing	*Factors influencing decreased gazing*
You are physically distant from the other person.	You are physically close to the other person
You are discussing uncomplicated or impersonal topics.	You are discussing difficult, complicated or intimate topics.
There is nowhere else to look.	You have other objects, people or background to look at.
You are interested in the other person's reactions to your words or behaviour.	You are not interested in the other person.
You are interested in the other person - you like or love the other person, and you are looking for reciprocal visual activity.	You are talking rather than listening.
You are of lower status than the other person, in which case you may be looking for reinforcement, acceptance, validation, or some other form of positive feedback.	You are not interested in the other person's reactions to you, your words or your physical activity.
You are trying to dominate or influence the other person.	You perceive yourself as having higher status or of greater importance than the other person.
You are from a culture that values visual contact in human interactions, as in most parts of N.A.	You are from a culture that does not encourage visual contact during human interaction (e.g. an Asian culture)
You are an extrovert or wish to appear so.	You are an introvert or wish to appear so.
You want or need to be liked or accepted by the other person - you are "looking for acceptance."	You don't need or want to be liked or accepted by the other person, or wish to appear so.
You are dependent on the other person - for love, for food or for a job.	You are detached or distracted from the conversation.
You are listening rather than talking.	You have a mental disorder.
	You are embarrassed, ashamed, sorrowful, submissive, trying to hide something or attempting to be deceitful.
You are female.	You are male.

Table 4.2: Activities that increase or decrease eye contact. (from Alberts [1])

moment on the nature of the movement properties we have previously described.

It is a mistake to blindly apply these properties in a prescriptive way. They are not recipes that can be followed to give a specific result, but abstract principles that give insight to the nature of movement. As Shawn says,

> We have to extract [the laws of Delsarte], make simple statements of them for study purposes; but in actuality, in the final use, a vast number of these laws are operating simultaneously and mutually modifying and affecting each other to produce complex yet seemingly simple results. [146]

There is a level of artist interpretation that is fundamental to these principles: "[N]o two artists using the same principle would produce forms of expression identical with each other, or with anyone else." [146, p.26]

Even when the principles being applied are well understood and seem straightforward, the results may not be correct. Thomas writes:

> Even the moves that seemed to define attitudes or mood proved to be annoyingly elusive. To make a figure sad, we slumped the body, dropped the shoulders and the head, dragged the feet when walking, and timed the actions slowly. But these elements of sadness could also mean despair, or listlessness, or exhaustion. [158, p.21]

This illustrates the level of ambiguity inherent in descriptions of movement properties and the difficulty of the task.

It is exactly these aspects of ambiguity, interpretation and synthesis inherent in descriptions of movement properties that makes it difficult to put them into computational form. They are not hard, objective laws and they do not form a consistent calculus. At the same time, these principles discovered in the arts probably provide the best insight available into the nature of expressive movement. Due to the amount of subjectivity that remains, it is extremely valuable to have a human artist, capable of making decisions, involved in the process.

Although Delsarte's Law of Correspondence suggests a direct, causal link between outward appearance and internal feeling [155, 146], the actual situation is no doubt more complicated than this. Appia likely hits closer to the mark when he states

> Gestures, however, do not express the life of our soul directly.... We can suffer for hours, and indicate it by a gesture of only a second's duration. Gesture in our daily life is a sign, a symbol, nothing more. Actors know this, and guide their acting by the very inconsistency between these two patterns of time duration: that is, between that of the inner life of our soul and that of our physical body - the latter of which is certainly different from the former. [7, p.22]

This chasm between the inner state of the character and what he/she decides to reveal to the world is another source of complexity in generating computational models of movement.

4.6.1 Creating a Specific Character

Defining a character's movement is difficult because so many factors come into play in deciding how a particular character moves in a particular situation. This dilemma is well captured in the following passage from Laban:

> So movement evidently reveals many different things. It is the result of the striving after an object deemed valuable, or of a state of mind. Its shapes and rhythms show the moving person's attitude in a particular situation. It can characterise momentary mood and reaction as well as constant features of personality. Movement may be influenced by the environment of the mover. So, for instance, the milieu in which the action takes place will colour the movements of an actor or of an actress. They will be different in the role of Eve in paradise, or of a society woman in eighteenth-century salon, or a girl in the bar of a public-house in the slums. All three women might be similar personalities exhibiting almost the same general movement characteristics, but they would adapt their behaviour to the atmosphere of an epoch, or a locality. [85, p.2].

Synthesizing these diverse aspects into a consistent whole and determining a way of movement that reflects that is the challenge faced by animators, actors and dancers when portraying a character. As will be seen elsewhere in this thesis, our work seeks to aid this process by allowing animators to sketch out various aspects of movement that seem appropriate for a particular scene. The character's actions in a scene will then be modified to match the character sketch. By trying different character sketches, the animator can freely experiment with different movement styles in order to evaluate what movement properties best reflect their intention for a given character in a given scene. This will be explored in more detail later, but for now we will look at the many aspects that influence a character's movement.

The first set of properties influencing a character is physical. What is the character's age? Old people and children will display vastly different movement patterns. What is his or her body type? Endomporphs are round, soft and fat. They are generally perceived as dependent, affable, generous, relaxed, tolerant, sociable, affectionate, emotional, cooperative and sympathetic. Mesomorphs are muscular, bony and athletic. They are generally perceived as dominant, determined, adventurous, confident, outgoing, optimistic, energetic, courageous, aggressive, enthusiastic, argumentative and temperamental. Ectomorphs are thin and fragile. Common perceptions of such characters include that they are tense, anxious, meticulous, self-conscious, thoughtful, considerate, shy suspicious, awkward, tactful, cautious and introspective [1]. It would be rare for a character to completely conform to the common personality traits of one type. Indeed, it can often be effective to play against the normal expectations associated with a given physical body. Nonetheless, it is useful to be aware of these expectations. A character's overall bearing, posture and physical attitude are both influenced by his character, and a reflection of his character.

Other physical attributes worth considering include if the character is sick, ill or tired. Does the character have injuries that affect its movement? What is the race of the character and how is this reflected in movement, such as attitudes towards personal space or the use of gestures [146]?

The second source of influences on a character's behaviour comes from his/her environment. What is the social standing of the character, particularly in reference to other characters in the scene? What role is the character playing? Is he a servant who must be subservient or a commander that must try to show confidence and leadership? What is the overall milieu of the action? Is it a ghost town saloon or society party? What is the character's life work? How comfortable is the character in the given setting?

A character may behave differently when in crowds or on his own. All characters have a public persona and a private one [1]. It is useful to show audiences both. Emotional expression can be free or repressed [1]. Except during very heightened moments, characters will repress or conceal their emotional expression in public. While on their own, or in heightened public exchanges, they are more likely to engage in free emotional expression. A character showing restrained or repressed emotions can be very captivating [1].

The psychological makeup of the character offers a third set of influences. This includes both what a character is thinking and how a character processes thoughts [1]. A character's motivation is a key influence. Motivation should always be positive. That is, it should be phrased in terms of what a character wants or needs, not what a character does not want [1]. Negative motivations are very difficult to play, whereas positive motivations are what often gives a scene its energy. Finding a character's motivation for each line of dialogue is a technique actors use to generate tension in a scene [154, 148]. It allows the subtext to be reflected in the physical manifestation of the acting.

A final set of influences on a character's movements come from his or her emotional state. Is the character happy, sad, impatient? Angry or content? Each of these emotional states will strongly effect a character's movements.

Through all of this is woven the thread of personality. Personality reflects a picture of a particular character's way of being in the world. According to Alberts, it is made up of the four main levels of influence - physical, environment, psychological and emotional - but determines how the character will react to each of these factors [1].

4.6.2 Character Properties

Although accounting for the wide range of factors affecting a character's behaviour and movement style is a daunting task, all hope is not lost. The constructive problem appears to be less difficult. In other words, while it is immensely difficult to understand the complex and multifaceted influences acting on a character, it is easier to put together a set of movement patterns that, at least at some level, capture the sense of the character. Indeed, many such mappings for stereotyped characters, like an old-man, already exist and there are many heuristics that guide actors in developing a character. These provide useful starting points in developing unique movement signatures.

Age is perhaps the easiest attribute to characterize in movement. Following Lawson [93], young people are more energetic, impetuous and quicker. Middle aged people tend to be more methodical, direct and bolder. Old women tend to make maternal gestures towards children and can walk quite quickly, almost on their tip toes, when moving towards them. Old men often have stiff feet and knees. They walk in a fussy manner with their feet turned out. Often they are short sighted and wear glasses. Their hands are often busy with finicky gestures and they are nearly always a little out of breath. Of course, these are all clichés, but they serve as potential starting points when developing movement while likely not being the final goal.

People will often have developed habits related to their work [93]. Labourers who lift heavy loads will often pull up their pants. Other people may mop their brows frequently.

Laban suggests that deciding on a person's inner attitudes towards each of the four effort parameters goes a long way towards defining their character [85]. At the very least, it is a good way to establish an initial, strong style of movement.

In group relations, status will normally come into play. One person will be higher status than another and this dictates how they behave towards each other. Playing up status issues is a common technique for increasing tension and interest in a scene. Alberts provides the chart summarized in Table 4.6.2 detailing the stereotyped behaviour expected of low and high status characters.

Status changes are often possible over the duration of a scene and can have a powerful, or more often, humorous effect. A traditional gag sees the butler ending up on top after

Higher Status	*Lower Status*
stands erect	body slightly hunched or bowed
body relaxed	body tense
good posture	body asymmetrical, unbalanced
many gestures	few gestures, restrained
large, expansive movements	small, limited movements
talks more, and more loudly	talks less, and more quietly
often interrupts inferiors	rarely interrupts superiors
often invades other's personal space	always "keeps his distance"
often touches subordinates	rarely, if ever, touches superiors

Table 4.3: Behaviour of low and high status characters.

a series of shenanigans.

It must be noted that such generalizations should only be seen as starting points. As Thomas notes, everyone moves differently [158]. There are contradictions and exceptions to every type [93]. Furthermore, generalized movement is never going to be enough to satisfy an audience. "Animation cannot afford ambiguity. It has to speak quickly and directly to its audience. General kinds of movement are not enough to sustain a film of this type and the moves that are used must be completely convincing, natural and fluid." [158, p.24] Further to this, Stanislavski argues that broad generalizations do not work on stage. A skilled actor will subdivide a general character category to produce something more defined and specific. Clichés do not work [149]. We need tools that allow the creation of strong, specific characters that can engage an audience.

When creating a particular character, it is important to avoid easy stereotypes. Stanislavski's description of an old man is illustrative of the dangers here. "[The joints of an old man] rasp and squeak like rusty iron. This lessens the breadth of his gestures, it reduces the angles of flexibility of his torso, his head. He is obliged to break up his larger motions into a series of smaller ones and each has to be prepared before he makes it." [149, p. 29] A young man might turn his hips 50-60 degrees while an old man moves 20 degrees and slowly. The tempo and rhythm of an old person's motions are slow and flaccid. Yet Stanislavski criticizes an actor that directly implements this: "You are keeping constantly to the same slow rhythm and pace as you walk to an exaggerated caution in your gestures. Old people are not like that." [149, p.30] Old people will change their rhythm, have periods of preparation followed by periods of speed. At times they are limbered up and there is momentum to their movements. They prepare actions much more than the young. For example, when sitting down they will feel for the chair and prepare to sit, sit, pause and then finally lean back. Once seated, the difficult part is done. The person can show more vigour, but while seated and with a limited joint range [149]. There is a varied texture to the movement. The old man does not perform all motions in the same way. Also, notice that both the sequence of actions performed and the manner change when an old man sits down versus a young.

There is an important lesson here for computer animation researchers. Any process that tries to modify a motion sequence through filtering, interpolation or extrapolation will not add new motions into the sequence. It will not for, instance, insert a motion of reaching back to gauge the location of the chair when taking a sitting motion and recreating it in an "old-man" style. A structural change must be made to the motion in order to take these differences into account. Both *how* the motion is performed and *what* motions are performed must be changed when the character is changed. This has repercussions for how we design tools.

In the final analysis, it is easier to build up a character through experimentation with different movement patterns than it is to create a universal mapping that will go from a definition of a character's inner state to its outer movements. In other words, constructing a character is easier than producing a final, a priori analysis of what movements a character will make because it is easy for a human to evaluate various movement approaches and make refinements. When building a character, artists may start from a stereotype or some basic ideas and then create an initial movement sequence. These movements can then be reviewed, either by the artist or by other members of the creative team, in order to answer one main question "Does the movement work?". Based on the resulting judgement, they can modify the movement, add nuances, make small adjustments, drop features that are distracting or even try a different starting point. The movement can again be evaluated and through this iterative process, a movement style is developed that captures the perceived essence of the character. This experimental, iterative approach has been the basis for the performimg arts for centuries. It is precisely what goes on during the rehearsal process for a play. Our system seeks to match the strengths of this time tested workflow. It keeps the animator in the creative loop and tries to make it easy for him to experiment with different movement styles and ideas.

To conclude, it is worth noting that similar findings are reported in the experimental psychology literature which studies the relationship between body movements and emotions. For example, leaning towards a subject is viewed as positive and leaning back has negative connotations [107]. Contracted poses featuring hunched shoulders, a forward leaning trunk and sunken chest have a sad or negative connotation [107, 108], whereas expansive motions are found with angry or happy emotions [108]. An erect posture seldom accompanies shame, sadness or boredom, which tend to feature more collapsed postures [169]. Lifting the shoulders is typical of joy and hot anger, and moving the shoulders forward is more frequent with disgust, despair and fear [169]. Body movement and postural activity is an important indicator of the quantity of emotion, as well as quality [169].

While it is reassuring that the psychology literature also recognizes the emotional impact of these movements, it is important to note that our system does not enforce particular bindings for particular emotions. We feel this choice is best left to the animator as deviations from the norm are often what make a character interesting. This freedom allows an animator to create characters with behavioural idiosyncrasies or characters that "go against type".

Chapter 5

Shape

Our whole body must adapt to every movement, however small. Everybody must proceed in his own way. No stereotype exercises can be imposed. If we pick up a piece of ice from the ground, our whole body must react to this movement and to the cold. Not only the fingertips, not only the whole hand, but the whole body must reveal the coldness of this little piece of ice.

Jerzy Grotowski, Towards a Poor Theatre, p.193

This and the following chapters will examine how the lessons taken from the arts literature can be used to develop a new set of computational tools that explicitly target expressive aspects of motion. The focus of this chapter is *shape*.

Shape refers simply to a character's pose. The system allows shape to be controlled in three ways. First, an animator can directly specify desired joint angles for a pose. Second, a *body shape solver* has been developed that will solve for a pose while combining reach, balance and aesthetic constraints. Both of these methods are used primarily for pose generation whereas the third method acts to modify a pose that has already been generated. Here, a pose that has been stored in the BRep is manipulated by applying additional properties such as *Vary Extent* or *Vary Amplitude.*

When the full system is discussed in Chapter 7 it will be seen that motion generation takes place in two phases: a motion plan is first created that specifies what the character is to do and how he is to do it, and then the motion plan is executed in order to generate the final motion. The body shape solver is used in both of these phases. During the planning process, the solver is used to compute a series of poses for the character. During motion generation, a component of the solver is run at each time step to actively control the Degrees of Freedom (DOFs) in the lower body in order to satisfy constraints placed on the character's lower body, such as a desired balance point or knee bends. Even if the body shape solver is not used to determine the pose, it can still be used to control the lower body during motion generation. An action specification can include a set of desired upper body joint angles along with a set of signal parameters to control the lower body and these will be seamlessly combined in the final full body animation.

The lower body plays a significant functional role in maintaining a character's stability. This reduces the number of ways it can be used expressively. More automatic control of the lower body is both acceptable because of the reduced range of movements of this section of the body and required because of the functional role it plays. For this reason, we will generally automatically control the lower body to satisfy a reduced range of aesthetic constraints while also maintaining balance.

The body shape solver will be employed differently at the various phases of the creation process. During the early exploration phase, the solver can be used to interactively explore pose space and develop new shape properties, referred to as *shape sets*. When specifying an animation sequence, the solver can be invoked with a specific set of parameters in order to calculate a desired pose. Finally, the solver can be used interactively as needed to refine any poses in the animation sequence.

The body shape solver is primarily designed for determining poses where the character's support is his feet. These poses include one and two foot standing postures, crouches and twists.

5.1 Direct Specification of Joint Angles

As part of any action description, the animator can explicitly specify the desired joint angles for any subset of the character's DOFs, including hand specifying entire poses. This is done by specifying a *SetDOFAngle* property for each DOF to be controlled. These properties can also be used to set parameter signal values to control movement aspects such as the character's desired balance point or pelvic twist. As with other properties, different instances of SetDOFAngle can be layered together to determine the desired value for a DOF.

5.2 Body Shape Solver: Overview

As discussed above, the body shape solver is primarily designed for determining poses where the character's support is his feet. This makes it well suited for a very significant range of movements including posture adjustments, balance shifts, gestures and reaching motions. A character's posture can reveal his inner state and also indicate how he feels about his environment. A tired character will often hunch over whereas a proud character will arch his back and puff his chest. A character afraid of an object will often pull back from it whereas an interested character may lean toward the object. Similarly, stance adjustments play a key role when a character reaches out to touch objects by providing an indication of both the character's personality and the character's feelings towards the object.

Such posture adjustments are not generally the main actions that define a movement sequence. They are, however, often the movements that convey the most information and clarify the intentions of a character. Many inverse kinematics (IK) tools help an animator achieve reach and foot constraints, but they provide limited support for varying the manner in which these constraints are achieved in order to reveal the character's personality and intentions. The intent of the body shape solver is to allow an animator to efficiently and effectively control the aesthetic aspects of character pose that are necessary for telling a story, while also satisfying hard world space constraints for foot and wrist locations. The solver does this by simultaneously solving for world space constraints, balance constraints and aesthetic constraints within a unified schema.

A significant focus of this work is to allow animators to easily explore aesthetically meaningful portions of posture space. To this end, both low and high level interfaces have been developed. Animators can interact with our system by directly modifying the low level parameters of the shape solver. These parameters have been designed based on lessons from the arts literature to relate to key expressive aspects of posture. Alternately, animators can work more efficiently at a high level of abstraction by selecting a *shape set*. Shape sets are procedurally defined and encapsulate a particular aesthetically meaningful range of poses. Example shape sets will be shown for recoil motions and a posture range. Shape sets make use of the parameters in the body shape solver and provide animators with a simple interface consisting of a scalar intensity value and possibly one or more world space constraints. Switching shape sets allows an animator to efficiently explore the range of expressive postures, and the animator can always switch to the low-level interface to fine tune a given posture. Shape sets are fully extensible and new shape sets can be developed as needed during a production.

The body shape solver has been presented in [116] and [118].

5.3 Body Shape Solver: Kinematic Balance Control

Balance control is based on a feedback strategy and works in conjunction with the lower body IK algorithm described below. The support polygon is the convex hull of the monitor points attached to the character's feet. The balance algorithm attempts to keep the projection of the character's centre of mass (COM) onto the ground plane at a particular location within the support polygon. Provided the character's momentum is low, she will remain statically balanced as long as the COM projects into the support polygon. The desired location of the projection can be changed either by a shape set or directly by the animator, allowing for balance adjustments.

One foot is set to be the dominant foot and balance control is actuated through this foot. The dominant foot can be changed arbitrarily during a simulation and the other foot can be lifted off the ground. The current X and Z error in the location of the COM projection is fed back into the Z and X axes of the ankle in order to minimize the error. The other joint angles in the lower body are then calculated by the IK algorithm in order to keep the other foot rooted.

A similar feedback approach has previously been used in control of dynamic characters by Wooten [173], but their work fed the error into the desired value of both ankles and both hips. If this approach were used to control the actual angles directly instead of the desired values, it could lead to errors in the lower body joint angles, depending on the width of the stance and location of the balance point. In the dynamic case, ground reaction forces will prevent floor penetration and maintain foot contact. When these values are used directly in kinematic control, however, errors in joint angles can cause foot sliding and floor penetration. We avoid these errors by controlling only one ankle and using our IK system to solve for the remaining angles in the lower body to maintain the position of the other foot.

Feedback based balance control is used when generating an animation. As discussed below, calculating a pose makes use of an optimization procedure that must update the character's balance. For this routine, a simple search is performed to directly find the correct X and Z ankle offsets rather than using the feedback procedure which adjusts these values over multiple time steps.

The kinematic balance controller is also used for dynamic simulation when the sticky ground floor model is used. This allows a wide range of lower body adjustments to be achieved within a dynamic setting.

5.4 Body Shape Solver: Low-level Parameters

Our shape algorithm will solve for a character's pose whether or not reaching constraints have been specified. Foot placements are always maintained and world space targets can be specified for either or both wrists. If no wrist targets are specified, the system returns the pose that satisfies the character's shape constraints.

Our algorithm and implementation is specific to the human body. This allows us to construct the controls that we find most desirable for making expressive adjustments to human pose and also allows us to take advantage of our knowledge of the human form. An expressive animation system for general creatures is a related but separate goal.

Our algorithm breaks the body into four parts which are dealt with separately: the lower body (legs and pelvis), torso, arms and head. This breakdown is the same as is used when analyzing movement in mime [93], and is based on the different aesthetic contributions made by each of the four zones. Analytic methods are used for the lower body, arms and head. An optimization method is used for the torso and to help meet reach constraints. Our treatment of each of these zones is described in detail below.

5.4.1 Lower Body

The input to the lower body algorithm is the location and orientation of each foot, desired knee angles, balance adjustments to the root ankle from the balance routine, and desired pelvis X, Y and Z orientations. Default values are used for any parameter that is not specified. The pelvic Y orientation is relative to the line that passes through the centre of the two ankles. Note that while the knee angles are part of the skeleton state, the rest of the values are constraints the algorithm will try to achieve. Balance point, stance width, pelvic orientation and knee bends are the key aesthetic parameters for the lower body and they are all directly controllable in this model. The algorithm outputs a pose that maximizes satisfaction of these constraints.

In brief, the algorithm starts with the position and orientation of each foot and then calculates the centre of each of the two hip joints. Starting at the root ankle the root hip position is calculated based on the ankle adjustments from the balance routine and the requested knee angle. It then solves for the free hip position in order to keep the free ankle in position and achieve the other pelvic and knee constraints. The skeleton is then subsequently fit to these points, maintaining the foot orientations, which requires calculating the orientation of both legs in order to set the hip angles.

Detailed Lower Body IK Algorithm

The input to the lower body algorithm is the location of each foot, desired knee angles, balance adjustments to the root ankle from the balance routine, and desired pelvis X, Y and Z orientations. The goal is to calculate the left and right hip locations and all the angles in the lower body.

The desired angle of the root knee is taken as the actual value for this joint. The length of the vector from the root ankle to the root hip is solved for using the knee angle and thigh and shin lengths. The X and Z offsets provided by the balance algorithm are applied to this vector to determine the root hip location relative to the location of the root foot.

Consider the triangle shown in Figure 5.1 made up of the hip length (side B), the line from the free foot to the root hip (side A) and the line from the free foot to the free hip (side C). The location of the root hip and the free foot are known and we want to determine the location of the free hip. In general, it is not possible to satisfy the pelvic constraints and the knee constraint simultaneously. The animator can specify how these constraints are to be traded off. We will consider the case here where the algorithm tries to exactly achieve the knee angle and minimize the error in the requested pelvic values, which is the compromise we normally use in practice. Other compromises involve straightforward variations of this solution.

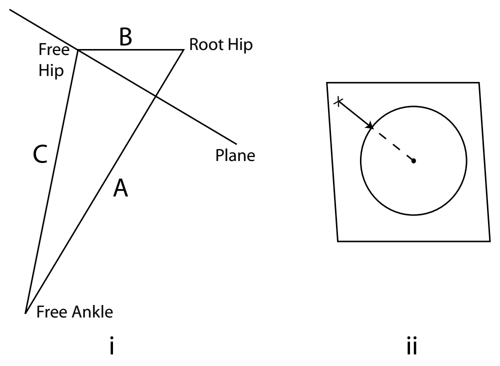


Figure 5.1: i) The location of the free foot and root hip are known and the location of the free hip must be solved for. ii) The initial estimate of the hip vector intersects the plane at the point marked with an X. This point is moved to the closest point on the circle of valid hip locations.

The free knee angle determines the distance between the free foot and free hip, or the length of side C. There is a plane that is perpendicular to the line A and passes through the free hip location. All potential values for the free hip lie on a circle in this plane. The circle is centred where line C penetrates the plane and its radius is the height of the triangle, taking C as the base.

To determine the free hip location, the desired pelvis Y and Z rotations are first applied to the hip vector. The vector is then (extended and) intersected with the plane. In extreme poses, it is possible for the hip vector to lie in or close to the plane. Intersection is not well behaved at this point, so we instead project the end point of the vector onto the plane and use this as the intersection point. The intersection point is moved to the closest point on the circle of possible values, as shown in Figure 5.1, and this is used as the location of the free hip. Once the free hip location is known, the angles of the free ankle can be calculated.

The final step is to determine the angles of the two 3-DOF hips. Given the frame of the root thigh, a rotation can be determined that will align the hip with the calculated location for the hip vector. This defines two degrees of freedom. The third degree of freedom is a rotation around the hip vector that controls the pelvic X rotation. We compute a view matrix that looks along the hip vector and has the pelvis's Y axis as

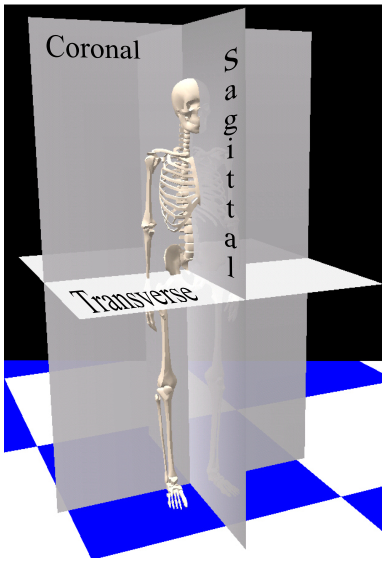


Figure 5.2: The coronal, sagittal and transverse planes are displayed as they relate to the human skeleton. In our coordinate frame, the transverse plane is the XZ plane, the sagittal plane is YZ and the coronal plane is YX.

its *up* vector. The pelvic X rotation is used to determine a desired *up* vector. This is projected onto a plane perpendicular to the hip vector and the projected location is used to determine the necessary rotation that must be applied around the hip vector. Quaternions have proved a useful representation for doing these calculations and the final result can be converted to whatever representation is necessary for the animation system.

The process for determining the free hip degrees of freedom is similar. The frame of the pelvis and the location of free thigh are now known so a rotation that will align the thigh with its desired location can be calculated. Again, this uses two degrees of freedom, the third being the amount of rotation around the thigh. This rotation is similarly calculated to ensure that the shin and foot are correctly located.

5.4.2 Torso

The torso is comprised of two components: the spine, including the neck, and the collar bones. The spine is parameterized by three values and the collar bones by two. We analyze spine shape in terms of deformations in the coronal plane, deformations in the

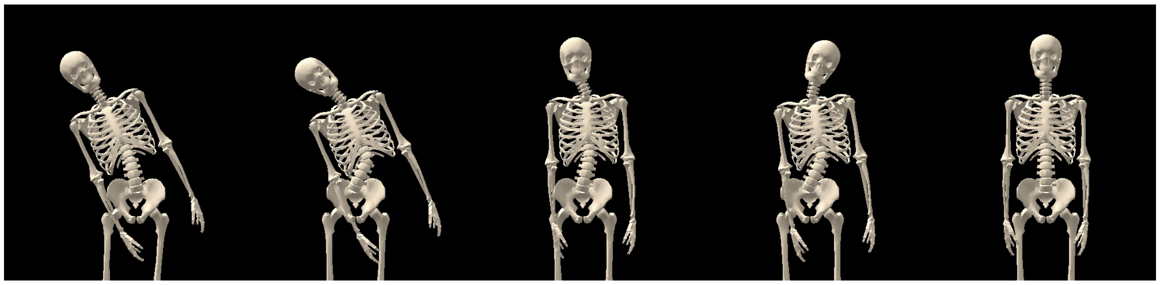


Figure 5.3: Emphasized for illustrative purposes, the five coronal shape classes for the spine are shown from left to right: Large C, Small C, Large S, Small S, Straight. Additional weight shifts have not been added.

sagittal plane and the amount of twist or transverse motion. These planes are shown in Figure 5.2. Sagittal and coronal configurations belong to one of five classes depending on the shape they generate: large S, large C, small S, small C and straight. The straight class indicates no deformation in that plane. Large S and C make the shape of the letter including the lower body. Small S and C make the shape totally within the spine. These are illustrated in Figure 5.3 for the coronal plane.

The configuration of the spine is determined by a coronal shape class and amplitude, a sagittal shape class and amplitude, and a transverse amplitude. Each shape class defines a relative weight for each DOF in the spine that is part of the affected plane. The final DOF value is calculated as the product of the weight and the amplitude. For example, the DOFs that control movement in the sagittal plane are calculated as follows if the LargeS shape class is active:

$$AbX = sagittalAmplitude * LargeSWeight_{AbX} \tag{5.1}$$

$$ChestX = sagittalAmplitude * LargeSWeight_{ChestX} \tag{5.2}$$

$$NeckX = sagittalAmplitude * LargeSWeight_{NeckX} \tag{5.3}$$

$$HeadX = sagittalAmplitude * LargeSWeight_{HeadX} \tag{5.4}$$

The weight values are summarized below. For the transverse DOFs, the amplitude is evenly distributed amongst the spinal joints.

Sagittal Weights				
Class	*Degree of Freedom*			
	AbX	ChestX	NeckX	HeadX
Large S	.15	-.1	-.12	-.1
Large C	.15	.15	.15	.1
Small S	.15	-.2	.06	-.1
Small C	-.15	.3	.15	.1
Straight	0	0	0	0

Table 5.1: Sagittal Weights

Coronal Weights				
Class	*Degree of Freedom*			
	AbZ	ChestZ	NeckZ	HeadZ
Large S	.08	-.1	-.12	-.06
Large C	.15	.15	.12	.06
Small S	.15	-.2	.12	.06
Small C	-.15	.3	.12	.06
Straight	0	0	0	0

Table 5.2: Coronal Weights

These shape classes provide sufficient control to achieve the spinal properties outlined in the background section, namely curling, shifting of the central torso, Albert's posture range, the beauty line and control over what part of the body is leading a pose. For instance, the large C class naturally models hunched behaviour, whereas the S classes can be used to thrust the chest out or generate the beauty line.

5.4.3 Spinal Shape Class Weights

The coronal and sagittal shape classes specify a weight for each DOF of the spine in their respective planes. Final DOF values are obtained by multiplying the coronal and sagittal amplitudes by the weight factors associated with each DOF. The weight factors are summarized in Tables 5.1 and 5.2.

5.4.4 Collar Bones

The collar bones consist of Y and Z rotations. In each of the Y and Z dimensions, the two collars are set to either align with or oppose each other. If they are aligned that means that both shoulders move in the same direction; opposed indicates that the shoulders move in opposite directions. A shrug in which both shoulders are raised is an example of an aligned motion. Collar movement is characterized by a Y and a Z amplitude and Y and Z states indicating whether the motion is aligned or opposed.

The collar movement works in conjunction with the spinal movement to generate the desired torso pose. Aligned backward movement of the collar bones opens up the torso, while moving the collars forward closes the chest. Aligned movement upwards raises the shoulders, indicating tension or apprehension. Conversely, downward movement is more relaxed or dejected. When the collars are opposed, they act as single rigid unit. This is particularly useful to emphasize a transverse twist in the spine by rotating in the same direction or to oppose the twist by acting in the opposite direction.

We also found it occasionally useful to allow an offset to be calculated for the X angle of the abdomen. This occurs when a particular amplitude gives the desired spinal shape, but due to the location of a world space object, one may wish to rotate the entire torso closer or further from the object. In this case, equation 5.2 becomes:

$$AbX = sagittalAmplitude * LargeSWeight_{AbX} + abXAmplitude. \tag{5.5}$$

An animator, either directly through an interface or by invoking a shape set, specifies which shape class to use for each dimension of the collarbones and the sagittal and coronal components of the spine. An optimization routine is then used to solve for the coronal, sagittal and transverse amplitude; the X and Z amplitude of the collar bones; and if it is included, the abdominal offset. The shape classes and the amplitudes operate together to determine the value of each DOF in the torso.

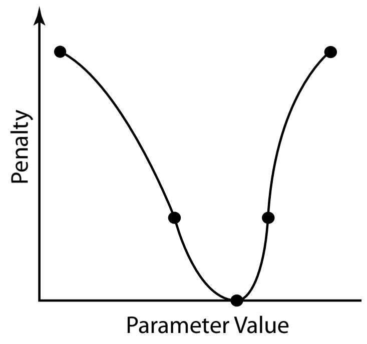


Figure 5.4: The penalty function consists of desired point (shown by the filled circle that lies on the parameter value axis) and four cubic Hermite curve segments that increase the penalty as the parameter moves away from the desired point.

The optimization routine is necessary to resolve two types of conflicts: conflicts between desired internal body values and world space reach constraints; and conflicts over the desired X abdomen value, which is controlled by both the sagittal class and the abdominal offset. The latter of these is more significant and motivates the use of optimization.

A penalty function is defined for each amplitude using four cubic Hermite curves as shown in Figure 5.4. A desired value is specified for each parameter and two curves on either side of this value increase the penalty as the parameter varies from this set point. The penalty functions have three desirable features: they are smooth; they can have flat basins when any value within a range is acceptable or steep sides when a certain value is desired; and they can easily be made asymmetrical which allows solutions on one side of the desired point to be favoured. An additional penalty function is included for each wrist constraint that is active. These penalty functions specify a desired distance from the shoulder of the reaching arm to the world space constraint. The objective function to be minimized is defined as the weighted sum of penalty functions f:

$$obj = \sum_i w_i f_i \ . \tag{5.6}$$

Additional penalty functions can be added to enforce other desired world space constraints. The penalty functions indicate how easily the different aspects of torso motion can be adjusted in order to achieve desired reach constraints, as well as specifying the importance of the reach constraints themselves and their desired accuracy.

We use a simple direction set optimization procedure that performs well for our application and does not require the computation of derivatives. A *line* minimization minimizes a one dimensional function. *Direction set methods* operate in n-dimensional space. They find a minimum by performing a sequence of line minimizations on n-dimensional direction vectors [132]. Different strategies can be used to determine the best set of directions. The dimensions of our space correspond to the amplitude terms discussed above. We use the simplest set of search directions: the unit vectors corresponding to each amplitude.

Our algorithm uses Brent's Method to perform line minimization. This uses inverse parabolic interpolation to speed the search for the minimum. Details and code are available in [132]. The core of the optimization procedure performs a line minimization for each amplitude term in sequence. This is done iteratively until the objective is within tolerance of zero or the objective stops improving, indicating a local minimum.

In our experience, the optimization procedure generally converges in a small number of iterations, rarely taking more than five. This indicates that the shape of the function being optimized is reasonably well aligned with the search directions. If this were not the case, an alternate set of directions for line search would perform better, but since our iteration count is low, we have not investigated more complicated search strategies.

5.4.5 Arms

If there is a wrist constraint on the arm, we solve for the elbow angle given the location of the constraint and the location of the shoulder. A rotation around the vector from the shoulder to the world space constraint is then applied to the arm triangle. This controls how far the elbow is from the chest, which as discussed is expressively important. The amount of rotation is parameterized and is either specified by the animator or the active shape set.

If there is no wrist constraint, we simply determine the elbow angle and rotate the arm out from the body based on a parameter provided by the animator or shape set.

5.4.6 Head

The key contribution of the head is to indicate gaze direction. When desired, we solve for the head orientation in order to have the character look at a specified world space point.

Gaze involves a combination of head movement and eye movements. It can be unnatural for a character to turn his head to completely align with a gaze direction when in real life the gaze would be achieved by a combination of head and eye movement. This is particularly true if the character is indifferent, casual or frightened. The system includes a weight that allows head position to be varied from the rest position to being fully aligned with the gaze direction.

5.5 Body Shape Solver: Shape Sets

The controls in the body shape solver were carefully designed to allow animators to directly alter the most aesthetically important aspects of a character's pose as evidences in the arts literature. In practice, the interactive control palette allows an animator to arrive quite quickly at a desired pose. Nonetheless, there are still many significant parameters to specify and it is often beneficial to allow an animator to more quickly explore the shape space. *Shape sets* are procedurally defined and encapsulate a particular class of body shapes. They take a small set of parameters, generally the name of the desired shape, an intensity value, and if required, one or more world space points that are needed to solve for the shape.

Shape States	
State	*Possible Values*
CorClass	LargeS, LargeC, SmallS, SmallC, Straight
SagClass	LargeS, LargeC, SmallS, SmallC, Straight
Collar Z	oppose. align
Collar Y	oppose. align

Table 5.3: The torso configuration is defined by four states and an amplitude associated with each state.

Shape sets are simply another type of movement property in the system. Composite properties may invoke both shape sets and other classes of properties, such as timing changes or transition warps.

When using the body shape solver, a particular pose is defined by a set of states and a set of numeric values. The states define the configuration of the spine and are defined in Table 5.3. Each state will have a numeric value associated and their are additional numeric values that defines features such as the X dimension balance offset and the orientation of the elbow relative to the spine. The state set defines the interpolation space in which an artist is working. Different state sets lend themselves to different aesthetic meanings, providing a particular tunnel through pose space. Working within a shape set allows an animator to more quickly explore a particular pose range and switching shape sets allows an artist to more efficiently explore the total artistic range of pose space.

The numeric values can be viewed as defining a shape vector. If all the numeric values are zero, for any state set, the character will stand in a neutral pose with his arms at his side. By interpolating from the origin to a given shape vector and then extrapolating beyond it, one obtains a continuous range of “intensities” of a pose. This is the basic idea behind the shape sets and the intensity parameter. Shape sets are based on one or more shape vectors and state sets and the intensity parameter interpolates between these. The interpolation can be between the origin and a given vector or it can be between two or more vectors.

Simple interpolation between vectors is not always sufficient, and for this reason shape sets are defined procedurally. Often certain aspects of shape will relate to the location of the world space object the character is reacting to. If a person is recoiling from a large spider, the direction of the recoil will depend on the location of the spider. By procedurally defining shape sets, the directional or other contextual aspects of a pose can be varied based upon the world space constraints specified by an animator. In other circumstances, some aspects of shape will change at a different pace than others, making regular interpolation inappropriate. Consider a posture range from an over-erect army officer standing at attention, to someone badly hunched over with their knees slightly bent. There may be a continuous interpolation of the spine and collarbones between the two poses, but the legs will be straight over most of this range and the knee bend should only be introduced near the end of the interpolation range. This is straightforward to do with a procedural representation where both the method for determining a shape vector and the interpolation rules for moving between shape vectors can be varied. Furthermore, shape sets facilitate the use of different interpolation functions for different parameters, allowing different parts of the body to change at different rates over an interpolation range. The interpolation functions can also be biased to emphasize a certain area within interpolation space.

Shape vectors have other uses as well. By making small perturbations to the state vector, the pose of the character can be varied while still keeping the character in the same general area of “shape space”. This makes it easy to have a character maintain a consistent set of poses without having the character always hold the *same* pose. Negating and reducing the amplitude of a shape vector allows a character to move backwards to wind up for a motion before moving forwards. This is an important anticipatory effect.

5.5.1 Building Shape Sets

To create a shape set, a user would normally begin by experimenting with the low-level interface. Through this, one or more appropriate sets of states and corresponding shape vectors will be determined. These are tied together procedurally in a shape set by specifying what interpolation functions will be used to move between them as the user varies the input parameters: an intensity scalar and possibly a world space location. The intensity value, denoted i, is normally defined over $[0, 1]$ or $[-1, 1]$.

The shape vector components are typically divided into two categories: those parameters that have a world space dependency and those parameters that are primarily used for aesthetic modification of the character's posture. The latter are varied based on the intensity value and the former are varied based on both the world space location and the intensity value. Parameters with a world space dependency are primarily used for orienting the character. The pelvic Y and transverse twists, weight shifts and knee bends can all be used for orientation. Arm length, collar twists and in some cases, spinal deformation, can also have a world space dependency, but also have a strong aesthetic impact.

Shape sets are designed to operate over a particular working range. For example, reach sets will generally operate well for reach targets that are a comfortable distance in front of or beside the character, but may not gracefully handle outlying target locations.

Example: Keen Reach shape. This shape set uses a short arm constraint in order to pull the character close to a given world space target and adjusts the character's body to show intense interest. An example generated with this shape set is shown in Figure 5.5. The generated poses involve a forward lean, so the working range of the shape set includes targets in a reasonable range in front of the character. An example set of poses generated with the shape set is shown in Figure 5.6, illustrating the coverage of the shape set.

The parameters of this shape set are summarized in Table 5.4. Notice that the

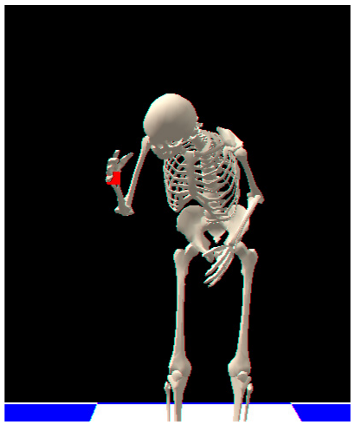


Figure 5.5: An example generated with the Keen Reach shape set.

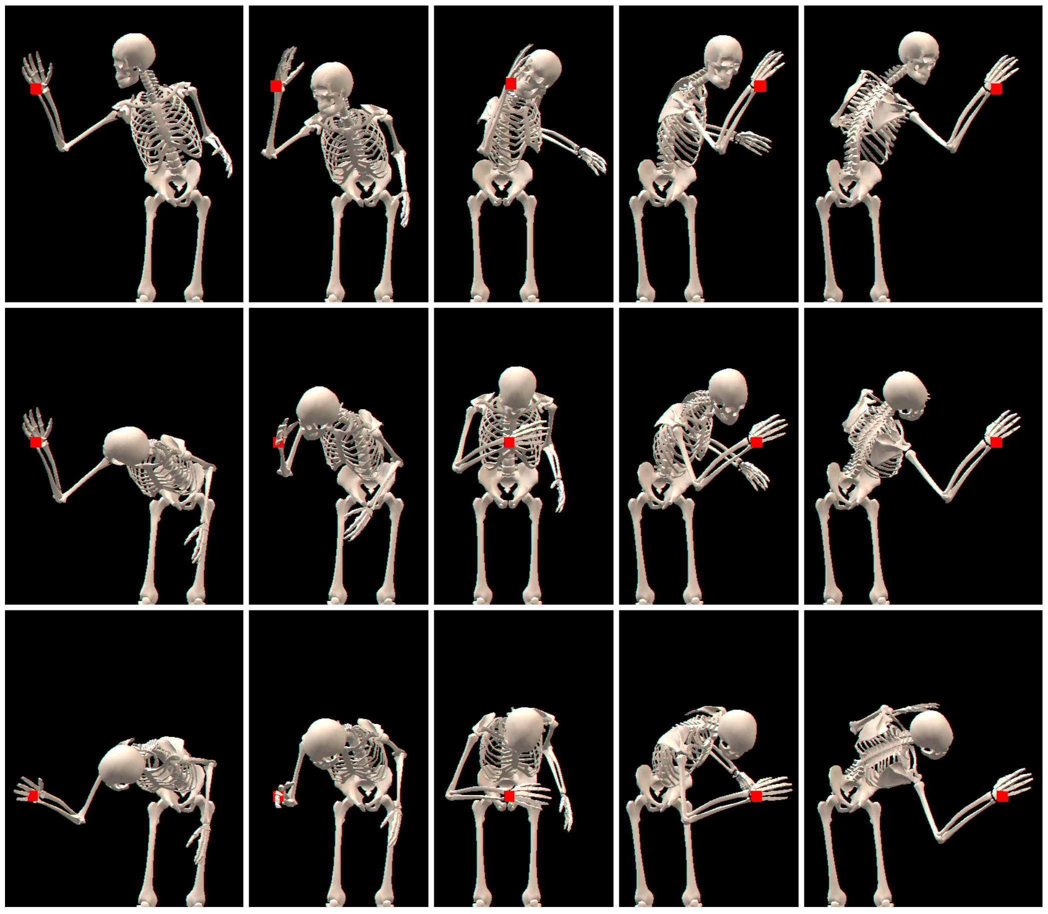


Figure 5.6: Examples poses for various reach locations generated with the *keen reach* shape set. The intensity for each pose is 0.7. The targets are shown as red boxes. The vertical spacing between targets in each row is 20 cm. The horizontal target spacing is -50 cm, -30 cm, 0 cm, 30 cm and 50 cm.

Shape Set Parameters for Keen Reach			
CorClass	LargeS	CorAmp	0
SagClass	LargeC	SagAmp	$3.7i$
TranAmp	$(5.7\theta + 0.15)i$	PelvisY	$(0.7\theta + 0.05)i$
RArmLength	$1.0 - (.7i)$	LArmLength	$1.0 - (.57i)$
GazeFactor	1.0	RArmAngle	$0.5i$
Collar Z	oppose	Collar Y	oppose

Table 5.4: State sets and desired shape vector values for the *keen reach* shape set. The desired value for any parameter not shown is zero. $i \in [0, 1]$ is the intensity value. If the world space location is projected onto the horizontal plane with its origin at the centre of the character, θ is the angle that point is away from the axis coming directly forward from the character. All angles are in radians.

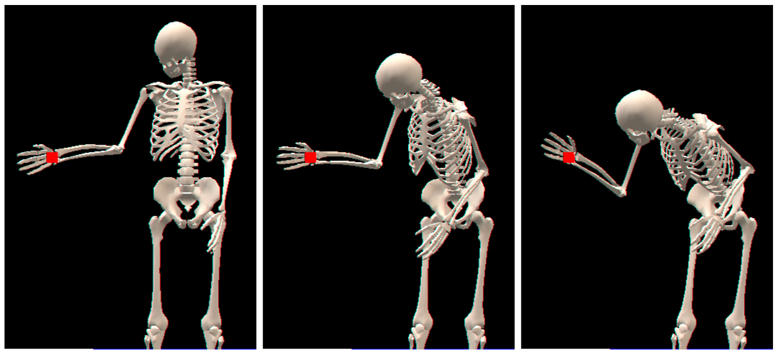
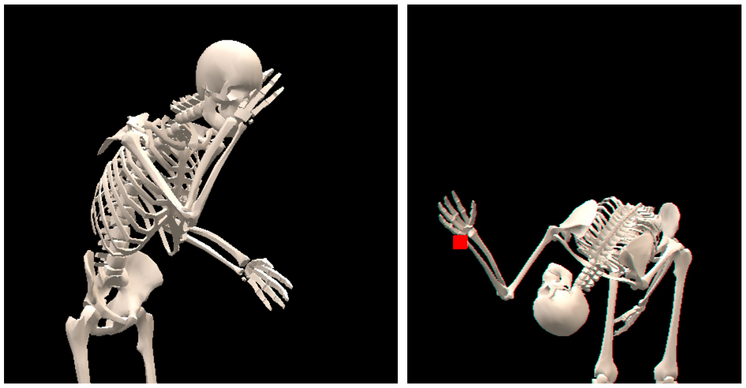


Figure 5.7: The *keen reach* shape set is used with intensity values 0.1, 0.4 and 0.7 respectively. The reach target is shown as a red square.

world space location is used only to determine the angle θ in the horizontal plane that the target makes with the forward vector. This in turn is used to modify the pelvic and spinal twists to orient the character towards the desired location. The intensity value will determine the desired shape vector. The final values of the shape vector are generated by the optimization process as it works to satisfy the reach constraint on the character's right arm. The reach constraint plays a significant role in deforming the torso. Since the gaze factor is 1, the character will look directly at the target. The impact of changing the intensity value is shown in Figure 5.7.

If a target is outside of the shape set's working range, errors can occur. Figure 5.8 shows two types of error. The system does not deal with collision detection and hence the character can self-intersect for certain configurations. This is shown on the left for



(a) Interpenetration Error (b) Unnatural Pose Error

Figure 5.8: This figure shows two potential types of error. Because there is no collision detection, interpenetration can occur. If the requested posture and reach constraint are not reasonable, unnatural poses may result.

a reach target that is too close to the character. In the top row of Figure 5.6, the reach targets for the middle three examples were moved out to avoid this error. For the middle example, the target was moved 10 cm and it was moved 5cm for the examples to the right and the left. The other targets in the figure are at the same depth. A second type of error occurs when the reach constraint is too difficult to satisfy and the optimization procedure forces the character into an unnatural pose in order to achieve it. This is shown on the right of Figure 5.8. This example is for the target in the lower left of Figure 5.6. The error was corrected in that pose by reducing the arm length constraint by ten percent. That was the only pose in the figure that required such an adjustment.

The optimization process will not automatically change the lower body posture. This was a design decision because it was felt that lower body posture changes are part of a higher decision process. Figure 5.9 shows the *keen reach* shape set being applied to a low target. By default, the character will remain standing and will bend over to reach the target. The shape set can be augmented to have the character automatically crouch for targets below a certain height. Both examples are shown in the figure. Note that neither option is necessarily right nor wrong, but the choice to crouch is likely related to

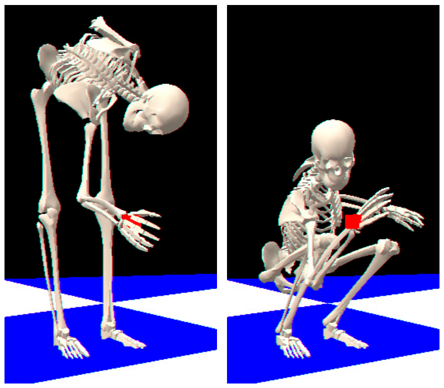


Figure 5.9: The *keen reach* shape set is applied to a low target. By default, the character will remain standing. An augmented version of the shape set has the character automatically crouch for low targets.

the personality and health of the character.

Finally, it should be noted that at any point, the animator can flip back to the low-level interface to fine-tune the solution determined from the shape set. This allows shape sets to be used for rapid search, followed by a refinement phase to achieve the needed pose.

5.5.2 Combining Shape Properties

Shape data may be specified through the various input channels available in the system and must be combined when solving for a final character pose. Shape data of the same type, i.e., with the same set of states, can be blended in a straightforward manner by combining their desired numeric values. The animator can specify edits (*animator edits*) that are used for refinement and apply individual shape parameters that override parameters previously defined in a shape set. This allows animator edits to augment or adjust the behaviour of shape sets. A particularly useful technique is to use the character sketch to specify a default posture for a character. The default or base posture can be specified either using a shape set with an intensity value, or by specifying a series of low-level shape parameters. The default posture can include both parameters that are used in

Blending Rules for Default Postures	
Attribute	*Blend Rule*
Arm Angle to Side	Add offset to current value.
Arm length	Use default if shorter.
Keep arms vertical	Done if requested by either the default posture or the active shape set.
COM offsets	Use default if the current shape set does not request an offset.
Pelvic twist	Use default if the current shape set does not request a twist.
Knee bend Amp	Use default if the current shape set does not request a knee bend.

Table 5.5: Rules for combining shape values used in analytic routines. The "default" values refer to a default posture specified in the character sketch which must be blended with whatever shape parameters are specified as part of the current action.

the optimization process and parameters are that are used in the analytic routines. The parameters used in the optimization process provide an offset to each of the torso DOF values. For example, the equations governing the sagittal shape should a LargeS shape be requested become:

$$AbX = sagittalAmplitude * LargeSWeight_{AbX} + AbXAmplitude + base_{AbX} \tag{5.7}$$

$$ChestX = sagittalAmplitude * LargeSWeight_{ChestX} + base_{ChestX} \tag{5.8}$$

$$NeckX = sagittalAmplitude * LargeSWeight_{NeckX} + base_{NeckX} \tag{5.9}$$

$$HeadX = sagittalAmplitude * LargeSWeight_{HeadX} + base_{HeadX} \tag{5.10}$$

where the *base* values are the default posture. The AbX and sagittal amplitudes are both solved for by the optimization routine. Combining shape data in this way allows a default posture to be defined with one set of states and an action to specify a shape set that uses a different set of states.

Blending rules define how the analytic properties are combined. The current rules are summarized in Table 5.5.

5.5.3 Body Shape Solver Output

Shape properties are associated with a particular pose in the action hierarchy discussed in Chapter 7. Once the properties are merged, their application sets values in a data store that is accessed by the body shape solver. The solver determines a set of joint angles that meet the constraints and stores these angles as SetDOFValue properties attached to the pose's DOFEntries. The solver may also add signal parameters to the pose.

5.6 Pose Modifying Shape Properties

Once a pose has been determined, it can be useful to vary it in well defined ways. *VaryExtent* and *VaryAmplitude* are two properties that operate in this manner that were presented in [114]. Both read a pose from the base representation, modify it, and then store it back into the base representation[1].

5.6.1 Amplitude

An amplitude edit acts in a similar manner to a linear scaling operation in modeling: it compresses or expands the joint ranges over which a motion occurs. As the amplitude increases, the joint ranges spanned by a motion are increased and as amplitude decreases, the joint ranges are decreased. There are numerous ways to define an amplitude edit. Our current edit scales movement relative to an interpose average, as discussed in the implementation section below.

Bold and excited gestures often have large amplitude. Shy or nervous characters often will make small amplitude movements. The amplitude edit allows an animator to very quickly change the feel of a movement. Subtle yet expressive movements can often be obtained by taking a large motion and reducing it to a small proportion of its full amplitude while maintaining the same energy.

[1]Details of the base representation are provided in Chapter 7

Implementation

Amplitude edits take a positive real value a which specifies the degree of the amplitude adjustment. A value of one indicates no change, less than one a reduction and greater than one an increase. This adjustment must be done with respect to a reference pose, the semantics of which we now describe.

By default, an amplitude edit will calculate the inter-pose average $\theta_{\mathbf{avg}}$ between end values and varies the amplitude relative to this:

$$\theta^{\mathbf{i}}_{\mathbf{avg}} = \frac{\theta^{\mathbf{i}} + \theta^{\mathbf{i+1}}}{2}, \tag{5.11}$$

where $\theta^{\mathbf{i}}$ and $\theta^{\mathbf{i+1}}$ are consecutive pose vectors and $\theta^{\mathbf{i}}_{\mathbf{avg}}$ is their average. Two deltas are calculated, one measuring the distance from the average to the end of the pose and the second measuring the distance from the average to the end state of the previous pose:

$$\mathbf{\Delta}_{\mathbf{prev}} = \theta^{\mathbf{i}} - \theta^{\mathbf{i}}_{\mathbf{avg}}, \tag{5.12}$$

$$\mathbf{\Delta}_{\mathbf{next}} = \theta^{\mathbf{i}}_{\mathbf{avg}} - \theta^{\mathbf{i+1}}. \tag{5.13}$$

$$\tag{5.14}$$

The deltas are multiplied by the user provided amplitude a and added to the average to determine new suggested end and start values; a start value simply being the end value of the previous pose. Formally,

$$\theta'^{\mathbf{i}} = \theta^{\mathbf{i}}_{\mathbf{avg}} + a\mathbf{\Delta}_{\mathbf{prev}} \tag{5.15}$$

$$\theta'^{\mathbf{i+1}} = \theta^{\mathbf{i}}_{\mathbf{avg}} + a\mathbf{\Delta}_{\mathbf{next}} \tag{5.16}$$

$$\tag{5.17}$$

where $\theta'^{\mathbf{i}}$ and $\theta'^{\mathbf{i+1}}$ are the updated pose vectors. If the pose is among a sequence of poses, there will be a suggested new value calculated relative to the average on either side of it. These are averaged to generate the final value. Joint limits can also be enforced here. Whenever quaternion joints are used, spherical linear interpolation is used instead of regular linear interpolation to determine joint angles.

An amplitude edit can also take a reference pose. In this case the amplitude is varied relative to that pose rather than relative to the computed averages.

5.6.2 Extent

The concept of extent was discussed in Chapter 4 and refers to the proximity of an action to a character's body [85]. It is generally applied to arm movements. There are three extent ranges: near, mid and far. Near movements take place within a few inches of a character's body. These actions often suggest a character is timid, nervous or shy. Mid extent movements include most daily activities such as shaking hands. They occur at a medium range from a character's body and appear relaxed and normal. Far extent movements occur as far from the body as possible. The arms are straight and stretched out from the body or above the head so that the character is occupying as much space as possible. Such actions suggest excitement and confidence. They also read more clearly if a character is viewed at a distance and are often used on stage or in long shots in film.

Extent edits and amplitude edits are particularly effective when used in conjunction with each other.

Implementation

Two different extent edits are defined. The first examines how closely the arms are held to the body. It blends the poses in an action with a pose that has the arms held straight down, close to the torso, in order to vary the shoulder angle and either pull a movement closer to the body or move it out into space.

The second edit examines the distance of the hand from the shoulder and allows an animator to pull the hand closer to the shoulder or move it further out into space. A similar averaging process is used here as with the amplitude edit above. When multiple poses are edited, an average extent value is calculated over all the poses and then an offset from this average is calculated for each pose. The extent edit varies the distance of this average and maintains the same offsets relative to it. In this way, the extent of an action like a wave can be varied without needing to vary each individual pose in the wave.

5.7 Results

The results for the body shape solver and the pose manipulating properties will be discussed separately. In general, it is possible to layer the latter over the former. All animations discussed in this section are available online [112].

5.7.1 Body Shape Solver

To emphasize the role played by shape, we have not changed the timing, succession, transition envelope or other expressively important aspects of the motion for the animations discussed in this section. A simple ease-in, ease-out curve is used for all transitions.

To demonstrate the correctness of the balance algorithm and lower body IK routine, we have the character complete a complicated set of motions including a crouch, pelvic Y twist and balance adjustments. This sequence is performed with both a narrow and a wide stance.

The posture range proposed by Alberts is shown in Figure 5.10. The parameters that define the two extreme poses in the sequence are described in Table 5.6. The interpolation scheme will move from one parameter set to the other, passing through the neutral zero pose. State changes are made at the neutral pose. The spine's sagittal shape class changes from a SmallS, used for the over-erect posture, to a LargeC used for the hunch

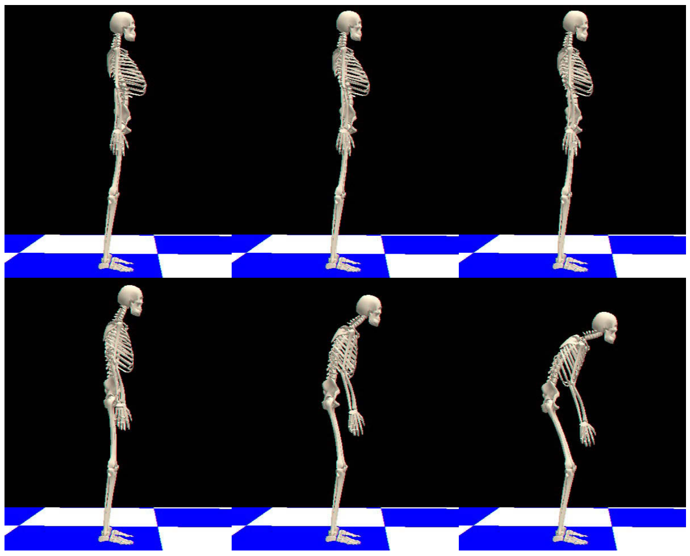


Figure 5.10: A continuous range of postures.

as the sagittal amplitude passes through zero. It is also here that the old man's bent knees and arms begin to be included in the interpolation. Similarly, for the beauty line posture displayed in the second frame of Figure 5.13, the knee that is bent changes as the amplitude passes from positive to negative. These changes are easy to effect with a procedural implementation of the interpolation scheme. Alberts' posture scale and the beauty line posture represent two posture shape sets. Other potential postures include a slovenly posture in which the pelvis is thrust forward and straight spine postures where the abdomen angle is varied, but the spine is held straight.

Four poses generated by different version of the recoil shape set are shown in Figure 5.11. These show different reactions to an object, represented by the red dot. The top two poses show different forms of recoil, indicating apprehension towards the object. The bottom two poses show interest in the object.

It should be noted that the four poses are not simply linear interpolations within a pose space, but each is structurally different. The upper left recoil uses Large C shape classes for the coronal and sagittal planes, opposes the collar movement in both dimensions and adds a weight shift and pelvic Y rotation. The recoil in the upper right uses a Large S coronal class and a Small C sagittal class. The collars are opposed in the

Extreme Posture Vectors		
	Soldier	Old Man
SagClass	SmallS	LargeC
SagAmp	2.2	2.6
Collar Y	align	align
Collar Y Amp	-1.7	1.1
Collar Z	align	align
Collar Z Amp	0.9	-1.2
Ab X	-0.7	0
Arm Length	0 (straight)	0.08
Arm Angle	4 degrees	0
CentreOfMassZ	0.3	-0.2
Knee Bend	0	0.9

Table 5.6: The two extreme postures in Alberts' postures range, denoted here as "soldier" and "old man". The desired value for any parameter not shown is zero.

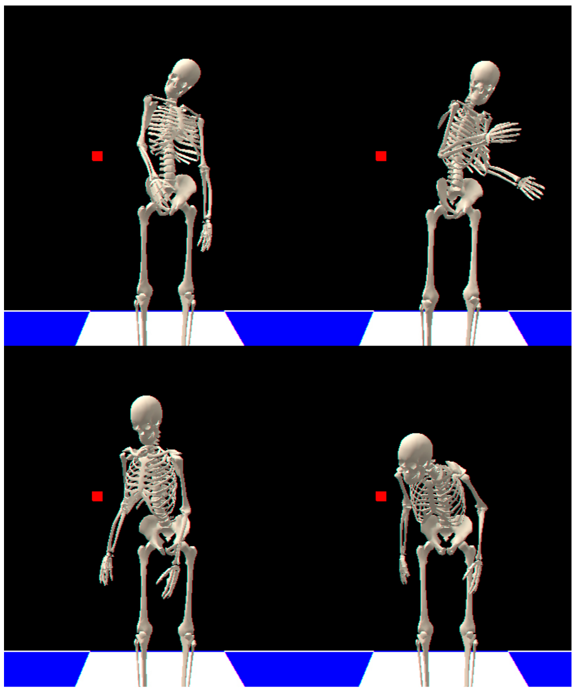


Figure 5.11: Reactions to an object, indicated by the dot.

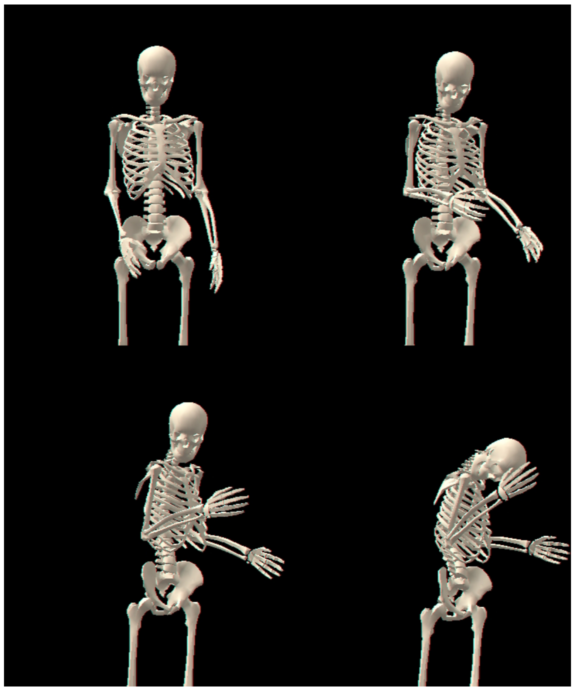


Figure 5.12: Four different intensities of the strong recoil shape set.

Y dimension, aligned in the Z dimension and the weight shift and pelvic Y rotation are increased. The lower left interest shape set uses Large S coronal and sagittal classes with aligned collars in each dimension. Finally, the lower left interest class uses a Large C for the coronal and sagittal shape classes and the collars are aligned in each dimension.

Four different intensities of the strong recoil (upper right in Figure 5.11) are shown in Figure 5.12. Table 5.7 summarizes the parameters that define the shape set. By selecting a shape set and providing a scale parameter, an animator can quickly experiment with both the type of reaction and the vehemence of that reaction.

Figure 5.13 shows four different reaching poses for the same target and initial character position. The first pose is the system's default solution. While it is reasonably "natural", and not surprising for an IK system, it communicates no information about how the character is feeling and hence does nothing to help the animator tell his story. The three subsequent poses are generated based on shape sets. They all feature different relationships with the object and convey more information about the character's state.

Shape Set Parameters for Strong Recoil			
CorClass	LargeS	CorAmp	$3.1if(\theta)$
SagClass	SmallC	SagAmp	$2.7i$
TranAmp	$-3.9if(\theta)$	PelvisY	$0.7if(\theta)$
Left Knee	$0.55 - .05f(\theta)$	Right Knee	$0.55 + .05f(\theta)$
RArmLength	$1.0 - i(0.54 - .18f(\theta))$	CentreOfMassX	$-0.86if(\theta))$
LArmLength	$1.0 - i(0.54 - .18f(\theta))$	CentreOfMassZ	$-0.43i$
Collar Z	align	Collar Y	oppose
Collar Z amp	$2.4i$	Collar Y amp	$-1.7i$
Ab X	$-2.4i$		

Table 5.7: State sets and desired shape vector values for the *strong recoil* shape set. The meaning of θ is the same as above. $f(\theta)$ is an interpolating function that biases the values away from the origin (currently $\pm\sqrt{fabs(\theta)}$).

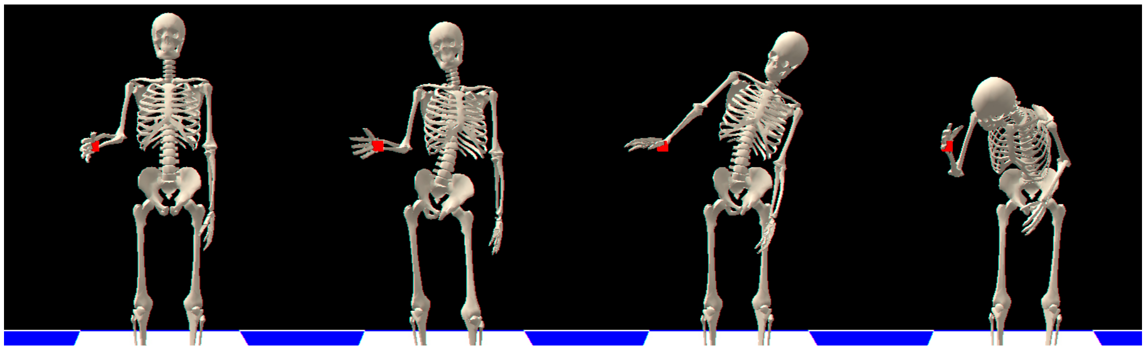


Figure 5.13: Four different reach postures satisfying the same constraint. The last three use shape sets to modify the expressiveness of the stance.

This makes them far more useful in an animation sequence. It also illustrates how an animator can quickly explore pose space by switching shape sets.

Consider the second, "beauty-line" pose in Figure 5.13. It is unlikely that such a pose would be found with a traditional IK or optimization algorithm. The joint angles are intentionally and continuously varied from their central value. Furthermore, in a single kinematic chain, some limbs are rotated toward a goal while others are rotated away from it. While not "optimal", such movements have a clear expressive impact.

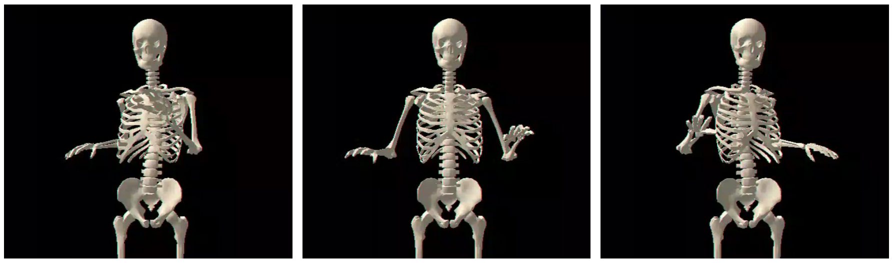


Figure 5.14: Three frames from the basic twist dance show the left and right pose along with an interpolated pose in the middle.

5.7.2 Pose Modification

A simple animation based on the sixties dance “The Twist” is generated by cycling two poses as shown in Figure 5.14. The various edits are applied over multiple repetitions of the dance as shown in Figures 5.15 and 5.16. The corresponding squares in each of these figures is from the same cycle of the twist with the same properties applied. Due to the programmatic representation used to define the edits, it is a straightforward task to vary the intensity of the edits over a movement sequence such as this, allowing the dance to be built up to a wild crescendo, or reduced to a shy bob.

When creating a realistic piece of acting, sometimes a subtle piece of motion is needed to colour a scene. What is called for is often not a broad gesture that would distract from the scene, but a small piece of motion that does not draw attention to itself, but helps to set a mood for a character. These very subtle gestures, while clear in intent in an animator’s mind, are difficult to envision and animate. One effective way to generate them is to take a broad piece of motion and then apply edits to both adjust the flow through succession changes and to drastically scale down the motion using amplitude and extent. We illustrate this in an online video of a twisting dance motion that has been reduced to generate a subtle, but expressive “twitch” that can be applied to a character.

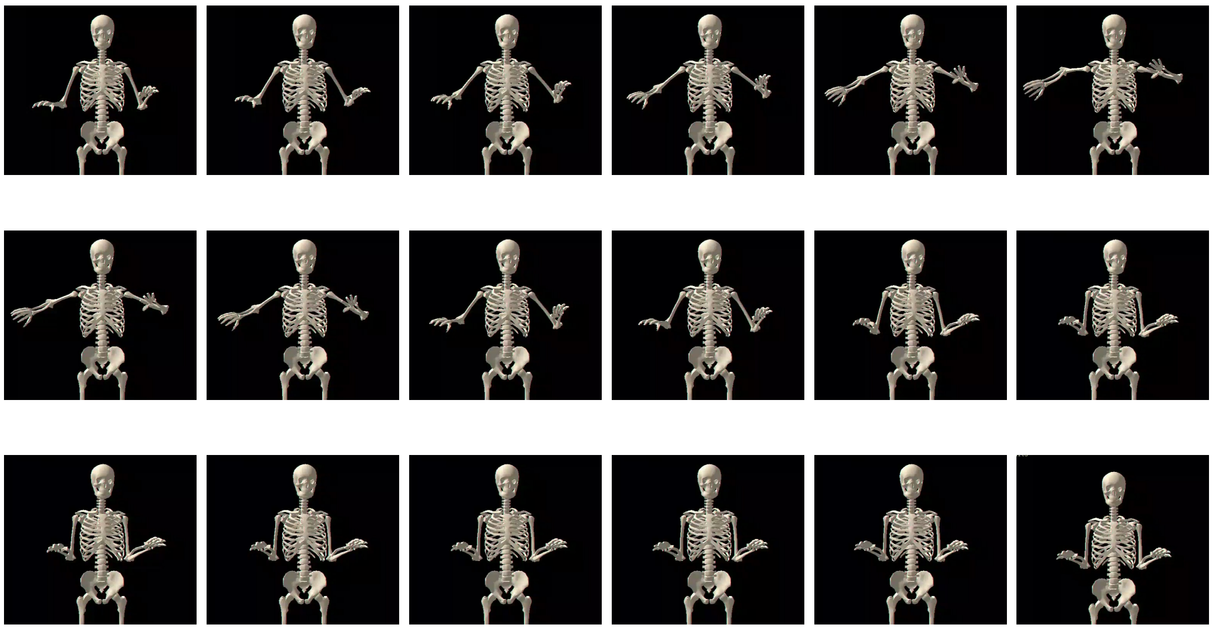


Figure 5.15: These frames are taken from a long twist dance sequence in which amplitude and extent are varied. Each frame is from the middle of the twist motion. In the top row, both the amplitude and extent are increasing as you move from left to right. In the middle row, the extent is being reduced but the amplitude remains the same. In the bottom row, the amplitude is reduced.

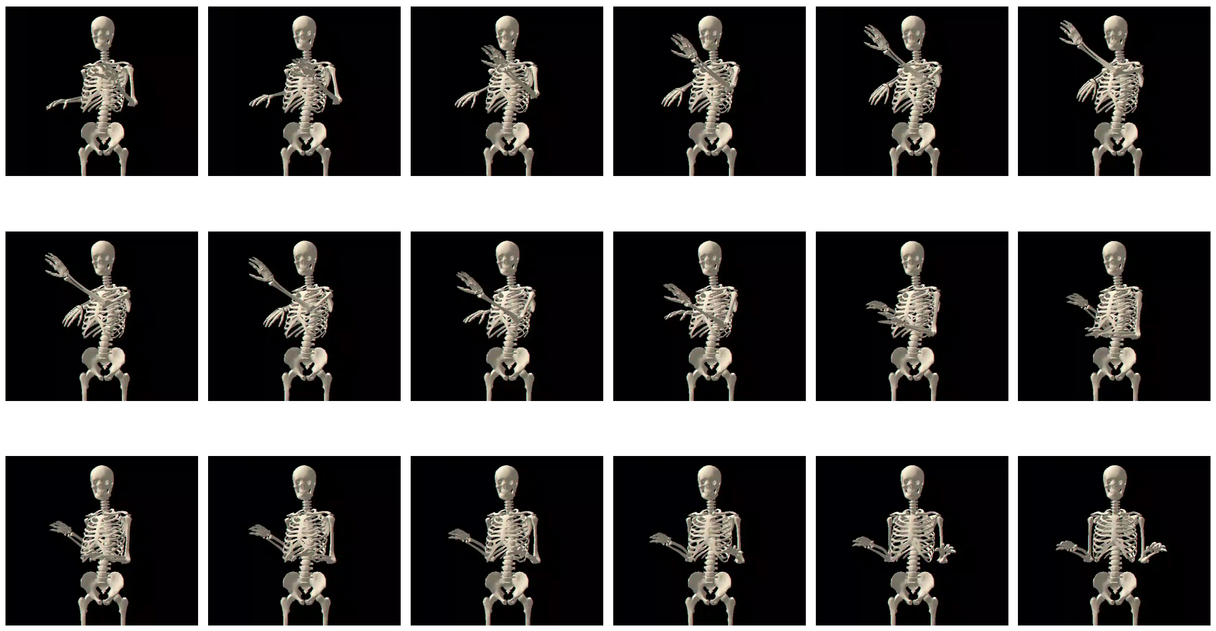


Figure 5.16: These frames are taken from a long twist dance sequence in which amplitude and extent are varied. Each frame is from the right extreme of the twist motion. In the top row, both the amplitude and extent are increasing as you move from left to right. In the middle row, the extent is being reduced but the amplitude remains the same. In the bottom row, the amplitude is reduced.

5.8 Conclusion

A range of methods for specifying shape have been presented, including direct specification of the joint angles, a body shape solver that takes high and low-level aesthetic constraints, and movement properties that manipulate an existing pose. The body shape solver provides an integrated model for body shape, balance and reach constraints. The controls in the solver and the pose manipulation properties have been carefully chosen to align with the aesthetically salient aspects of body shape. We have much to learn from the performing arts on how to communicate emotion and intent through variation in posture. One of the main hopes for this work is that it will make it easier for animators to actively explore the different postures, and different meanings, available when determining poses.

5.8.1 Body Shape Solver Evaluation

Experience with the system has revealed a number of strengths and weaknesses. The low-level controls were carefully chosen based on the arts literature to provide an aesthetically meaningful parameterization. In the practice of an experienced user, they appear to meet this task and allow desired poses to be rapidly created. The parameterization performs a degree of freedom reduction on pose space, simplifying the user's task. Shape sets offer a means for very rapid exploration and experimentation. Since shape sets are defined using the low-level parameters in the system, a pose arrived at through a shape set can be further refined by simply propagating its parameters into the low-level interface. The two level interface thus supports both exploration and refinement. The shape representation is extensible and customizable, allowing the tool to grow over time to better meet an individual animator's needs.

The disadvantages include the limited working range of shape sets and that the system does not prevent self-penetration. Some labour is required to create new shape sets, although it was generally not significant and shape sets could potentially be built from captured poses. More complicated shape sets could be built to cover a very wide range of movements, but at the expense of additional labour. Occasionally, the weights in the optimization system need to be adjusted to achieve a desired pose. This is quite rare and arises when an arm constraint strongly opposes the body shape constraints.

5.8.2 Future Work

We have only begun to explore the expressive uses of balance. Greater use could be made of changes in stance width and the greater range of balance adjustments used in dance would be interesting to explore. It would be worthwhile to extend our lower body IK routine to allow characters to go up on their toes. Adding in hand and other world space constraints would also be worthwhile.

The large number of matrix operations used in the lower body IK routine suggests it might be amenable to implementation in graphics hardware.

It would be interesting to explore other possible pose modification properties.

Chapter 6

Transitions, Tension and Physics-Based Animation

You cannot ... have any conception of the evil that results from muscular spasm and physical contraction.

Constantin Stanislavski, An Actor Prepares, p.91

Transition effects deal with how a character moves from one pose to the next. Important transition effects involve altering the motion envelope and adjusting muscle tension during a movement.

The envelope of a motion refers to how speed is varied over the duration of a motion. This can be visualized by plotting joint position vs. time. Some motions start quickly and end slowly, such as a careful reaching motion. Other motions may start slowly but speed up at the end, like a dabbing motion or a punch. Gentle motions often ease-in toward the end position, while more aggressive striking motions will maintain a high velocity near the end. Motions can be exaggerated by either recoiling backwards before starting the forward path of the motion or overshooting the end position before returning to it.

Some models of movement will have a distinct signature in the motion envelope they produce. For example, when step changes are used with actuated PD control[1], the movement will start very quickly and move slowly towards the end. This is because a large initial error caused by changing the desired value of the joint will generate a large torque to reduce the error, leading to rapid movement. As the error is reduced, the torque is reduced, and the movement slows down. A goal of our work is to allow animators to freely vary the motion envelope as needed to achieve a desired expressive effect.

We provide two methods for warping the motion envelope. The first uses Hermite curve transition functions, the second changes tension during a movement. Both techniques can be used separately or in conjunction. Tension modeling also allows control over other effects, such as controlling pendular motion at the end of a movement or varying how a character reacts to external forces. Some of the results of the tension work have been presented in [113].

There is a strong relationship between transition effects and timing effects. We divide the two property classes based on whether time is the defining aspect of the effect. The motion envelope is sometimes considered to be a timing property, but it has no sense of absolute time. Modulo momentum effects, the motion envelope can be scaled to arbitrary durations, and hence we class it as a transition effect. Other properties like duration, tempo and rhythm are defined by their time attributes.

6.1 Transition Curves

In traditional cell animation, film makers would emphasize the key poses (*keyframes*) by placing more frames close to them and using fewer frames for the in-between motion. Interpolating splines are used for the same purpose in computer animation [88], with the classic ease-in, ease-out curve taking more time near the keyframes and less on the in-between poses.

[1]PD control will be formally defined in Section 6.2.2. For now it is enough to know that PD-control applies torque to a joint based on the error between the joint's actual position and desired position.

An ease-in ease-out curve provides a useful default motion envelope, but is somewhat vanilla, and is only one element in the range of motion envelopes that are important in expressive animation. For instance, a punch will contain a much higher acceleration and a tai-chi move will have an acceleration close to zero and thus a more constant velocity.

In designing our transition curve representation, the following affordances needed to be accommodated:

- A simple ease-in ease-out curve.
- Linear and semi-linear transitions.
- Transitions with an initial high acceleration and long deceleration.
- Transitions with a low initial deceleration and high acceleration at the end of the movement.
- Anticipation effects, where the character moves back before moving forward.
- Overshooting effects, where the character moves past the desired point and then comes back to it.
- A combination of overshoot and anticipation.

This list was built based on the arts literature, and Laban's Effort notation was particularly relevant (Section 4.3.4). These affordances are most easily shown visually and are illustrated in Figures 6.1 to 6.7. The tangent values are for our Hermite formulation as explained in section 6.1.2 below. The reader may wish to pay attention to the shape of the figures for now and review the details of the tangents once the mathematics have been discussed.

6.1.1 Previous Work on Pose Interpolation

Much work has been done on interpolating keyframes. Three previous efforts are particularly relevant.

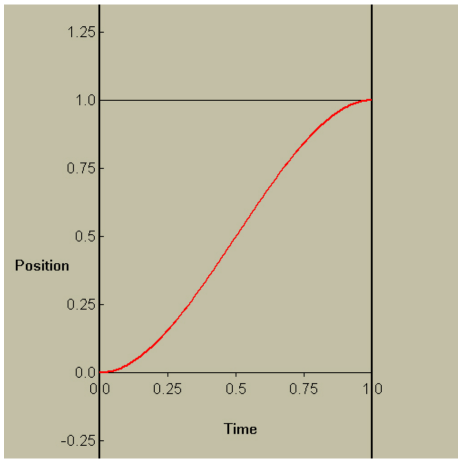
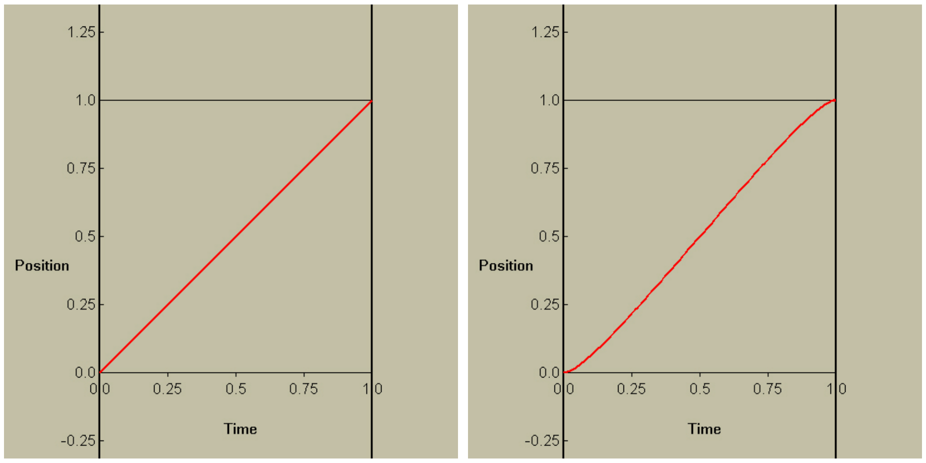


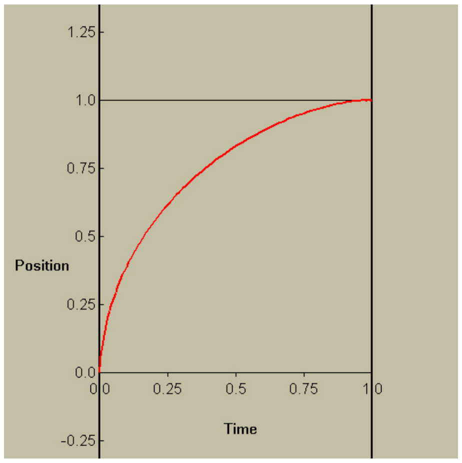
Figure 6.1: A basic ease-in ease-out curve. Tangents 1, 0, 1, 0.



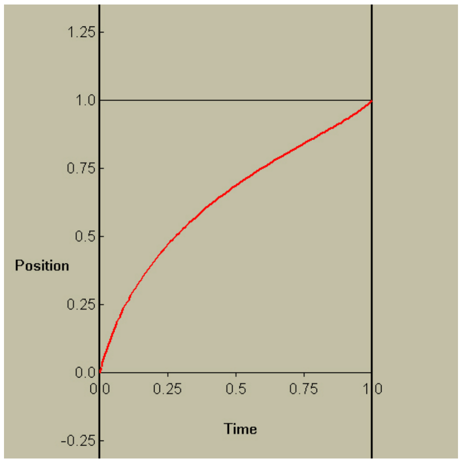
(a) Tangents: 0, 0, 0, 0.

(b) Tangents: 0.4, 0, 0.4, 0.

Figure 6.2: Linear and semi-linear transitions.

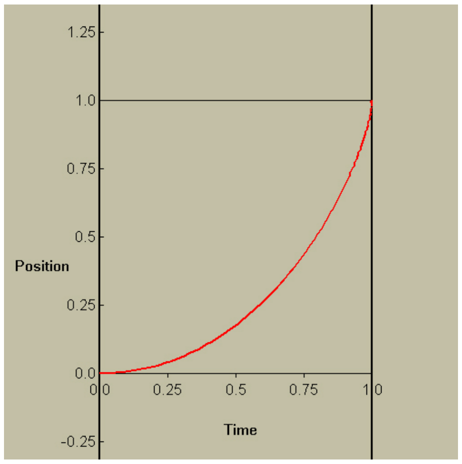


(a) Tangents: 0.1, 2, 1, 0.

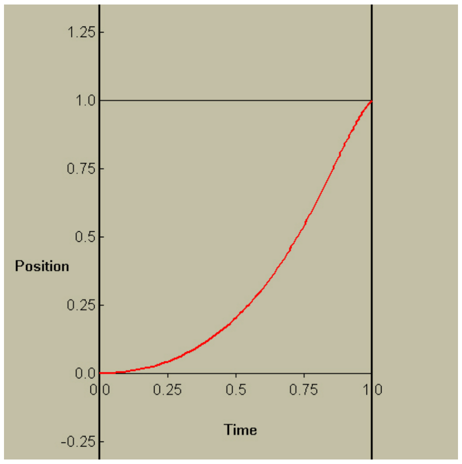


(b) Tangents: 0.5, 2, 0.5, 0.5.

Figure 6.3: Transitions with an initial high acceleration and long deceleration.

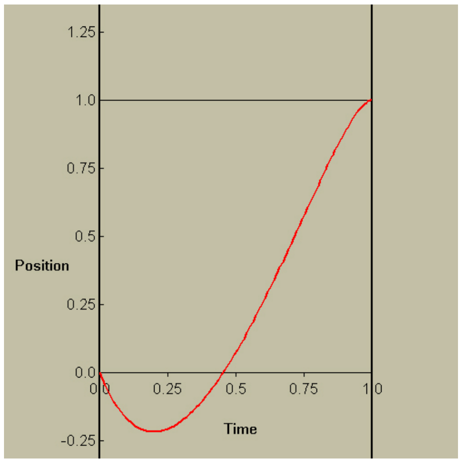


(a) Tangents: 2, 0, 0, 0.5.

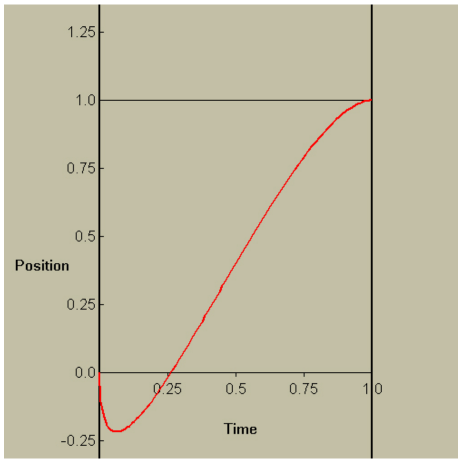


(b) Tangents: 2, 0, 0.5, 0.5.

Figure 6.4: Transitions with a low initial acceleration and high acceleration at the end of the movement.

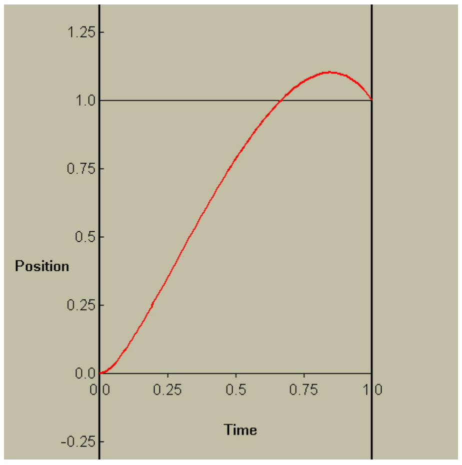


(a) Tangents: 1, -2.5, 0.5, 0.

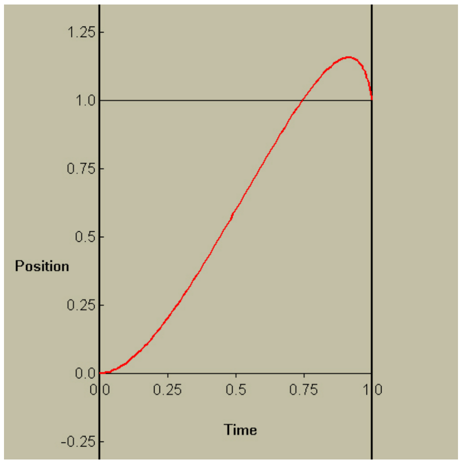


(b) Tangents: 0, -2.5, 1, 0.

Figure 6.5: Anticipation effects, where the character moves back before moving forward.



(a) Tangents: 0.5, 0, 1, -1.5.



(b) Tangents: 1, 0, 0.3, -2.

Figure 6.6: Overshoot effects, where the character moves past the desired point and then comes back to it.

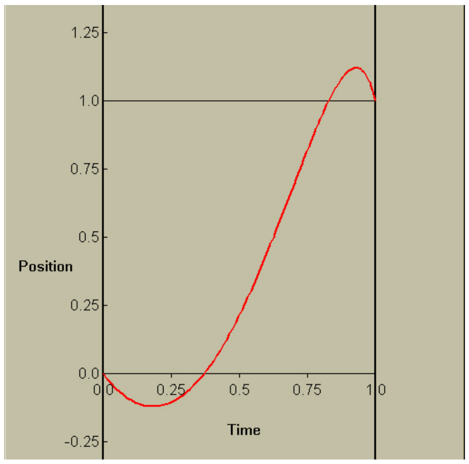


Figure 6.7: Anticipation and overshoot. Tangents: 1.5, -2, 5, -2.

Kochanek and Bartels introduced interpolating splines with local tension, continuity and bias control [82]. Their spline formulation uses the cubic Hermite interpolation functions as a basis. A spline segment is set between each key point to be interpolated. These segments are defined by their two endpoints and a tangent at each endpoint. Kochanek and Bartels use the tension, continuity and bias parameters to vary the tangents at each interpolated point. Tension controls how sharply the curve bends at a point. The tension parameter acts to vary the length of the tangents on either side of a point. Reducing the lengths to zero will create a cusp, increasing the lengths generates more slack in the curve. The continuity parameter controls whether the tangents on either side of the point are aligned or not. The further the tangents vary from being aligned, the more discontinuous the animation will appear at the point. The tangents are determined by averaging the chords on either side of the point. The bias term allows one chord to be weighted more heavily than the other and can be used for overshoot effects.

Steketee and Badler introduced a double interpolant method, using cubic B-splines as the interpolant [151]. The B-Spline formulation ensures second order continuity. They use a separate interpolant for time and space. The time or kinetic interpolant describes the time of keyframes with no reference to position. The position interpolant describes how position changes as a function of keyframe. In their EMOTE system, Chi et al.

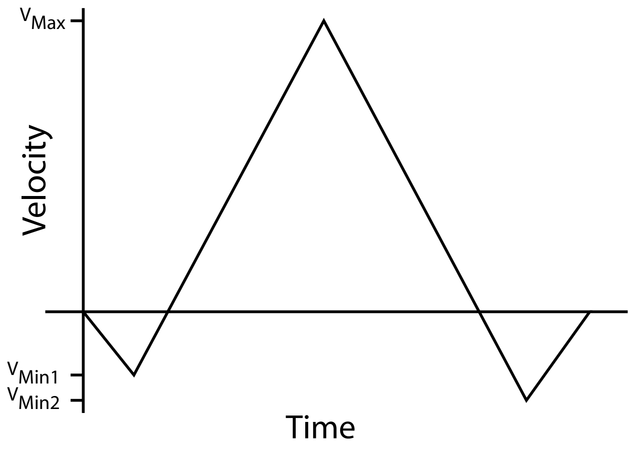


Figure 6.8: Velocity profile used by Chi [35].

use a double interpolant approach which is a combination of the above approaches [36]. They control the path curvature through space using the tension parameter from the formulation by Kochanek and Bartels. Focusing on reaching motions, they interpolate one of the end-effector position, joint angles or elbow position. For timing, they define a transition function between 0 and 1 that has zero velocity at the two end points. As shown in Figure 6.8, there is an initial constant deceleration to a negative velocity, followed by a constant acceleration to a positive velocity, followed by another constant deceleration to a negative velocity, and finally a constant acceleration to zero velocity. By shifting the position of the maximum velocity, they can warp the transition curve to either emphasize acceleration or deceleration. They can also include anticipation and overshoot effects.

6.1.2 Chosen Formulation

Three guidelines were used to select a transition function formulation: it had to satisfy the desired affordances, it should involve a small number of parameters and it should be easy to understand. A parametric cubic Hermite curve embedded in time and space was selected.

The Hermite basis consists of four cubic polynomials. Following [51], the componentwise Hermite blending function is:

$$H(u) = (2u^3 - 3u^2 + 1)P_0 + (-2u^3 + 3u^2)P_1 + (u^3 - 2u^2 + u)R_0 + (u^3 - u^2)R_1 \quad (6.1)$$

where P_0 is the starting point, R_0 is the starting tangent, P_1 is the end point and R_1 is the end tangent. Since we normalize both time and position to $[0, 1]$, the equation can be simplified to:

$$H(u) = -2u^3 + 3u^2 + (u^3 - 2u^2 + u)R_0 + (u^3 - u^2)R_1. \quad (6.2)$$

Finally, we represent our time and position space curve as the tuple (t, p), defined as:

$$(t, p) = (H_t(u), H_p(u)). \quad (6.3)$$

A transition function is thus defined by four values: the initial and final tangents of p and the initial and final tangents of t. In script files, we specify this as:

2DHermite t_{R_0} p_{R_0} t_{R_1} p_{R_1}

The same transition functions are used to specify the desired value for a DOF as well as the actuator gains for that DOF, which will be described below.

At each time step, the system must calculate either gain or position values, which are given as the *position* value on a space curve. This value is determined by the current time. First, the actual simulation time is normalized for the current transition to a value t in the range $[0, 1]$. Equation 6.3 gives:

$$t = H_t(u). \quad (6.4)$$

Given t, we can solve for the parameter u, using an analytic solution for the roots of a cubic. Provided the tangents are defined such that the transition curve does not fold back on itself (i.e. it should behave like a function, having only one value for any $t \in [0,1]$), a simple and necessary condition, there will be only one root in the interval $[0,1]$. This root is u, which can then be substituted into $H_p(u)$ to yield p. Note: for some tangent values, the transition function becomes quadratic or linear. These are solved for respectively by using the quadratic formula or finding the root of the line.

When performing physical simulation, the transition functions are used to calculate a torque at each time step. This torque creates an acceleration that must then be twice integrated in order to update the positional value of the DOF. This double integration adds two degrees of continuity to the final position curve, so for G^0 continuous input, the final motion will be C^2 continuous. Continuity is hence not a concern when generating physically simulated motion. As will be discussed below, low tension movements will further "smooth" or add additional slack to the transition curves.

Interestingly, in practice continuity was never a concern when generating kinematic animation either, even though only G^1 continuity is guaranteed. A significant deviation from C^1 continuity would only occur when requested by the animator, and did not appear objectionable in these cases.

6.1.3 Transition Function Property

The property SetTransitionFunction is used to control the shape of the transition function. It takes as input the four tangent values associated with the two Hermite curves and either an *ABS* or *AVG* flag. The *ABS* flag will cause the property to override any lower level transition functions, using the current property's tangent values. The *AVG* flag will cause tangent values to be averaged.

6.1.4 Offset Curves

Sometimes additional perturbations of the transition functions are desired. For example, someone who is very old or has Parkinson's disease may shake slightly during movements.

Such perturbations are accomplished by the use of an offset curve which is added to the main transition function. For example, the shaking behaviour can be accommodated by adding a high frequency sinusoid to the transition frame.

6.2 Torque Generation and Physical Simulation

One advantage of dynamic simulation is that it places the animation in the same framework as real motion, generating the same set of constraints and helping to create the secondary effects that would naturally occur. In a forward dynamics approach to physics based animation, the fundamental problem is to generate the necessary torques at each DOF so that the character successfully completes desired movements. This is also referred to as the *control problem*, since the required torques over time must be resolved and applied.

A control strategy can be divided into three levels. There should be an organizing structure, a torque generation strategy and a low level system to actually generate the correct torques. The organizing structure coordinates the activity of various limbs to achieve a movement. *Actions*, with associated SetDOFAngle properties, perform this role in our system. The torque generation strategy refers to how the necessary torques will be generated (e.g. using feedback, as impulses etc.). Our system is based on the *equilibrium point hypothesis* which we discuss below. Finally, the low-level system, or actuator, is responsible for applying the correct torques. We develop an antagonistic actuator which is a variant of PD-control and represents a simple approximation to muscle.

There is biological motivation behind our choices at each level of the control strategy. The relevant background will be reviewed next, followed by a detailed account of our control strategy, then a discussion of the physical simulation process.

6.2.1 Biological Inspiration

Biological inspiration can guide both how actions are organized and how low-level control is achieved. The following three sections will detail control organization, strategies for torque generation and muscles as biological actuators.

Motor Programs

In human movement, if the brain is directly involved in controlling a movement, there are long feedback paths by which a signal must be sent to the muscles, feedback must return from the muscles to the brain, and then new signals must be sent to make adjustments. This introduces latency caused by the transfer time of each signal. A complicated action such as a punch involves the activation of the tricep, followed by its deactivation and activation of the bicep, followed by its deactivation and activation of the tricep again. In such rapid, complicated actions there are too many degrees of freedom for a high level executive function to manage in the allotted time for the action [145]. Schmidt and Lee [145] suggest that one hypothesis used to explain how humans deal with this limitation is that instead of direct control, the executor invokes a motor program that then controls the individual degrees of freedom. Motor programs are open loop processes made up of a series of instructions that coordinate joint behaviour. Motor programs are believed to be used for fast actions where there is no time for sensory feedback or for actions where sensory feedback is not required. When sensory feedback does come into play it can operate to alter a motor program or trigger a new one, but there will be some lag involved. Motor programs are responsible for grading, timing and coordination of muscular activities required for skilled behaviour [175]. David Zeltzer championed the idea of using motor programs as a building block for controlling animated characters [178].

Detractors of the motor program theory argue that the brain does not have the capacity to store a motor program for every action people can perform and that motor programs do not provide a method for novel actions to arise, even though people can perform some movements that they have never tried before. The concept of generalized motor programs was developed to respond to these concerns [145]. Generalized motor programs are essentially parameterized version of motor programs that can cover a range of similar movements.

Certain properties of generalized motor programs are invariant, including the order of events, the temporal structure or relative timing of various events and the relative force produced by various muscles [145]. Amplitude may also be invariant [175]. Parameters include an overall duration and overall force [145]. The degrees of freedom controlled by a generalized motor program are not fixed. For example, the similarity between people's writing done on a piece of paper using the wrist and fingers, and their writing on a blackboard using their upper arm has led researchers to propose that the same generalized motor program is being used for both tasks [145].

Some researchers have suggested hierarchical representations are used for motor programs [101, 50]. More complex activities are built out of simpler muscle synergies that coordinate the activity of small groups of muscles. It has also been suggested that motor control consists of short term, specific action plans that depend on the state of the system and long term, more abstract representations related to the general properties of a motor action [50].

While we have made no explicit attempt to precisely implement motor programs, they nonetheless provided an important early inspiration for the action representation in our work. Like motor programs, our actions provide coordinated control of a set of joints; although they specify a fixed set of joints, which generic motor programs may not. Actions also control the relative timing of events, as do motor programs. Most significantly, both representations are open loop. Our actions are pre-planned, stored and executed, with no feedback during the execution of the motion. In other words, there is no monitoring and correction during a movement. While this seems to be required for certain actions in biological systems, in our work it represents a simplifying assumption and allows us to explore what movements can be effectively modeled without corrective feedback.

For kinematic simulation in a static world, there will be no difference between the movement plan and the final motion. A motion will always precisely hit its desired target if the target is achievable. In dynamic simulation, or if interaction targets in the world are moving in ways that are not precisely known, there may be some error between the plan and the actual movement – either the movement may deviate from the plan or the target may have moved to a different spot than was anticipated. These types of deviations suggest that for some high precision movements, such as touching a particular location on an object or catching a moving object, the character may need to either monitor a movement during execution and make adjustments or complete the movement and make a second, corrective movement. Again, this appears to be consistent with biological movement. In aiming motions, people have an initial ballistic phase followed by a feedback based correction phase to refine the final position. The *optimized initial impulse* model which is used to explain Fitt's law is based on this idea [143]. Feedback is normally visual. The idea of adding in a monitoring and correction process to fine tune dynamic movements is an interesting area for future research. For the case of balance, where feedback did prove necessary, we included corrective adjustments in the system through the use of reactive controllers as discussed in Section 7.5.

Torque Regulation

Motor programs provide an organizing principle, but the problem of actually generating the movement remains. Again, inspiration can be taken from the natural world by examining the operation of muscles. As a first approximation, muscles behave like tunable springs. They are organized in agonist and antagonist groups that work in opposition around a joint, generating opposing torques. In the absence of external forces, the final position of the limb will be the equilibrium point at which the internal muscle forces are balanced, yielding zero net torque. This is the basis of the hypothesized *equilibrium-point model* of muscle control, originally proposed by Anatol Feldman in 1966 [48] for single joint systems. The model suggests that the central nervous system controls the position of limbs by adjusting the relative inputs to the muscles thereby varying the equilibrium point of the limb [41, 75]in contrast to models that directly control the amount of torque each muscle generates. The spring-like mechanics of the musculature will then effectively pull the limb towards the new equilibrium point. By not needing to provide muscles with a particular torque requirement, the work required by the brain is greatly simplified as the inverse dynamics problem is avoided [50]. The brain only needs to deal with kinematic trajectories, which presumably map to specific activation levels. According to Flash, several researchers have suggested that the equilibrium position can be determined from torque/angle curves for the agonist and antagonist muscles [50].

Early on it was thought that neural activity brought a rapid shift to the new equilibrium point, but experimental evidence showed this was not the case.

> Experimental observations showed that the path between two points exhibited a measurable degree of stability, not just the end-point. Thus, the production of the movement appears to be accomplished by a progressive movement of the neurally-defined equilibrium posture, which has been termed a *virtual trajectory.* [73, p. 162]

In addition, even at equilibrium, the position of the limb will in general vary from the neurally defined “equilibrium” position, due to the presence of external forces such as gravity, so the neurally defined virtual tranjectory will differ from the actual trajectory of the limb [74, 75]. This has implications for how equilibrium point control can be applied in practice, given that we wish to control the actual trajectory of a limb: the calculated virtual trajectory must be the set point trajectory that will yield the desired actual trajectory for a limb.

It should be noted that while the equilibrium-point model is well supported, there is evidence that suggests that muscle activation patterns in tasks are based on torque requirements, not directly on joint kinematics [3]. The opposing theory in biomechanics to the equilibrium-point hypothesis is known as the *impulse timing model.* It suggests that motor programs directly generate short impulses of torque to control joint movement, varying control by adjusting the length and magnitude of these impulses [145]. There is experimental evidence supporting both theories. There seems to be good agreement that the equilibrium-point hypothesis at least accounts for how limbs reach their final position [145]. Our work is compatible with the equilibrium point hypothesis less because it is a plausible control model for human motion than that it provides a convenient computational approach for producing nuanced synthetic motion. We explicitly include gravity and external forces in our calculations when deciding how to adjust our muscles so that the resulting equilibrium will accommodate all the forces acting on the limb and yield the desired position.

Recent work has characterized the motor control mechanism as the linear superposition of force fields, where the output behaviour of the motor control system is to organize this superposition to attain a desired stable posture[111]. This theory unites the inverse dynamics hypothesis and the equilibrium point hypothesis as these force fields can be seen as primitives the central nervous system can use to solve the inverse dynamics problem and a virtual trajectory can also be seen as a combination of these primitives. Other work has sought to characterize the nature of the basis functions that form these primitives [38].

Muscle Models

Now that we have settled on motor programs as an organizational framework and equilibrium point-control as a muscle activation strategy, the final step is to decide upon a muscle model to use that will generate the actual torques at each DOF. In developing the equilibrium-point hypothesis, the basic spring like behaviour of muscles was observed. Various muscle models will now be briefly described.

Muscles are complex entities, having significant biological, chemical and mechanical properties. Different models are used for different applications. Research in biophysics and biochemistry uses A. F. Huxley-type microscopic models [176]. These models include biochemical properties, modeling how contractions are actually generated. Biomechanics researchers rely almost exclusively on the three component Hill model [121, 176]. This model contains a Contractile Element (CE) and two non-linear elastic elements: the Parallel Elastic Element (PEE) and the Series Elastic Element (SEE). For their human jumping model, Anderson and Pandy modify the basic Hill model by adding an additional elastic series element to represent the tendon [5]. Muscle properties biomechanists normally model include the force-length relationship, the force-velocity relationship and activation dynamics [121] or neural input [73].

The spring-like nature of muscle is significant and an important contributor to postural stability.

> It also turns out that for stable equilibrium the neuromusculoskeletal system must possess the attributes of springs, and that for stability, these springs must exceed a certain critical stiffness. The passive, relaxed person is inherently instable at many levels – when someone faints, the head falls if the shoulders are supported, the torso falls if the body is supported at the pelvis, and the whole body falls if the person is freely standing. Certainly the central nervous system is partly responsible for this behaviour, but the primary mode of its influence can be considered to be adjusting muscle 'spring-like' behaviour. [6, p.384]

It is worth noting that, our system displays such instability when low spring stiffnesses are used with our actuator. The details are presented below.

The muscle model employed in our work is a simple antagonistic actuator that is similar to PD-control. This computationally simple model has a small set of parameters and captures the spring-like nature of muscles. Agonist and antagonist muscles can coactivate around a joint, which is captured by the actuator. The main impact of coactivation is to increase joint stiffness [73], which can be modeled using PD actuators or our antagonistic formulation. There are a number of differences, however, between real muscles and PD controllers or antagonistic actuators. Muscles are linear force generators whose torque arms, based on the point of attachment, change as limbs move [177]. This means that their torque generating capabilities vary throughout a movement. PD or antagonistic controllers are used to apply torque and there is no guarantee that the torques are within human limits [121]. A muscle's force output increases as it is stretched, which provides shock absorption, stabilization and damping of movement [121]. Some muscles span more than one joint and this may facilitate control [121]. Muscles are active objects that can continually generate force, whereas PD-controllers only generate force to correct errors [73]. Muscle use has a cost in terms of metabolic energy to which there is no equivalent in our virtual characters who never get tired.

Despite the number of differences between antagonistic control and real muscles, it is unclear how significant a perceptual impact these differences will have on multi-joint motion. This, combined with the simplicity and intuitive parameterization of antagonistic control, makes it a good candidate as the primary actuator in our system.

6.2.2 Equilibrium Point Control

The torque generating strategy employed in our system is *equilibrium point control*, based on the ideas of the equilibrium point hypothesis and virtual trajectories developed in biomechanics and summarized above. In order to further motivate this approach, it is important to first understand the shortcomings of straight PD control.

Probably the most common actuator used in physics-based animation has been the proportional derivative (PD) controller. PD control can be thought of as a spring and damper arranged in parallel and written as:

$$\tau = k_s(\theta_{desired} - \theta) - k_d\dot{\theta}, \tag{6.5}$$

where τ is the resulting torque, θ is the current angle of the DOF, $\theta_{desired}$ is the desired angle of the DOF, $\dot{\theta}$ is the velocity of the joint and k_s and k_d are the spring and damper gains respectively. Since here the value being tracked, θ, refers to joint angles, PD-control is a linear *angular* spring and damper.

When a control strategy such as PD control is used, there will in general be an error between the actual position of the joint and the desired position, even at steady state. This is due to the influence of gravity, which is pulling the joint away from the desired angle. The equilibrium point will be the angle at which the proportional term generates enough torque to balance the torque generated by gravity. (For an intuitive picture, imagine a rock hanging from a real spring. The amount that rock stretches the spring from its rest length is the error we are discussing here.) This same phenomenon occurs in the equilibrium point hypothesis used to characterize human movement and is why the *virtual trajectory* may differ from the actual trajectory, even at steady state [75].

Two strategies are commonly used to deal with this error. Spring tensions and set points can be carefully hand tuned to avoid excessive error while maintaining reasonable tension levels. Alternatively, there is a tendency when using PD control to reduce steady state and tracking error by increasing the proportional gain. This can lead to very stiff characters. Assuming gravity is the only relevant external force, when PD control is at equilibrium, there will be no damping force, and the net torque will be zero:

$$0 = k_s(\theta_{desired} - \theta) - \tau_G, \tag{6.6}$$

where τ_G is the torque due to gravity acting on the limb. Thus the error is given by

$$error = \theta_{desired} - \theta = \frac{\tau_G}{k_s}. \tag{6.7}$$

If the error is to be reduced by increasing the gain, the result will necessarily be overly stiff characters. Consider a 5kg solid cylinder, such as an arm, that is 60 cm long and rotated around a pin joint at one end. The gain to keep it within 3 degrees of horizontal, representing angles as radians, is approximately 280. To keep it within 0.5 degrees, the required gain is 1680. These large gains can cause the character to exceed the torque producing capabilities of real humans. They also override most momentum effects that would occur from normal arm movement, generating very stiff looking motion.

A relaxed, loosely moving character should have low gains, which means that there will be a large steady state error between the set-point of the PD controller and the actual position of the joint. It is unwieldy for an animator to try to anticipate the amount of error and provide a desired set point that will compensate for this error. For this reason, our system allows the animator to specify directly the actual angle for the joint and the desired stiffness. The final position of a joint will be the equilibrium point of all the internal and external forces acting on it. The system estimates the external forces and calculates the gains appropriately in order to realize the animator's requested angle with the requested amount of tension. This is what we refer to as *equilibrium point control*. The animator is given control over intuitive parameters: the actual angle and actual tension, and the system determines the gains and set points that will meet these constraints. Explicitly dealing with the external forces allows for accurate position control. The animator can use a transition function to specify how a DOF will change over time and a virtual trajectory will be calculated that will track that function.

Currently our system accounts for external forces due to gravity, but it is straightforward to add other forces. For instance, the mass of an object the character is carrying could be added, so the character could use the appropriate force to lift it. This allows room for some fun effects as well. If a character were "told" the object is heavier than it is, he would use too much force and jerk the box up quickly. This is the same behaviour humans exhibit when they overestimate the weight of an object they are about to lift. Making the character aware of external forces is equivalent to giving him or her a limited tactile sense.

6.2.3 Antagonistic Formulation

In place of classical PD control, we employ an antagonistic actuator. As discussed below, this is in fact a reformulation of PD-control, although interpolating the control parameters of the antagonistic actuator will not yield the same results as interpolating the control parameters of a PD-controller. This is discussed in Section 6.2.5.

Antagonistic control is implemented for each DOF using two ideal (linear) angular springs that are set in opposition to each other, and a damper that acts in parallel as shown in Figure 6.9. Unlike PD control, each spring has a fixed set point that remains constant throughout an animation. These set points are placed just past the joint limits for the DOF. The control law is written as

$$\tau = k_L(\theta_L - \theta) + k_H(\theta_H - \theta) - k_d\dot{\theta}, \tag{6.8}$$

where τ is the torque generated, θ is the current angle of the DOF and $\dot{\theta}$ is its current velocity. θ_L and θ_H are the low (L) and high (H) spring set points which serve as endpoints for the motion, k_L and k_H are the corresponding spring gains, and k_d is the gain on the damping term. The tension T or stiffness of the joint is taken as the sum of the two spring gains:

$$T = k_L + k_H. \tag{6.9}$$

The desired angle is not explicitly shown in the actuator, but is controlled implicitly by the value of the two gains.

Joint movement is achieved by varying the equilibrium point of a joint, which is the point at which all forces acting on the joint sum to zero. This is done by adjusting the gains on the two springs. At an equilibrium position, both the damping and the net force will be zero. The torque on the joint is then

$$0 = k_L(\theta_L - \theta_{eq}) + k_H(\theta_H - \theta_{eq}) + F, \tag{6.10}$$

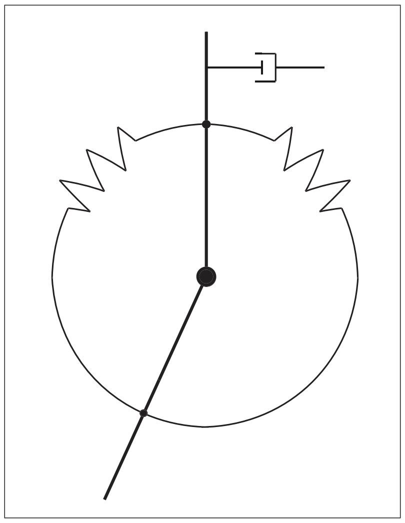


Figure 6.9: Located at each DOF, an antagonistic controller consists of two angular springs and a damper.

where θ_{eq} is the equilibrium point, and F represents gravity together with all other known external forces acting on the system. For any given equilibrium angle, the gains that will achieve it all lie on a line in k_L-k_H gain space given by

$$k_H = k_L \, \frac{\theta_L - \theta_{eq}}{\theta_{eq} - \theta_H} \;-\; \frac{F}{\theta_H - \theta_{eq}} \; . \tag{6.11}$$

This line is termed an *isoangle* line. Notice that the slope of the line depends only on the limits and the desired angle, and that the external force acts to vary the y intercept of the line. Also, because θ_{eq} is bounded, the slope of the line will always be positive. Since we want the springs to only act in tension, gains are only chosen from the positive quadrant of the k_L-k_H space. This insures that the springs in the model, as with real muscle, can thus never "push".

We draw particular attention to the linearity of the isoangle relation, and that it depends only on the gains, current joint angle and joint limits. This implies that the tension on any joint, given initial gains, can be adjusted without needing to know the external forces. An example set of isoangle curves is shown in Figure 6.10, illustrating the simple structure of the isoangle space.

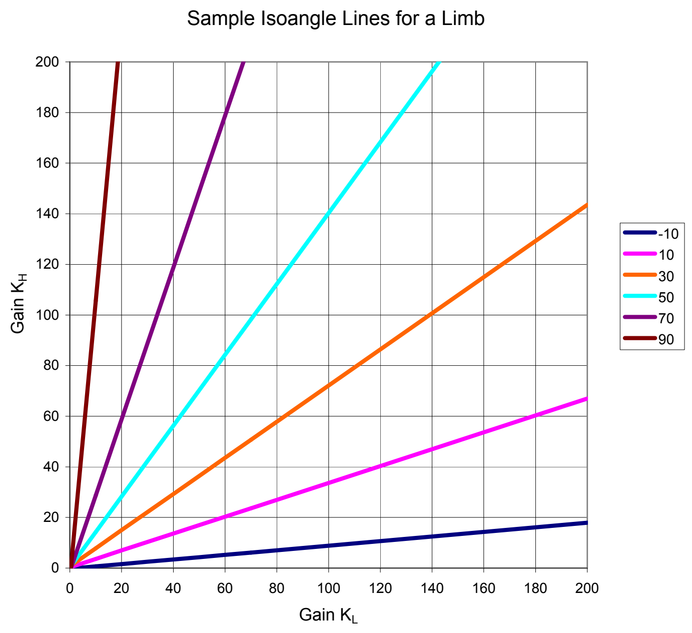


Figure 6.10: Isoangle lines spaced twenty degrees a part for a limb under antagonistic control. The set points on the two springs are −20 and 100. All angles are in degrees.

The antagonistic formulation reflects the organization of real muscles. This approach would remain suitable even as more complex muscle models are introduced. The formulation permits easy tension adjustment by moving along the linear isoangle line. It also makes explicit that the rest angle will necessarily be an equilibrium point determined by the spring gains.

6.2.4 PD Formulation of Equilibrium Control

Since the current version of antagonistic control uses linear springs, there is a mathematical equivalence between PD control and antagonistic control. For equivalent values, they will show exactly the same response to error. This equivalence is shown below. This does not imply, however, that interpolating control parameters in PD-space will have the same effect as interpolating parameters in antagonistic space, and indeed, as will be discussed below, it does not.

Equilibrium point control can be implemented using a PD actuator by rethinking the

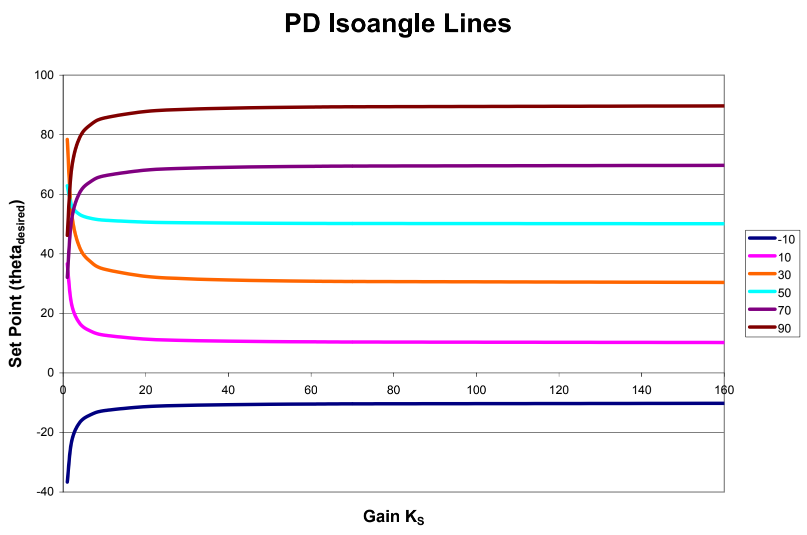


Figure 6.11: Isoangle lines spaced twenty degrees a part for a limb under PD control.

meaning of the desired angle set point in the PD control law. Instead of using the set point as the desired angle, the set point must be treated as a free parameter that is varied so that at equilibrium, the angle of the joint will be the angle requested by the animator.

The isoangle line for PD control is a function of stiffness and set point given by

$$\theta_{eq} = \theta_{desired} - \frac{\tau_G}{k_s}. \tag{6.12}$$

The isoangles thus lie on hyperbolic functions of the two terms. The shape of these curves depends on the external forces. The more complicated isoangle space is shown in Figure 6.11.

An advantage of antagonistic control is that a person authoring a controller that must deal with an external disturbance is given two intuitive options when using antagonistic control: the spring acting in the direction of the disturbance can be relaxed to reduce torque in the disturbance direction or the spring opposing the disturbance can be tightened to create a torque to resist the disturbance. Tensing or relaxing a joint to deal with a disturbance are the two options humans use, when for instance making balance adjustments (e.g. [49]). Achieving tension and force changes with PD control is less intuitive, and can include moving the set point from one side of the actual equilibrium point to the other.

Both antagonistic and PD-control have two free parameters that work together to determine the final equilibrium position. In antagonistic control, both parameters are spring stiffnesses that act together to define tension as described in equation 6.9. With PD-control, the parameters are a tension value and a set point. Antagonistic control allows any disturbance to be resisted by simply increasing the tension of the joint by moving outwards along an isoangle line in order to reduce the error. This joint stiffening behaviour has been observed experimentally in humans [73]. Increasing the stiffness will not generally work for PD control, as both the stiffness and set-point will need to be adjusted together. To make this clear, consider a limb under loose PD control that at steady state is leaning below its set point due to the influence of gravity. If a force is applied to the limb from below, it will move the limb closer to the set point. Increasing the gain on the PD controller will not restore the arm to its original location, and instead will pull it closer to the set point. Provided the disturbance was not strong enough to knock the arm past the set point, increasing the stiffness of the spring has exactly the wrong effect: it moves the limb further in the direction of the disturbance, away from the desired equilibrium point and closer to the set point. To generate the desired behaviour, the set point and tension must be moved in tandem along the isoangle line in PD space.

The convergence under disturbance of antagonistic control to the desired angle as tension increases can be shown more formally. Let θ_R be the requested position. The initial joint gains and slope for the tension isoline are calculated based on θ_R. Substituting equation 6.9, which defines tension, into equation 6.11 and assuming the equilibrium angle is the requested angle gives the following relation for the isoangle line:

$$k_H = (T - k_H)\frac{\theta_L - \theta}{\theta - \theta_H} - \frac{F}{\theta_H - \theta} \tag{6.13}$$

$$k_H = T\frac{\theta_L - \theta_R}{\theta_L - \theta_H} + \frac{rmg\sin\theta_R}{\theta_L - \theta_H} \tag{6.14}$$

where we are considering the case of a single limb of length r and mass m and the frame of the limb has the same orientation as the world frame. Now consider θ_A to be the actual value of the DOF when some disturbance is applied to the joint. Substituting equation 6.9 into equation 6.10 yields the following relation governing steady state behaviour.

$$0 = (T - k_H)(\theta_L - \theta_A) + k_H(\theta_H - \theta_A) + rmg\sin\theta_A + F \tag{6.15}$$

$$0 = T(\theta_L - \theta_A) + k_H(\theta_H - \theta_L) + rmg\sin\theta_A + F \tag{6.16}$$

where F is some additional disturbance torque acting on the limb. Substituting equation 6.14, which defines the isoangle line, into equation 6.16 yields

$$\theta_A = \theta_R + \frac{-rmg\sin(\theta_R) + rmg\sin(\theta_A) + F}{T} \tag{6.17}$$

Now note that as T goes to infinity, the oscillations in θ_A tend to amplitude zero, and thus approaches θ_R in the limit ($\lim_{T\to\infty}(\theta_A) = \theta_R$). Thus increasing tension along an isoangle line will act to reduce the impact of any disturbance acting on the limb.

Similar behaviour can be achieved with PD-control. To resist a disturbance, the stiffness and set point should be moved along the isoangle curve. At equilibrium

$$0 = k_{S0}(\theta_{S0} - \theta_R) + F, \tag{6.18}$$

and thus by rearranging terms,

$$k_{S0} = \frac{-F}{\theta_{S0} - \theta_R}. \tag{6.19}$$

Here θ_{S0} is the initial setpoint, θ_R is the requested value, and k_{S0} is the initial gain. If the stiffness is to be increased by a factor c such that

$$k_{S1} = c k_{S0}, \tag{6.20}$$

where k_{S1} is the new gain, then

$$\frac{-F}{\theta_{S1} - \theta_R} = c \, \frac{-F}{\theta_{S0} - \theta_R} \tag{6.21}$$

$$\theta_{S1} = \frac{\theta_{S0}}{c} + \theta_R - \frac{\theta_R}{c}. \tag{6.22}$$

Together, equations 6.20 and 6.22 define the update rules for the free PD-control parameters. As should be expected, as c is increased, the set point will converge to the desired joint value. In a manner similar to the one used with the antagonistic formulation above, it can be shown that as c is increased, any disturbance will decrease and the actual value will converge on the requested value.

Equivalence of PD and Antagonistic Control

Both PD and antagonistic control are linear in θ and $\dot{\theta}$ and hence are mathematically equivalent. This equivalence can be shown by multiplying out the two equations, including external forces, and equating the terms. The net torque with PD control is given as

$$\tau_{net} = k_s \theta_{desired} - k_s \theta - k_d \dot{\theta} + F, \tag{6.23}$$

and the antagonistic control equation becomes

$$\tau_{net} = k_L \theta_L - k_L \theta + k_H \theta_H - k_H \theta - k_d \dot{\theta} + F. \tag{6.24}$$

The damping term and external force terms are the same. Equating the θ terms from both equations gives

$$-k_s\theta = -(k_L + k_H)\theta, \tag{6.25}$$

$$k_s = k_L + k_H = T. \tag{6.26}$$

Equating the set point terms gives:

$$k_s\theta_{desired} = k_L\theta_L + k_H\theta_H \tag{6.27}$$

$$\theta_{desired} = \frac{k_L\theta_L + k_H\theta_H}{T} \quad \text{subst. eq. 6.26 .} \tag{6.28}$$

Together these relations allow any antagonistic formulation to be converted to an equivalent PD formulation.

6.2.5 Different Behaviour of PD and Antagonistic Formulations

Moving a limb with either PD-control or antagonistic control requires interpolating control parameters in a two dimensional space between two isoangle curves. With PD-control, the parameters are the set point and spring gain. With antagonistic control, they are the two spring gains. Perhaps surprisingly, when tension changes are present, interpolating in each of these control spaces will yield different results. This can be seen by examining the equilibrium behaviour of a joint as it is moved between two values in either formulation.

Let us consider first the PD formulation. Recall that its equilibrium position θ is defined as:

$$\theta = \theta_D + \frac{F}{k_s} \ . \tag{6.29}$$

where θ_D is the set point of the controller. In order to move a limb, we must change control parameters between two isoangle curves. This can be represented in parametric form as:

$$\theta = (\theta_{D1} - \theta_{D0})u + \theta_{D0} + \frac{F}{(k_{s1} - k_{s0})u + k_{s0}} \tag{6.30}$$

where $u \in [0, 1]$ represents the progress of the transition and the numeric subscripts represent the initial and final isoangle curves. The transition in antagonist gain space can be expressed similarly. At equilibrium,

$$\theta = \frac{k_L\theta_L + k_H\theta_H + F}{k_L + k_H} \ . \tag{6.31}$$

A transition can thus be represented in parametric form as:

$$\theta = \frac{[(k_{L1} - k_{L0})u + k_{L0}]\theta_L + [(k_{H1} - k_{H0})u + k_{H0}]\theta_H + F}{(k_{L1} - k_{L0})u + k_{L0} + (k_{H1} - k_{H0})u + k_{H0}} \tag{6.32}$$

where once again, u is the parameter.

In order to compare these two relations, they need to be written in terms of the same variables. For this we can use the equivalence between the two formulations shown in the last section in equations 6.26 and 6.28. We will express the two relations in terms of starting and ending tension, T_0 and T_1, and starting and ending set points, θ_{D0} and θ_{D1}. The interpolation in antagonistic space thus becomes (using Eq. 6.28):

$$\theta = \frac{(T_1\theta_{D1} - T_0\theta_{D0})u + T_0\theta_{D0} + F}{(T_1 - T_0)u + T_0} \ . \tag{6.33}$$

In a similar manner, the equation for interpolation in PD-space becomes (using Eq. 6.26 and forming a common denominator):

$$\theta = \frac{(T_1\theta_{D1} - T_1\theta_{D0} - T_0\theta_{D1} + T_0\theta_{D0})u^2 + (T_1\theta_{D0} - T_0\theta_{D0} + T_0\theta_{D1} - T_0\theta_{D0})u + T_0\theta_{D0} + F}{(T_1 - T_0)u + T_0} \ . \tag{6.34}$$

Three important points should be observed about these two relations. First, they are identical to each other if $u = 0$ or $u = 1$. Second, they are identical if $T_0 = T_1$. Third, they will yield different equilibrium points when the first two conditions are not met. In practice, what this means is they will generate different warpings of the motion envelope in response to tension changes.

Figure 6.12 shows a plot of the two relations for starting and ending set points of .379 and 2.62 radians respectively. Increasing tension transitions go from 100 to 400, decreasing tension transitions from 400 to 100 and constant tension transitions remain at 100. The external force is for a single 5 kg limb that is 60cm long and is acted on by gravity. It is characterized by $F = mrg\sin\theta$ Notice that tension changes with antagonistic control generate a clear warping to the motion envelope. This has proved to be expressively useful.

6.2.6 Instability at Low Gains

In the presence of gravity, certain low gain configurations can be unstable. This is because the torque due to gravity is proportional to $\sin\phi$, where ϕ is the worldspace angle of the limb (0 is up) and this can lead to multiple equilibrium points in the range $[-\pi, \pi]$. Consider a limb controlled by PD control at equilibrium. (The situation is necessarily the same for antagonistic control, but the PD analysis is easier to follow.) The net torque acting on it will be zero as given by

$$0 = k_s(\phi_{desired} - \phi) + rmg\sin\phi, \tag{6.35}$$

where r is the distance from the joint to the centre of mass of the limb, m is the mass of the limb and g is the acceleration due to gravity. This equation represents the sum of a linear spring force and a sinusoidal gravity force. For small values of k_s (low gains), the slope of the line will be small compared to the magnitude of the sine function. For a given gain and $\phi_{desired}$, there may be multiple values for ϕ that make the equation zero.

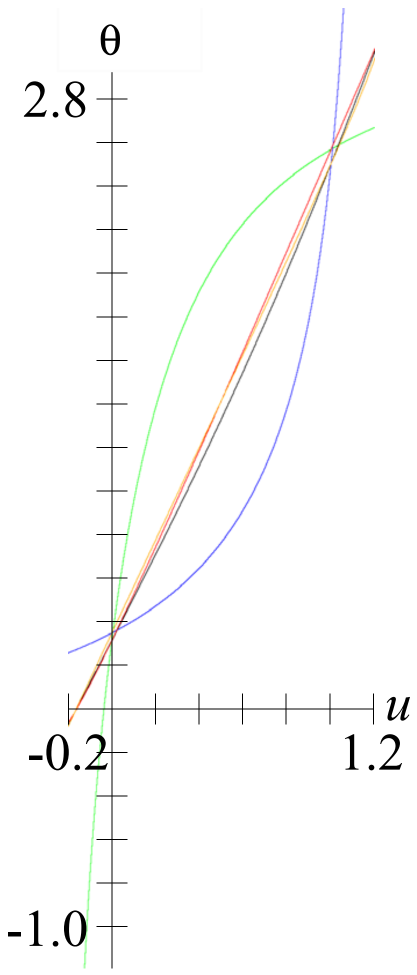


Figure 6.12: A comparison of tension changes with antagonistic and PD control. u is the transition parameter and θ is the equilibrium point of the limb. The green curve (top) represents an increase in tension with the antagonistic formulation, blue (bottom) a decrease in tension with the same formulation and black (middle) is constant tension in both formulations. The red and orange lines (middle) show an increase and decrease in tension respectively with PD-space interpolation.

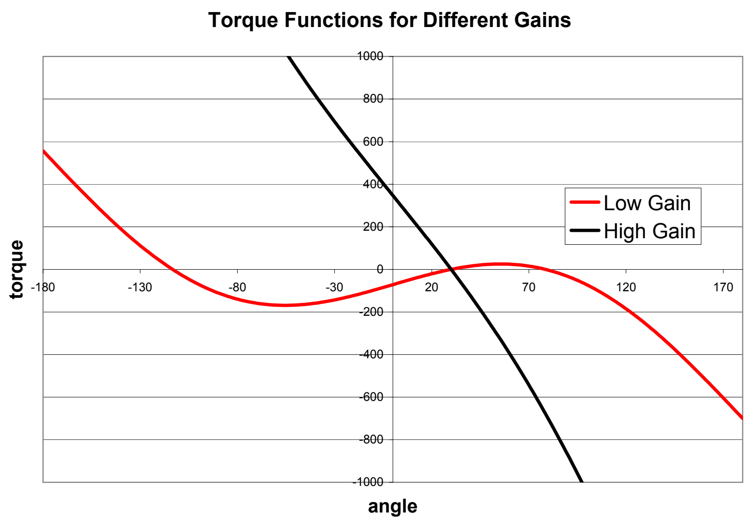


Figure 6.13: Comparison of the net torque for a limb given high and low gains: $\tau = k_s(\theta_{desired} - \theta) + rmg\sin\theta$. There are three zero torque equilibrium points for the low gain case.

In other words, there may be multiple equilibrium points within the limits of the joint. This behaviour is shown for low and high values of gain k_s in Figure 6.13. Notice, that for low gain, there are three equilibrium points where the torque on the joint will be zero. For high spring stiffness k_s, there is only one possible equilibrium point.

If the limb is at an equilibrium point that is higher in world space than another equilibrium point, the limb will fall to the lower equilibrium point given any slight disturbance. This of course is also a feature of the antagonistic representation, and is also characteristic of people, who would be unstable with low muscular "spring stiffness" [6]. The isoangles curves for both PD and the antagonistic control cross at low values of gain where there is instability. Figure 6.14 shows this behaviour for the antagonistic controller case. It is possible to test for unstable low gain configurations and thus avoid them by increasing the gain, but it is important for users of the system to be aware of this phenomenon.

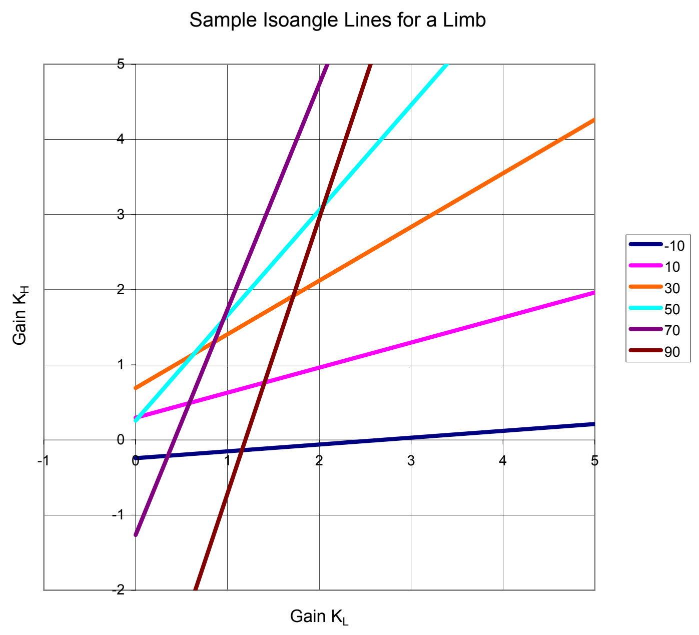


Figure 6.14: At low gains, the isoangle lines will cross. This leads to instability. The example shown here is for antagonistic control.

6.2.7 Setting Spring Gains

Given the control law and the transition functions that are to be tracked, the question remains as to how to update the gains of the springs to achieve the desired motion. The update process must take into account the external forces acting on the character to affect equilibrium control. This is done in two ways in the system: by sampling or by prediction, and they can be used together for different DOFs.

The sampling method takes samples of the forces acting on the figure at a given frequency, converts them to torques and recalculates the two actuator gains so that the equilibrium point at the sample time will be the angle indicated by the transition function. This approach is most appropriate for situations where forces may change in unexpected ways or if the force model used for a given external force does not lend itself to predictive use. In our implementation, sampling is used most for joints in the lower body that will experience significant variation in ground reaction forces, whereas the predictive approach is used for free upper body movements.

The predictive approach determines a set of gains that match the starting state of the pose, a set of gains that will achieve the end state of the pose and then simply transitions between the two sets of gains using the transition function interpolant to achieve the desired motion. The starting gains can be the gains that are currently active in the system, or they can be recalculated to adjust the character's stiffness. The amount of tension can also be varied during a motion.

Recall that the system described here is built on top of the DANCE framework which was designed to support physics-based animation. Within this framework, all forces acting on a character are applied by *actuators*. As detailed in Section 7.7, there are actuators that apply gravity, ground reaction forces, external forces like shoves, and muscle forces. For free upper body movements, the most significant external force will be gravity. The character's state at the end of the motion is estimated based on the transition functions stored in the BRep. The actuators can then be queried to provide an *a priori* estimate of the force they will apply to the character at that time. Linear forces can be converted to torques using the *sdequivht* function which is provided by SD/FAST. The torque compensation is quasi-static as it does not include inertial forces.

There are arbitrarily many ways to make a transition between the two lines in k_L-k_H space. The path chosen will affect the shape of the resultant motion. By default, a transition path is chosen that keeps the joint stiffness, T, constant. Recall that $T = k_L + k_H$. Using the standard slope-intercept equation for a line, $y = mx + b$, a constant tension line plotted in k_L-k_H gain space as follows. If

$$k_H = mK_L + T$$

then

$$k_H = mk_L + k_L + k_H$$

meaning that

$$0 = k_L(m + 1). \tag{6.36}$$

Therefore,

$$m = -1 \text{ for } k_L \neq 0. \tag{6.37}$$

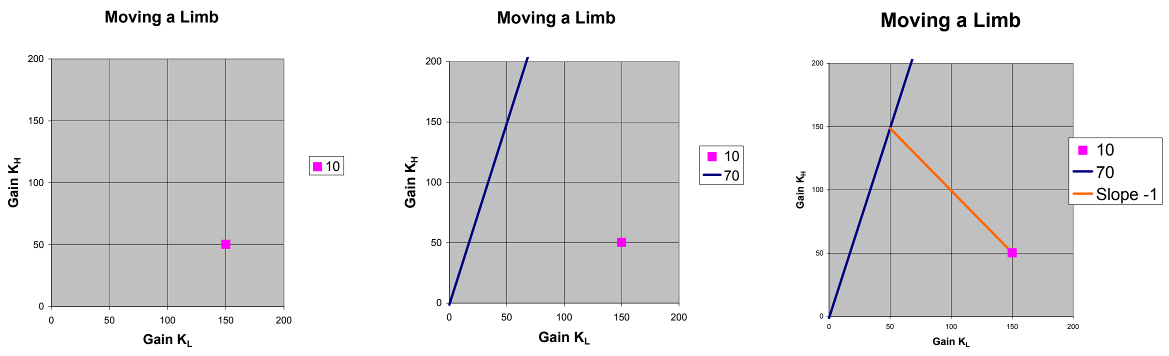
The constant tension line must therefore have a slope of -1. Movement along this line ensures that for sufficiently high gains, if the transition line is traversed at constant speed, the resulting motion will be linear, as is demonstrated below. In other words, a constant tension transition will accurately track the input transition function for sufficiently high gains.

The process of setting the gains is shown in Figure 6.15. First the starting gains for the joint are determined. In the example, they lie on the ten degree isoangle line and are shown with a pink dot. The isoangle line corresponding to the desired end value is calculated based on the torque estimates for this time. A line with slope -1 is then drawn between the starting point and the new isoangle line. During the motion, gain values are determined by interpolating along this line. The rate of progress along the line is determined by the transition function. This process produces gains over time that will generate a resulting motion that has the same shape as the transition function. As tension is increased, the provided trajectory will be more accurately tracked.

The position along the line in gain space is determined by the *position* component of the transition function, which is normalized to $[0, 1]$ [2]. Details on how this value is calculated are provided in Section 6.1.2. Once determined, the position value must be scaled based on the length of the transition line in gain space and then indicates the progress along the line.

The same approach is used for transitioning ball joints represented by quaternions, except that the progress between the end points must be scaled. If the progress between

[2]Note, for overshoot and anticipation effects, the position may move before 0 or past 1.



(a) Initial gains. (b) Final isoangle curve. (c) Transition line.

Figure 6.15: Starting with an initial point in gain space corresponding to the current value and stiffness of the DOF, the isoangle line for the desired value of the DOF is calculated. The gain values are updated by moving from the initial point to the calculated isoangle line along a line of slope -1.

the start and end (i.e., zero and one) is represented by p and the angle between the two end quaternions is θ_{span}, the update rule is

$$p = \frac{\sin(p\theta_{span})}{\sin(\theta_{span})}. \tag{6.38}$$

Due to the estimates made for the end state, small errors may occur in the end position if the estimates are not accurate. As discussed above, this is consistent with at least some forms of human behaviour such as reaching, where humans often have to make small corrections to position at the end of a motion.

Linear Transitions

It was claimed above that for significantly high tension, a linear movement along a line of slope -1 in k_L-k_H gain space will lead to a linear change in the value of the controlled DOF. This will now be formally shown. An initial point in k_L-k_H space is indicated by (L_0, H_0). A line of slope -1 passing through this point can be represented parametrically as:

$$k_H = H_0 + u \tag{6.39}$$

$$k_L = L_0 - u, \tag{6.40}$$

where u is the parameter determining progress along the line. Note that by definition, $H_0 + L_0 = T$.

The value of the DOF, θ, at equilibrium is governed by equation 6.10. Considering a single limb being acted on by gravity and substituting the relations for k_H and k_L above gives the following derivation:

$$0 = k_L(\theta_L - \theta) + k_H(\theta_H - \theta) + F \tag{6.41}$$

$$0 = k_L(\theta_L - \theta) + k_H(\theta_H - \theta) + rmg\sin\theta \tag{6.42}$$

$$\theta(k_L + k_H) = k_L\theta_L + k_H\theta_H + rmg\sin\theta \tag{6.43}$$

$$\theta = \frac{(L_0 - u)\theta_L + (H_0 + u)\theta_H + rmg\sin\theta}{L_0 - u + H_0 + u} \quad \text{subst. eq. 6.40} \tag{6.44}$$

$$\theta = \frac{(L_0 - u)\theta_L + (H_0 + u)\theta_H + rmg\sin\theta}{L_0 + H_0} \tag{6.45}$$

$$\theta = \frac{u(\theta_H - \theta_L)}{T} - \frac{L_0\theta_L + H_0\theta_H}{T} + \frac{rmg\sin\theta}{T}. \tag{6.46}$$

Note as well that the range of u is proportional to T. Specifically, if one is to make a maximal length move along the transition line from one set point to the other, u will cover the interval $[0, T]$. As T is increased, the range of u is proportionally increased. On the other hand, $rmg\sin\theta \leq rmg$ and does not vary as T increases. This suggests that as the tension in the movement, T, increases, the first term of equation 6.46 will dominate and

$$\theta \propto u. \tag{6.47}$$

Thus for sufficiently high tension, the change in the value of a DOF will be proportional to the amount of movement along a line of slope -1 in k_L-k_H space.

6.2.8 Physical Simulation

In the preceding sections, our method for determining joint torques has been described. At each time step in the simulation, the torques are applied and used to generate joint accelerations. These are twice integrated to update the position of each DOF in the character. The equations of motion for an articulated figure are based on classic Newtonian mechanics. We use a commercial package, SD/FAST, to generate simulation code [76]. The input to SD/FAST is a description file of the articulated figure to be modeled. For each limb, it specifies the length, mass, moment of inertia, type of joint at the limb's root and connectivity information. SD/FAST produces C-code that implements an order(N) formulation of the equations of motion, where N is the number of DOFs.

Two integration schemes are used in the system. A fourth order Runge-Kutta integrator is provided by SD/FAST and is the default integrator. It is somewhat slow, but stable with small time steps. A semi-implicit Rosenbrock integrator with a variable step size was implemented based on [132]. If the step size in the Rosenbrock integrator becomes too small, the integrator will revert to RK4 for a few time steps to move past a difficult portion of the integration. This integrator does not perform well when there are discontinuities as the foot monitor points make and break contact with the ground. Therefore, it is not used with the free standing full-body. The integrator does allow larger time steps and faster performance, and is the default integrator for the torso only model and full body model when using “sticky ground”. To give an idea of performance, it takes about four seconds of computation time per second of simulation time for the full skeleton with the sticky ground floor model and Ronsenbrock integrator running on a 2.2 GHz Xeon.

6.3 Aesthetic Aspects of Tension Control

Modeling tension has several important aesthetic impacts on the nature of motion. Tension control:

- Provides intuitive control over the transition envelope of a motion.
- Allows variation of end effects after a transition, such as controlling overshoot, or creating pendular motion when a character brings his arm to his side.
- Controls how precisely the character tracks an input control trajectory.
- Varies a character's reactions to external forces.
- Controls how force is transferred between joints.
- Varies the influence of gravity and momentum on movements.

In addition, dynamic simulation acts as a low-pass filter, smoothing the control input. This input regulates torque over time, which is used with Newton's laws to calculate the angular acceleration at each DOF. This is twice integrated to update the DOF values. Provided the control input is continuous, which it always is in our system, the resulting DOF curves will naturally be C^2 continuous.

6.3.1 Warping the Motion Envelope

The importance of the transition envelope was discussed in Section 6.1. Our system provides two intuitive ways to warp the motion envelope and these can be used together. The first is to vary the transition curve as discussed above and the second is to vary the tension over a movement. This combination has proven to be very flexible and powerful in practice. An animator can precisely control a movement when that is required. Often however, it is sufficient to provide a very simple trajectory and a satisfying shape can be achieved by specifying a tension change that should occur during the motion. It appears easier to generate a desired motion by varying the tension parameters than by trying to anticipate the correct transition curve. While we have not found evidence in the

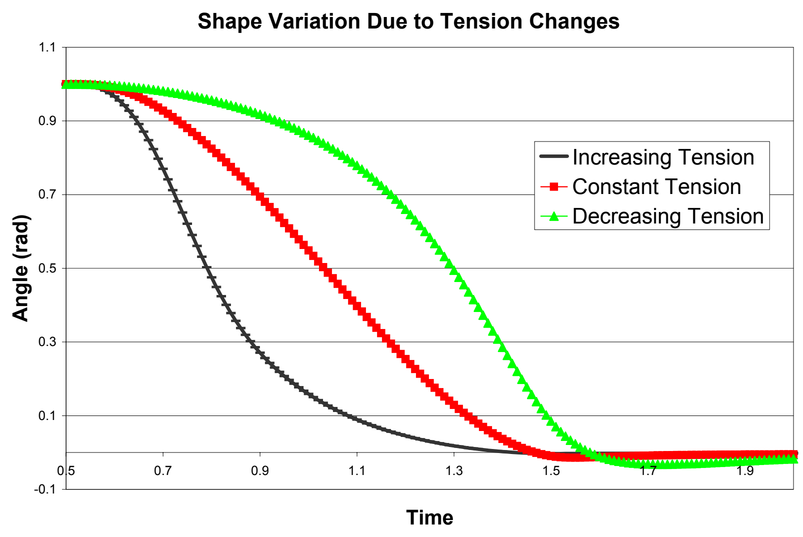


Figure 6.16: Varying tension during a motion warps the motion's shape. The trajectory function is an ease-in-ease-out curve.

biomechanics literature as to whether or not the warping induced by tension changes when using the antagonistic formulation is consistent with human movement, it does appear correct when used to generate animations. Numerous examples have been generated where a desired motion envelope is achieved by specifying an appropriate tension change. This is appealing because movement is a very natural activity for people and they can easily develop an intuition about the tension changes they make during a movement. Trajectory curves are a more abstract concept to work with.

An animator can provide tension values for the start and the end position for each DOF in a pose. These are used when the start and end gains are calculated for the motion.

Lines in gain space having a slope of other than -1 will induce a warping on the trajectory as discussed in Section 6.2.5. Transitioning from a high gain to a lower gain will cause the motion to start slowly and accelerate towards the end. Moving from a low gain to a high gain will cause the motion to accelerate quickly at the beginning and ease in towards the end. An example of this is shown graphically in Figure 6.16. It is in this way that tension changes warp the motion envelope. Often this warping is enough

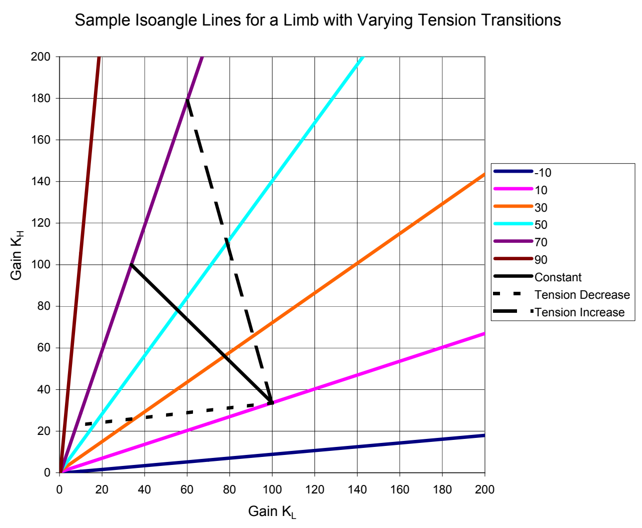


Figure 6.17: Isoangle lines for an antagonistic actuator with transition lines displayed in black over top. The three transition lines correspond to a tension decrease, constant tension (slope -1) and a tension increase.

to provide the desired shape for the motion, using either a linear transition function or simple ease-in ease-out curve as the initial transition function. Posture changes are especially well suited to modeling with this method. The animator is still free to use arbitrary transition functions to warp the motion if additional control is needed.

The explanation for this warping follows directly from the antagonistic parameterization as shown above. Figure 6.17 shows the isoangle curves overlayed with three different transition lines: one for increasing tension, one for constant tension (slope -1) and one for decreasing tension. Notice that as one follows each of these transition lines, the spacing of the isoangle lines vary. Specifically, for a tension decrease, most of the isoangle lines are crossed at the end of the transition line. The opposite behaviour occurs for a tension increase. This induces a warping on the rate at which the DOF value is varied as the character transitions from the starting to ending value.

If additional control is required, the animator can specify a more complex transition function. Once again, the transition functions control the rate of progress along the transition line in k_L-k_H space.

It is also possible for the animator to request a tension change without the accompanying warping. In this case, the transition function is used to determine the desired angle at each time step. Given the desired angle, the joint limits, an estimate of the external torque and the requested tension value, the necessary gains can be calculated from equations 6.9 and 6.10. Although supported, this method is rarely used in practice because the motion warping caused by the tension change is normally aesthetically consistent with the reason the tension was change was requested.

6.3.2 Accuracy of Trajectory Tracking

With high gains, the resulting motion will track the transition functions closely. As gains are reduced, the momentum of the limbs starts to contribute more strongly to the motion and the trajectory deviates from the transition function. Figure 6.18 shows examples of both downward and upward movements, each at two different gains. The high gain curve precisely tracks the kinematic transition function. For the downward motion, the lower gain function at first lags behind the high gain function as it takes longer to overcome the inertia of the limb. During the middle section of the movement, the limb leads the kinematic curve under the effect of gravity. The two motions reach the end point at approximately the same time, but the lower gain motion slightly overshoots the desired position. This overshoot is often visually desirable. The results are similar for an upward motion. Once again the high gain motion precisely tracks the kinematic transition function. The lower gain motion again exhibits a lag at the beginning and an overshoot at the end. Since gravity is now acting against the motion, the lower gain transition trails the motion throughout.

6.3.3 End Effects

Tension changes will affect the behaviour of a limb at the end of a motion. As discussed above, low tension at the end of a movement will lead to overshoot, whereas high tension will bring a more precise stop to a motion. Varying tension during a motion can be used

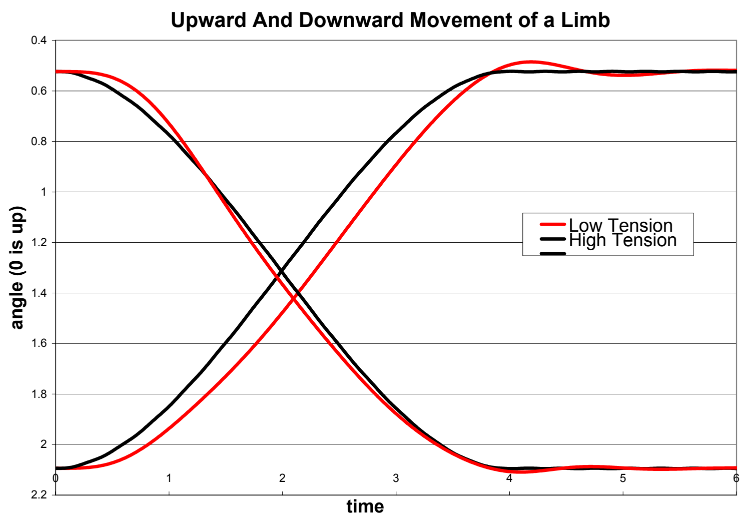


Figure 6.18: Upward and downward motions for a limb comparing high and low gains.

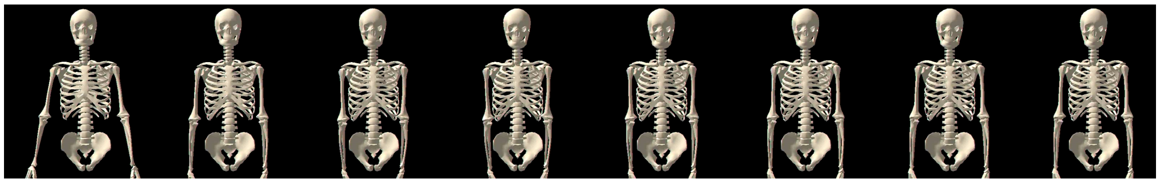


Figure 6.19: A very loose character brings his arms to his side. There is some pendular motion before the arms come to rest. The spacing between frames is 0.13 sec. This is subtle effect is very difficult to appreciate in still frames, but reads strongly when seen in the animation.

to achieve desirable pendular effects at the end of a motion. For instance, an arm gesture may start by being tightly controlled and end in low tension to allow the arm to sway loosely at the character's side. Figure 6.19 shows a character bringing his arms to his side under low tension. Note the pendular sway in the motion. This is a much stronger effect in the animation than in the still frames. It should be noted that simply lowering tension at the start of the movement is not effective. This would create an unnatural drop in the arms, whereas a gradual reduction in tension shows controlled movement at the beginning and loose swinging at the end of the motion.

In some situations it is useful to keep joint tension loose throughout a motion and to not balance for external forces. A limb controlled in such way will act passively in response to the rest of the body's movement. For instance, an animator may wish to loosely hold the character's wrist straight, and allow gravity and momentum to cause it to sway from side to side during a movement. An example of passive wrist movement is shown in Figure 6.23.

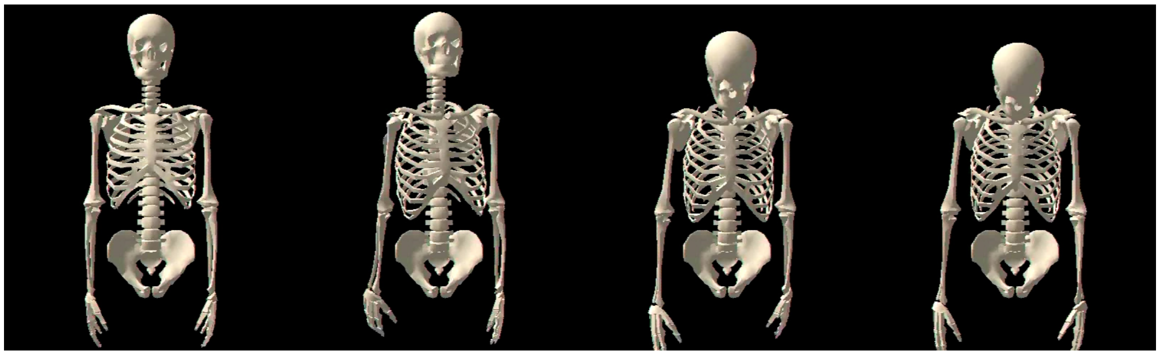
6.3.4 Force Transference

Joints that are less stiff will also be more affected by the movement of adjacent limbs. The antagonistic controllers do not compensate for these transference forces, so the effect the forces have depends solely on the tension in the neighbouring joint. Transitory effects will become much more apparent as the character relaxes and this can yield richer movement. A very tense character will behave more like a rigid body, whereas a loose character will be more fluid.

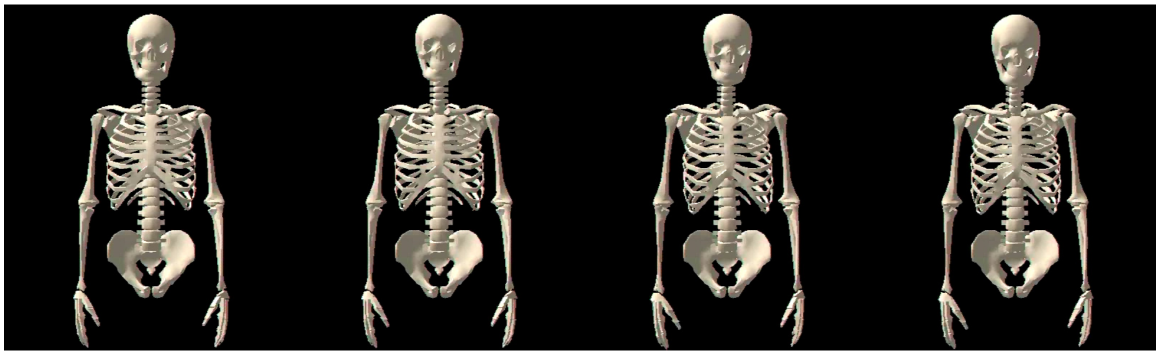
Figure 6.20 shows several frames from two animation sequences of a character being hit on the back shoulder. In each case, the same force was used for the shove. In the top row, the character was very loose. Notice the large deformation that occurs. In the bottom row, the character had tensed before impact. The deformation is almost not visible in the still frames, although there is clear movement in the animation. The tensed character behaves much more like a rigid body. (Note, the difference in the base posture was specified by hand to emphasize the idea of the character "bracing" for impact.)

6.3.5 Calibration

In an authoring system, it is important to give an animator sufficient control over the motion in order to achieve the effects she desires. This control is not useful, however, if it is unintuitive. While deciding between an increase or decrease in stiffness is normally simple, determining exact gain parameters can be unintuitive. Rather than automating the gain setting and limiting animator control, a calibration method was established that gives the animator some intuition for what gain values mean. The calibration process is performed on the simulated torso skeleton and makes the entire skeleton very stiff except



(a) Loose Character



(b) Tense Character

Figure 6.20: The figures show frames from two animation sequences in which the character is hit in the shoulder with the same force. The character is loose in the top series, but tense in the bottom. The frame spacing for the top series is 0.23 sec and the spacing in the bottom is 0.1 sec. The movement of the tense character is more visible in the animation than in the still frames.

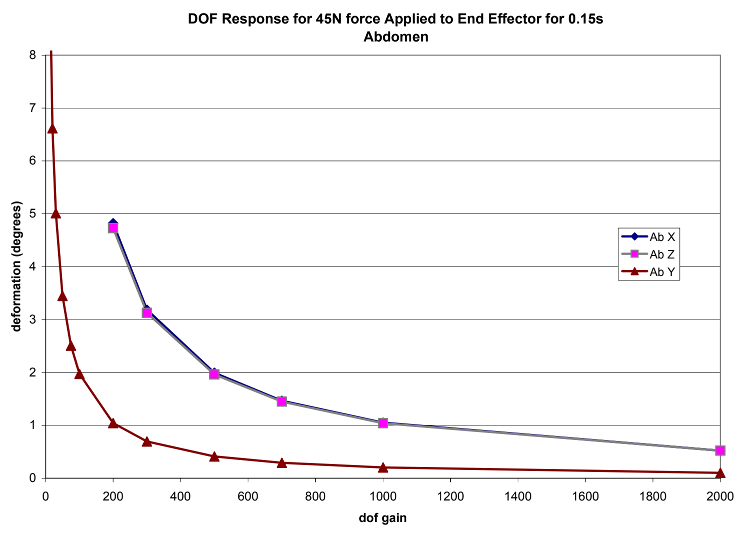


Figure 6.21: Calibration chart for the abdomen joint showing how much the abdomen joints deform when a 45N push is applied to the character's head from an appropriate direction.

for one DOF. The skeleton is then perturbed by pushing it at an end effector appropriate for the DOF with a force of 45N. This corresponds to a firm push. The test is repeated for several different gain values for the DOF and then for the rest of the DOFs in the torso. The character's response is thus sampled for different tension values at each joint. The animator can then access a set of curves that show the maximum deformation that a push will give at each DOF as a function of gain. These curves have hyperbolic shape as shown in Figure 6.21. This information provides a clear picture of how stiffness values range across different joints of the body, as well as the effect of changing the gain, and allows the parameters to be used in a knowledgeable way.

6.3.6 Damping

Damping is the final control parameter in the system. It can be varied to reduce overshoot and transitory effects in a motion. It also generates some lag. If damping is too low and the movements are quite fast, ringing can occur where the motion exceeds the equilibrium point during the transition and then gets pulled back toward it. Increasing the damping

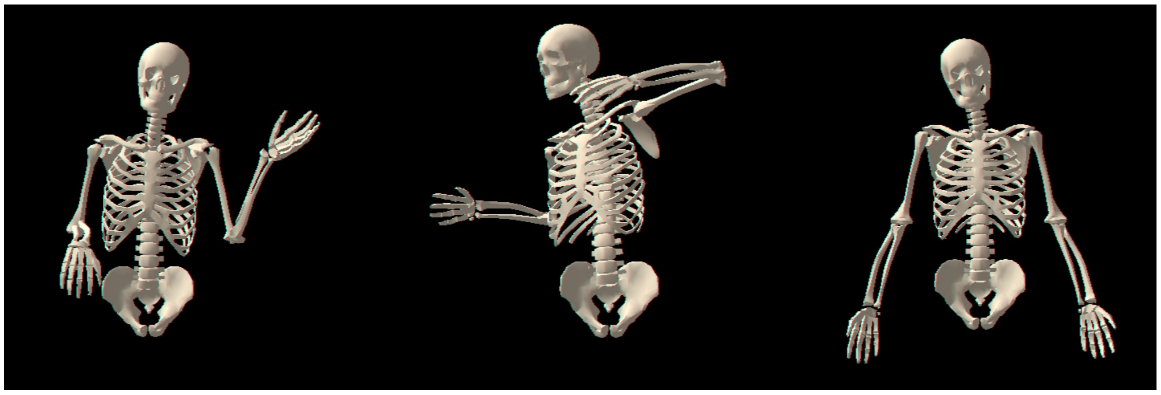


Figure 6.22: A montage of frames from the animations discussed in this section.

can cure this. Generally, damping values that are approximately one tenth of tension values work well.

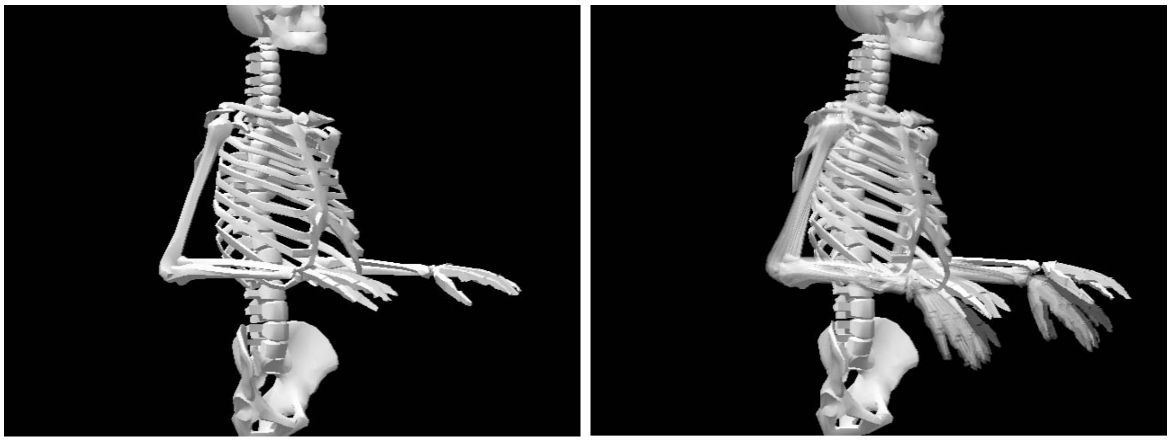
6.4 Results

A number of sample animations have been generated to illustrate the impact of tension control. As with all animations discussed in this thesis, they are available online [112].

All the animations described consist of a small number of poses, the richness coming from the impact of tension variation on the dynamics simulation. Some iteration was involved in generating the animations. In general, less iteration is required when tension changes are the dominant shaping function than when the transition functions are carefully tweaked.

6.4.1 Gestures

Gestures are a prime candidate for expressive variations in tension. An animation is shown that contrasts a relaxed, flamboyant character and a police officer giving the direction "You should go left." The police officer's movements are quick and precise. The flamboyant character's movements are loose and flowing. His arms swing slightly at his sides, his wrists flick, his torso tilts slightly, the movements accelerate and decelerate



(a) Kinematic (b) Dynamic

Figure 6.23: Comparison of kinematic and dynamic motion: The left image shows 0.4s from a kinematic version of the "go left" animation after the character has brought his arms down to his side and the right image shows the same 0.4s from a dynamic version of the animation. Note the overshoot and wrist motion that occurs with a low-tension dynamic simulation.

based on tension changes. All of these effects are due to tension modeling, not the kinematic trajectories. It should be further emphasized that these effects would not occur for uniformly high tension. In this case, the system would closely track the kinematic trajectories and most of the subtle nuance would be lost.

Figure 6.23 shows a 0.4 sec section from the animation. During this clip, the character has just brought his arms down to his sides such that his forearms are parallel to the ground. The left image shows five frames from the kinematic version of the animation. There is no movement during this period. The right image shows five frames from the dynamic version of the animation. The bounce in the spine, shoulders, elbows and wrists can be clearly seen. This additional motion greatly adds to both the sense of looseness captured by the animation and the realism of the clip.

A wave motion was generated using pure actuated control, kinematically, using high tension control and varying the tension through out the movement. The actuated control is quite unnatural as the movement accelerates quickly and then eases in toward the end of each section of the motion. The kinematic wave is substantially better, but still looks stiff and not particularly lifelike. Using tight control produces an animation that looks very similar to the kinematic motion. Varying tension produces a significantly more lifelike wave. The wrist is allowed to move passively creating a swaying hand motion, the force transference generates some movement in the collar bone, the trajectories are smoothed and warped due to the influence of gravity and momentum, the arm sways slightly at the character's side at the end of the movement. These effects rely on being able to vary the tension during a movement. For instance, the arm is relaxed as it is brought to the character's side, allowing it to sway slightly at the end. This animation uses kinematic trajectories for shape control, but enhances them by adding tension variation.

6.4.2 Postures

Posture adjustments are ideal candidates for tension control. If a tired character is to sag downwards, a pleasing motion shape can be achieved by simply providing a linear transition function and reducing the tension during the motion. Other posture adjustments can be made in a similarly straightforward manner. A key point here is that the animator retains the ability to provide arbitrary transition functions to customize a motion. Examples were prepared that contrast a formal greeting nod with a tired character sagging downwards. A shrug is also shown.

6.4.3 Anticipation

People make tension changes when they anticipate an event. This affects how they react. To demonstrate this, a character is shoved with different forces and different amounts of tension in his body. The effect of the shove becomes quite muted as he braces for impact. Another animation combines tension changes and gestures as the character reacts to being pushed from behind.

6.5 Future Work

Additional transition properties could be added to the system. *Path in space* is the most significant transition property not currently modeled, and relates to how a character moves through space on his way from a starting pose to a destination pose. This is related to the *space* parameter in Laban's Effort notation, ranging from direct to indirect. Direct movements follow a straight line towards the target, minimizing the use of space. Indirect movements take a curved path towards the target, indulging in space. A slash is an indirect movement, whereas a straight punch is a direct movement. For another example, imagine a boxer with his fists raised, about to make a punch. He might twist his body forward and throw a direct punch at his opponent, or he might drop to the side and swing back up towards his opponent with an upper cut.

The standard technique to deal with path in space effects is to add additional *via* poses through which the character must pass on his way between the initial and final pose, thus modifying his path through space [36]. The tangents of the transition curves must be modified to create a smooth movement through the poses. It may also be possible to vary the path in space by controlling the interpolation path on the hypersphere of the quaternions used to define the shoulder joints.

The effect of tension changes is non-linear and hence difficult to accurately predict. The calibration system provides assistance to animators, but investigating more intuitive mappings would be worthwhile.

Currently, a *body shape solver* and *time planner* have been implemented, but the *transition planner* has not. The envisioned transition planner would enforce continuity constraints between sequential transition functions. There has been little need for this in practice.

Chapter 7

Animation System Architecture and Workflow

The thing with high-tech is that you always end up using scissors.

David Hockney, British Artist

The primary objective in developing this system was to explore how the lessons contained in the arts literature can be used to inform and improve the design of an animation system. This has forced a reductionist approach to design. The lessons from the arts literature must be given representation in order to be used computationally. The literature identifies a long list of different aspects of movement that are expressively important and provides some indication as to how these aspects can be varied for specific effect. Taken together, these aspects form a language for describing movement. By finding computational representation for a part of this language, we are reducing movement to a series of handles that each control one aspect of movement and these aspects must then be joined together to provide a complete movement piece. This requires the construction of a "common language" in which the various aspects can be brought together. This is done through the creation of a "base representation" to which all higher level constructs in the system are mapped.

The system builds on a number of threads from previous research. Character poses have proved to be a useful abstraction for animators and are the basis of the keyframe animation systems used for most high end production work. Poses provide animators with an effective set of handles on a motion sequence [80, 124]. They are also a useful abstraction when creating hand tuned controllers for dynamic simulation. For both these reasons, we have adapted a motion representation based on generalized poses. As discussed in previous chapters, the system also makes use of inverse kinematic techniques, principles developed for hand-tuned controller synthesis and feedback based balance control.

The system also differs in many important ways from previous work in the field. If this system has a desideratum, it is that animators need control, and it tries to provide this by creating a rich set of handles. In particular, the system tries to create handles for specific aspects of movement that are identified in the arts literature as being important. This direct control of aesthetic aspects of movement has seen limited attention elsewhere and lends itself to more expedient editing of a motion sequence. It requires that a precise and explicit semantics be given to each principle taken from the arts. It is also necessary to provide handles at different levels. Some ideas are more abstract and will modify the entire animation sequence, as for example an instruction like "Do that as if you are an old man.", whereas other handles are very concrete and low-level, such as adjust the elbow in the third pose by thirty degrees. To work effectively, an animator needs such a range of handles, including the many levels in between, that support both the rapid exploration of abstract ideas and the precise refinement of particular aspects of motion. Furthermore, to support refinement, it must be possible to adjust particular aspects of broad edits in a well defined way without destroying the effects of the broad edits. The system supports this by building the more abstract handles out of the more concrete, low-level handles so that the animator can always increase their control by moving down the representation hierarchy. To our knowledge, other systems have not placed such an emphasis on both exploration and refinement.

Related to this is the idea of rehearsal, which is fundamental to the system's workflow. The animator will not move directly to the final animation, but pass through multiple iterations, or *rehearsals*, of the sequence, during each exploring different approaches, adjusting aspects of the motion, and refining his subjective knowledge of what the final animation sequence should be. This process is important for the animator as he will not in general know what the final motion sequence should look like until he has experimented with it. This process also requires both broad edits that support exploration early in the rehearsal process and precise control that supports refinement late in the process. Related to this is the need for the various input channels to manifest themselves in the final motion sequence and for later edits to layer on top of earlier edits.

Given this background on the unique requirements for the system, the following sections will provide a precise definition of it. An overview of the system was provided in Section 1.6.1, which serves to introduce this material. An early version of the system was presented in [115] and a more complete version in [117].

7.1 Workflow

User interaction in the system follows a basic Perform-Review-Update cycle that is similar to how a rehearsal process operates in theatre. The system is given a set of instructions and generates an animation sequence. The animator then reviews the animation and considers possible adjustments. The animator can then issue additional instructions, or start over with different instructions, and generates a new animation sequence. This sequence is repeated until the animator converges on a desired movement sequence.

The animator and the system progress through the following steps.

1. The animator provides a script specifying the actions to occur in the animation sequence along with additional instructions that specify how these actions should

be performed (see Section 7.3)

2. The system develops an executable motion plan based on these instructions (see Section 7.4)

3. A dynamic or kinematic simulator executes the motion plan in order to generate an animation, satisfying low-level constraints such as balance (see Section 7.5)

4. The animator reviews the animation and refines the script, iterating until he is satisfied with the result

Each of these steps will be outlined in detail below. An animator will normally iterate through this process, initially making broad changes as he explores different possibilities and then making fine adjustments as he converges on a final motion sequence.

The channels by which an animator can interact with the system are summarized in Table 7.1. Each of these are described rigorously and in more detail in Section 7.3. Each channel serves a different purpose and different channels are often used at different points in the production process. *Properties* define the handles by which an animator can change the feel of a piece of motion. New property definitions may be developed during character set up if the existing properties are deemed insufficient. Such extensibility is necessary because the list of different aspects of movement one might want to vary – indeed the language used to describe movement – is both large and open ended. High level terms in particular will need specific definition for a given character

A script must always be provided to specify the actions a character performs. This allows the animator to provide a terse, high-level specification of what should happen. The action descriptions themselves provide a framework for a specific movement onto which properties can be attached. Actions provide animators with a handle on a particular "chunk" of movement.

Character sketches make global changes to the movement sequence and are normally applied in the early stages when an animator is exploring different approaches for a

Channel	*Summary*
Script	Specifies a time series of actions.
Action Description	Defines the structure of a movement, generally with an initial set of properties.
Character Sketch	Generates a global change to the movement sequence.
Animator Edits	Used to fine tune an animation sequence.
Property Definitions	Provide interaction handles.

Table 7.1: Interaction channels.

character. Sketches allow an animator to very quickly modify the motion sequence, supporting prototyping, exploration and experimentation. In short, they allow an animator to play with ideas. Character sketches define properties that are typical of a particular character type and then propagate them to all actions in the script. For instance, a character sketch might increase the tempo and the tension a character exhibits. Changing character sketches allows an animator to rapidly explore broad characterizations.

Animator edits allow the animator to directly change, add or fine-tune properties at any point in the action hierarchy for any action in the script and can also be applied globally to all actions. These are necessary for refinement, providing a complement to the exploration supported by the character sketch. An animator must be able to explore and then converge on the precise animation he desires.

7.2 Base Representation

Although conceptually the Base Representation comes into play later in the motion planning process, it is simpler to specify it first as higher level constructs can then be defined by reference to it.

All higher level constructs in the system must be mapped to the elements in Base Representation (BRep). The BRep is a random access data store that provides all the information necessary for either a kinematic or dynamic simulator to generate a final animation. Two actuator formulations are supported with dynamic simulation: traditional

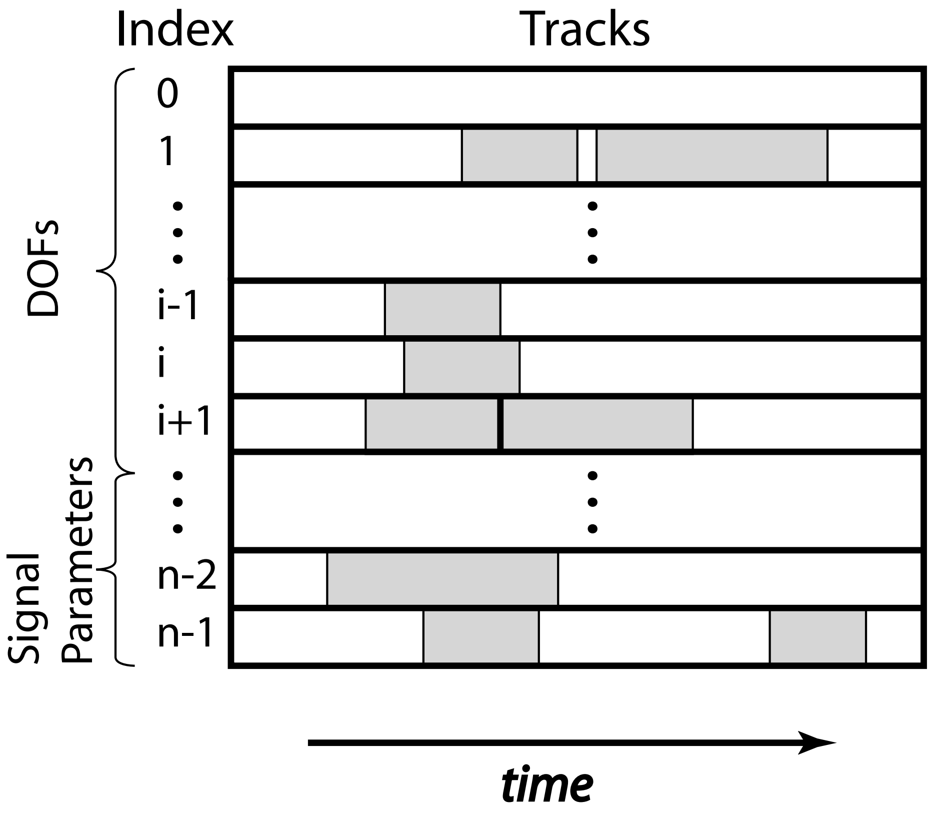


Figure 7.1: Animation can be generated from the base representation shown here. There are time ordered tracks for each DOF in the skeleton and additional tracks for signal parameters that are used to specify higher level effects like a balance shift. Each track is populated with transition elements, which are shown as grey boxes here.

PD-control and the two spring variant, *antagonistic control*, discussed earlier. A BRep contains a track for every DOF in the skeleton and additional *signal parameter* tracks that act as blackboard space. These are used to specify continuous variations in higher level attributes controlled by the system, such as a centre of mass offset or the amount of pelvic twist. Each track is time ordered and can thus be seen as inducing a time series of activities for every degree of freedom. The BRep is illustrated in Figure 7.1. A track T can be populated with either *Transition Elements* or *Control Parameters*:

$$T = \langle a_0, a_1, a_2, \cdots, a_k \rangle, a_i \in \{ControlParam, TransitionElement\}.$$

We define $\mathbb{T}$ to be the set of all such sequences T. *TransitionElements* are set by the planning process, described below (Sec. 7.4). *ControlParams* are valid for one time step and are set by reactive controllers during the motion generation process (Sec. 7.5). They can override the values of a TransitionElement for that time step.

Control parameters provide the data necessary for either a dynamic or kinematic simulator to advance the state of the character one time step. Formally:

$$\mathit{ControlParam} = (v, K_L, K_H, K_d;\quad \theta_L, \theta_H).^1$$

where the *value* $v \in \mathbb{R}$ is the joint angle in a kinematic simulation; it is the set point in PD-control and it has no meaning in antagonistic control. This value is constrained to $\theta_L < v < \theta_H$, where θ_L and θ_H are set just past the joint limits of the DOF, and they act as the set point for the springs in antagonistic control (see Section 6.2.2). The terms $K_L, K_H \in \mathbb{R}, K_L, K_H \geq 0$, are the two spring gains used in antagonistic control. Only K_L is used in PD-control and they have no meaning in kinematic simulation. The term $K_d \in \mathbb{R}, K_d \geq 0$, is the damping gain. It has no meaning in kinematic simulation.

[1]We use a semicolon to distinguish between variable parameters to the left and constant parameters to the right of the semicolon. We also employ the standard notation in which "$(a_0, a_1, \cdots, a_n)$" denotes an ordered n-tuple of elements, while "$\langle a_0, a_1, a_2, \cdots, a_n \rangle$" denotes a sequence of elements.

Transition Elements cover a small period in time for a DOF and specify how to make a transition from the DOF value at the beginning of the element to a desired value at the end of the element. Transition Elements can also include hold times at the end of the transition. More formally, a transition element can be defined as follows:

$$\mathit{TransitionElement} = ([t_0, t_1], d, h, f_v(t), f_{K_L}(t), f_{K_H}(t), f_{K_d}(t),\ \mathit{tags};\ L),$$

where $t_0 \leq t_1 \in \mathbb{R}$ define the interval of "track time" over which the activity occurs. The terms $d, h \geq 0 \in \mathbb{R}$ define the duration and hold time of the activity; $t_1 - t_0 = d + h$. The intervals of the TElements are (currently) presumed to be disjoint, but their union may leave gaps in the timeline for that DOF, which indicates that no actions are active in that interval. The four functions f are interpolation functions. The values they return correspond to the element in the control parameter definition with the corresponding subscript. For example, $f_v(t)$ returns the value of the DOF at time t. These functions are parameterized on $t \in [0, 1]$ and then scaled and translated to $t \in [t_0, t_1]$. We normally use a cubic Hermite interpolant embedded in space and time as our interpolation function. More detail on the interpolant is provided in Section 6.1.2). The parameters of these functions can be specified and varied by higher level processes in the system (e.g. properties, planners and filters). *tags* are additional data that can be used by filters that operate on the BRep, but they are not available to the simulators. *tags* are set by properties. For instance, a synchronization tag might be added by a property and then affected by the Time Planner. L is the label of the action that generated the transition element and is used to facilitate search and query operations (labels are defined in Section 7.3.2). There is currently a one-to-one correspondence between Transition Elements in the BRep and DOFEntries in the action description.

Any instance B of a *base representation* is an m-tuple of tracks containing at minimum one track for each DOF. Thus $B \in \mathbb{T}^m$, or

$$B = (T_1, \cdots, T_m).$$

The index i identifies the DOF in the character's state to which track T_i is bound. Index values above the number of DOFs in the character state correspond to signal parameter tracks. We define the space of all base representations $\mathbb{B}$ to be all such B.

The BRep is incrementally and iteratively constructed during a motion specification process. This creates a sequence of refinements of valid BReps

$$\langle B^0, B^1, \cdots, B^n \rangle, \tag{7.1}$$

where $B^i \in \mathbb{B}$ and i indicates the i^{th} iteration on the BRep. Any such B^i may well be executable by the simulator, but the final element B^n should be seen as a "converged" base representation. The initial base representation, B^0, is the null-operation, and is defined to be an m-tuple of empty tracks:

$$B^0 = (T_1, \cdots, T_m) \ \mid \ T_j = \varnothing, j = 1, \cdots, m. \tag{7.2}$$

With each iteration, more information is added or existing information is modified in the representation. This represents a motion planning process as described in Section 7.4.

The base representation is generally invisible to a user, who instead works with the more germane and powerful animation primitives described above. All such primitives, however, have a semantics that is defined by their effect on a BRep.

7.2.1 Commands

The Base Representation supports a set of commands that it can execute. These commands define how the BRep can be updated. All properties act through them.

Command	Description
SetControlParams	Sets a control param that is valid for a single time step
AddTransitionElement	Creates and adds a transition element to a given track of the BRep at a given time.
InsertPose	Like AddTransitionElement, but inserts the pose into the midst of a current action. For instance, if an action was defined to take five seconds, a new pose could be added that would occur two seconds into the action.
RemoveTransition- Element	Removes a transition element from the BRep.

Table 7.2: Structural commands acting on the Base Representation

The main commands are outlined below. It should be noted that these are not the complete list of commands available to properties. Some commands configure a data store associated with a particular solver, which in turn writes into the BRep. For instance, there are a set of commands associated with the body shape solver such as SetGazeDirection and SetSagClass. These configure how the shape solver works and it in turn generates a set of DOF values.

In addition, properties support composition – properties can invoke other properties. Each new property effectively defines a command that can be used within a composite property.

Structural Commands

These commands add structural data (Transition Elements or Control Params) to the BRep. The current set are summarized in Table 7.2.1. There are also commands that work on the script, rather than the BRep, as discussed in Section 7.3.1.

Attribute Commands

Attribute commands modify a transition element attribute that is used to generate the final animation. They are summarized in Table 7.3.

Command	Description
SetStartTime	Sets the time the TElement becomes active.
SetStopTime	Sets the time the TElement ceases to be active.
ShiftTimes	Shifts the whole transition element (start and stop times).
SetDuration	Sets the duration of a transition from the initial value of the DOF to the desired value.
SetHold	Specifies how long the DOF value should be held for.
SetDamping	Sets the damping value used during the transition.
SetEndTension SetStartTension	Sets the tension values used in dynamic simulation.
SetDOFValue	Sets the desired DOF value for the end of a transition.
SetStartValue SetStopValue	Used for control parameter transition curves, specifying both the initial and final value of the DOF.
SetTransitionTangents	Set the four tangent values that define a Cubic Hermite transition curve.
ApplyOffsetCurve	Specifies an offset curve that is added to the transition curve.

Table 7.3: BRep commands that update animation attributes.

Command	Description
SetActionLabel	Sets the label that is used to identify the TElement. The label links it back to the generating action.
SetTag	Allows a generic tag to be added to a given TElement. The tag has a name and a parameter. Tags can be set to either override previous tabs with the same name or to blend with them.

Table 7.4: Commands for adding meta data to TElements.

Tag and Label Commands

Table 7.4 summarizes the label and tag commands. The SetActionLabel command is used to link transition elements to the actions which generated them. The SetTag command writes meta (tag) data into a transition element. The tags are not used to generate the final animation and have no meaning to the simulator. They are used by planners/filters that are run across the BRep before it is passed to the simulator. For instance, a tag item might indicate that the start time of subsequent items should be increased by 0.5 sec. A time planner will actually make the adjustments to satisfy this.

Query	Description
GetTElementsWithName	Returns all the transition elements that match the given label. This can be used to get a specific Transition Element, all the Transition Elements for a given pose etc. Refer to the definition of labels in Section 7.3.2.
GetDOFValue	Returns the desired value of the Transition Element.
GetStartTime	Returns the start time of a specified Transition Element.
GetStopTime	Returns the stop time of a specified Transition Element.
GetDuration	Returns the duration of a specified Transition Element.
GetTag	Returns the tag data given a tag name.

Table 7.5: Queries supported by the BRep.

7.2.2 Queries

The BRep also defines a set of query operations that allow properties to obtain data. A preliminary set of queries is outlined in Table 7.5. Additional query commands can be added as needed.

7.3 High Level Constructs

This section will develop formal descriptions of the high level objects in the system. It is through these objects that the animator interacts with the system to specify and refine an animation sequence, as shown in Figure 1.1. The high level objects include:

- A *script* which specifies the actions a character is to perform
- *Actions* that provide a handle on particular movements, like a shrug
- *Properties* which encapsulate a particular aspect of motion and give meaning to actions. For example, properties would control the joint angles of the shrug and specify that the movement was done in a “loose” manner.
- A *character sketch* which defines global aspects of the movement. For instance, an old man sketch could automatically add a hunched back to the shrug and slow its timing.
- *Animator edits* which vary the properties acting on the actions. For instance, animator edits could be added to decrease the amplitude of the shrug and change the motion envelope.

Command	Description
AddAction	Adds an action to the script. This command can be used both to build the initial script and to add an action in response to a particular property.
DeleteAction	Deletes an action from the script.
InsertAction	Like Add action, but inserts the action into the midst of a current action.

Table 7.6: Commands supported by the script.

Each of these high level objects in the system represents a reusable resource. Resources developed during one production can be used in future productions.

7.3.1 Script

The script S is used to specify the actions a character is to perform. It contains a time ordered series of actions that may overlap. For example, a script might specify a shrug followed by a grand sweeping gesture to the right. More formally, a script consists of an unordered set of tracks R which each contain a time ordered sequence of actions:

$$R = \langle a_i \rangle, a_i \in \mathit{Action} \wedge \mathit{StopTime}(a_i) \leq \mathit{StartTime}(a_{i+1}),$$

$$S = \{R_1, \ldots, R_n\}, n \geq 1.$$

Normally, n is about 5.

The animator defines an initial script which outlines the animation sequence and can directly edit it during any iteration of the animation. It can also be modified by *generator properties*. These edits produce a sequence of scripts S^i where i indicates the current iteration. The script refinement process is described in Section 7.4.1.

The script supports a number of commands which are summarized in Table 7.6.

7.3.2 Actions

An action is the basic unit of movement in the system. For instance, the shrug and sweeping gesture are each actions. Actions are pose based and defined hierarchically with the following *action levels*:

- Action
- Cycle
- Pose
- DOFEntry

A *DOFEntry* controls an individual degree of freedom of a character as a function of time. The DOFEntry level can control Signal Parameters as well as character DOFs. A *pose* is a set of DOFEntries. A *cycle* is a sequence of poses that can be repeated. An *action* is a sequence of cycles. A pose, and hence an action, can control any subset of a character's DOFs. It need not specify a full body configuration. Multiple actions controlling different parts of the body can be superposed by placing them on separate tracks in the script. The term *action level* is used to refer to a specific instance of a cycle, pose, or DOFEntry within a particular action. For instance, an action level might be the second pose in an action named "wave".

An action defines the structure of a movement, but does not define the content. More specifically, an action defines which DOFs make up a pose, how many poses are part of a movement, and the sequencing of those poses. It does not define what value each DOF should hold for a pose, the transition time to a pose, the shape of the transition, etc. All of these details are provided by properties which are attached to the action and will be described shortly. In this sense, actions can be thought of as structured containers which hold properties. As an example, the "shrug" action might specify that the movement consists of a sequence of two poses involving the shoulders, torso and head. The actual joint angles and timing information are provided by properties that are attached to this basic framework.

Each level in the action has a similar formal definition. An action, A, can be specified as:

$$action = A = (\langle cycle_1, \ldots, cycle_n \rangle, \langle P_1, \ldots, P_m \rangle, L)$$

where $n \geq 1$, P refers to a property attached to the action and m can equal zero (there are zero or more properties attached to any action). L is an *action label.* Action labels are applied to each level in the hierarchy and allow for quick referencing. Any field in the label that is not relevant for a given action level is set to -1. L can be defined as:

$$L = (name, cycleCnt, poseCnt, DOFIndex, cycleRep)$$

where *name* is the name of the action, $cycleCnt \in \mathbb{N}$ is the number of the cycle containing this entry, $poseCnt \in \mathbb{N}$ is the number of the pose containing this entry, $DOFIndex \in \mathbb{N}$ specifies the DOF that a DOFEntry controls and $cycleRep \in \mathbb{N}$ refers to which repertition of a cycle this entry is associated. For an action, all entries but *name* will be set to -1.

In a manner similar to an action, a cycle is defined as:

$$cycle = (\langle pose_1, \ldots, pose_n \rangle, \langle P_1, \ldots, P_m \rangle, L)$$

and a pose:

$$pose = (\{DOFEntry_1, \ldots, DOFEntry_n\}, \langle P_1, \ldots, P_m \rangle, L)$$

where $n <$ num tracks in B. Note that the ordering of DOFEntry within a pose is not important because each DOFEntry defines what DOF it controls. Finally, a DOFEntry is defined as:

$$DOFEntry = (\langle P_1, \ldots, P_m \rangle, L).$$

Actions divide a movement sequence into meaningful units with which the animator can work. They allow an animator to define an edit on a particular piece of motion, without necessarily affecting surrounding or overlapping pieces of motion, and without committing to a particular emotional presentation.

Currently, the semantics of a property does not change if it is applied to different actions. In other words, actions do not affect properties. A higher level process which is applying properties to actions, say a process that adds beats to motions, might change the property it applies based on the name of the action. Adding functionality to actions so that they could modify the meaning of applied properties may be an interesting avenue to explore in the future.

All actions are mapped to at least one Transition Element in the Base representation. The Transition Element can either directly control a joint angle or a signal parameter. There is currently a one to one correspondence between the Transition Element in the BRep and the DOFEntry in the action. For most actions, many transition elements will be created.

Initial action descriptions are specified in small text files. The general idea of these descriptions is to give the rough skeleton of a movement that can then be varied by the application of additional properties. These text files will normally include a few initial properties to give the actions form. The minimum information needed to create a transition element is the number of the track it is to be stored in and the start time. Default data will be generated for all other parameters, but it is normal to also provide a duration, hold time, desired value and a transition function for each Transition Element.

7.3.3 Properties

Properties are designed to encapsulate a particular aesthetically meaningful aspect of movement. Properties provide an interpretation for actions. One of the system design goals is to allow properties of the same type to be blended together. This allows edits from various channels to be layered, rather than overwriting earlier edits.

Properties can be divided into numerous classes. These classes can be distinguished by the input the property requires, the type of output the property generates and the property attachment. Properties are either attached to a particular action, or stand alone. If a property is applied to multiple actions, it will be replicated and a separate instance of the property will be applied to each action. There are two types of standalone properties: *generator properties* and *applicator properties*. Generator properties act to introduce new actions into the script, such as idle motions, nervous ticks, or movement sequences that are more easily defined procedurally. Generator properties can also attach other properties to the actions they create. Applicator properties attach properties to a series of actions. For instance, a rhythm property can attach properties that will offset timing information to a set of actions in order to generate a particular rhythm. Most properties are attached to actions.

Properties that attach to actions can be applied at any level in the action hierarchy. By default, properties will cascade to lower levels in the action description. If they encounter a property of the same type at a lower level, the lower level property will be given precedence. It may override the higher level property or blend with it, depending on the property's settings. For instance, a transition curve applied at the Pose level will cascade to all DOFEntries that are part of that pose. If a DOFEntry has a transition curve property applied to it separately, that property can either override or blend with the higher level property.

Properties can be differentiated on the type of input they require. Most properties require no input, aside from the basic data that defines them (see Table 7.9). A subset of properties needs to query the BRep for data that has been previously written into it. These properties are called *post-burn in* or *post-commit* because they must be executed after other properties have written the data they need into the BRep. This introduces an ordering constraint on property application. An example of such a property is *VaryAmplitude*, which varies the joint range used in a motion and must be called after the affected

Output Type	*Formal Action on System*
Modify Transition Element data.	$P : TransitionElement^i \rightarrow TransitionElement^{i+1}$
Modify Transition Element tags. The tags are used by solvers that work to achieve the effect requested by the property.	$P : TransitionElement^i = TransitionElement^i \wedge newTag$
Stores constraints in a separate data store D. These constraints modify the action of a solver, which in turn will write data into the BRep.	$P : D^i \rightarrow D^{i+1}$
Modifies or adds to the properties associated with an action	$P : A^i \rightarrow A^{i+1}$
Adds transition elements to a track or tracks.	$P : T^i \rightarrow T^{i+1}$
Generates new actions which are added to the script.	$P : S_i \rightarrow S_{i+1}$

Table 7.7: The system generates different forms of output. The effect of each type of output is shown formally on the right. In all cases the i and $i+1$ refer to iteration counts.

DOF values have been written into the BRep.

Finally, properties can be grouped based on the kind of output they generate, or in other words, how they act on the system. The various output types are listed in Table 7.7.

It is also important to distinguish between *Base properties* and *Composite properties. Base properties* directly modify low-level attributes exposed by the system by invoking commands exposed through the Base Representation. *Composite properties* combine several base properties. They provide for higher level interaction and co-ordination.

Properties are defined procedurally. It is not a goal of our work to propose a static, exhaustive set of properties. Rather, the property set is designed to be extensible by an animator or technical director. If a new aspect of movement needs to be controlled for a particular character, or an animator needs a particular handle to vary a piece of

Property	*Description*
SetDOFValue	Specifies the value for a DOF
SetTension	Adjusts the amount of joint tension during a motion.
Synchronize	Sets a timing relationship between two actions
SetDuration	Varies the transition time to a pose.
SetTransitionCurve	Varies the transition envelope.
VarySuccession	Adjusts the relative timing of joints in a motion.
VaryExtent	Adjusts the amount of space a pose occupies
VaryAmplitude	Adjusts the angle range covered by a motion.
GenerateWeightShifts	Generates idle weight shifting behaviour.
SetReachShape	Varies posture during a reaching motion.
SetPosture	Varies character posture.
AdjustBalance	Alters the balance point of the character.
SetRhythm	Coordinates actions to occur with a certain rhythm.
CreateRecoil	Varies all aspects of movement to recoil from point.

Table 7.8: A few of the properties defined in the system.

animation, new properties can be defined to achieve this. All manner of properties are permissible as long as they can translate into explicit effects on the base representation. Properties can also be modified, so the manner in which a particular aspect of movement is varied can be customized for a given character. Seen in this light, developing new properties is a part of character design, much like rigging. The use of repositories of properties will facilitate re-use in future productions.

A representative sample of properties is shown in Table 7.8. Note the wide range of granularity at which properties act. *SetDOFValue* will control the desired value for a single DOFEntry. Properties like *VaryExtent* or *SetPosture* vary several DOFEntries within a single pose. Synchronization and Rhythm properties create relationships between multiple actions. *CreateRecoil* is a high level property that will change a character's pose, warp the transition function, increase his tension and increase his tempo. Properties like *CreateRecoil* normally support a number of different forms of recoil.

Properties take parameters that define both how they behave and how they should be combined with other properties of the same type. (The property merging process is discussed in Section 7.4.) *SetDuration* for instance, takes a value and a flag. If the flag is set to ABS, the property will override previous *SetDuration* properties attached to the action level and set the item's duration to be *value*. If the flag is REL, the property acts in a relative manner and multiplies the action level's current duration by *value*. *SetReachShape* takes a world space location to be touched, a shape class that defines the type of stance that should be sought during the movement, and an intensity that controls how strongly this stance should be adopted.

Properties must regulate both how they are to be combined with other properties and how they will update the base representation. In order to do this, they must define the data outlined in Table 7.9 and also implement the functions described in Table 7.10. The system for merging and applying properties is described below. Each property has a precise semantics, many of which are discussed either in the preceding chapters or the chapters ahead.

7.3.4 Animator Edits

The animator can modify a motion sequence by specifying edits. An edit is simply a property along with a specific set of parameters and, if the edit is to be applied to an action, a label specifying the action. Formally,

$$AnimatorEdits = \langle edit_1, \ldots, edit_n \rangle,$$

$$edit = (name_P, L, merge, params)$$

where $name_P$ is the name of the property to be applied with the edit, L is the label of action that receives the edit, $merge$ specifies how the property should be combined with previous properties of the same type (average, override etc.) and $params$ are any parameters associated with the edit. Some properties are not applied to a specific action. In this case, L is not required. If the edit does include a label, the effect of the edit is to attach a new property of the given type, with the given parameters, to the specified

Data	*Description*
Combination Type	Specifies how the property should be combined with other properties of the same type. Examples include ABS, which will override earlier properties; AVG, which will average with the values of previous properties; and REL, which will linearly scale previous property values.
Category	Specifies which aspect of movement the property affects. Categories include: shape solver, shape, timing, transition, post-burn in, generator, applicator, reactive controller and other.
Priority Level	Priority level is used to determine the order properties are merged in. It is based on the source of the property and the level in the action hierarchy at which the property is applied.
Minimum Property Level	This indicates the lowest level in the action hierarchy at which this property makes sense (e.g. cycle or DOFEntry).
bCombinable	Flag that states whether this property should be merged with other properties of the same type or should just be directly applied.
parameters	Most properties will define their own set of parameter values.

Table 7.9: Properties must include the above data. Many of these data items are used in property merging.

Function	*Description*
GetParamString	Properties are normally customized by specifying values for parameters that are defined as part of property authorship. A property must be able to return these values as a string. The strings can be passed between properties of the same type, which can decode them. Merge operations are performed on these strings.
Combine	Properties define how they can be merged, including what combination types they will support. This is done by writing a *combine* function.
Apply	Every property must define this function. It is called to execute the property. This triggers the property to generate its output.

Table 7.10: Properties must implement these three functions. Properties that are not combinable can implement stubs for GetParamString and Combine.

action. As with action labelling, not all fields in L need to be complete. For instance, if the action name and cycle count are specified, the property will be applied to that cycle and cascade down to all the poses below. The label can also include the * wild card, which will accept any value for that field. This allows an edit to be applied to all actions, or to all shoulder DOFEntries etc.

7.3.5 Character Sketch

A *character sketch* is used to make global changes to a movement sequence in order to allow an animator to quickly explore different character types for a sequence. The character sketch currently consists of a list of edits that are applied globally:

$$CharacterSketch = \{edit_1, \ldots, edit_n \mid \forall edit, edit.L.name = *\}.$$

The character sketch is the natural repository for all global character information. In the future, data such as strength limits, body parameters such as limb lengths, joint ranges, etc., can all be added to the character sketch.

The use of both the animator edits and the character sketch will be discussed at some length in Chatper 9.

7.4 Generating a Motion Plan

Once the animator has completed the motion specification process via the channels described above, there are a number of active sources of information in the system that will all effect how the final animation is generated. These instructions must be combined in a consistent and predictable way in order to generate the final motion specification. This is the job of the *Movement Property Integrator*, or *MPI* for short, which is the central intelligence of the system. It is responsible for mapping the various forms of input into an executable motion plan, stored in the BRep. The stages involved in this process are shown in Figure 7.2 and described below. The script is first refined, all active properties in the system are then resolved and applied, and then the BRep goes through a final refinement stage. In a single iteration, the system transforms the user's input to an animation sequence. The system *always* generates an animation.

The MPI operates by applying properties, triggering solvers and running filters. Most of the "knowledge" of how to modify movement is contained in these entities and the MPI's job is rather one of coordination and arbitration. Multiple properties may attempt to modify the same underlying data and the MPI determines how to correctly merge these properties. It must also sequence all operations so that data created by one process will be in place before it is needed by another. All this must be done in a manner that is consistent and predictable to ensure that high level animator adjustments have the expected result.

Due to the complex interactions among primitives and the current state of the motion plan, operators are required that clarify how a high level animation is decomposed into an executable base representation. The MPI makes use of four classes of operators: one for script refinement ($\mathbf{Q}$), one for attaching properties to actions (α), one for executing properties ($\mathbf{M}$) and one for refining the BRep ($\mathbf{F}$).

Algebraically, any operator $\mathbf{Q}$ is a mapping from the script, together with information about how the script is to be modified, to a script. More formally, the operator $\mathbf{Q} : \mathbb{S} \times \mathbb{P}_G \to \mathbb{S}$, where $\mathbb{P}_G$ is the set of generator properties. The other operators can be defined in a similar manner. The operator α attaches properties to actions in order to support animator edits and the character sketch. It can be represented as $\alpha : \mathbb{A} \times \mathbb{P} \to \mathbb{A}$ where $\mathbb{P}$ is the set of animation properties and $\mathbb{A}$ is the set of actions that are modified by these properties. The M operators apply the properties to update the BRep: $\mathbf{M} : \mathbb{B} \times \mathbb{A} \times \mathbb{P} \to \mathbb{B}$. Finally, the F operators also update the BRep, but take only the BRep as input: $\mathbf{F} : \mathbb{B} \to \mathbb{B}$. The MPI thus functions by composing a sequence of mapping

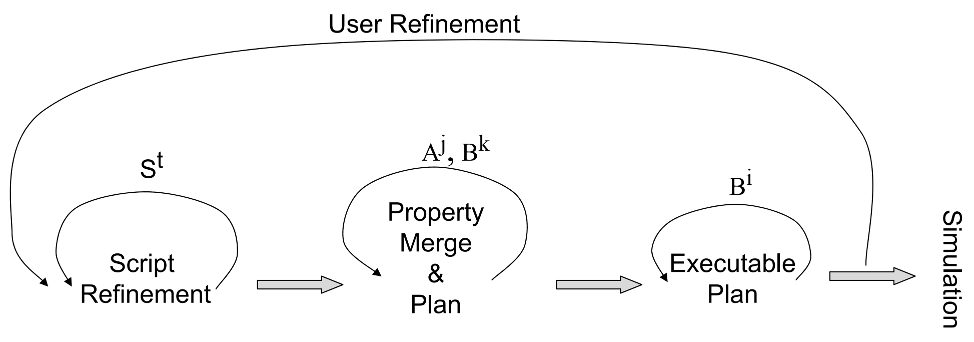


Figure 7.2: Generation of an executable movement plan, showing the three stages: script refinement, property mapping and BRep refinement.

operations: $\langle Q_1, \ldots, Q_k, \alpha_1, \ldots, \alpha_l, M_1, \ldots, M_m, F_1, \ldots, F_n \rangle$ where the comma is taken to mean the composition of functions. It is the MPI's job to specify, re-order and trigger these mapping operations. The ordering described here, and implemented in the current system, is designed to minimize conflicts between the various properties and to ensure that properties are applied ahead of when they are used. For instance, it will be seen below that signal parameter properties are applied before shape calculator properties because they will be needed by the shape calculator.

In the following section, more detail will be provided on each stage in the mapping process. The crucial point of the notation is that it captures the essentially iterative nature of the construction of animation.

7.4.1 Script Refinement

The *Script*, denoted S^i and superscripted with i to denote the iteration count, defines the movements a character is to perform. As described previously, it consists of a set of time ordered tracks. Each track is populated with actions. The animator defines the initial script and can directly edit it during any iteration. It can also be modified by *generator properties.*

In building the motion plan, the MPI begins by applying all the generator properties. Generator properties can either be included as part of the character sketch or specified as animator edits. As mentioned above, these properties add new actions to the script, which can be represented very simply as

$$MPI : S^i \rightarrow S^{i+1} \; .$$

This can be further refined by considering the set of all generator properties $\{P_1, \ldots, P_n\}$ where P_j is some element of this set and

$$P_j(S^i) = S^{i+1} \; .$$

The operator Q is then quite simply the application of a generator property that will add one or more actions to the script:

$$\langle Q_i \rangle = \langle P_j \rangle \; .$$

Once the generator properties have been dealt with, the remaining edits in the character sketch and animator edit lists act to attach properties to actions in the script. The MPI next performs this binding through the α operators. This can be represented as:

$$\alpha : S^{i+1}(\mathbf{A}^j), CharacterSketch, AnimatorEdits \rightarrow S^{i+1}(\mathbf{A}^{j+1}) \; ,$$

where $\mathbf{A}$ is the initial set of actions and $\mathbf{A}^{j+1}$ is the evolving set of actions as they are augmented with further properties.

Again, we can further refine the action of our operator. After the application of the generator properties, all the remaining edits will specify attachment properties. Consider the set of all remaining character sketch edits, $edit_c$, and the set of all remaining animator edits, $edit_a$. Each edit specifies a property and a set of actions to which the property is attached. The character sketch edits can thus be represented as the sequence of operators

$\langle \alpha_1, \ldots, \alpha_m \rangle$ denoted $\alpha_{\mathbf{c}}$ and the animator edits can be represented as the sequence of operators $\langle \alpha_1, \ldots, \alpha_m \rangle$ denoted $\alpha_{\mathbf{a}}$. Hence

$$\langle \alpha_1, \ldots, \alpha_l \rangle = \langle \alpha_{\mathbf{a}}, \alpha_{\mathbf{c}} \rangle$$

where for any α_i,

$$\alpha_i(P, \{A\}) = \{A'\} \ .$$

Thus for an edit achieved by the operator α_i, P_i is the property specified by that edit, $\{A\}_i$ is the subset of all of the actions to which the property is attached (as specified by the action label), and $\{A'\}$ is the same set of actions with P_i attached to each item in the set. The α operator thus enacts an edit by applying the appropriate property to the appropriate action(s).

Note that the ordering in which properties are attached does not matter because each property contains a priority tag which indicates the source of the property. Before properties are merged, they are sorted based on these tags. Therefore the desired order can be established as long as all properties are attached before the merge operation.

7.4.2 Property Resolution

Once the script has been completed and all properties have been attached, the MPI uses this information to develop the executable Base Representation. This is done by performing a sequence of mapping operations: $\mathbf{M} = \langle M_1, M_2, \ldots, M_n \rangle$, where once again, the operator has the form $\mathbf{M} : \mathbb{B} \times \mathbb{A} \times \mathbb{P} \rightarrow \mathbb{B}$. The ordering of these operators is based on the type of property being applied as some property types need to be applied ahead of others. The current mapping order is as follows:

1. Generate signal parameters (M_1).

2. Apply shape solver properties (M_2).
3. Apply other shape properties (M_3).
4. Apply timing properties (M_4).
5. Apply transition properties (M_5).
6. Apply properties that need to query the BRep (M_6).

Each operator normally consists of a *merge* phase followed by an *apply* phase. Composite properties are handled somewhat differently. The merge phase is the same as for other properties, but the apply phase acts to tag the actions with a new set of low-level properties rather than writing directly into the BRep. Composite properties are processed first so that the low-level properties they generate can be merged with other active properties.

The effect of the six operators can be more precisely specified. M_1, M_3, M_4 and M_5 all take the basic form:

$$M(B^i, A, P_A) = B^{i+1},$$

where M is the mapping operator, A is an action included in the script and P_A is a set of properties of a given type associated with A. These mappings take all of the actions of a particular type that are attached to the specified action and map them to the BRep, generating its next iteration. In these mappings the properties are blind: i.e., they have no knowledge of the data that may already be stored in the BRep. The mapping can both store transition elements into the BRep and set or modify parameter values for these transition elements. Every transition element is labelled based on the action that generated it to facilitate query operations. The mapping operations are repeated for every action in the script. Some timing and transition properties will write tags into Transition Elements to request effects such as synchronization. Once all properties of this type have been applied, the timing and transition planners must be run in order to realize them.

The shape solver operator M_2 is slightly more complicated and involves multiple steps. The first operator to be applied is:

$$M(D^0, A, P_A) = D^1,$$

where D is the data store associated with the shape solver. The data store is initialized for each action and then all shape solver properties for the current action write parameters into D. The shape solver is then invoked, which solves for a pose and writes the data back into the action as SetDOFValue properties. This can be represented in operator form as:

$$\alpha(P_{ss}, A) = A'$$

where P_{ss} are the properties determined by the shape solver. The final step is to merge and apply these properties using the M_3 operator:

$$M(B^i, A, P_A) = B^{i+1}.$$

The M_6 operator has the form:

$$M(B^i, A, P_A(B^i)) = B^{i+1}.$$

Here, the properties can query the BRep and condition their behaviour based on current state of the BRep, as established by previous iterations.

Merge

In general, an individual action will be tagged with multiple versions of a property, such as *SetDuration*. These different versions of the properties come from the different input sources: the initial action description, the character sketch and animator edits. It is important that all the applied properties can potentially affect the final motion, rather than one property replacing all earlier properties of a given type. This is what, for instance, allows the character sketch to *adjust* an existing movement sequence, rather than *defining* it. To provide a simple, concrete example, the sketch for a high tempo character could include a *SetDuration* edit that decreased the action durations by 20 percent, acting relative to the original duration properties that were specified as part of the action descriptions.

Before a property is written into the Base Representation, all properties of a given type acting on a specific action level[2] are merged. This resolution process effectively leaves a single property of each type attached to an action level. This merge process involves three steps. First, properties are pushed down to the lowest level of the action hierarchy at which they can act. This ensures that all properties of a given type will be at the same node when the merge operation is performed. Second, properties are sorted based on priority level. Priority level is determined based on the property source (default order: action description first, then character sketch, animator edits are last) and the level in the hierarchy the property is define at (higher levels being used first). Finally, the sorted property list is merged. Prototype merge functions are available which support merges that overwrite (ABS), scale by a factor (REL), average (AVG) or add (ADD) parameters and properties can also define their own merge semantics. This allows the property designer to decide how a property can best be merged. As an example, the *SetDuration* property supports absolute and relative duration specifications, the latter acting to scale a lower priority absolute duration. These edits can also be defined to behave locally or cascade down the timeline, shifting future TElements to accommodate the local duration change.

[2]e.g. pose two of an action called “wave”

Apply

Once the properties have been merged, the application process is quite straightforward. The one property of each type that is still active at an action node has its *apply* method called and simply writes its data either into the BRep or a data store associated with a solver. The various output behaviours of properties were summarized in Table 7.7. Once data has been committed to the BRep, it can be queried by future properties and filters.

7.4.3 BRep Refinement

Once the action-based properties have been applied, the BRep can be further refined through filtering or post-processing. These operators act in a similar manner, where for a filter F,

$$F(B^i) = B^{i+1}.$$

These processes are distinguished from the M operators by the fact that they no longer make any reference back to the script, working solely from the information contained in the BRep. Examples from this stage include a filter that smoothes transition curves of a particular DOF or the Time Planner described below which enforces the timeline semantics. Filters are a powerful notion as they admit the full extent of signal processing to be incorporated into the framework.

7.5 Executing a Motion Plan

This section will outline the basic process of executing a motion plan. The details of generating the correct torques and controlling a physical simulation are discussed in Chapter 6.

The process of executing the motion plan for one time step is shown in Figure 7.3.

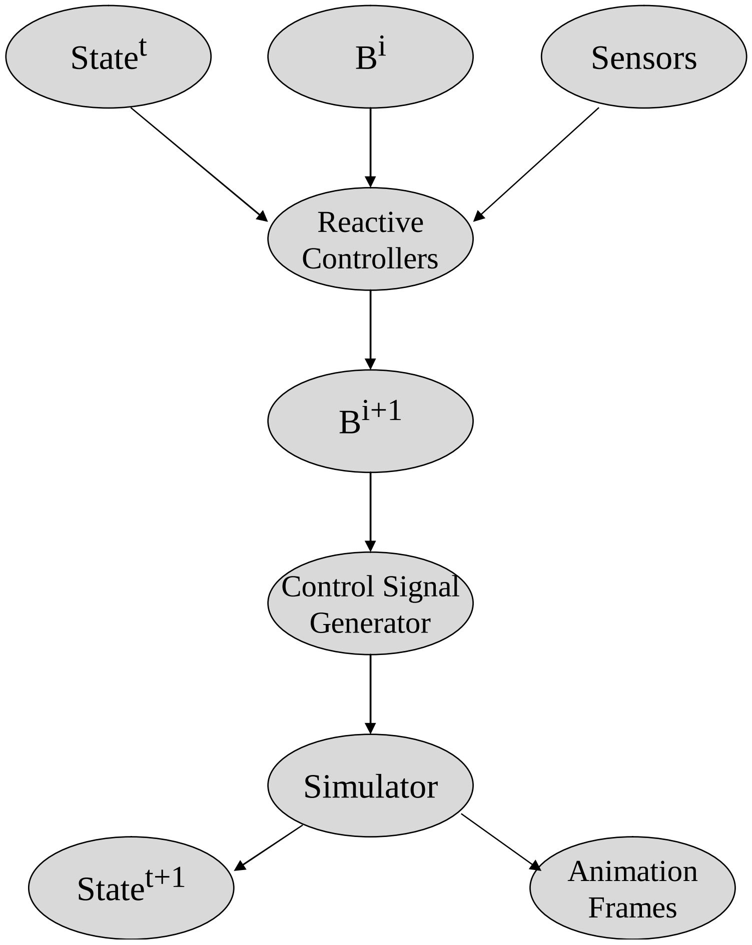


Figure 7.3: Data flow for one time step of simulation.

Reactive controllers allow the character to adjust its behaviour based on its state at the beginning of the time step. This is particularly important in dynamic simulation, where the affect of an action is not completely known ahead of time and adjustments may need to be made in order to ensure that a motion completes successfully. The main reactive controller in our system is used for balance control.

Reactive controllers receive the system state, the current BRep, and sensor information as input. *State* is a vector of position and velocity values for every DOF in the character. Reactive controllers can then update the BRep to attempt to better achieve the requested motion. Expressing this more formally, for the time step starting at t, a reactive controller R will update the BRep:

$$R(B^i, s, t) = B^{i+1}$$

where s is the current state of the character. As an example, the balance controller adjusts the desired DOF values for the charcter's lower body at each time step in order to achieve the balance adjustments, knee bends and pelvic twists specified in the signal parameter tracks of the BRep. Reactive controllers update the BRep by inserting *control params* into the DOF tracks that change the control parameters for the current time step.

Once all the reactive controllers have made their adjustments, the *Control Signal Generator* takes the DOF tracks of the updated BRep as input and produces the control information required by the current simulator. For the kinematic simulator, this information consists of the value of each DOF. The dynamic simulator requires a torque for each DOF. The torque values are generated by either a proportional derivative or an antagonistic actuator positioned at each DOF. The details of this process were described in Chapter 6.

The simulator takes the input from the control signal generator and advances the simulation Δt seconds, updating the character state and producing animation frames. The dynamic simulator is based on code generated by the commercial software SD/Fast [76]. The kinematic simulator implements a standard articulated body hierarchy.

7.5.1 Balance Control

The balance controller provides a useful example for understanding how a reactive controller functions. At each time step, the balance controller will look at the actual state of the character and compare this with the desired state. It will then insert control params into the tracks controlling the lower body DOFs in order to bring the character closer to the desired state.

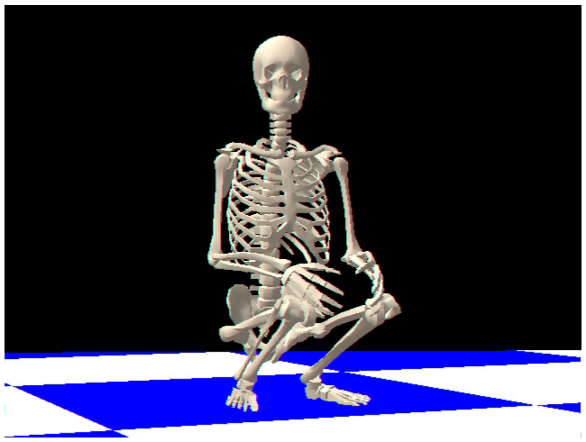


Figure 7.4: Frame from a crouch animation using a feedback based balance controller.

Movements are often most interesting near their balance limits [85], which makes balance control for expressive movement particularly difficult. Our system provides two different dynamic balance controllers, and one for kinematic balance control. All the controllers employ feedback based balance algorithms. The character's centre of mass is projected onto the floor plane and the error between the actual and desired location of the projected centre is used to adjust the character's ankle angles in order to reduce the error. As long as the projection remains within the support polygon of the character – the convex hull of the edge of its feet – the character will remain balanced. Balance is implemented as a reactive controller so that the control signal can be updated each time step in reaction to the current state of the character.

The first dynamic controller, based on [173], provides true balance control in the sense that the character will fall over if it fails, but the controller has a stability region smaller than the range of movements we wish to control and only supports limited lower body adjustments–forward and side sway, but no pelvic twists, or unequal knee bends. The controller operates by adding an offset to the desired ankle angles in order to reduce the error in the location of the centre of mass. The offsets are calculated using the following proportional derivative relations:

$$\text{offset}_z += k_{p1}(X_D - X) - k_{d1}\dot{X} \tag{7.3}$$

$$\text{offset}_x += k_{p2}(Z_D - Z) - k_{d2}\dot{Z} \ , \tag{7.4}$$

where offset$_z$ is the adjustment made to the Z DOF of the ankles, X_D is the desired location of the projection of the COM, X is the actual location, $\dot{X}$ is the rate the projected location is moving at and k_p and k_d are gains on the proportional and derivative terms respectively. The terms in the offset$_x$ relation are defined similarly. A frame from a crouch animation based on this controller is shown in Figure 7.4.

The other dynamic controller and the kinematic controller are both based on the kinematic balance algorithm described in Section 5.3 which supports a full range of lower body movements. For dynamic simulation, the controller makes use of "sticky ground" which uses simulated springs to help hold the feet in position, thus making the balance control task easier. This controller allows us to simulate a wider range of movements, taking advantage of tension effects, while not needing to worry that the character will move out of the controller's stability region and fall over. The ground models are described in Section 7.7.

7.6 Planners and Solvers

When designing properties, there is a trade-off between embedding all the functionality required to achieve a particular effect within a particular property versus creating a common code base for functionality that will be shared by multiple properties. The former maximizes encapsulation, while the latter encourages code reuse and makes it simpler to achieve planning effects that will affect multiple actions. It is not desirable to replicate complex functionality like the body shape solver in multiple properties. Both approaches are used in the system. Many individual properties completely define how they update the BRep. Other properties work by triggering either a planner or a solver. Two planners/solvers have currently been implemented, one for shape and one for timing. The term *planner* is used when the process receives the entire BRep as input and may operate across several actions in order to achieve the desired constraint. *Solvers* act locally on a single action or pose. The *body shape solver* is described in Section 5.2 and the *time planner* is described in Section 8.2.2.

Each solver or planner is controlled via properties that either write constraints into a data store associated with a solver, or place tags into the Transition Elements upon which the planner is to act. Multiple properties parameterizing a solver for a given action, potentially from different channels, can be merged before the solver is invoked. This provides for an efficient layering of properties.

While most properties in the system are associated with a particular action or group of actions, planners have a view of the entire BRep. This allows them to more efficiently arbitrate between conflicting constraints associated with different actions. Considering the time planner for example, individual properties can request synchronization constraints, scaling operations, shifts etc. and the planner can balance these constraints to determine a solution that is an appropriate trade off where the constraints produce conflicts.

A related category of processes that would be worth investigating in the future is *dynamic tuners*. Some dynamic properties are only manifested once the final dynamic simulation is complete. For instance, precise timing for loose movements or whether a character falls during a difficult move will not be apparent until the final simulation is complete. If the simulation does not meet the animator's expectations, the BRep will need to be adjusted so that the final animation meets the animator's desires. This could be achieved through an automatic, iterative process of adjusting the BRep and simulating until the correct parameters are found. *Dynamic tuners* could be used to run this process. Note that such tuners would have a longer feedback loop than other

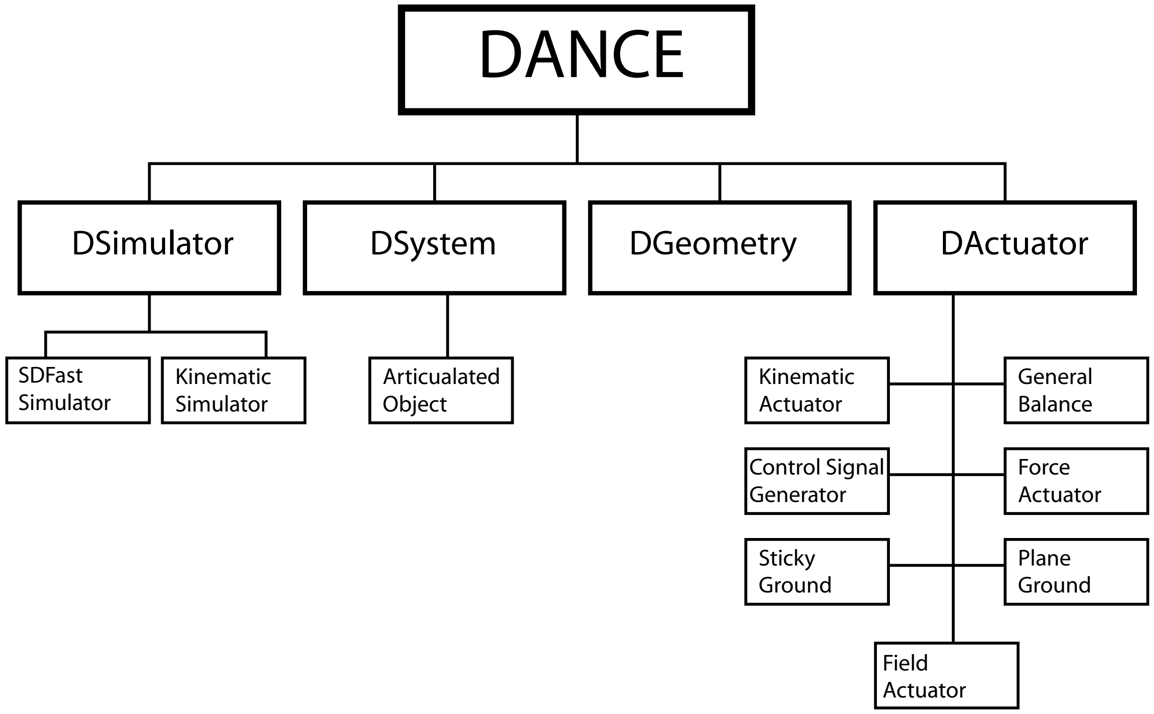


Figure 7.5: Integration of our software framework into the DANCE architecture. The plugins used in our system are linked to four the main plug-in categories

processes that modify the BRep as multiple simulations would need to be generated in order to correctly update the BRep. The other processes in the system do not analyze the final animation sequence, instead providing it to the animator for evaluation. It might also be possible to create a filter that made adjustments to the BRep based on a priori knowledge of the limits of the simulator.

7.7 Integration with DANCE

Our software is built on top of the Dynamic Animation aNd Control Environment (DANCE) developed by Ng-Thow-Hing and Faloutsos [119]. DANCE is structured around four main plug-in categories: DSimulator, DSystem, DGeometry, DActuator. Figure 7.5 shows the basic DANCE architecture with the four main plug-in categories. The plug-ins used in our system are shown linked to these base classes.

DSimulator

Simulators are used to advance the state of the system being modeled. The *SdFast-Simulator* was provided with the system. SD/FAST provides an O(n) implementation of the equations that govern a skeleton's motion and these are integrated using a fourth order Runge-Kutta integration scheme. The SdFastSimulator was extended to provide the option of semi-implicit Rosenbrock integrator [132]. The *KinematicSimulator* simply updates the character's state using DOF values received from the ControlSignalGenerator.

DSystem

A *system* in DANCE is any entity that the user might want to animate. In our work, the skeletons are the main systems. These are generated using the *ArticulatedObject* plug-in which implements a generic skeleton based on an SD/FAST file and was provided with the system. The articulated object consists of *links*, which are the limbs of the skeleton, and *joints*, which are the attachment points of the links. In the implementation, each link is also an instance of a system.

DGeometry

The DGeometry plug-ins are used to hold and manipulate character geometry. No additional geometry plug-ins have been added to the system. Each limb in the ArticulatedObject class contains a DGeometry object that defines the limb geometry.

DActuator

All forces in the system are applied through *actuators* (instances of DActuator classes). Any class that inherits off DActuator must implement a BeforeStep function that will be called before each time step in the simulation and an ExertLoad function that will be called by the simulator to cause the actuator to apply its forces. The *Force Actuator* can apply arbitrary forces on the system at arbitrary times. It is used to shove the character. The *Field Actuator* will apply a uniform force on every system. It is used to implement gravity. The *Kinematic Actuator* does not actually generate any forces, but rather acts to record the simulation for playback. All three of these actuators were included with the DANCE distribution.

PlaneGround and *StickyGround* both implement floor models. PlaneGround is based on an actuator received with the system and implements any piece-wise linear terrain [44]. Collisions between a system and the ground are detected through monitor points that are attached to the system. While in collision, PlaneGround generates reaction forces based on a penalty method that implements a Coulomb friction model. When a monitor point penetrates the ground, a reaction force is generated from the entry point to the current location of the monitor point. This force is generated by a linear spring and damper. Friction cones are used to model both static and sliding friction, allowing a character to slide along the ground if the tangential forces exceed a friction threshold. No force is generated for monitor points above the ground. StickyGround works to hold monitor points that are in contact with the ground in a fixed location. A spring and damper is again associated with each monitor point that is touching the ground, but unlike with PlaneGround, they generate forces if the monitor point rises above the ground as well as if it penetrates the ground. Sticky ground can be used to expand the stability range for movements near the balance limit.

The *GeneralBalance* actuator is used to adjust the character's balance point, maintain a standing posture, implement torso twists, knee bands and crouches. All of these attributes are specified through parameter signal tracks in the BRep. GeneralBalance will normally control all the lower body DOFs of the character while working to achieve these attributes. More detail on the balance algorithm is provided in Section 7.5.1. GeneralBalance does not directly apply forces. Instead, it is called before each time step, calculates the correct control information for each lower body DOF for that time step and writes this information into the BRep as a set of control params.

Most of the system architecture is embedded in DANCE through the *ControlSignalGenerator*. This actuator contains the BRep, and at each time step, generates the actuation torques for every DOF in the skeleton. The script, properties and MPI are also all associated with this actuator plug-in.

7.8 Summary

The concept of a property allows the system to encapsulate particular ideas from the arts literature. Key to the operation of the system is the use of explicit representation and hierarchical construction. These allow an animator to refine high level edits, such as the character sketch or shape sets, by adjusting the components that define them. This allows the system to effectively support both exploration and refinement.

The set of properties is diverse and open ended. Defining a common base representation allows these properties to be combined in a final motion sequence. Indeed, *any* property can be defined as long as it has a precise semantics that indicate how it changes the base representation. There is a well defined process that moves from the high level inputs that are useful to the animator to the low-level base representation. This base representation can then be executed to create an animation sequence. Reactive controllers allow the character control to be adjusted to reflect the actual state of the character during simulation.

In summary, the system provides a method whereby ideas can be taken from the arts literature, given a precise semantics and incorporated into an animator's workflow. The system is open ended, so it can be continuously evolved to better meet an animator's needs.

Chapter 8

Timing

Time is the measure of movement.

Auctoritates Aristotelis

In our earlier discussion of the Arts, we found that the three main timing properties are succession, tempo and rhythm. Rhythm describes the metric structure of a movement sequence, whereas tempo describes the speed at which the sequence is completed. For instance, the rhythm of a movement series could be to spend less time moving between various poses (short beat) and then hold those poses for a relatively long time (long beat). This pattern could be done either at a fast or slow speed, thus reflecting a change in tempo. While tempo and rhythm reflect the overall pattern of a sequence of movements, succession involves the timing of joints within an individual movement. In forward succession movements, the motion begins at the base of the character's spine and progresses out to the extremities. Joints further from the spine will begin the motion later than joints near the base of the spine. Reverse successions move from the extremities to the base of the spine and if there is no succession, all the joints begin to move at the same time. Successions relate to a sense of flow. Finally, it should be noted that while rhythm and tempo relate to the overall structure of a sequence, in practice we often adjust the duration of a particular movement independently of the movements around it.

In the system, all actions get mapped to a set of transition elements that are placed in the BRep; a sequence of time ordered tracks for each DOF in the skeleton. Transition elements require three pieces of temporal data: a start time, a duration and a hold time. The duration specifies how long it should take to make a transition from the value at the start of the TElement to the desired value for that TElement. The hold time indicates how long this value should be held before a new transition begins. Thus a transition element is active from $startTime$ to $stopTime$ where $stopTime = startTime + duration + holdTime$.

All timing effects are affected by changing the length and position of the TElements in the BRep. Different relationships can exist between the TElements and the animator may or may not want to maintain these relationships with any given time edit. TElements that are part of a single action will have a defined offset relative to one another. For example for a given transition, all the joints in the arm may start and end at the same time, or the wrist DOFs may lag the forearm by 0.2sec, or twenty percent of the duration of the forearm's movement. In a similar manner, there may be relationships between the TElements of different actions. For instance, if a punch and lunge movement are layered together, the punch may need to start a certain time relative to the start of the lunge to make the motion appear natural. With a given edit, an animator may wish to maintain or break these relationships. For example, he may wish to maintain the relative timing while increasing the tempo of a set of actions, or he may wish to break the relative timing to redefine the relationship between actions or between joints within an action. Edits of the latter type are normally designed to refine the definition of an action.

Given these potentially conflicting demands, a set of invariant rules cannot be defined for how time edits should cascade to other TElements that are part of the same action or elsewhere on the timeline. Consider sequential actions that may have a fixed duration pause between when one action stops and the next one starts. If the first action has its scale reduced, is the second action shifted to maintain the same spacing, or does it stay where it is to maintain the same absolute start time? There is no single, correct answer to questions like this as they depend on the context of the movement and the intent of the animator. For this reason, the animator must be allowed to decide which effect is intended in a given situation.

Most timing properties can be applied at arbitrary levels within the action hierarchy. They can also be applied across actions, over a particular time span, or over the entire timeline. When applied to a single Transition Element or a small subset of Transition Elements that make up an action, they can vary the relationship of these transition elements with the rest of the action. Applied at the action level, time properties will be uniform across the action, acting to scale it or shift it while maintaining the relative timing between all the transition elements that make up the action. Whether the effects of an edit on one action should propagate to change the timing of actions around it depends on the animator's specification. Properties that enforce a relationship between actions do so by storing information in the TElements associated with these actions that can be used by the time planner, as discussed below.

Two mechanisms are used to modify time data in transition elements: properties, and the *time planner*. All timing edits are controlled through the application of properties, but some effects may actually be enforced by the time planner. During generation of the motion plan, all of the time properties will first be merged and applied, and then the time planner is invoked. Time properties may directly change the time data in the TElements or they may tag TElements with additional information that is used by the time planner. The time planner is used to enforce relationships among transition elements. It also provides the semantics for an elastic timeline, which will stretch and compress either to create room needed for longer actions or to fill gaps.

8.1 Time Properties

There are several time properties that we wish to control. These are described below.

8.1.1 Length and Scaling

Scaling and *length* edits can be applied to adjust or set the hold time and duration. These edits can be applied as properties to any level in the action hierarchy. For instance, they could be applied to the entire action, a particular pose or a single DOF in a pose. The animator can control whether the edits just affect the TElements to which they are applied or whether they also shift the start times of related TElements to maintain relative timing. This is discussed in more detail in the implementation section below.

Two properties are used to control length and scaling: *SetDuration* and *SetHoldTime*. Both take a value and either a *REL* or *ABS* flag to indicate how they should be combined with previous edits. Consider the following sample edits:

1. SetDuration Wave 0 1 ABS 4
2. SetDuration Wave 0 1 ABS 2
3. SetDuration Wave 0 1 REL 1.5

The result would be to make the duration of pose 1 in cycle 0 of the wave action 3 sec.: the first edit sets the duration to be 4 sec., the second overrides this and sets it to be 2 sec., the third scales this by a factor 1.5 yielding the final duration of 3 sec..

8.1.2 Placement and Succesions

Placement edits shift the position of TElements in the time-indexed BRep tracks. These edits can also be applied at arbitrary levels of the action description hierarchy.

Succession edits are simply a particular form of a placement edit. The *VarySuccession* property takes a forward or reverse flag, along with the amount of the offset to be applied in seconds.

8.1.3 Synchronization

Synchronization edits act to either synchronize actions to each other or fix them to particular points in time. Three forms of action-action synchronization are supported: TElements in different actions can be set to start or end at the same time, or actions can be linked to each other to maintain the same relative spacing.

Synchronization with time code presents a more interesting situation as it will normally either leave gaps, create overlap or require scaling to maintain a dense timeline. The current implementation does not support overlap, although it could be extended to either blend or arbitrate between overlapping TElements. There are three scaling options that can be used with synchronization commands:

1. Do not scale. Leave gaps. Raise an exception for overlap.
2. Scale only to avoid overlaps
3. Scale to either avoid overlap or fill gaps.

8.1.4 Elastic Timeline

The BRep timeline is elastic and this is enforced by the time planner. Timing edits can potentially cause TElements to overlap or gaps to emerge between actions. The system does not currently support overlapping TElements and additional gaps are generally not desired. The elastic behaviour of the timeline is designed to deal with these two problems. The time planner will locally stretch the timeline to eliminate overlap and locally compress the timeline to eliminate gaps.

More precisely, the following behaviour is enforced. If a TElement overlaps another TElement, all TElements that are part of the overlapping pose and all TElements that occur after the end of the overlap will be moved forward in time by the amount of the overlap, thus eliminating it. If there is a gap, all future actions are shifted back so that the first TElement to start after the gap now starts at the beginning of the gap.

Occasionally, empty gaps may be desired in the timeline. This is achieved with a separate track in the BRep that contains *spacers*. These are TElements that simply serve as space fillers, preventing the elastic behavior of the BRep from collapsing a gap. Spacers can be affected by time edits, such as a duration scale, but have no effect on the final animation.

The initial script may include gaps between actions. It is assumed that these gaps are intentional and spacers are automatically generated to maintain these gaps.

8.2 Implementation

This section explains how the properties and time planner work together to achieve the various time properties and how the elastic timeline behaviour is achieved.

8.2.1 Properties

Length and Scaling Edits

SetDuration and SetHoldTime operate in the same manner, directly changing the duration or hold time of a TElement when they are applied. They do not change the start time of the TElement. This can lead to unnatural behaviour if the start time of different DOFs is offset within an action because a scale applied to the action will change the relative timing of these DOFs. By default, the Time Planner accommodates this by shifting the start times of TElements to affect a local linear scale on the action with the scale's origin at the start time of the action. The scale properties can override this functionality if the animator wishes to change the relative timing within the action.

To complete its work, the time planner needs more information than the final duration or hold time of the TElement. For this reason, the properties will update the value of the duration or hold time and also store how much they have changed this value from its base value. This information is used to determine the scale factor for the action which is used to determine the local linear scale.

Successions

The *VarySuccession* property takes two parameters: whether the succession is normal or reverse and how much of a time offset (t) to use between the joints involved in the motion. The edit determines all of the transition elements involved in the pose and shifts their starting time based on their location in the character's joint hierarchy. For instance, a normal succession would not modify the first joint in the spine, but it would offset the next joint by t, the following joint by $2t$ etc. The succession traces down all branches in parallel, for instance, modifying the start time of both collar bones, then both shoulders and then both elbows etc. Reverse successions work in the opposite way. They do not shift the TElements associated with the extremeties, shift the TElements one joint away from the extremeties by t, etc until they arrive at the base of the spine.

Synchronization

Synchronization properties enforce a relationship between two actions by creating a link between them. The time planner examines these links and determines the correct time for the actions. For instance, consider a punch that needs to start at the same point relative to the start of a lunge. This is enforced by creating a link between the *base* (e.g., lunge) and *target* (e.g., punch) actions. The synchronization property creates this link and stores it in a list of synchronization links. The time planner adjusts the start time of the target action based on the average scaling of the base action. The target action is also locally scaled. Different link types can be created for different types of synchronization, enforcing relative timing, or forcing the start or stop time of TElements in different actions to be the same.

Edits that synchronize actions to particular points in the timeline work in a similar manner. They add a link to a synchronization list that specifies the time at which the

```
Apply time properties.
Apply successions.

//enforce local scaling and action based synchronization constraints
start at time 0, for each action:
    ScaleLocal()

//Enforce timeline semantics
StretchTimeline()  //eliminate any overlap
ContractTimeline() //fill gaps

//Enforce timeline synchronizations
SynchWithTimeline()
```

Table 8.1: Pseudocode for the time planner.

target action must occur. The actual placement of the action in the timeline is achieved by the time planner.

8.2.2 Time Planner

As was seen above, the various time properties are responsible for specifying the desired timing behaviour, but the much of the work of actually achieving the time constraints is achieved by the time planner. Time planning is a feature that benefits from a global perspective on the timeline. Properties working on individual actions can request edits that have repercussions for the rest of the timeline. The *Time Planner* implements a timeline semantics. It works directly on the BRep as a post-process, making use of tag data added to the TElements by properties, as well as data stored in the synchronization lists.

Universal scaling of the entire timeline can be easily achieved by means of filtering or by applying scale edits to all actions and invoking the timeline.

The pseudocode in Table 8.1 gives an overview of the time planning process. In the first stage, all the time properties are applied. The successions are applied last, although

```
//adjust start times to ensure a locally linear scaling
//and enforce action based synchronization constraints
ScaleLocal()
    scaleOrigin = startTime of the first TElement in action
    locally scale the action
    {
        shift start times to effect a linear scale

        if ( action synched to previous action )
            shift entire action to meet synchronization constraint
    }

    //recurse on all the actions that are syncrhonized to this one
    For all synchronization target actions
        ScaleLocal()
```

Table 8.2: Pseudocode for the ScaleLocal() method used in the time planner.

this is not strictly necessary.

After the properties have been applied, the time planner is run. It begins by adjusting the start time of TElements to enforce local scales and inter-action synchronization. The planner starts at the beginning of the timeline and processes actions in sequential order. For each action, it calls ScaleLocal, the behaviour of which is shown in Table 8.2. The origin of the local scale is the start of the first TElement in the action. For each TElement, the startTime relative to the origin is scaled based on the amount the TElement has been scaled in order to maintain the same relative timing with respect to the other TElements in the action.[1] This local scaling can be overridden if the animator's intent is to adjust the relative timing of the action instead. Next, if the action currently being processed is the target of a synchronization constraint with a previous action, the entire action is shifted to satisfy the constraint. ScaleLocal then recurses on all the actions that are synchronized to the current action.

Once the local scales and action synchronizations have been completed, the elastic

[1]This is approximate if different scales have been applied to different TElements.

timeline behaviour is enforced. StretchTimeline() starts at the beginning of the BRep and scans forward in time, looking for overlapping TElements. If an overlap is found, the earlier action is held in place, and all the TElements of the later action, along with all the TElements of subsequent actions, are moved forward in time by the amount of the overlap. This eliminates the overlap. ContractTimeline() is then called which works in a similar manner by scanning the entire timeline, but it is looking for gaps. If it finds a section of the timeline that includes no TElements, including no spacers, it contracts the timeline to eliminate the gap.

The final step is to shift actions so that they satisfy syncrhonization constraints with particular points in time. As discusses above, this can create gaps or overlap. Depending on the requested behaviour, the actions will either be scaled to eliminate the overlap and/or gaps or exceptions will be thrown when an overlap occurs.

8.3 Results

The animations discussed below are available online at http://www.dgp.toronto.edu/people/neff.

A simple bowing animation in which a character draws his arm across his body and out to the side shows the power of the succession edit. The animation consists of two poses on top of the rest pose, one of the character bowed forward and a second of the character gesturing off to the right. The basic animation generated by interpolating the poses has a stiff, robotic feel. The animator applies the succession edit with an offset of 0.2 sec for the first transition and 0.3 sec for the second transition. As can be seen in a side by side comparison of the animations, the application of the succession edit gives the movements a remarkable sense of flow. A few frames from the end of the animation are shown in Figure 8.1. The character's lower body is automatically controlled by the reactive balance controller.

Motion capture and learning approaches to animation tend to view animation as a

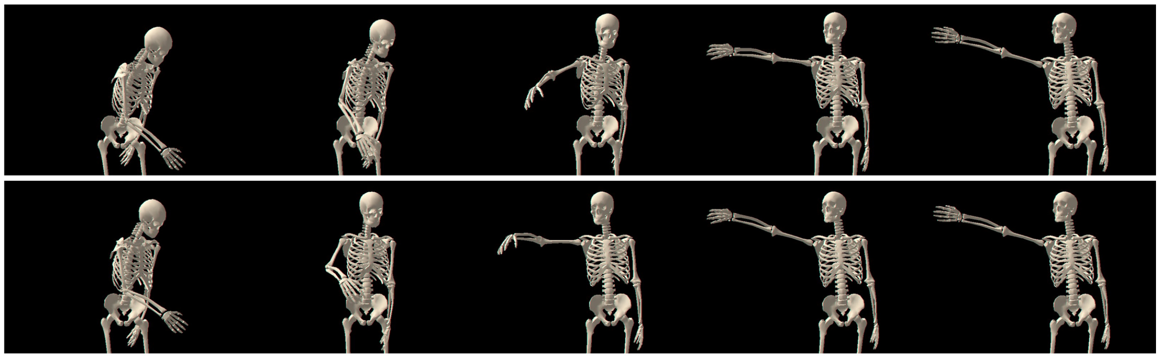


Figure 8.1: Succession Edits: The top image sequence shows frames from an unedited animation. The bottom shows the same sequence after a succession edit has been applied. Note the greater sense of flow in the lower animation. Frames are evenly spaced within the transition.

time series of vectors, each vector containing the entire body state. This view suggests a correlation between all joint angles that in general is not necessary. In our system, actions can use a subset of the characters DOFs and then be overlapped in time. An action thus provides a handle by which an animator can modify the movement of certain correlated joints independently of other joints.

To illustrate how this works in practice, two versions of simple twisting dance are presented. The first is correlated with a slow tempo 4/4 rhythm. Both the lower body twist and upper body twist and arm swings are aligned with each beat. The second sequence is based on a faster tempo 3/3 rhythm. The lower body twist is still aligned with each beat. The upper body movement from the first sequence is used only on the third beat of each measure and its motion envelope is warped to better punctuate the beat. New actions are inserted on beats 1 and 2 that simply feature a small elbow bend and head twist. The result is two very different sequences, based on the same action descriptions. This type of retargeting differs significantly from what can be achieved by motion warping a state vector.

Chapter 9

Animator Exploration and Refinement

> Fix this for all times in your memories: *On the stage there cannot be, under any circumstances, action which is directed immediately at the arousing of a feeling for its own sake.*
>
> **Constantin Stanislavski, An Actor Prepares, p.38**

One of the main objectives of the system is to support both exploration and refinement. Character work in theatre is marked by exploration and play. Finding the best way to portray a character is not a linear process. In general, actors will try, discard and possibly combine many approaches to a character before settling on a particular plan for the performance. Very often, exercises are used to help actors experiment with different potential characterizations.

In animation, the cost of experimentation is much higher than it is in theatre. While an actor can try a different approach to a scene in a few minutes, it may take an animator several hours to rough out a scene's character motion. This limits the amount of character exploration that can be undertaken in animation production and makes it much more difficult to find ideal approaches to a character.

Our system provides animators with a low-cost method for exploring different approaches to a scene. It does so in two ways. First, it provides high-level properties that require very few parameters but change many aspects of the motion in aesthetically meaningful ways. Shape sets and composite properties are examples of these. They allow an animator to rapidly explore different aesthetically meaningful areas of the movement space. Second, character sketches enact global changes to an entire movement sequence. They allow an animator to completely change the style of the piece by simply switching character sketches. The goal of both of these techniques is to provide an animator with the opportunity to explore and the option of playing with very different ideas for a character. High risk approaches can be tried at a low cost, expanding the range of possibilities that are considered when searching for exciting and original characterizations. More detail is provided below on both of these methods for exploration.

High level tools are useful for prototyping and exploration, but on their own, are unlikely to be sufficient for final production animation. Animators demand control. They want to be able to refine their animations to achieve the exact nuances that they desire. It is likely for this reason that many high level tools that do not support refinement have not seen penetration into the production environment. Tools must provide animators with the control they demand. The system was designed to support refinement. All of the high level tools are built out of low level controls. There is always an explicit representation that defines how a high-level tool achieves a desired effect by manipulating the low-level controls. Animators are free to either redefine the behaviour of the high-level tools, or more likely, switch to a low-level interface to fine tune the motion. The twin goals of exploration and refinement are probably best achieved in the body shape solver, where shape sets provide a high level interface for rapid exploration and the values set by shape sets can then be loaded into a low-level interface for fine grain editing.

The exploration and refinement framework relies on the ability to layer different properties together. Three main input channels combine to determine the style of a motion sequence: the initial set of properties included with action descriptions, the character sketch and the animator edits. Progressive refinement requires each input channel to build on, rather than replace, the work done through other channels. The character sketch, for instance, is effective because it only needs to specify a small set of adjustments to already defined actions, it does not need to provide the full action definition itself.

The layering process also allows motion to be broken down into different aspects, for instance aspects related to time or shape. Each aspect of the motion can then be defined and modified separately. For instance, the posture of a character can be modified without affecting the motion envelope. This allows character sketches and animator edits to describe specific differences between characters and modify particular aspects of the motion.

9.1 High Level Properties

Much has already been said on the high level properties in the system, so only a brief overview will be provided here. *Shape sets* and *composite properties* are the two main approaches to high-level interaction. Shape sets are a specific type of composite property and are described in detail in Section 5.5. Each shape set is designed to allow an animator to explore a particular aesthetically consistent region of pose space. They take a small number of parameters, generally an intensity value and possibly a world space constraint. Varying the parameters will vary the posture within the aesthetic region. To make whole scale changes to the motion, the animator simply changes shape sets.

Composite properties share the same goal as shape sets – to allow rapid exploration of a particular movement idea – but they can contain any combination of movement properties. Composite properties are always built out of lower level movement properties. This hierarchical definition builds in support for refinement.

A composite property such as *CreateRecoil* encapsulates a full set of movement properties, and is suitable for rapidly exploring a movement idea. It takes as input the location of the object to react to, a parameter that indicates one of several styles of reaction and an intensity value that indicates the vehemence of the reaction. The property will adjust the pose based on the location and alter pose, timing and transition features based on the intensity value. An animation is available online that shows three different style recoils.

9.2 Property Layering

A simple animation sequence which is available online illustrates how properties are layered together to generate a final motion piece. The character reaches for an object, brings it to her in order to inspect it and then shows it to the audience. An initial version of the animation has the basic timing and defines the reach constraints. To each pose, we then add the *beauty line* reach property. This takes the constraint and one additional parameter that controls the intensity of the pose. The tempo is reduced to give the animation a more languid feel and forward successions are added to increase the sense of flow. Transition warps are used to give the animation a bit more pop, and finally, it is dynamically rendered.

We use our shape calculator to solve for the poses, making for a very compact action description. The *ReachForObject* action, using the beauty line property, is defined as follows. An additional hand bend is also specified.

```
<Pose>
Time transition 1
Time hold 0.1

2DHermite 1 0 1 0

<dof2> Right_Hand_z
DesiredAngle -.3
</dof>
```

```
<CalcPose>
SetReach beautyLine -.5 -.3 -.9 .8
</CalcPose>

</Pose>
```

Ultimately, such descriptions would be generated as the output of interactive tools.

This layering example shows how a final piece of motion is iteratively defined. It also illustrates how the definition of various aspects of the movement can be varied independently.

9.3 Character Sketch

A character sketch performs global edits on the entire script. It will make the same modifications to every action. These modifications will usually behave relative to the properties that are already associated with the actions. In this way it varies action definitions rather than specifying them. A character sketch can also add new actions to the script by specifying generator properties.

Globally modifying all actions in the same way is an abstraction of how an actor or an animator would normally create a character. Nonetheless, it is worth noting that psychological research suggests that people do indeed have "style factors", certain movement tendencies that are consistent across all the normal activities in which they engage [52]. The character sketch builds on this idea.

Normally a character sketch will modify each of the three aspects of motion: shape, timing and transition. An animator, however, is free to emphasize the aspects of motion he chooses. For instance, one character might be characterized by a hunched back while another by a tendency to make movements at a high tempo. A character sketch is defined in the same manner as the animator edits and indeed a character sketch is simply a set of animator edits that are applied globally. Individual character sketches can be combined and they can also be combined with sets of animator edits.

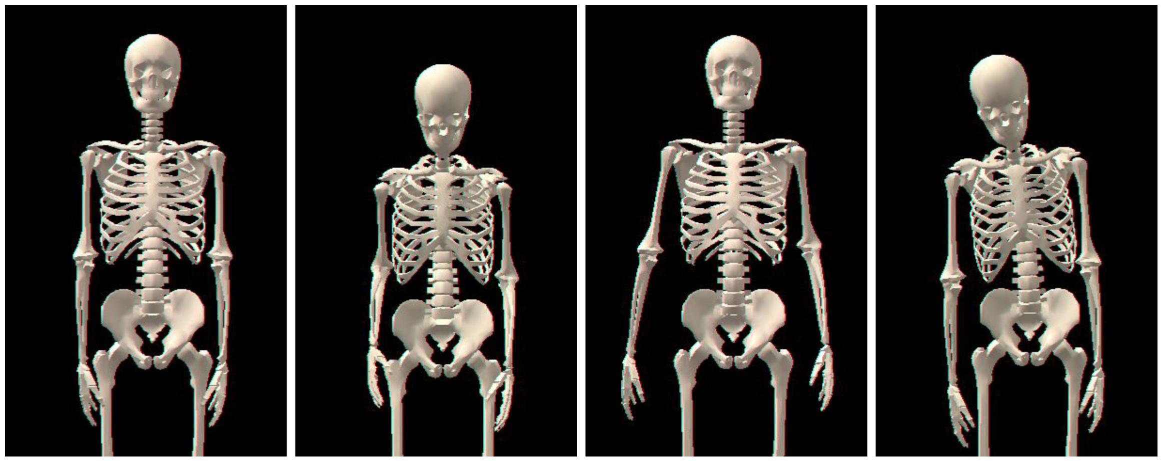


Figure 9.1: Default posture for different character sketches. From the left, the first figure is the default posture, the second is the *old man* posture, the third is the *energetic* posture and the last is the *dejected posture.*

Most character sketches contain a default posture. Posture is a powerful indicator of personality and mood, and can go along way towards generating a particular characterization. Default postures are defined using the low-level parameters of the body shape solver. They thus use the same language as shape sets and other body shape solver properties. As with the rest of the sketch, it is normal for a posture to have an incomplete specification that includes only those features that are most important for a character. In one case, this may be a particular curve in the spine. For another character, it may be the collars held up and the arms held out from the character's side. These default shape parameters are blended with shape parameters associated with actions, either through a shape set or through the application of low-level shape properties. The default posture can thus modify the character's pose, combining with pose requirements of each action, rather than defining the final pose. The system for combining shape properties is described in Section 5.5.2.

In the following sections, three different character sketches will be described. One is for an *old man*, one generates a younger, *energetic* character and one generates a *dejected* character. The default posture associated with each sketch is shown alongside the default posture in Figure 9.1.

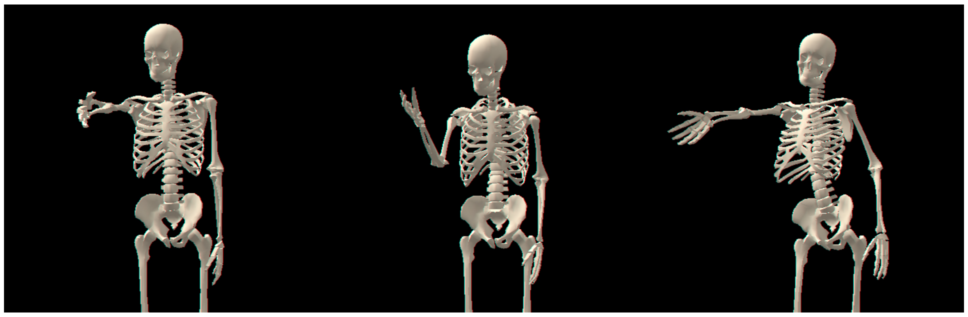


Figure 9.2: Three frames from an animation in which a character shows an object to the audience. Left: from the base animation. Middle: an “old man” character sketch has been applied. Right: beauty shape edits and an “energetic” character sketch have been applied.

A character sketch can be used to prototype different approaches to a character. This is shown with a simple animation sequence that shows a character greeting a friend, beckoning him over and then showing him an object to the right. Two different character sketches are applied, one for an old man, one for a more energetic character. They produce two very different animation sequences, allowing the animator to quickly play with different styles. Due to the global nature of the character sketch, some of the adjustments may not be appropriate for some of the actions. In a subsequent pass, the animator can layer in additional edits to fine tune the motion.

Edits and sketches can be freely layered together. For instance, we combine the beauty line shape edits with the “energetic” character sketch to generate the third frame in Figure 9.2. The first frame shows the same pose from the base animation and the middle frames shows the “old man” sketch applied to the base animation.

9.3.1 Old Man Character Sketch

The character sketch for the old man is shown in Table 9.1. On any row, the first term is the property to apply, the curly braces specify the recipient action (* indicates a global edit) and the rest of the entries are parameters. The specifier {* * * 16} associated with the SetDOFAngle edit specifies that the edit should be applied to DOF 16 in any pose, in any cycle, in any action. DOF 16 is the y-angle of the neck. It should be noted that in a production system such information would be entered through a user friendly GUI in which the joint to control would be selected on the skeleton. The prototype system described here relies on script files.

The extent variation is applied post-burn in, after a pose has been solved for that may have involved a world space constraint. It should not be used on actions with such touch constraints as there would be no opportunity to maintain them. The rest of the properties do not interfere with constraint satisfaction. By using a different scaling on the hold and duration times, both the character's rhythm and timing can be adjusted. The old man for example takes longer to prepare for a movement than the default character.

The edits are generally consistent with the stereotype of an old man. He moves more slowly, takes more time to prepare his actions, moves over a more limited joint range and makes less extreme movements. The transition functions have been flattened to give a steadier pace to his movements. Some high frequency shaking motion has also been specified for the forearms, representing what would often occur with age or the onset of Parkinson's disease.

The default posture for the old man is shown in Table 9.2. He has a bad hunch in his back, his collar bones are curled inward and down, his knees are slightly bent and his arms are bent as well. This default posture is shown in the second frame of Figure 9.1. The posture actually looks better when the action specifies a *look at* constraint, a common occurrence, and the character lifts his head while otherwise maintaining the posture.

9.3.2 Energetic Character Sketch

The energetic character sketch is shown in Table 9.3. It is largely the opposite of the old man sketch. The duration of movements is decreased and the hold time is almost eliminated. Extent is increased so the character takes up more space with its movements.

```
#Scale duration (increase it)
SetDuration {*} REL 1.8

#Scale hold time (increase it)
SetHoldTime {*} REL 3

#adjust shape settings
SetCharShapeParam {*} 1 load oldMan.shp

#flatten the transition functions
SetTransitionFunction {*} AVG 0 0 0 0

#reduce extent
VaryExtent {*} 0.8

#add a normal succession
VarySuccession {*} normal 0.01

#reduce neck y rotations
SetDOFAngle {* * * 16} REL 0.5

#add some shake to the right forearm
SetShake {* * * 33} ABS 0.017 7

#add some shake to the left forearms
SetShake {* * * 24} ABS 0.017 7
```

Table 9.1: The *old man* character sketch.

```
#add a slight bend to the arms
RArmLen .9
LArmLen .9

#create a hunch in the spine
SagClass SmallC
SagCentre 1.6

#create a forward and downward hunch with the collar bones
ColYCentre 1
ColZCentre -2.6

#arms should hang down when not in use
ArmsVertical true

#add some bend to the knees
rKnee .5
lKnee .5
```

Table 9.2: The default posture for the *old man* sketch.

Transition functions are warped so that motions start more quickly.

Table 9.4 shows the default posture that is used with the energetic sketch. The character thrusts its chest out and pulls its shoulders up and back. The arms are held out from his side so the character can indulge in the use of space. Arms are bent slightly to suggest some tension in the elbows.

9.3.3 Dejected Character Sketch

Table 9.5 shows the *dejected* character sketch. *Dejected* is an interesting example as it shares much in common with the old man, and yet it should still make a distinct impression. Recall Frank Thomas's comment on the difficulty of applying motion stereotypes:

> Even the moves that seemed to define attitudes or mood proved to be annoyingly elusive. To make a figure sad, we slumped the body, dropped the shoulders and the head, dragged the feet when walking, and timed the actions

```
#increase the speed of movements
SetDuration {*} REL .8

#almost eliminate the hold time between movements
SetHoldTime {*} REL .02

#specify a default posture
SetCharShapeParam {*} 1 load energetic.shp

#warp motion envelope to start more quickly
#this could also be done with a tension edit
SetTransitionFunction {*} AVG 0 0 3 0

#increase the extent of actions
VaryExtent {*} 1.3
```

Table 9.3: *Energetic* character sketch.

```
#slightly shorten the arms
RArmLen .95
LArmLen .95

#keep the arms out from the character's side (slight asymmetry)
#angle of arm relative to side
RArmAngle 10
LArmAngle 6.5

#arch backwards in the sagittal plane
SagClass SmallS
SagCentre 1.4

#raise the shoulders and pull them back
ColYCentre -.9
ColZCentre 1

#arms should hang down when not in use
ArmsVertical true
```

Table 9.4: Base posture for the *energetic* sketch.

> slowly. But these elements of sadness could also mean despair, or listlessness, or exhaustion. [158, p.21]

Many of the basic adjustments for the aging character and the dejected character are the same, so care must be taken to distinguish the two. Posture adjustments will be discussed below, but the rest of the character sketch will be discussed first.

The dejected sketch does feature an extent edit that reduces the use of space, but this is less so than the old man sketch and joint ranges are not restricted. Similarly, both sketches slow the timing, but the exact value is different. The most significant difference is in how the motion envelope is warped. The old man sketch contains an edit that generates slightly flattened kinematic transition functions. The dejected sketch instead uses tension reduction to warp the motion envelope. For each transition, the tension will start high and end quite low. This causes the motion to start slowly and speed up towards the end. The low final tension also leads to overshoot effects as the arm will waver in its final position or sway at the character's side. This looseness indicates a sense of indifference that is consistent with dejection. The tension decrease also gives the sense that the character hurls himself into each motion, but looses energy part way in. Again, this is consistent with dejection. Many tension edits must be specified as different tension values should be used with different joints. It would be good in the future to parameterize the tension range so that the edit merely had to say "move from high to low" and the correct numerical values would be inserted for each joint.

The default posture for the dejected sketch is shown in Table 9.6. Like the old man, the collars are dropped, but they are not curled forward. Also a larger curve for the spine is used to suggest less of a hunch and more of a sense of weight. While the old man is fairly symmetrical, the dejected posture is assymetrical to help generate a sense of unease.

```
#increase the duration
SetDuration {*} REL 1.3

#increase the hold time by a factor of 5
SetHoldTime {*} REL 5

#set the default posture
SetCharShapeParam {*} 1 load dejected.shp

#apply a slight forward succession
VarySuccession {*} normal 0.1

#reduce the extent of actions
VaryExtent {*} 0.9


#decrease shoulder tension during each movement
SetTensionValues {* * * 20-22, 29-31} ABS 400 100
#scale damping to match tension
SetDamping {* * * 20-22, 29-31} ABS 10

#decrease tension for abdomen and chest joints in the spine
SetTensionValues {* * * 6-11} ABS 800 500
#scale damping appropriately
SetDamping {* * * 6-11} ABS 60

#decrease tension for neck and head joints in the spine
SetTensionValues {* * * 12-17} ABS 100 50
#scale damping appropriately
SetDamping {* * * 12-17} ABS 8

#decrease tension in the collar bones
SetTensionValues {* * * 18-19, 27-28} ABS 200 60
#set an appropriate damping
SetDamping {* * * 18-19, 27-28} ABS 8

#decrease forearm tension
SetTensionValues {* * * 23-24, 32-33} ABS 400 200
#set appropriate damping
SetDamping {* * * 23-24, 32-33} ABS 30

#decrease hand tension
SetTensionValues {* * * 25-26, 34-35} ABS 10 1
#set appropriate damping
SetDamping {* * * 25-26, 34-35} ABS .5
```

Table 9.5: *Dejected* character sketch.

```
#arch the character to the right in the coronal plane
CorClass LargeC
CorCentre .4

#arch the character forward in the sagittal plane
SagClass LargeC
SagCentre 1.


#adjust the arms so they hang close to the character's side given the torso position
#angle of arm relative to side
RArmAngle 4
LArmAngle -3


#hunch the shoulders down
ColZCentre -1

#arms should hang down when not in use
ArmsVertical true
```

Table 9.6: Base posture for the *dejected* sketch.

9.3.4 Discussion

As previously noted, the character sketch is merely a first approximation to what an animator would do while generating a character. Indeed, it is meant as a method for quickly exploring different character types, realizing that further edits will be required to generate a final animation sequence. Nonetheless, it would be worthwhile expanding the capabilities of the character sketch.

There are two significant areas in which the character sketch falls short. The first is that it blindly applies the same edits to every motion. An actor or animator will normally vary movement adjustments based on the intended mood of the character or the particular nature of the action. Knowledge representation could be used to greatly expand the capabilities of the character sketch by allowing the sketch to vary the edits it performs based on both the action that it is editing and the current mood of the character.

The second shortcoming of the character sketch is that it does not change the structure of the motion; that is, it does not add additional poses or actions to the sequence, or remove them, based on the nature of the character. A nervous sketch was created that automatically adds weight shifts and knee bends to the character. This is interesting, but shows no awareness of the actual motions the character is performing. Recall the example of an old man sitting down. One would expect him to add the motion of reaching to feel the location of the chair before sitting. This change is not a mere warping of an existing motion. It is a more significant change to its structure. Our system is well positioned to support these kinds of edits, but knowledge representation or another technique would need to be employed to automatically determine which actions should be added or removed from a sequence.

Chapter 10

Conclusion

People call me the painter of dancers, but I really wish to capture movement itself.

Edgar Degas

10.1 Discussion of Results

This thesis presents a general approach, along with a prototype system, for generating expressive character motion. Procedurally defined properties provide direct handles on aesthetically important aspects of motion. High level interfaces such as the character sketch and composite properties support rapid exploration, while animator edits and low-level properties allow for detailed refinement. From a refined motion plan, both dynamic and kinematic outputs are supported. The system is validated in two ways: first, by grounding the ideas in the field that traditionally studies expressive movement, namely the performing arts; second, by demonstrating how these ideas can be implemented, how they alter animation sequences and how they can be combined for expressive effect.

A number of specific contributions have been made. The aesthetic importance of tension modeling has been introduced to the computer graphics community. Not only does tension modeling alter how a character responds to external forces, it also generates important secondary and end effects such as pendular motion at the end of arm swings, passive movement of loosely held limbs, and overshoot. In addition, specifying a tension change during a motion is an intuitive method for warping the motion envelope.

A new system for exploring character shape has been introduced that includes both an aesthetically meaningful low-level interface for shape modeling and a high-level interface based on shape sets that supports rapid exploration of the aesthetic space. Shape sets allow an animator to change a character's pose based on the intent of the character, for instance to reflect if a character is interested in an object or afraid of it. Shape parameters can be blended so that a character sketch can specify a default posture for a character which will serve as deformation for the posture associated with a particular action.

Various sample movement properties have been built. Along with tension and shape effects, these include properties that control extent, amplitude, succession and timing. A time planner has been designed and implemented that enforces a timeline semantics, supporting elastic timeline behaviour, synchronization and a range of scaling behaviour.

Overall, a new workflow has been presented that encourages animators to explore different movement possibilities and allows them to refine the animation sequence as needed. This is supported by a number of architectural decisions, such as the use of explicit representations that support both customization and interaction at different levels.

A formalization of the system has been developed that allows precise definitions to be given to the various components. It provides a method for a range of movement properties to be rigorously defined and then combined and mapped to a well defined base representation that is executable. This formalization allows questions to be answered about the system such as "What can and cannot be represented in it?". It represents a first step towards being able to more quantifiably compare different theories of motion.

Finally, an analysis of the arts literature has been provided that shows a method by which this rich but problematic information source can be used to improve computer animation tools.

10.2 Future Work

Given the breadth of the problem, it is a truism that much work remains to be done.

10.2.1 Expanding the System

The system presented here is a prototype. It is designed to be open and extensible. The main mechanism for extension is through the addition of new properties to the system. A core but nonexhaustive set of aesthetically meaningful properties has been implemented based on the arts literature. There are at least three reasons why an animator might wish to extend the property set. First, the animator might want to implement new ideas from the arts literature or experience. Second, she may wish a certain type of control that is well suited to her particular animation style. Third, high level properties may need to be more character specific. For instance, there are completely different ways a character can *recoil* from an object. One character may choose to turn away, and in the extreme, cover his head. Another character may stiffen his torso and lean slightly back while still maintaining eye contact with the object. Different *recoil* properties could be instantiated to cover these different ranges of movement. Designing custom properties like this is seen as part of the character development process.

Production work will require a larger range of motions. Standing motions have been used as the test-bed here, but general animation tasks will require locomotion, lying down, jumping, dancing etc. Many of these motions will require better dynamic control algorithms. Indeed, stronger balance control would be useful even in the limited domain of standing motions. Physics based control remains a difficult problem and limits the range of tasks that can be simulated using forward dynamics, so there remains much work to do here.

10.2.2 Resource Management

This system introduces a number of new animation resources. Properties, actions and character sketches are all reusable resources. As the system is used over time, libraries of these resources will be developed. Animators can draw on these repositories when creating new animation sequences. If the system sees widespread use, effective resource management will become an issue. Methods will need to be developed for cataloguing and searching these resources. Comparison metrics will be useful and for some resources such as character sketches, it may be helpful to develop editing operations such as merges and blends.

10.2.3 Interface Design

The focus of this work has been on conceptual design, where the driving issue has been what aspects of movement are important to represent in order to aid in the character animation task. Limited consideration has been given to interface design – how an animator could best interact with these movement concepts. Indeed, many interface possibilities exist.

The current interface is based largely on script files, augmented with a large control panel for the low-level interface to the body shape solver and an interactive timeline editor for specifying the script. While this interface is effective for experimenting with the system, it is not suitable to present to an animator.

A first step would be to implement a standard menu and widget interface as is used with Maya and other commercial animation packages. This would provide a comparable interaction style to most other animation packages, putting it in the comfort range of most animators.

It would be more interesting to experiment with different forms of interactive control that give animators real-time feedback on the effect of various adjustments. One way to do this would be to provide a series of predictions of the impact of different versions of an edit. Laszlo et al. [89] use predictions that look ahead in time to help control a physically simulated character. The idea is to present a series of previews around the current input location that indicate the state of the character if the input is moved to the location of the preview. The same idea could be applied to aesthetic properties by mapping the input parameters to a 2D space and displaying the effects of changing input on top of the space. Other input mappings could make use of the information in how an animator performs an input gesture to modify the aesthetic nature of the movement.

A very different form of interface would arise if our system was used as a low-level subsystem for a higher level AI character controller. Our system is appropriate for such applications as it provides an appropriate set of handles by which an AI system can vary a character's movements based on her current mood and personality profile. For instance, a high level system might decide a character should show disdain or possibly affection for another character. Our system can be used to define a mapping for how such high level concepts should make concrete adjustments to the low-level movement. Eventually, such a high level system might be able to accept verbal direction, as a director would give an actor, and this could be mapped to low-level movement generation.

10.2.4 User Validation

Given that the long term goal of this work is to make it easier for an artist to generate animations that satisfy his intensions, it will be ultimately important to have artists test the system. There are several barriers to testing, including the lack of a polished interface, the need to extend the system to a full range of motion, the need to control appropriately for the different backgrounds of test subjects and account for their learning curve with the system, the open ended artistic task the system is designed for and the difficulty in defining a success metric. Nonetheless, several testing strategies can be proposed that would be worth investigating:

- Use the system with a group of artists to produce an extended animation sequence or sequences. This is the ideal test as it would allow the system to be used throughout the different stages of the production process. New properties could be developed during character rigging and the animation process would include both exploration and refinement.

- Extend the system to provide more complete coverage of a narrower domain, such as gestures, and build a customized interface for this domain. By limiting the scope of the problem, a more complete solution could be generated and this could be tested in a more constrained manner.

- Examine a video sequence of an actor and reproduce it in the system. This would provide an indication of the range of motion that the system could accommodate.

10.2.5 Using Motion Capture

Motion capture can be combined with our system in two ways: by using motion capture as a source to develop new properties and actions, and by using our system to edit motion capture data.

Research is currently under way to explore how movement properties can be learned using a combination of motion capture data and machine learning. This appears to be a promising avenue by which to expand the property set. Features of a particular person's movement style can potentially be learned and represented in our system.

Developing a translator that can map motion capture data into our representation would allow our system to be used as an editing tool for this data. Motion data clips could also be used to define new actions. Games often rely on sequencing very short motion clips. Given the relatively simple action representation in our system, it should at minimum be possible to map such clips into our representation.

10.2.6 Other Extensions

Currently, specifying tension changes requires the animator to provide numeric gain values. While the calibration data helps give these numbers some meaning, specification of an arbitrary numerical value is unintuitive. It is desirable to have a system that automatically calculates numeric gain values for each joint in a particular skeleton and maps them to a standard range, say *very loose* to *very tense*. Damping could be automatically tuned as well. In general, a system that were to allow an animator to specify the desired behaviour and then automatically calculate the physical control parameters to achieve it would be of significant benefit.

The prototype is designed for offline authoring. In this setting, it is reasonable to generate a complete motion plan before executing it. It is straightforward to extend the system to interactive applications for games and virtual environments. The system simply needs to plan one or two actions ahead of the current time. In this way, the motion plan can be continuously extended on the fly, reacting to other events occurring in the environment.

Exploring inter-character systems is a worthwhile objective. Coordination would require synchronization and querying across the BReps of multiple characters.

10.2.7 Long Term Trajectory

Properties represent a step towards a *language* for movement, with the ultimate goal being the ability to provide concrete definitions for terms like "graceful". Such terms will in general not have a single definition as they encompass a range of movement nuances. Having concrete representations that cover a good portion of this range, within an extensible system, would be of immense value. Our system allows the discussion of movement properties to be made rigorous. Each movement property is given a concrete definition and the effect it has on movement can be demonstrated by generating animations and evaluated quantitatively and qualitatively. Our language and system can act as a bridge across artistic and technical communities to resolve ambiguities regarding the language of motion.

Bibliography

[1] David Alberts. *The Expressive Body: Physical Characterization for the Actor*. Heinemann, Portsmouth, N.H., 1997.

[2] Alias1Wavefront, Toronto. *Using Maya: Animation*, 1999.

[3] Gil L. Almeid, Di an Hong, Daniel Corcos, and Gerald L. Gottlieb. Organizing principles for voluntary movement: Extending single joint rules. *Journal of Neurophsiology*, 74:1374–1381, 1995.

[4] Kenji Amaya, Armin Bruderlin, and Tom Calvert. Emotion from motion. *Graphics Interface '96*, pages 222–229, May 1996.

[5] Frank C. Anderson and Marcus G. Pandy. A dynamic optimization solution for vertical jumping in three dimensions. *Computer Methods in Biomechanics and Biomedical Engineering*, 2:201–231, 1999.

[6] Gunnar B. J. Andersson and Jack M. Winters. Role of muscle in postural tasks: Spinal loading and postural stability. In Jack M. Winters and Savio L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*. Springer-Verlag, New York, 1990.

[7] Adolph Appia. *The Work of Living Art: A Theory of the Theatre*. University of Miami Press, Coral Gables, Florida, 1962 (original 1921), fifth edition 1982. Translated by H.D. Albright.

[8] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[9] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, July 2003.

[10] Yahya Aydin and Masayuki Nakajima. Balance control and mass centre adjustment of articulated figures in interactive environments. *The Visual Computer*, 15(3):113–123, 1999.

[11] Norm I. Badler, Diane Chi, and Sonu Chopra. Virtual human animation based on movement observation and cognitive behaviour models. In *Computer Animation 99*, 1999.

[12] Norman I. Badler. A computational alternative to effort notation. In J. Gray, editor, *Dance Technology I*, pages 23–44. AAHPERD Publications, Reston, VA, 1989.

[13] Norman I. Badler, Barry D. Reich, and Bonnie L. Webber. Towards personalities for animated agents with reactive and planning behaviors. In Robert Trappl and Paolo Petta, editors, *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture Notes in Artificial Intelligence 1195, Berlin, 1997. Springer-Verlag.

[14] Paolo Baerlocher and Ronan Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6):402–417, 2004.

[15] Eugenio Barba. Actor's training (1959-1962). In Jerzy Grotowski, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[16] Eugenio Barba. Dilated body. In Eugenio Barba and Nicola Savarese, editors, *A Dictionary of Theatre Anthropology: The Secret Art of The Performer*. Routledge, London, 1991.

[17] Eugenio Barba. Theatre anthropolgoy. In Eugenio Barba and Nicola Savarese, editors, *A Dictionary of Theatre Anthropology: The Secret Art of The Performer*. Routledge, London, 1991.

[18] Eugenio Barba. Theatre anthropology: First hypothesis. In Eugenio Barba and Nicola Savarese, editors, *A Dictionary of Theatre Anthropology: The Secret Art of The Performer*. Routledge, London, 1991.

[19] Eugenio Barba and Nicola Savarese. *A Dictionary of Theatre Anthropology: The Secret Art of The Performer*. Routledge, London, 1991.

[20] Joseph Bates, A. Bryan Loyall, and W. Scott Reilly. An architecture for action, emotion, and social behaviour. In *Artificial Social Systems: Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*. Springer-Verlag, Berlin, 1994.

[21] P. Becheiraz and D. Thalman. A model of nonverbal communication and interpersonal relationship between virtual actors. In *Computer Animation 96*, pages 58–67. IEEE Computer Society Press, June 1996.

[22] Bruce M. Blumberg and Tinsley A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. *Proceedings of SIGGRAPH 95*, pages 47–54, August 1995.

[23] Ronan Boulic, Ramon Mas-Sanso, and Daniel Thalmann. Complex character positioning based on a compatible flow model of multiple supports. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):245–261, jul - sep 1997.

[24] Matthew Brand and Aaron Hertzmann. Style machines. *Proceedings of SIGGRAPH 2000*, pages 183–192, July 2000.

[25] David Breen, 2003. Personal communication.

[26] Lynne Shapiro Brotman and Arun N. Netravali. Motion interpolation by optimal control. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):309–315, August 1988. Held in Atlanta, Georgia.

[27] Armin Bruderlin and Thomas W. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):233–242, July 1989. Held in Boston, Massachusetts.

[28] Armin Bruderlin and Tom Calvert. Knowledge-driven, interactive animation of human running. *Graphics Interface '96*, pages 213–221, May 1996. ISBN 0-9695338-5-3.

[29] Armin Bruderlin, Chor Guan Teo, and Tom Calvert. Procedural movement for articulated figure animation. *Computers & Graphics*, 18(4):453–462, 1994. ISSN 0097-8493.

[30] Armin Bruderlin and Lance Williams. Motion signal processing. *Proceedings of SIGGRAPH 95*, pages 97–104, August 1995.

[31] Tom Calvert and Sang Mah. Choreographers as animators: Systems to support composition of dance. In Nadia Magnenat Thalman and Daniel Thalman, editors, *Interactive Computer Animation*, pages 100–126. Prentice Hall, London, 1996.

[32] J. Cassell and H. Vilhjalmsson. Fully embodied conversational avatars: Making communicative behaviors autonomous. *Autonomous Agents and Multi-Agent Systems*, 2(1):45–64, 1999.

[33] Justine Cassell, Catherine Pelachaud, Norman Badler, Mark Steedman, Brett Achorn, Tripp Bechet, Brett Douville, Scott Prevost, and Matthew Stone. Animated conversation: Rule-based generation of facial expression gesture and spoken intonation for multiple converstaional agents. *Proceedings of SIGGRAPH 94*, pages 413–420, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[34] Marc Cavazza, Rae Earnshaw, Nadia Magnenat-Thalmann, and Daniel Thalmann. Motion control of virtual humans. *IEEE Computer Graphics and Applications*, pages 24–31, Sept./Oct. 1998.

[35] Diane M. Chi. *A Motion Control Scheme for Animating Expressive Arm Movements*. Ph.D. dissertation. department of computer and information science, University of Pennsylvania, 1999.

[36] Diane M. Chi, Monica Costa, Liwei Zhao, and Norman I. Badler. The emote model for effort and shape. *Proceedings of SIGGRAPH 2000*, pages 173–182, July 2000.

[37] Michael F. Cohen. Interactive spacetime control for animation. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):293–302, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

[38] O. Donchin and R. Shadmehr. Linking motor learning to function approximation: Learning in an unlearnable force field. *Advances in Neural Information Processing Systems*, 14:197–203, 2002.

[39] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Transactions on Graphics*, 22(3):409–416, July 2003.

[40] Jean Dorcy. *The Mime*. Robert Speller and Sons, Publishers, Inc., 1961. Translated by Robert Speeler, Jr. and Pierre de Fontnouvelle.

[41] M. Dornay, J. McIntyre F. Mussa-Ivaldi, and E. Bizzi. Stability constraints for the distributed control of motor behaviour. *Neural Networks*, pages 1045–1059, 1993.

[42] Sergei Eisenstein. On recoil movement. In Alma Law and Mel Gordon, editors, *Meyerhold, Eisenstein and Biomechanics: Actor Training in Revolutionary Russia.* McFarland and Company, Inc., Publishers, Jefferson, North Carolina, 1996.

[43] Sergei Eisenstein and Sergei Tretyakov. Expressive movement. In Alma Law and Mel Gordon, editors, *Meyerhold, Eisenstein and Biomechanics: Actor Training in Revolutionary Russia.* McFarland and Company, Inc., Publishers, Jefferson, North Carolina, 1996.

[44] Petros Faloutsos. *Composable Controllers For Physics-Based Character Animation.* Ph.D. dissertation. department of computer science, University of Toronto, 2002.

[45] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. *Proceedings of SIGGRAPH 2001*, pages 251–260, August 2001. ISBN 1-58113-292-1.

[46] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics*, 25(6):933–953, December 2001.

[47] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics*, 22(3):417–426, July 2003.

[48] A. G. Feldman. Functional tuning of the nervous system with control of movement or maintenance of a steady posture - ii. controllable parameters of the muscles. *Biophysics*, 11(3):565–578, 1966.

[49] R. C. FitzPatrick, J. L. Taylor, and D. I. McCloskey. Ankle stiffness of standing humans in response to imperceptible perturbation: Reflex and task-dependent components. *Journal of Physiology*, 454:533–547, 1992.

[50] Tamar Flash. The organization of human arm trajectory control. In Jack M. Winters and Savio L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*. Springer-Verlag, New York, 1990.

[51] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Prniciples and Practices*. Addison-Wesley Publishing Company, second edition, 1990.

[52] Peggy E. Gallaher. Individual differences in nonverbal behavior: Dimensions of style. *Journal of Personality and Social Psychology*, 63(1):133–145, 1992.

[53] Michael Girard and Anthony A. Maciejewski. Computational modeling for the computer animation of legged figures. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):263–270, July 1985.

[54] Michael Gleicher. Retargeting motion to new characters. *Proceedings of SIGGRAPH 98*, pages 33–42, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[55] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: Assembling run-time animation. *ACM Transactions on Graphics*, 22(3):702–702, July 2003.

[56] Athomas Goldberg. Improv: A system for real-time animation of behavior-based interactive synthetic actors. In Robert Trappl and Paolo Petta, editors, *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture Notes in Artificial Intelligence 1195, Berlin, 1997. Springer-Verlag.

[57] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, August 2004.

[58] Jerzy Grotowski. The actor's technique. In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[59] Jerzy Grotowski. Actor's training (1966). In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[60] Jerzy Grotowski. American encounter. In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[61] Jerzy Grotowski. Skara speech. In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[62] Jerzy Grotowski. Statement of principles. In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[63] Jerzy Grotowski. The theatre's new testament. In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[64] Jerzy Grotowski. Towards a poor theatre. In Eugenio Barba, editor, *Towards a Poor Theatre*. Routledge, A Theatre Arts Book, New York, 1968.

[65] Radek Grzeszczuk and Demetri Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 63–70, August 1995.

[66] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. *Proceedings*

of SIGGRAPH 98, pages 9–20, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[67] Jason Harrison. *Measuring and Comparing Human Walking Motions for Computer Animation.* Ph.D. dissertation. department of computer science, University of British Colombia, 2001.

[68] Barbara Hayes-Roth, Lee Brownston, and Erik Sincoff. Directed improvisation by computer characters. Technical Report KSL-95-04, Knowledge Systems Laboratory, Stanford University, 1995.

[69] Barbara Hayes-Roth, Robert van Gent, and Daniel Huber. Acting in character. In Robert Trappl and Paolo Petta, editors, *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture Notes in Artificial Intelligence 1195, Berlin, 1997. Springer-Verlag.

[70] Chris Hecker. Physical controllers: Re-inventing game animation. *Game Developer*, pages 33–40, April 1999.

[71] Jessica K. Hodgins and Nancy S. Pollard. Adapting simulated behaviors for new characters. *Proceedings of SIGGRAPH 97*, pages 153–162, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[72] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. *Proceedings of SIGGRAPH 95*, pages 71–78, August 1995.

[73] Neville Hogan. Mechanical impedance of single- and multi-articulator systems. In Jack M. Winters and Savio L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization.* Springer-Verlag, New York, 1990.

[74] Neville Hogan. An organizing principle for a class of voluntary movements. *The Journal of Neuroscience*, 4(11):2745–2754, November 1984.

[75] Neville Hogan and Jack M. Winters. Principles underlying movement organization: Upper limb. In Jack M. Winters and Savio L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*. Springer-Verlag, New York, 1990.

[76] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. *SD/FAST User's Manual*. Symbolic Dynamics Inc., 1994.

[77] Paul M. Isaacs and Michael F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):215–224, July 1987. Held in Anaheim, California.

[78] Keith Johnstone. *IMPRO: Improvisation and the Theatre*. Methuen Drama, 1981.

[79] Prem Kaira, Nadia Magnenat-Thalmann, Laurent Moccozet, Gael Sannier, Amaury Aubel, and Daniel Thalman. Real-time animation of realistic virtual humans. *IEEE Computer Graphics and Applications*, pages 42–56, September/October 1998.

[80] Sacha Kapijimpanga, 2005. Professional animator. Personal communication.

[81] Hyeongseok Ko and Norman I. Badler. Animating locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, pages 50–59, March 1996.

[82] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3):33–41, July 1984. Held in Minneapolis, Minnesota.

[83] E. Kokkevis, D. Metaxas, and N. I. Badler. User-controlled physics-based animation for articulated figures. In *Computer Animation '96*, pages 16–26, June 1996. ISBN 0-8186-7588-8.

[84] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[85] Rudolf Laban. *The Mastery of Movement.* Northcote House, London, fourth edition, 1988. Revised by Lisa Ullman.

[86] Alexis Lamouret and Marie-Paule Cani. Scripting interactive physically–based motions with relative paths and synchronization. In *Graphics Interface '95*, pages 18–25, May 1995.

[87] Alexis Lamouret and Michiel van de Panne. Motion synthesis by example. *Computer Animation and Simulation '96 – Proceedings of the 7th Eurographics Workshop on Simulation and Animation*, pages 199–212, August 1996.

[88] John Lasseter. Principles of traditional animation applied to 3d computer animation. *Proceedings of SIGGRAPH 87*, 21(4):35–44, July 1987.

[89] Joe Laszlo, Michael Neff, and Karan Singh. Predictive feedback for interactive control of physics-based characters. In *Eurographics*, 2005.

[90] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. *Proceedings of SIGGRAPH 2000*, pages 201–208, July 2000. ISBN 1-58113-208-5.

[91] Joseph F. Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH 96*, pages 155–162, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[92] Alma Law and Mel Gordon. *Meyerhold, Eisenstein and Biomechanics: Actor Training in Revolutionary Russia*. McFarland and Company, Inc., Publishers, Jefferson, North Carolina, 1996.

[93] Joan Lawson. *Mime: The Theory and Practice of Expressive Gesture With a Description of its Historical Development*. Sir Isaac Pitma and Sons Ltd., London, 1957. Drawings by Peter Revitt.

[94] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[95] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, pages 39–48, August 1999.

[96] Philip Lee, Susanna Wei, Jianmin Zhao, and Norman I. Badler. Strength guided motion. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):253–262, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[97] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, July 2002.

[98] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416, July 2002.

[99] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical spacetime control. *Proceedings of SIGGRAPH 94*, pages 35–42, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[100] A. Bryan Loyall, W. Scott Neal Reilly, Joseph Bates, and Peter Weyhrauch. System for authoring highly interactive, personality-rich interactive characters. In *Computer Animation 2004*, pages 59–68. Eurographics Association, 2004.

[101] J. M. Macpherson. How flexible are muscle synergies? In D.R.Himphrey and Freund H.-J, editors, *Motor Control: Concepts and Issues*. 1991.

[102] Maja J Mataric, Matthew Williamson, John Demiris, and Aswath Mohan. Behavior-based primitives for articulated control. In *Proceedings, From Animals to Animats 5, Fifth International Conference on Simulation of Adaptive Behavior*, pages 165–170. MIT Press, 1998.

[103] Maja J. Mataric, Victor B. Zordan, and Matthew M. Williamson. Making complex articulated agents dance: An analysis of control methods drawn from robotics, animation and biology. *Autonomous Agents and Multi-Agent Systems*, 2(1), july 1999.

[104] Michael Mateas. An oz-centric review of interactive drama and believable agents. Technical Report CMU-CS-97-156, School of Computer Science, Department of Computer Science, 1997.

[105] Tim McGovern. The use of live-action footage as a tool for the animator. In Bill Kroyer and Phillipe Bergeron, editors, *3D Character Animation By Computer*, volume 5 of *Siggraph 87 Course Notes*. ACM Siggraph, 1987.

[106] Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):29–38, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[107] A. Mehrabian. Significance of posture and position in the communication of attitude and status relationships. *Psychological Bulletin*, 71(5), 1969.

[108] Joann Montepare, Elisa Koff, Deborah Zeitchik, and Marilyn Albert. The use of body movements and gestures as cues to emotions in younger and older adults. *Journal of Nonverbal Behavior*, 23(2):133–152, 1999.

[109] Sonia Moore. *The Stanislavski System: The Professional Training of an Actor*. Penguin Books, 1984.

[110] Claudia L. Morawetz and Thomas W. Calvert. Goal-directed human animation of multiple movements. *Graphics Interface '90*, pages 60–67, 1990.

[111] Ferdinando A. Mussa-Ivaldi. Motor primitives, force-fields and the equilibrium point theory. In N. Gantchev and G. N. Gantchev, editors, *From Basic Motor Control to Functional Recovery*, pages 392–398. Academic Publishing House, Sofia, 1999.

[112] Michael Neff. http://www.dgp.toronto.edu/people/neff.

[113] Michael Neff and Eugene Fiume. Modeling tension and relaxation for computer animation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 81–88, July 2002.

[114] Michael Neff and Eugene Fiume. Aesthetic edits for character animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 239–244, July 2003.

[115] Michael Neff and Eugene Fiume. Artistically based computer generation of expressive motion. In *Proceedings of the AISB 2004 Symposium on Language, Speech and Gesture for Expressive Characters*, pages 29–39, 2004.

[116] Michael Neff and Eugene Fiume. Methods for exploring expressive stance. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49–58, August 2004.

[117] Michael Neff and Eugene Fiume. Aer: Aesthetic exploration and refinement for expressive character animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, July 2005. to appear.

[118] Michael Neff and Eugene Fiume. Methods for exploring expressive stance. *Graphical Models*, 2005. to appear.

[119] Victor Ng-Thow-Hing and Petros Faloutsos. Dynamic animation and control environment. http://www.cs.ucla.edu/magix/projects/dance/index.html.

[120] J. Thomas Ngo and Joe Marks. Spacetime constraints revisited. *Proceedings of SIGGRAPH 93*, pages 343–350, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

[121] Benno M. Nigg and Walter Herzog, editors. *Biomechanics of the Musculo-skeletal System.* John Wiley and Sons, Ltd., second edition, 1999.

[122] S. Oore, D. Terzopoulos, and G. Hinton. A desktop input device and interface for interactive 3d character animation. *Graphics Interface '02*, page ?, 2002.

[123] Masaki Oshita and Akifumi Makinouchi. A dynamic motion control technique for human-like articulated figures. *Computer Graphics Forum*, 20(3):192–202, 2001.

[124] Jeff Panko, 2005. Professional animator. Personal communication.

[125] Ken Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, March 1995.

[126] Ken Perlin and Athomas Goldberg. Improvisational animation. White Paper.

[127] Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. *Proceedings of SIGGRAPH 96*, pages 205–216, August 1996.

[128] Paolo Petta and Robert Trappl. Peronalities for synthetic actors: Current issues and some perspectives. In Robert Trappl and Paolo Petta, editors, *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture Notes in Artificial Intelligence 1195, Berlin, 1997. Springer-Verlag.

[129] Cary B. Phillips and Norman I. Badler. Interactive behaviors for bipedal articulated figures. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):359–362, July 1991.

[130] Robert Playter. Physics-based simulations of running using motion capture. In Jessica Hodgins and Zoran Popovic, editors, *Animating Humans by Combining Simulation and Motion Capture*, volume 33 of *Siggraph 2000 Course Notes*. ACM Siggraph, 2000.

[131] Zoran Popovic and Andrew Witkin. Physically based motion transformation. *Proceedings of SIGGRAPH 99*, pages 11–20, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[132] William H. Press, Saul A. Tukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[133] K. Pullen and C. Bregler. Animating by multi-level sampling. *IEEE Computer Animation Conference 2000*, 2000.

[134] Katherine Pullen and Christoph Bregler. From motion capture to motion texture. In *Conference Abstracts and Applications*. ACM Siggraph, 2000. Siggraph 2000 Sketch.

[135] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics*, 21(3):501–508, July 2002.

[136] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):349–358, July 1991. ISBN 0-201-56291-X. Held in Las Vegas, Nevada.

[137] William T. Reeves, Samuel J. Leffler, and Eben F. Ostby. The menv modelling and animation environment. *Journal of Visualization and Computer Animation*, 1(1):33–40, August 1990.

[138] C. W. Reynolds. Computer animation with scripts and actors. volume 16, pages 289–296, July 1982.

[139] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):25–34, July 1987. Held in Anaheim, California.

[140] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, September - October 1998.

[141] Charles F. Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. *Proceedings of SIGGRAPH 96*, pages 147–154, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[142] Charles F. Rose III, Peter-Pike J. Sloan, and Michael F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3):239–250, 2001.

[143] David A. Rosenbaum. *Human Motor Control*. Academic Press, Inc., 1991.

[144] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3):514–521, August 2004.

[145] Richard A. Schmidt and Timothy D. Lee. *Motor Control and Learning: A Behavioral Emphasis*. Human Kinetics, Champaign, IL, third edition, 1999.

[146] Ted Shawn. *Every Little Movement: A Book about Francois Delsarte*. Dance Horizons, Inc., New York, second revised edition, 1963.

[147] Karl Sims. Evolving virtual creatures. *Proceedings of SIGGRAPH 94*, pages 15–22, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[148] Constantin Stanislavski. *An Actor Prepares*. Theatre Arts, Inc., 1936. Translated by Elizabeth Reynolds Hapgood.

[149] Constantin Stanislavski. *Building a Character*. Theatre Arts Books, 1949. Translated by Elizabeth Reynolds Hapgood.

[150] Constantin Stanislavski. *Creating a Role*. Theatre Arts Books, 1961. Translated by Elizabeth Reynolds Hapgood.

[151] Scott N. Steketee and Norman I. Badler. Parametric keyframe interpolation incorporating kinetic adjustment and phasing control. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, volume 19, pages 255–262, July 1985.

[152] Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. Speaking with hands: creating animated conversational characters from recordings of human performance. *ACM Transactions on Graphics*, 23(3):506–513, August 2004.

[153] Seyoon Tak, Oh young Song, and Hyeong-Seok Ko. Motion balance filtering. *Computer Graphics Forum*, 19(3):437–446, August 2000.

[154] Jennifer Tarver and Kate Bligh. The physical art of the performer, November 1999. Workshop: A 30 hr. intensive introduction to Laban and Grotowski held at the Nightwood Theatre studio, Toronto.

[155] George Taylor. Francois delsarte: A codification of nineteenth-century acting. *Theatre Research International*, 24(1):71–81, 1999.

[156] Demetri Terzopoulos, Xiaoyuan Tu, and Radek Grzeszczuk. Artificial fishes: Autonomous locomotion, perception, behaviour, and learning in a simulated physical world. *Artifical Life*, 1(4):327–351, dec 1994.

[157] Daniel Thalman and Hansrudi Noser. Towards autonomous, perceptive, and intelligent virtual actors. In *Artificial Intelligence Today*, number 1600, pages 457–472. Springer-Verlag, 1997.

[158] Frank Thomas. Can classic disney animation be duplicated on the computer? In Bill Kroyer and Phillipe Bergeron, editors, *3D Character Animation By Computer*, volume 5 of *Siggraph 87 Course Notes*. ACM Siggraph, 1987. Originally appeared in Computer Pictures, July/Aug 1984.

[159] Frank Thomas. The future of character animation by computer. In Bill Kroyer and Phillipe Bergeron, editors, *3D Character Animation By Computer*, volume 5 of *Siggraph 87 Course Notes*. ACM Siggraph, 1987.

[160] Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Abbeville Press, New York, 1981.

[161] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics*, 23(3):424–431, August 2004.

[162] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. *Proceedings of SIGGRAPH 95*, pages 91–96, August 1995.

[163] Michael van de Panne, Ryan Kim, and Eugene Fiume. Virtual wind-up toys for animation. *Graphics Interface '94*, pages 208–215, May 1994. Held in Banff, Alberta, Canada.

[164] Michiel van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, pages 40–49, March 1996.

[165] Michiel van de Panne and Eugene Fiume. Sensor-actuator networks. *Proceedings of SIGGRAPH 93*, pages 335–342, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

[166] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):225–234, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[167] Michiel van de Panne and Alexis Lamouret. Guided optimization for balanced locomotion. *Computer Animation and Simulation '95*, pages 165–177, September 1995. ISBN 3-211-82738-2.

[168] M. Alex O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *Proceedings of the International Conference on Pattern Recognition (ICPR 02)*, volume 3, pages 456–460, August 2002.

[169] Harald G. Wallbott. Bodily expression of emotion. *European Journal of Social Psychology*, 28:879–896, 1998.

[170] Doug Wiley and James Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphic and Applications*, 17(6):39–45, Nov-Dec 1997.

[171] Andrew Witkin and Michael Kass. Spacetime constraints. *Proceedings of SIGGRAPH 88*, 22(4):159–168, August 1988.

[172] Andrew Witkin and Zoran Popovic. Motion warping. *Proceedings of SIGGRAPH 95*, pages 105–108, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[173] Wayne L. Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans.* Ph.D. dissertation., Georgia Institute of Technology, 1998.

[174] Katsu Yamane and Yoshihiko Nakamura. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):352–360, jul-sep 2003.

[175] Douglas E. Young and Richard A. Schmidt. Motor programs as units of movement control. *Making them move: mechanics, control, and animation of articulated figures*, pages 129–155, 1991.

[176] George I. Zahalak. Modeling muscle mechanics (and energetics). In Jack M. Winters and Savio L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*. Springer-Verlag, New York, 1990.

[177] Felix E. Zajac. Muscle coordination of movement: A perspective. *Journal of Biomechanics*, 26:109–124, 1993. Supll. 1.

[178] David Zeltzer. Task-level graphical simulation: abstraction, representation, and control. *Making them move: mechanics, control, and animation of articulated figures*, pages 3–33, 1991.

[179] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

[180] Liwei Zhao and Norman I. Badler. Gesticulation behaviours for virtual humans. In *Pacific Graphics 1998*, pages 161–168, 1998.

[181] Victor B. Zordan and Jessica K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. *Computer Animation and Simulation '99*, September 1999. ISBN 3-211-83392-7. Held in Milano, Italy.