# INTERACTIVE CONTROL OF DEFORMABLE-OBJECT ANIMATIONS WITH INTUITIVE MOTION PATTERN ADHERENCE

by

SHANE MICHAEL TRANSUE

B.S., Metropolitan State University of Denver, 2011

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Computer Science

2014

This thesis for the Master of Science degree by

Shane Michael Transue

has been approved for the

Computer Science Program

by

Min-Hyung Choi, Chair

Gita Alaghband

Ellen Gethner

May 2, 2014

Transue, Shane Michael (M.S., Computer Science)

Interactive Control of Deformable-Object Animations with Intuitive Motion Pattern Adherence

Thesis directed by Associate Professor Min-Hyung Choi.

## ABSTRACT

Recent advances in the control of physically simulated objects have provided precise methodologies for deriving a desired physical state of a deformable object; however most of these approaches have severely limited the interactive component from real-time animation modification. To increase the level of interactivity in the generation of physically-based animations, we present an adaptive and intuitive methodology for controlling the localized deformation of physically simulated objects using an intuitive motion-based control interface. We achieve this control through the dynamic recording of physically simulated deformable objects and the development of high-level motion controls that provide effective manipulation techniques for altering the animation of deformable objects. To maximize the interactive component presented in this approach we consolidate existing feedback mechanisms in deformable-body control techniques to provide an intuitive simulation editing environment. We introduce the notion of control metaphors as the abstract formulations of primitive motions enacted by deformable-bodies when external forces modify the physical state of the object. As an application of this proposed control methodology we develop a practical solution for interjecting local deformations into dynamically recorded deformable objects. The effectiveness of this approach is demonstrated through interactively generated compound movements that introduce complex local deformations of the objects in existing physically-based animations. Additionally we validate the resulting movements imposed by the control metaphors by using directed behavior demonstrations through physical animations.

The form and content of this abstract are approved. I recommend its publication.

Approved: Min-Hyung Choi

## DEDICATION

I dedicate my work to my continuously supportive parents that have shaped my academic career and accepted my extensive research efforts, the additional motivation and support that I receive from my siblings, and eternal support provided by my beloved Malia has encouraged me to aim higher than I would have ever thought possible.

## TABLE OF CONTENTS

CHAPTER

I. INTRODUCTION ..... 1

Physical Simulation in Animation ..... 2

Towards Dynamic Simulation Control ..... 3

II. RELATED WORK ..... 5

Physically Simulated Object Motion Control ..... 6

External Force Motion Control ..... 7

Dynamic Keyframes ..... 9

Rest Shape Adaptations and Elastic Models ..... 16

Primary Incorporated Contributions ..... 20

III. DYNAMIC SIMULATION OF DEFORMABLE MODELS ..... 24

Deformable-body Simulation ..... 25

Mass-Spring Systems ..... 28

Mass-Spring Visual Representations ..... 34

Collision Detection ..... 36

Collision Response ..... 38

Mass-Spring Simulation Stability ..... 41

IV. HIGH-LEVEL DEFORMABLE-BODY MOTION CONTROL ..... 43

Control Metaphors ..... 44

Local Coordinates ..... 47

Robust Deformable Body Local Transformation Estimation ..... 49

Orthogonal Regression and Static Analysis ..... 50

Dynamic Local Coordinate Updates ..... 56

Control Coordinates ..... 59

Compound Motions with Control Metaphors ..... 61

V. PRIMITIVE CONTROL METAPHOR IMPLEMENATIONS ..... 62

Stretch Control Metaphor Implementation ..... 63

Bend Control Metaphor Implementation ..... 65

Twist Control Metaphor Implementation ..... 69

VI. DYNAMIC RECORDING OF PHYSICALLY-BASED ANIMATIONS ..... 72

Real-Time Simulation Recording ..... 73

Recording Invalidation ..... 76

VII. CURVE-BASED FORCE MAGNITUDE AND DURATION ..... 78

VIII. PHYSICAL SIMULATION DYNAMIC PREVIEW GENERATION ..... 84

IX. INTUITIVE CONTROLS FOR EDITING ANIMATED SIMULATIONS ..... 88

X. CONTROL METAPHOR IMPOSED MOTION VALIDATION ..... 93

Objective Evaluation of the Twist Control Metaphor ..... 94

Objective Evaluation of the Bend Control Metaphor ..... 96

Objective Evaluation of the Stretch Control Metaphor ..... 98

Evaluation of Localized Deformation Control ..... 101

XI. IMPLEMENTATION ..... 105

Architectural Overview ..... 105

Sx Library Modules ..... 108

Archetypes ..... 111

Viewport Controllers ..... 112

Collision Models ..... 113

Performance ..... 114

XII. RESULTS ..... 116

Compound Control Metaphors ..... 118

Compound Deformation Edits ..... 121

XIII. EVALUATION AND DISCUSSION ..... 124

Control Metaphor Limitations ..... 125

Control Metaphors and Dynamic Keyframes ..... 127

XIV. CONCLUSION ..... 130

XV. FUTURE WORK ..... 132

REFERENCES ..... 133

## LIST OF FIGURES

FIGURE

2.1. Local Deformations Derived from External Forces .....9

2.2. Conveying Motion with Keyframes and Interpolation .....11

2.3. Inverse Dynamics for Rigid Body Simulation .....13

2.4. Inverse Simulation for Deformable-body Simulation .....14

2.5. Non-trivial Deformable-body Keyframe State .....15

2.6. Rest-pose Adaptations and Elastic Potential .....18

2.7. Cage-based Deformable Object Control .....19

3.1. Deformable Mass-spring Cloth Representation ..... 32

3.2. Cross-sectional View of a Mass-spring Torus .....34

3.3. Deformable-body Self-collision .....37

4.1. Planar Orthogonal Regression..... 51

4.2. Furthest-point Projection Heuristic for Local Coordinate Generation ..... 52

4.3. Rotationally Instable Local Coordinate Derivation ..... 53

4.4. Point-Cloud Subset Selection for Rotationally Stable Coordinates ..... 54

4.5. Local Coordinate Rotation Axis..... 55

4.6. Accurate Alignment of the Generated Local Coordinate Systems .....57

4.7. A Rotationally Stable Local Coordinate System .....58

4.8. Control Coordinate System ..... 60

5.1. Stretch Control Metaphor Node Selection .....63

5.2. Stretch Control Metaphor Unique Force Pattern .....64

5.3. Cage-based Control Widget for the Stretch Control Metaphor .....65

5.4. Bend Control Metaphor Node Selection .....66

5.5. Bend Control Metaphor Unique Force Pattern .....67

5.6. Bend Metaphor Control Widget and Effected Node Sets .....68

5.7. Unique Force Pattern for the Twist Control Metaphor .....69

5.8. Twist Control Widget with Modified Orientation..... 70

6.1. Dynamic Recording Algorithm for Physical Simulations .....75

7.1. Conceptual Force-Curve Editor ..... 79

7.2. Beizer Force-Curve Editor Implementation ..... 82

8.1. Dynamic Simulation Preview Generated for 200 time-steps with $m = 20$ ..... 85

8.2. Dynamic Preview of a Highly-Deformable Object ..... 86

9.1. Real-time Deformable Animation Studio ..... 88

9.2. List of Active Control Metaphors Acting upon the Selected Mesh ..... 89

9.3. Flexible Simulation Navigation ..... 90

9.4. Interactive Animation Time-line ..... 91

9.5 Animation Media Controls..... 92

10.1. Twist Control Node Sets ..... 94

10.2. Twist Control Metaphor Deformation Validation ..... 95

10.3. Demonstration of Validated Twist Deformation..... 96

10.4. Bend Control Metaphor Validation – Pre-Deformation ..... 96

10.5. Bend Control Metaphor Validation – Post-Deformation ..... 97

10.6. Demonstration of the Validated Bend Control Metaphor ..... 98

10.7. Stretch Control Metaphor Validation ..... 99

10.8. Demonstration of the Validated Stretch Control Metaphor ..... 100

10.9. Demonstration of the Stretch Metaphor utilized for Compression ..... 100

10.10. Localized Stretch Deformation ..... 102

10.11. Localized Bend Deformation ..... 102

10.12. Localized Twist Deformation ..... 103

11.1. Basic Simulation Architecture Overview ..... 106

11.2. Complete Sx Library Architecture ..... 107

11.3. Sx Library Extensions ..... 107

11.4. Graphics Archetype Formal Definition ..... 112

11.5. Potential Archetype Definitions ..... 112

11.6. Viewport Controller Hierarchy ..... 113

12.1. Localized Twist Control Metaphor Deformation ..... 116

12.2. Compound Control Metaphor Application ..... 120

12.3. Inverse Force Curves ..... 121

12.4. Compound Deformation Edit Configuration ..... 123

12.5. Compound Deformation Behavior: Ear Flap ..... 123

13.1. Improper Bend Control Metaphor Application ..... 126

## LIST OF EQUATIONS

EQUATION

3.1. Total Deformable-body Mass ..... 28

3.2. Deformable-body Center of Mass ..... 29

3.3. Hooke's Law for Linear Springs (restoring) ..... 29

3.4. Simulated Object Force, Mass, and Velocity Relation ..... 29

3.5. Initial Value Problem Formulation for Velocity and Position ..... 30

3.6. Implicit Euler (general form) ..... 30

3.7. Net Force Acting Upon a Node within a Mass-spring System ..... 31

3.8. Mass-spring System Spring Count Bounds ..... 32

4.1. Net Force Imposed by Multiple Control Metaphors ..... 61

# CHAPTER I

# INTRODUCTION

The simulation of physically-based systems in computer graphics is a well researched topic that has rapidly developed over the course of several years. The most notable progress in this field relates to the increase in the visual fidelity and physical accuracy of modern physics simulations. Research in this field extends the usability of physical simulations to countless domains including architecture, biology, civil engineering, and computer animation. In computer animation the general purpose of a physical simulation is to aid in the visual quality of some desired motion, to increase the accuracy of complex collisions and to greatly reduce the amount of work required to form a believable animation. Specifically we look at how physical simulation can be used to develop and refine computer generated animations. The field of computer animation develops rapidly and new authoring techniques are constantly introduced with the hope of improving the visual quality of the animation, increasing the physical accuracy, and making the overall process efficient and intuitive.

Physical simulation is a very powerful tool in the field of computer animation. The number of animated shorts, films, and various other media that are computer generated has drastically increased since the introduction of new tools that make computer animation tangible and intuitive to a larger number of artists. The development of systems that simplify the interface for producing a computer animation has allowed artists to utilize computer animation to effectively express their artistic intent. For computer graphics as a whole, this has revolutionized the process of producing animation and has lead to a significant amount of research in the field of physical simulation.

**Physical Simulation in Animation**

One of the primary reasons that physical simulation has become so popular in the field of computer animation is because of the accuracy of the resulting movements and the drastic reduction in the effort required achieving those results. Fundamentally an animation consists of a series of frames that are rapidly displayed to depict the illusion of motion. This basic idea dictates that every frame must be properly authored to depict the state of an object or figure at some instant in time. In traditional animation this is an extremely time-consuming process. With the introduction of computer animation, automated processes have been introduced to eliminate this vast amount of work. While the ability to utilize concepts such as keyframes, artists can now focus on the important aspects of the animation and allow for the direct computation of the intermediate frames.

The introduction of physically-based simulations takes this basic concept and extends it further to provide a robust, efficient technique for generating physically plausible motions. Even with computer-aided animation and techniques like keyframe interpolation, there is a significant amount of responsibility left to the artist. This is due to the fact that while keyframes reduce the amount of input required from the artist, they do not specify any notion of correct movement or physically plausible interactions of the animated objects or figures. While this gives artistic freedom to the animator, due to the flexibility of the technique, most animations require the accurately physical interaction and motion of the animated objects. A direct solution to this problem is the introduction of a physical simulation. The physical simulation removes most of the burden acquired by the artist when they wish to animate an object with physical properties. This alone has drastically altered how most modern animations are created.

**Towards Dynamic Simulation Control**

Considering the current state of physically-based simulations and their use in modern computer animation, we focus our attention to the development and control of the systems that artists can use to efficiently generate realistic animations. The process of allowing an animator to author an animation with the assistance of a physical simulation, the fundamental requirement that must be addressed is artistic control of the simulation. When authoring an animation an artist must be able to effectively control the physical simulation to achieve the intended motion of the animated objects. However, specifically noting that the physical simulation should be utilized to generate most of the intermediate motions between important events in the animation, we look at the possible ways an artist may want to modify the result of a physical simulation. These define the methods of control that should be exposed to the artist. Ideally these are high-level controls that contribute to generalized behaviors within the physical simulation. The artist should not be required to move every point on the surface of a simulated object to achieve some intended result; rather, the artist should be able to specify a high-level command that corresponds to the general desired behavior (motions such as bending, stretching, twisting, etc). In this research we propose an animation modification technique that provides these high-level metaphors to an animator and interprets the desired behavior of the object and transforms the motion of the simulated object to match the artists visional of this intended behavior.

The proposed technique builds on the existing research in computer animation and physical simulation to provide artists with high-level control metaphors that allow them to quickly and effectively author complex physical behaviors in animations. In this work we allow the artist to create a physically-based animation based on recorded

deformable objects and then provide an extensive set of modification metaphors that can be used to alter the resulting recorded animation. Utilizing these tools the artist can then modify the global trajectory and local deformations of all deformable-bodies within the animation. This is achieved through the implementation of the high-level metaphors that utilize the introduction of artificial external forces that act upon the simulated objects to modify their behavior.

We take an extensive look at the related work in this field to tailor this solution to the needs and requirements of the artist composing the animation. We use the some of the high-level concepts introduced by these prior research efforts to provide a flexible and robust design framework to artists. We additionally analyze the importance of feedback to the artist and include several effective means of conveying this feedback as the animation is composed and modified. We develop this framework into a functional animation studio that allows artists of varying experience to modify deformable-bodies in physically simulated animations.

# CHAPTER II

# RELATED WORK

Physically-based dynamic system simulation is a well researched field within computer graphics. As a primary sub-field of computer graphics, the development of physically-based simulations of both rigid- and deformable-body systems has been extensively studied. Within this extensive amount of research, the visual fidelity [10] and physical accuracy [11] of simulated physical objects has improved over the course of several years. Most existing methods provide stable, physically plausible results for rigid-body simulations in controlled environments [3]. These simulations, based on numerical integration, approximate geometric representations, and collision resolution have provided solution implementations that deliver robust, physically plausible rigid-body motion [14].

Similar to the research into rigid-body simulation, numerous prior research efforts have attempted to address the stable and accurate simulation of deformable-objects [8, 9]. However due to the intrinsic complexity of deformable-object behavior, several additional considerations must be addressed by the simulation. Unlike rigid-bodies, deformable-bodies in a simulation are prone to an extensive number of invalid states. This is simply due to the fact that the total number possible interactions between two deforming objects are much greater than in the rigid-body case. This problem is further compounded by the introduction of deformable-body self-collisions. Various techniques have been developed [7] to address some of these complex self-interactions; however research into effective solutions to these problems is ongoing [14]. Additionally, deformable-object behavior is harder to control both from a simulation standpoint and

from the perspective of a user interacting with a simulation. In the rigid-body case, the object is defined by one location and the possible inter-object interactions are simple, therefore the numerical stability of the system is easily maintained. For an interactive rigid-body simulation the user can simply modify the position and initial velocity of the object and gain a reasonable result, yet this only provides a limited level of control.

However, these problems are further complicated when considering deformable-body simulation. In this case, several issues must be addressed: what is the provided physical representation of the deformable-body, how can the numerical stability of the simulation be maintained, and how can a user control or manipulate a deformable-object in an effective way. While the first two issues have been reasonably addressed, and previously proposed techniques provide reasonable behaviors in deformable-body simulations, the third (deformable-object motion control) remains one of the most difficult aspects of deformable-body simulations to effectively address.

**Physically Simulated Object Motion Control**

One of the primary research fields within physically-based simulation is dynamic object control. Research within this field aims at defining various methods that allow for the manipulation of object motion to reach some desired state. Intuitively, the lowest form of control over the resulting state of a physical simulation is based on modifying the initial physical parameters of the simulated objects. Generally this is accomplished by simply modifying the initial position and velocity of the simulated object.

These properties will determine the global trajectories of the objects and their interactions with other objects as the simulation is progressed at some interval. While this level of control allows for the behavior of the objects within the simulation to be

modified, this approach does not provide any information about the future state of the objects. When changing the initial parameters of the simulated objects, the global trajectories can be estimated, however the state of the object at some arbitrary point in the simulation is difficult to predict. Adjusting the parameters of the object and the simulation to achieve some desired output may even become tedious to an animator or in the worst case; the desired state may not be obtainable. Ideally for some form of control it is desirable to provide a constraint on the behavior that is desired at some point within the simulation that simply allows the animator to specify their intent. To achieve this, several approaches have been developed that utilize the idea of animation keyframes to control the future state of a simulated object. These approaches provide insight into how the objects in a simulation can be controlled by modifying their future state and mathematically deriving the motions used to obtain it.

**External Force Motion Control**

An intuitive approach to modifying the behavior of a deformable body is to simply alter the objects global trajectory or create a local deformation in the surface of the object at some predefined point in the simulation. This can be thought of as a push or pull operation that acts on the deformable-object. The simple idea behind this approach is that as the simulation progresses, the behavior will initially match the expected outcome defined by the initial conditions of the simulation, until a specific point in time is reached. At this point in time the introduction of an external force (a force that is not responsible for maintaining the linkage between the objects nodes) will alter the physical state of the deformable-object. This provides a means to controlling the behavior of the objects motion during the course of the simulation after the specified time.

There are however several things that must be considered in order to perform this application of an external force to the deformable-object. First it is assumed that the physical simulation can properly handle external forces that are applied at some initial time $t_i$ for a duration $d$ ending at a final time defined by $t_f$ where $t_f = t_i + d$. This is due to the fact that in the physical system, the force must act over the duration of time according to the discrete steps of the simulation. Therefore in general terms $d$ would be any positive real value. However in the simulation, the process of discretization will be applied to the time-step ($dt$) which will result in $d$ representing a natural number corresponding to the number of time-steps the force is applied. Second, to create the external force that will be interjected into the system, the direction and magnitude of the force must be specified. The image in Figure 2.1 illustrates this process as a function of time and the corresponding relation to the force generated in the simulation. The directional force $F$ illustrated in the image depicts the artists intended modification of the global trajectory of the object.

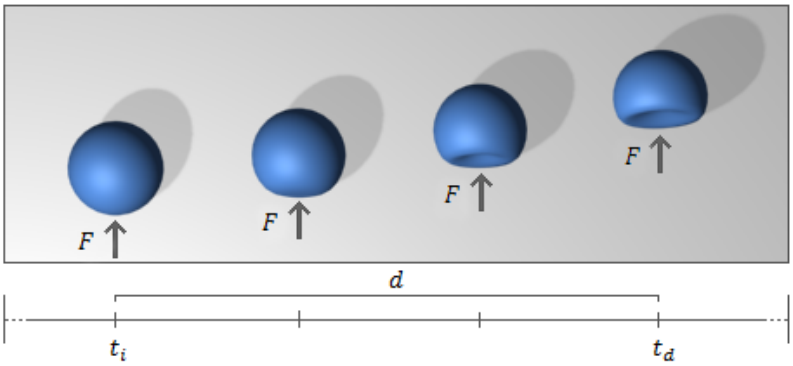



**Figure 2.1. Local Deformations Derived from External Forces**

Application of the constant external force $F$ over the duration $d$ resulting in a large localized deformation.

The final consideration is enforced by the highly-interactive nature of this approach. The instant in time the force is applied, the direction and magnitude of the

force, and the total duration must all be effectively translated from the artist's original intent. This mapping is not direct and presents one of the largest hurdles to over-come if this technique is utilized to modify a simulations behavior. Previous attempts at this approach demonstrate concrete examples and show extensive promise [25]. However due to the required highly-interactive interface required to properly translate the artist's intentions into external forces; extensive research must be performed in usability engineering. This type of development requires not only an effective means of translating the artist's modifications into a meaningful edit of the external forces but it must also actively convey accurate feedback to the artist. This mapping between the intended action and the resulting external force has to be properly addressed for this method to be effective.

While the research in usability engineering may be an additional requirement, the potential of this solution is derived from the high level of interactivity and the potential for immediate feedback. This can provide an iterative feedback loop that allows the artist to refine the outcome of the simulation to suit their original intent. This approach forms an active simulation editing technique, that is, direct changes to the simulation result in the desired motion. The next two sections cover methods that provide more of a passive realization of the artist's intent, that is, they rely on an initial specification from the artist and then internally calculate the required force modifications.

**Dynamic Keyframes**

In traditional animation, important motion states are captured in keyframes. A keyframe simply stores the information about the object it describes at some point in the

animation timeline. An animator must provide a sequential set of keyframes in order to define the information required to reproduce a desired motion over time in an animation.

This process has been developed to alleviate the process of manually defining each frame of an animation. Considering the creation of each frame of the set $F$ that compose an animation, the task is inherently time-consuming. As the number of moving objects and scene complexity increase, smooth motion becomes increasingly difficult to achieve. This is the problem that keyframes aim to address. Since the important motions are described through a smaller set of keyframes $K$ where $|K| < |F_{total}|$, an automated process can fill-n the remaining required motion states. This is simply a process of interpolating the data between each keyframe pair based on the two selected keyframes. The set of keyframes can be defined by $k_0, k_1, \dots, k_n$ where $n$ is the total number of keyframes considered in the animation and the corresponding values $t_0, t_1, \dots, t_n$ indicate the position in time of the keyframe. Each frame is then defined between each pair of keyframes such that $k_p < f_i < k_s$ where $k_p$ represents the primary keyframe, $k_s$ is the secondary keyframe and $f_i$ represents the intemediate frames that will be generated in the interpolation process. Once all $i$ frames have been generated, the process will continue considering the next keyframe pair. This process continues until the full animation has been achieved.

This greatly reduces the amount of effort required to author the appearance of smooth motion during an animation, refining the scope of the task to defining the set of keyframes. The image in Figure 2.2 illustrates how keyframes in animation are typically defined and also illustrates the intermediate steps that will be provided between set

keyframes. This provides an informative appearance of motion that is noted as an effective means of conveying the future state of an object or figure.

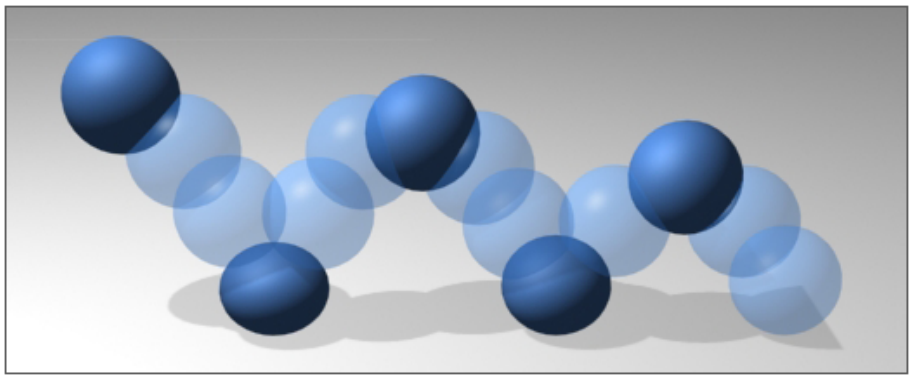


**Figure 2.2. Conveying Motion with Keyframes and Interpolation**
Conveyed motion based on the intermediate steps between keyframes. The keyframes are illustrated in dark blue and the interpolated frames are transparent.

The resulting appearance of motion is derived from the intermediate states defined by the interpolation process. Typically the resulting generated frames provide a smooth motion or transition between keyframes using a positional interpolation scheme. Considering a basic linear interpolation, the desired appearance of linear motion is achieved when the individual frames are illustrated in rapid succession. The interpolation process of the data used to compose the required frames is domain dependent and based on the desired motion behavior of the animation. Utilizing these basic concepts from animation, this process can easily be applied to the domain of physically simulated objects to achieve controlled physically plausible motion.

The application of keyframe animation as a basis which provides a basic control system for physical simulation has been extensively researched [15]. Keyframes and the process of interpolation that can be utilized within a physical simulation have been showed to effectively allow for the control of dynamically simulated objects. The fundamental concept behind these approaches is that the previously introduced concept of

keyframes remains valid; however the implementation of the abstract interpolation process is modified to suit the requirements of the domain. This introduces the concept of dynamic keyframes. In the domain of physical simulation these keyframes are considered dynamic due to the fact that they not only describe the position of the object but they almost must describe the entire physical state of the object. These additional physical properties must be correctly derived for the intermediate frames in order to generate the desired physically plausible motion. In the simple case, a rigid-body object (where all deformations are neglected), the state and movements of the object are defined by the object's position $p$, velocity $v$, and it's rotation along with its angular velocity $\omega$. The interpolation process can then be defined based on these physical properties.

Given some primary $k_p$ and secondary state $k_s$ of a simulated object, the objective of the interpolation process is to determine the dynamic state of the object for each frame $f_i$ between $k_s$ and $k_p$. Progressing from the primary state to the secondary state is achieved with forward simulation, that is, the force, velocity, and physical properties are utilized to generate the next position of the object. However, since the physical control system requires that the secondary state is provided, the inverse of this approach is required. This technique is defined as inverse simulation [4]. Inverse simulation revolves around the notion of defining a control parameter that will effectively reach a predefined output state of the system. In this case we consider the output state of the system to represent the secondary state of the simulated object defined by $k_s$. The objective then coincides with inverse dynamics: determine the required input control parameters (eg. the force, velocity, etc.) that will effectively reach the desired secondary state provided by the keyframe. The proposed method of achieving this desired state is

effectively achieved in rigid-body simulation [17]. Iteratively this process can be used to determine the intermediate states between each keyframe pair. A brief example is illustrated in Figure 2.3. This approach provides an animator with a much more precise control of the simulated objects behavior.

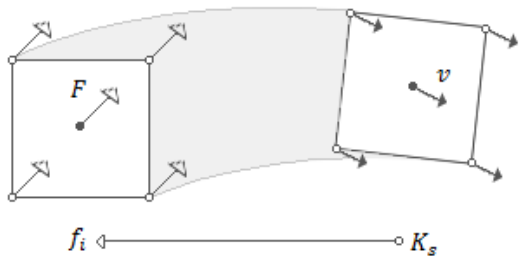



**Figure 2.3. Inverse Dynamics for Rigid Body Simulation**
Inverse dynamics can be utilized to derive previous force states. The intermediate frame $f_i$, the position and the force $F$ are calculated from the state of the secondary keyframe $K_s$ and its definition of the physical state of the rigid-body.

Building on the introduction of the interpolation process of a rigid-body between animation keyframes, the principle can again be applied to a more sophisticated domain: deformable-object simulation. The application of the interpolation process to deformable-object simulation remains consistent with the rigid-body formulation; however the complexity of deriving the intermediate steps drastically increases. With the rigid-body case, the interpolation process revolves around the calculation of the control parameters that will achieve the desired state. For the deformable-body interpolation process, the overall concept remains the same however the number of control parameters that must be determined drastically increases. This is due to the fact that for the rigid-body case, the set of control parameters can be uniformly applied to all nodes that define the geometry.

When considering a deformable-body and the motion of its surface (and internal structure depending on the physical representation), this is not the case. To fully describe the behavior of the deformable-body, the control parameters must be determined for each

discrete location defined in the representative geometry of the object. This task is inherently much more complex due to this requirement. Therefore based on a nodal-based geometric definition of the deformable-object, this process must be performed on each of the $n$ connected nodes. Due to the deformable characteristic of the physical representation, each node will have several degrees of freedom, therefore solving for the required force, velocity, and position of all nodes at some frame prior to the desired secondary keyframe is not a trivial task. Figure 2.4 illustrates an overview of the interpolation process based on inverse dynamics for a simple deformable cloth model with nine surface nodes that approximate the desired deformable object and its behavior.

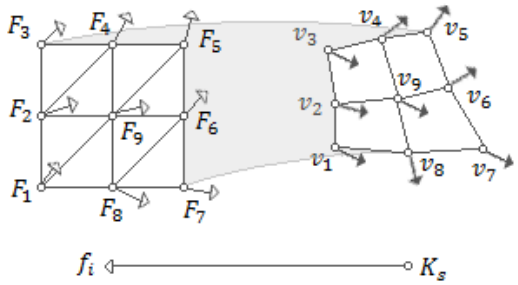



**Figure 2.4. Inverse Simulation for Deformable-body Simulation**

Calculating an intermediate step using inverse dynamics in deformable-body simulations must calculate the positions, velocities, and forces of all nodes to determine the frame $f_i$, given the secondary keyframe $K_s$.

Current research techniques [1] propose efficient applications of this concept. Control via general spacetime constraints such as keyframes, velocities, and forces provide the artist with the ability to modify the behavior of the deformable-object. Additional techniques have been developed [13] that build on these developments and improve the quality of the simulation and the animation it produces. This includes the preservation of detail from the provided animation, smoothness of the edits produced by the control provided to the user, and the incorporation of secondary motion introduced by

the edits. Additional approaches have improved the performance of this technique by reducing the degrees of freedom of the simulated nodes [16].

These techniques in dynamic keyframing provide physically plausible motion for the provided simulation states. However due to the complexity of the optimization required for determining the previous control states, real-time (60[Hz]) interaction is unobtainable when all nodal degrees of the deformable are considered. These techniques drastically reduce the control parameters of the optimization by reducing the degrees of freedom of all nodes. Typically the reduced space is acquired using a dynamic modeling technique such as modal analysis, then a mapping is made between this reduced space and the original simulation. This is typically acceptable due to the fact that the visual fidelity of the animation is not reduced by this marginal sacrifice in accuracy.

The fundamental problem with deriving control of a deformable-simulation from dynamic keyframes is framed by a simple question: how are the dynamic keyframes that will define the movement of the deformable-body over time initially derived? Consider the example keyframe shown in Figure 2.5. This complex state of the deformable object would be incredibly difficult and time consuming if not impossible to accurately produce.

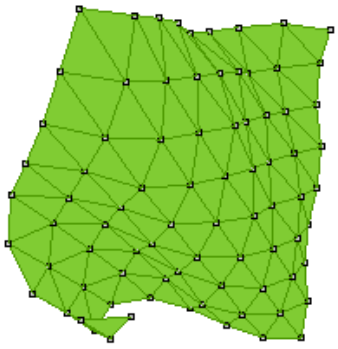


**Figure 2.5. Non-trivial Deformable-body Keyframe State**
The complex state of a keyframe that would be would be ideal for an inverse dynamics solution, yet the derivation of this initial motion is not a trivial task.

This technique does not directly address this problem. When considering the keyframes required by a rigid-body simulation, the requirements are intuitive. Since a rigid-body object cannot deform the animator only has to provide a position, velocity, rotation, and angular velocity. This will describe the complete state of the object for the required secondary keyframe. Yet in the case of a deformable-body this information must be applied to every discrete node that composes the object. To further compound this problem, the influence of all connected nodes must also be accounted for.

From an artist's point of view, manually defining all of these parameters is almost impossible and furthermore, this is the reasoning behind the inclusion of a physics engine in simulated animation. If the animator could simply define all of these parameters for each state then the physical simulation is no longer required. Therefore when considering control for a dynamic simulation, the provided technique should be able to properly address this initial requirement.

**Rest Shape Adaptations and Elastic Models**

Methods that utilize external forces to modify the behavior of the deformable objects rely on the interactions between the introduced forces and the internal forces maintained by the physical system representation to achieve the desired motion. Similarly dynamic keyframes provide the basis from which the required external forces are calculated via inverse dynamics. However, providing artificial external forces is not the only means to modifying the behavior of a deformable-object during the course of a simulation.

Alternative methods to modifying the behavior of deformable-bodies that deviate from utilizing external forces have been proposed in an effort to provide more natural

behaviors [2]. This research proposes that the addition of external forces is a detriment to the physical quality of the resulting animation. This is based on the fact that the externally applied forces do not sum up to zero, therefore they introduce unwanted rotation. Additionally the body can accumulate unwanted momentum through the application of the external forces, leading to inaccurate physical behavior. This research proposes that a more intuitive approach based on internal elastic forces should be used. This is based on the idea that real creatures utilize internal energy to move throughout their environment (through muscle movement). In an attempt to match the intuitive principle behind this fact, internal forces are considered instead of an introduction of artificial external forces. Based on the shared principle the hope is that the resulting motion of derived from this approach will provide a more natural resulting motion throughout the animation. While the results of this approach are impressive, they are generally limited to this natural motion and somewhat limit the possibilities for an artist.

Essentially the approach defines two rest positions of the deformable-object which will lead to a difference in elastic potential energy. To determine the contribution of internal forces, the approach calculates this elastic potential energy between two rest shapes of the deformable-object by optimizing the provided equations of motion. The utilized equations are similar to those considered in all approaches but have been optimized for performance. This is due to the introduction of the optimization problem that is utilized to calculate the required internal forces need to attain the target rest shape.

The image in Figure 2.6 shows two rest poses of a deformable object that will be used to calculate the elastic potential between the two rest poses A and B. The elastic potential energy will be calculated based on these two poses and converted to internal

forces that will define the movement of the objects nodes as it transitions from the initial state to the desired state.

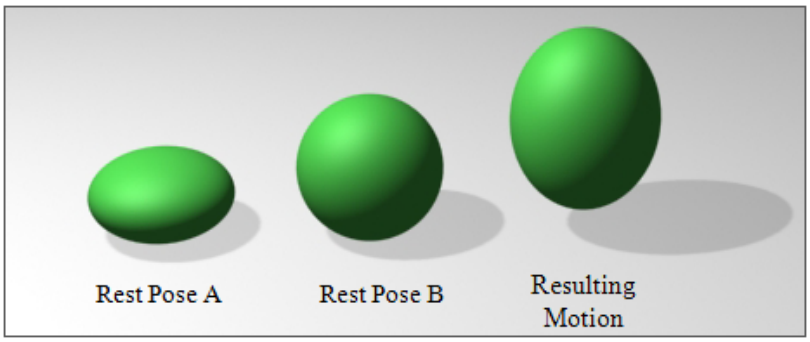



**Figure 2.6. Rest-pose Adaptations and Elastic Potential**
Elastic potential energy derived from the Rest Pose A and Rest Pose B. The elastic potential energy is then converted to a net internal force. The resulting motion generated by this force is illustrated by the jumping ball.

Rest pose adaptations are also introduced provide the animator with various methods to effectively specify the rest poses of the deformable object. The initial case where the target motion is specified by two rest poses illustrates an example-based rest post adaptation. The initial rest pose is the current state of the deformable-object and the target pose is the desired state of the same object. The target provides an example of the desired state to reach based on the potential difference calculated using the original rest position. The objective function will then consider the example during the derivation of the elastic potential energy which will then be converted into internal forces. Again this can be thought of as an implementation of the abstract interpolation process between two states of the deformable-body by using the potential elastic energy as the means to reach the target rest position of the object.

Another powerful rest pose adaptation is the cage-based adaptation approach [18]. This provides a lower-resolution wireframe model around the deformable-body that allows an artist to have high-level control over the deformable-body. Cage-based

adaptations are particularly expressive and allow the artist to develop customized bounding structures for their deformable models. The cage-based model allows the user to develop the customized bounding structure that surrounds their geometry. The artist can then link the cage to the deformable-body (which generally contain too many nodes to individually modify) and modify the general state of the deformable body. This approach is particularly effective since it allows a higher level of control over the nodes in the deformable model. The image in Figure 2.7 illustrates the simplification of modifying the pose of a deformable-body by utilizing the cage that bounds the geometry of the object.

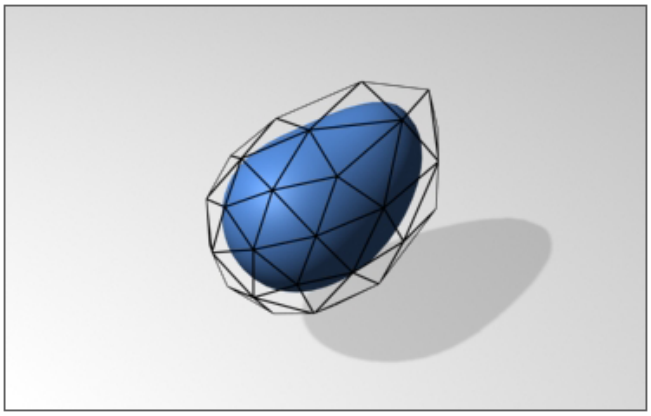


**Figure 2.7. Cage-based Deformable Object Control**
The cage based modification of deformable-bodies provides a high-level descriptive tool that allows the artist direct control over the higher resolution deformable model.

The concept introduced by this approach is similar to modifying the behavior with dynamic keyframes; however it relies on modifications to the internal forces calculated by the elastic potential. Utilizing the elastic potential to calculate the internal forces also ensures that momentum is properly preserved. Using these internal forces to drive the modification of the behavior between rest states may initially seem like the same concept introduced by dynamic keyframes, however this is not the case. This approach leads to

distinct behavioral results that differ from the dynamic keyframe technique. The resulting behavior of the deformable-object based on the forces modified by the elastic potential will be bounded to the derived potential energy. This presents a visually distinct result in the behavior of the modified objects. Specifically the objects appear autonomous due to the fact that simple motions like jumping require elastic potential energy, which for most objects and natural creatures includes a form of compression or crouching. This leads to a very natural form of simulated motion. Yet while the intended goal of providing natural looking motions was presented, this type of motion may not provide the artist with the tools to attain their intended motion. The results of this research provide an optimal solution to the described behavior, but the approach is also limited by these constraints.

**Primary Incorporated Contributions**

Prior research efforts related to deformable object control and behavior modification have provided several techniques for modifying the global behavior of a deformable-body. Based on these results, animators and artists no longer have to rely on arbitrary adjustments to the simulation parameters to achieve a desired state of the simulated deformable objects. These methods provide a drastic improvement in the interface between the artist and the control mechanisms required for modifying deformable-body motion.

The approaches identified in this chapter effectively allow an artist to achieve the motions described in the in the published results however the requirements for using these approaches are fairly stiff. While they do allow for interchangeable underlying dynamic engine implementations and adjustable simulation parameters the lack flexibility in the interface provided to the artist. Additionally these approaches do not provide an

effective feedback mechanism as part of the toolset provided to the artist. The ability to refine and make additional adjustments to the resulting simulation state depends on this feedback and from an artistic standpoint, the flexibility to view the resultant motion from a defined edit is critical to the overall editing process.

The progress made by these research efforts do however provide several key features upon which an effective deformable-body motion control can be derived. Since each of the methods previously explored are relatively similar but address slightly different desired behaviors, the selection of the approach that provides the most overall flexibility should be considered. This selection is made based on the ability for the technique to deliver feedback effectively to the artist. The most flexible approach will allow for the most flexible toolset to be presented to the artist. If most of the control is removed from the abilities of the artist then they will lose the ability to effectively express their artistic intent. While the resulting animation should be accurate, most animations only require a physically plausible result. Therefore choosing an approach that is less accurate but more flexible will result in a solution that will allow for the largest amount of artistic expression. This approach slightly diverges from the previous research in that it shifts the responsibility from the optimization process into the domain of usability engineering. However the ability to perform this shift means that the proposed approach will not intrinsically be limited to the formulation of the objective function in the optimization, rather an extensible and flexible tool system shifts the possible resulting motion determination to the artistic creativity of the artist.

The proposed deformable-body animation modification approach utilizes several of the techniques previously reviewed to provide an artist with an extremely flexible

interface and set of behavior modification tools. While the proposed technique is based on the introduction of external forces, the key concepts contributing to the success of the other approaches are adapted to this solution. The technical implementation of these key concepts have been modified however they still properly address the problem they were designed to address. As an example we consider two of these transformed behavior control mechanisms.

The first control mechanism that provides an extensive amount of feedback to the artist is the introduction of a visual representation that depicts the state of the deformable object at some desired point in the simulation. This is similar to both dynamic keyframes and rest pose optimization techniques. However in this approach we consider how these concepts can be applied to the more flexible external force motion control technique. However in contrast to the dynamic keyframe requirement on the artist to define the desired state of the deformable object, this concept is instead translated into a feature provided by editing environment. This is due to the fact that the derivation of this desired pose is extremely difficult, if not nearly impossible to manually create. Therefore we utilize the result of the dynamic simulation to display the possible future states of the deformable object. The possible future states of the object will depend on the artistically applied edits; however the artist is not directly responsible for deriving the desired state. Alternatively this allows the artist to generate a potential state based on their guided edit and select the most appropriate state presented by the simulation preview. This effectively eliminates the unaddressed requirement of the dynamic keyframe approach previously discussed. The artist can define an edit that will generate a solution that is

close to the intended motion. The generated preview will allow the artist the view the future state of the object and keep the introduced edit if it produces the desired state.

Another important control mechanism that will be translated into the proposed solution is the notion of a cage-based bounding volume used to modify the behavior of the deformable object. Instead of providing a higher level of control of the position of the deformable object, the translation of the cage-based bounding volume will be utilized to illustrate the area of influence defined by a pending edit. As the artist introduces an edit they will be presented with a visual metaphor that will depict the outcome of the desired edit. Since the artist may wish to only apply this change to a localized area of the deformable body, the cage-based bounding volume is utilized to effectively convey to the artist the selected area to identify what nodes of the deformable object will be affected by the edit. This information is critical to properly editing the object in the desired way. The form and type of the cage (as well as its visual appearance) depends on the selected metaphor that will modify the motion of the deformable object. An open extension to this design allows for additional motion modification metaphors to be developed and added to the flexible toolset that this approach provides.

# CHAPTER III

## DYNAMIC SIMULATION OF DEFORMABLE MODELS

Fundamentally there are two general types of physical simulations. The first presents the accurate simulation of rigid-body objects. This type of simulation is derived from the assumption that all objects with physical properties that can dynamically interact with an environment, completely neglect all deformations. Essentially this locks the objects graphical representation to the objects center of mass (or some arbitrary point) and simulates the object as a point-mass with a rotation and angular velocity. Effective solutions for rigid-body motion control (for trajectories) have been derived [6], and implemented in both open source and commercial products that offer practical solutions for animators. In contrast, we focus the development of this approach around the control and modification of object behaviors in the second type of physical simulation: deformable-body simulations.

In the process of developing a dynamic simulation that defines the physical representation of a deformable-body, there are several considerations that must be accounted for during the process of simulating highly interactive objects. Specifically, each deformable object must maintain an extensive amount of information about the current state of its mass distribution, its connectivity, and the elastic properties striving to maintain equilibrium. The most common form of representing models with these properties is through the discretization of the mass distribution into individual point-masses that are commonly referred to as nodes. In this chapter we provide a brief overview of discrete nodal-based physical representations utilized to demonstrate the

proposed method of deformable object control and identify the imperative components of deformable-body simulation that must be included for this type of physical simulation.

To simulate a deformable-body accurately, each node in the physical representation of the object may move independently. This concept is introduced as the degrees of freedom of the node. The degrees of freedom of each node in a deformable-body are generally unbound. For instance, each point-mass $p \in R^3$ has three unique degrees of freedom: $x$, $y$, and $z$. Therefore the process of simulating and controlling these objects is inherently much more complex than that of rigid-body control. In order to build the foundation of this approach we look at the most appropriate existing deformable-body simulation techniques and utilize them to generate the unedited motions of the simulated deformable objects. This approach will then define how these techniques can be utilized to define how local deformations can be applied to the simulated deformable objects. Furthermore we alter the process of deformable-body simulation to provide a basis upon which an artist can interject high-level controls that will be interpreted and translated to the physical representation of the simulated objects. This will allow for the modification of object behavior to match the intent of the artist. The development of this framework and process plays a pivotal role in the ability to accurately control objects in deformable-body simulations.

**Deformable-body Simulation**

The underlying concept in deformable-body simulation is that the simulated objects physical representation and mass distribution can be altered during the course of the simulation. Intuitively this dictates that the relationships between all parts of an objects physical definition can move in relation to one another. Deformable simulations

aim to provide the behavior of elastic materials that stretch, bend, and twist based on the forces exerted within the object (internal forces) and from their environment (external forces). Two effective and common physical representations for deformable-bodies have been proposed and continuously developed: mass-spring systems (MSS) and finite element-based (FEM) models. These representations produce model behaviors similar to various kinds of real world elastic objects depending on parameters established for a model instance. These represent the most prominent representations for deformable objects in physical simulation. From the extensive development and stabilization of these models over the course of several years, the development of a framework that facilitates a method of interchanging these physical representations is required for effectively simulating and animating deformable objects. Based on this required interchangeability, a stable framework has been developed to allow the proposed animation generation and editing technique to be used with existing deformable-body simulation models. Therefore the approach presented for deformable object control can be effectively used with existing physics libraries. Since this abstraction has been made, we can consider the internal dynamics of the physical representation of the deformable object as a black-box, that is, the proposed approach is independent of the internal dynamics of any simulated object. The only requirement that this approach of deformable-body control imposes on the physical representation and internal dynamics is that it must be able to accept direct external forces. Direct external forces affect a specific part of the deformable-bodies' surface and contribute to the modification of both the internal state of the object and the surfaces position in space over time. This requirement is general enough to be applied to all physics libraries that either mass-spring systems or FEM-based models.

For the enforced deformable-body simulation interface, the standard procedural overview of physical simulation is maintained. This includes the process of updating the positions of the deformable-body based on a provided time-step, efficient collision detection, and a collision resolution process. Since the deformable-body representation is considered a black-box, we ensure that the updated positions, velocities, and forces of the nodal components are properly passed to the abstract collision detection and response handlers through consistent plain data types. This outlines the standard update process present in most physical simulations around which the proposed method of deformable object control is built. This standard procedure is flexible enough to incorporate any existing physical simulation methodology.

Since the physical representation is independent of this approach, we utilize a standard mass-spring system to model the deformable objects within the implemented physical simulation. Mass-spring systems provide a stable and robust deformable physical representation that can effectively emulate most elastic materials with physically plausible results. Since we include the ability to feature several independent objects in the animations generated with this approach, we consider each object as an individual mass-spring system with a single continuous surface. The flexibility of this design contributes to the effectiveness of the editing approaches we provide and simplifies the internal implementation of the editing process. To obtain the proposed method of deformable-body control, we also analyze the basic properties of mass-spring systems to define how generalized object deformations can be created through high-level control patterns or metaphors.

**Mass-Spring Systems**

The proposed formulation of a mass-spring system that defines the structure and behavior of a deformable-body introduces the notion of unique point-masses that are connected by springs. The precise definition of how the individual point-masses are connected with springs depends on three commonly defined connectivity patterns; however the basic principles behind this representation are uniformly defined for all mass-spring systems. In this section, we first provide the basis upon which nodal mass-spring systems are defined and then provide the derivation of the dynamics utilized to simulate this form of deformable object. We then provide an overview of the three commonly used connectivity patterns used to define mass-spring objects.

To define the general structure of a mass-spring system we let $N$ represent the set of point-masses (nodes) in the system where $\forall n \in N$, $n$ is a point in three dimensional space ($n \in R^3$). Each node and its' associated mass represents a fractional portion of the deformable-body where the total mass of the object is defined in Equation 3.1.

$$\sum_{i=1}^{|N|} mass(n_i)$$

**Equation 3.1. Total Deformable-body Mass**

In the proposed approach (and typically with most common definitions of mass-spring systems), we can consider the special case where the mass of each node has a unit value of 1.0. From this we can simplify the total mass of the deformable object as $m = |N|$, where $m$ represents the total mass. Given that during a simulation the distribution of this mass will vary over time, we also define the center of mass $CM$ (updated at each simulation time-step) using this simplification in Equation 3.2.

$$CM = \sum_{i=1}^{|N|} n_i$$

**Equation 3.2**. **Deformable-body Center of Mass**

The next required step in defining the general structure of a mass-spring system is the set of springs that are used to provide the connectivity of the nodes within the system. At this point we will assume that these springs are simply modeled after Hooke's law (Equation 3) where the displacement from the rest state is related through a spring constant $k$.

$$F = -kX$$

**Equation 3.3. Hooke's Law for Linear Springs (restoring)**

With this general outline of a mass-spring system, a brief derivation of the systems dynamics is presented to provide the basis upon which external forces are introduced to provide localize deformation control. Fundamentally the dynamic motion of each point-mass is derived from basic Newtonian laws, simplified through discretization, and then formulated as an initial state problem. Generalizing with vector notation provides a complete illustration of this derivation in three dimensions.

From the general vector form $\vec{F}(t) = m\vec{a}(t)$ we simply substitute equivalent statement $\vec{F}(t) = m\, d\vec{V}(t)/dt$ and rearrange to derive the differential equation that relates the force applied to a point-mass and its velocity over time. This final equation is presented in Equation 3.4.

$$\frac{\vec{F}(t)}{m} = \frac{d\vec{V}(t)}{dt}$$

**Equation 3.4. Simulated Object Force, Mass, and Velocity Relation**

This differential equation provides the basis of all physical simulations.

Given this differential equation we formulate an initial value problem of which the solution will be approximated using standard numerical techniques. The presented initial value problems that will be approximated for a particles velocity and position over time are presented in equation 3.5.

$$\frac{d\vec{V}(t)}{dt} = \frac{\vec{F}(t)}{m}, \;\; \vec{V}(t_0) = v_0 \;\; : \;\; \frac{dX(t)}{dt} = \vec{V}(t), \;\; X(t_0) = x_0$$

**Equation 3.5. Initial Value Problem Formulation for Velocity and Position**

Initial value problems given the initial velocity (left) and initial position (right) of a point-mass particle where $v_0$ is the initial velocity and $x_0$ is the initial particle position.

In the approximation of the solutions to these differential equations, considerations must be made with respect to the target domain of this approach. Specifically, the generation of animations based on recording a simulation at an artist defined frame rate may introduce numerical instability in this numerical approximation. Therefore we consider the possible artistic requirement for large time-steps in the simulation. To support this we forego the more efficient explicit numerical integration techniques and utilize the implicit (or backward) Euler method. The general form of this equation is presented in Equation 3.6.

$$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1})$$

**Equation 3.6. Implicit Euler (general form).**

For the application to the animation domain, we allow the artist to define what frame-rate they would like to record the simulation. The inverse of a provided frame-rate is a representative value of the time-step $h$. Due to the requirements of deriving the solution provided by the implicit Euler approximation, a conjugate gradient solver [32] is utilized to solve the sparse system of linear equations derived from the definition of the

mass-spring system. This implementation of this method is provided through the VegaFEM library.

With this derivation of particle dynamics we consider the final formulation of the mass-spring system. Since the velocity and position of the particles can be determined in relation to the set of forces acting on an individual particle we partition the forces to two sets: internal and external. The internal forces are those acting upon the nodes of the system by only the springs of the system. All other forces (such as gravity and friction) that may be applied to any of the nodes within the system will be considered external forces. Therefore the net force acting upon a particle in the mass-spring system is defined in Equation 3.7. In this proposed method we utilize this differentiation in forces to introduce external forces that will modify the motion and deformations of physical simulated objects.

$$\mathcal{F} = F_{int} + F_{ext}$$

**Equation 3.7. Net Force Acting Upon a Node within a Mass-spring System**

This overview provides the generalized form of mass-spring dynamics that are utilized by this approach to physically simulate deformable objects. Given that through this derivation we have defined the set of nodes and the set of springs of the system and their physical interactions, we must finally consider the connectivity within the system. Here we make a parallel with the set of $N$ nodes and the set of springs $S$ with a graph. The set of nodes represents the vertices of the graph and the set of springs represents the edges of the graph. Based on this analysis we can provide a theoretical bound on the connectivity of all mass-spring systems. While the presented approach to deformable object control can be utilized to modify the behavior of any mass-spring mesh on $n$

nodes, we only provide a demonstration of the approach on manifold meshes. The lower and upper bounds for the total number of springs (edges) within any mass-spring system is provided in Equation 3.8.

$$(n-1) \leq |S| \leq \frac{n(n-1)}{2}$$

**Equation 3.8. Mass-spring System Spring Count Bounds**

Bound on the number of springs within any mass-spring system. The lower bound can be characterized as a string of nodes and the upper bound is the complete graph $K_n$.

Based on the extensive number of potential spring configurations on $n$ nodes, and the required use of manifold meshes we can now characterize the three common forms of connectivity patterns that are used to define mass-spring meshes. Namely, these are cloth, shell, and solid. Since this approach utilizes the introduction of external forces to modify the motion of the mass-spring system, all common mass-spring representations are supported. Each of these generalized types of mass-spring meshes are characterized by a unique connectivity pattern. The first and most common form is the cloth connectivity pattern. This can be represented by a graph containing $(n\ x\ n)$ nodes. The connectivity of the nodes in a mass-spring system that approximates the behavior of cloth is conceptually illustrated in Figure 3.1

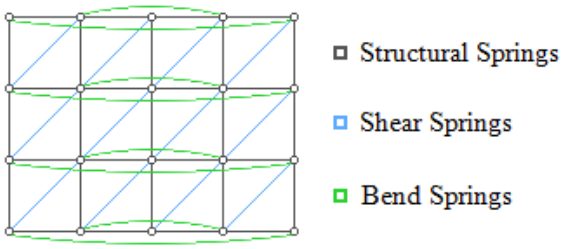



**Figure 3.1. Deformable Mass-spring Cloth Representation**

Connectivity of the nodes for a typical mass-spring mesh that emulates the behavior of cloth. The three types of springs generally only differ by their spring constant $k$.

This cloth representation is formed by three categories of springs that allow the nodes of the system to behave similar to the behavior defined by most cloth materials. Structural springs define the general layout of the cloth model (shown in Figure 3.1 as squares), shear springs allow for opposite ends of the cloth to be translated without rotation, and the bend springs provide folding and curl behaviors. Typically the spring constant $k$ of the structural and shear springs is much high than that used for the bend springs and generally results in a realistic simulation of cloth. Similarly, the other forms of connectivity provide unique spring constants that determine the overall behavior of the simulated object. Cloth mass-spring meshes are utilized extensively in animation and the presented approach for performing localized deformations can easily be expressed through the modification of cloth-based systems. Examples of these localized deformations are illustrated in the resulting application of the deformation controls presented for this method.

The second type of connectivity that defines a category of mass-spring systems is modeled after shell structures. General examples of the connectivity of a shell structure are provided by either a torus or sphere. The nodes of the mass-spring system in these examples represent the surface of the object and define an enclosed volume. While not as common as cloth models, shell-based mass-spring meshes can provide an additional class of unique deformable behaviors. Typically the deformations imposed on an objects surface from simple operations such as twisting and stretching are exaggerated due to the lack internal springs that could provide additional stability. The cross-section of the torus provided in Figure 3.2 illustrates the hollow volume enclosed by the surface of the objects shell.

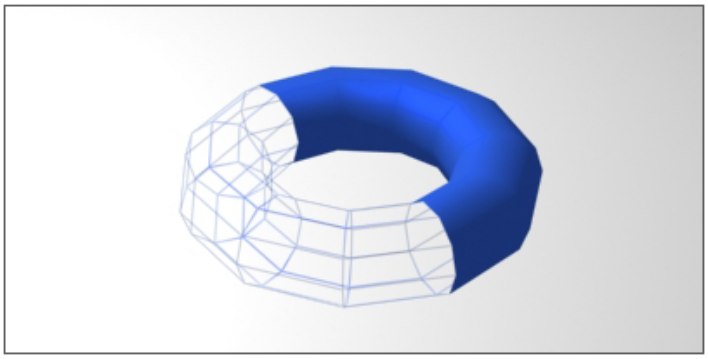


**Figure 3.2. Cross-sectional View of a Mass-spring Torus**
The shell represents the surface of the mesh and the visible edges represent the spring connectivity of the mass-spring system.

The third type (and second most common) form of mass-spring system connectivity pattern is defined as a solid tetrahedral structure. This form of connectivity attempts to provide a representation of a solid deformable-body. This is because the internal structure of a tetrahedral-based system is populated with internal nodes that approximate the internal distribution of mass. Typically a uniform distribution of mass is ideal but difficult to efficiently generate. The unique characteristic of the motions generated by tetrahedral-based systems is that they are more rigid due to the additional connections throughout the internal structure of the object. This pattern however leads to objects that are more stable for generalized simulations and are therefore commonly used for most physical simulations that utilize mass-spring systems.

### Mass-Spring Visual Representations

To this point mass-spring systems have been defined as a collection of point-masses connected by springs in one of three common ways. To be used in an animation however additional visual information (extended past the rendering of the nodes and the springs that connect them) must be provided to the artist and in the final animation. The ideal representation of the object would be an accurate depiction of the objects' surface;

therefore we introduce the notion of a visual representation that is defined by both the nodes and edges of the system to produce this surface. To visualize this representation, we consider that the shell of the object will represent its visible surface. This composes the ‘mesh’ portion of a mass-spring mesh and is inherently simple to generate for cloth and shell-based mass-spring systems; however tetrahedral-based systems are fundamentally different.

For the visualization of a tetrahedral-based mass-spring system we primarily consider the surface of the object as its visual representation while ignoring the internal nodes and springs. Typically the internal nodes and springs of these systems are not rendered and do not contain any face information. This provides the notion of a key concept that is developed as a result of this work: the distinct separation of the visual and physical representations of all simulated objects. Some objects may utilize representations in which the physical definition is the same as the visual definition, yet as stated for the tetrahedral-based systems that approximate a solid mass, the visual representation may not use the internal structural information. The flexibility of this design allows for the physical representation and visual representation of any simulated object to vary independently. This feature is built directly into the foundation of the implementation of the constructed simulation architecture and is covered in Chapter 11.

With the formal dynamics of a mass-spring mesh defined with respect to a set of constraint forces (springs), we can look at how the object will interact with both the static environment and other simulated objects. The next two sections provide a brief introduction to collision detection and resolution in physical simulations. The presented concept of collision handlers is then developed in Chapter 11.

**Collision Detection**

In any physical simulation the process of detecting collision is critical to the accuracy of the physical interactions between simulated objects and their environment. Again, initially considering the simple case of rigid-body collision detection, several research efforts have provided efficient and accurate collision detection algorithms [33, 34]. However, the challenge presented for collision detection involving deformable-bodies is greatly increased by the degrees of freedom of each node within the system. This problem is further exacerbated by the introduction of possible self collisions, that is, an objects surface may penetrate itself. A self-collision occurs when two parts of the single deformable surface penetrate each other and it not a present concern for rigid-body simulation. An example of an undeformed and self-penetrating deformable body is shown in Figure 3.3.

Self-collision not only requires that the collision detection scheme actively check all other objects for possible penetrations at each time-step, but it must also provide adequate support for self-collision detection. An additional consideration that must be carefully analyzed for the application of collision detection is the decision to utilize faster discrete collision detection (DCD) algorithms, or more accurate and less efficient continuous collision detection (CCD) algorithms. Due to the complex behaviors of deformable objects that will be simulated to generate animations, we utilize continuous collision detection to ensure that tunneling effect [35] will be removed and all self-collisions will be detected.

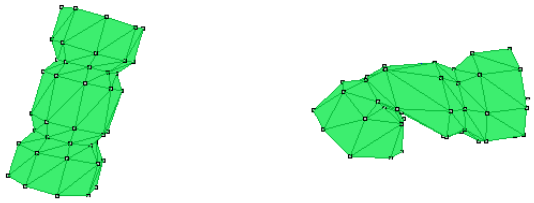


**Figure 3.3. Deformable-body Self-collision**

During extreme deformations, self-penetration can occur. When an object is in an undeformed state (left), initial interactions with will be with other objects. After this initial collision, the deformable body may collide with itself (self-collision, right image).

Recent research in deformable-body collision detection that includes self-collision checking (CCD) have been developed and now provide efficient solutions to handle the complex collisions [14]. In this approach we formulate the implementation of the physical simulation to match the model required by these techniques and employ a global collision detection model that considers the surfaces of all deformable objects. When any two surfaces (or two distinct parts of a single surface) penetrate each other, the collision will be marked by the collision detection handler. The CCD library that we utilize for deformable simulated objects is the Self-CCD collision detection library. This generalized polygon-soup [35] approach removes the categorization of 'objects' within the simulation and only considers the movement of primitives over time.

Collision detection represents the second stage in the standard method of simulation after the dynamics engine updates the positions of the nodes within the mass-spring meshes. Since various techniques to collision detection have been implemented, each with their own strengths, we account for generalization of the abstract collision detection process. This is illustrated through the introduction of modular and interchangeable collision detection handlers in Chapter 11. Utilizing this flexibility, we can interchange collision detection handlers for the physical simulation based on the

requirements of the artist. This allows for higher levels of accuracy when required and alternatively, the performance of the simulation can be increased by selecting more efficient collision detection algorithms.

**Collision Response**

In addition to the process of detecting collisions and interpenetrations between simulated objects, a collision response process must be defined to properly resolve these collisions. The collision response process is defined through enforcing the physical properties of the simulated objects, that is, it must prevent object penetrations to enforce the pseudo-solid state of the deformable objects. This is required for any physical simulation that must preserve the physical characteristics of the simulated objects. The interactions between the simulated objects determine the physical and visual fidelity of the entire simulation; therefore the quality of the resulting animation is directly affected.

Various approaches for enforcing these conditions have been proposed over the course of several years. Most of these common techniques fall within two collision resolution models: impulse-based and penetration-penalty reaction. Impulse resolution models apply small instantaneous forces to the penetration regions of the surface and artificially adjust the position of the surface areas that are causing the penetration to a valid non-penetration state. This means that penetrations are instantly corrected and the forces involved in the collision will properly repel the involved objects. This solution provides an instantaneous result due to each impulse being applied during an individual time-step. This provides the instantaneous impulse that changes the velocity of the surfaces involved in the collision, and as this change propagates through the deformable representation, secondary motions are introduced. This resolves the collision state of the

objects and allows the simulation to continue. This provides an efficient means of resolving collisions between objects in the simulation however; it is not the most physically accurate implementation of a collision resolution technique. Next we consider looking at penetration-penalty based resolution schemes and consider how the presented approach may be designed to handle other methods of collision resolution.

In the attempt to model more physically accurate collision responses, penetration-penalty systems have been proposed. These systems try to adhere to the laws of physics involved in real-world collisions. The force imposed on one object from a colliding object is met with an opposite and equal reaction force, thus surface penetration is prevented. Penetration-penalty based systems seek to follow this principle by applying this reaction force to the colliding object. While the penetration exists, this force is applied to the object to eliminate the penetration (typically these forces are imposed by temporary springs). A noticeable side-effect of this approach is that, due to the penalty force requiring a length of time ($dt$) to act over, the objects will momentarily in a penetration state. While this approach closely simulates the theory of real-world physical collisions, the visual result can be negatively influenced by the duration of the penalty force application (visually the objects remain in a penetrated state long enough to break the illusion of correct physical interaction). Recent modifications [36] to this approach have been made to reduce this reduction in visual quality; however this problem remains persistent.

The collision response scheme that was selected for the implantation of the proposed method of controlling deformable objects and generating animations in real-time is an impulse-based method. The implementation of this collision response scheme

provides a simple and efficient means of preventing object penetrations. Additionally it is fast enough to ensure interactive rates with the constructed editing environment that can be provided to an artist. The accuracy of the collision response method is orthogonal to the introduction of artificial external forces to modify the behavioral result of the animation. Therefore this decision does not directly contribute to the proposed methodology of modifying the physically-based system but rather is a general requirement of any physical simulation. We note that to achieve the required interactive rates required for real-time animation editing, the accuracy of the collision response mechanism has to be sacrificed. This sacrifice in accuracy provides an enormous performance gain but also increases the probability that a collision artifact may be introduced into the recorded animation.

In the development of this approach we incorporate flexibility in this decision through the creation of modular collision resolution handlers that can be dynamically interchanged for the simulation. The implemented collision response scheme can be replace with an algorithm that provides higher accuracy in the response to the detected collisions. The only constraint that we place on the introduction of a new collision response mechanism is that it adheres to the performance requirements of the interactive editing environment. This is critical due to the interactive real-time feedback loop that we provide to the user about the state of the simulation and currently recorded animation. The consequence of utilizing a collision response handler that provides a high level of accuracy is that the rate of interaction between the user and the application deteriorate as the accuracy increases. The provided implementation approximates collision reactions through an impulse-based method to ensure that the responsiveness of the application

allows the artist to receive immediate feedback related to their modifications to the current animation.

**Mass-Spring Simulation Stability**

While the compositions of most mass-spring systems lead to generally stable simulations, specific care must be taken to ensure that the forces within the modeled objects do not lead to diverging approximations in the applied numerical integration scheme. We have provided large stable time-steps through the use of the implicit Euler method; however several factors can contribute to an instable simulation. (1) The physical representation of the mass-spring system differs for each object, therefore the internal forces and their propagation will be unique to the connectivity provided. This can contribute to instability because of the following scenario: external force $f$ is applied to object $A$ that has connectivity $A_c$, this same force is then applied to object $B$ that has connectivity $B_c$ where $A_c \neq B_c$ but the same set of nodes are used for each object. The resulting behavior of each object will differ due to how the propagation of the applied force alters the state of the incorporated nodes. (2) The scale and mass of the object and the relative magnitude of the forces introduced by the internal springs can lead to instability. The application of an arbitrarily large external force can create large self-penetrations on light-weight objects. This makes the process of continuous self-collision required in the collision detection phase; otherwise a node can "tunnel" through itself causing an invalid self-collision state. (3) Inaccurate collision responses can break the assumed continuity of the approximated dynamics functions, thus introducing errors into higher-order approximation techniques. The last and most prominent contribution to an

instable simulation in the development of this technique is the introduction of arbitrarily large external forces.

While stability is a concern for all deformable-body simulations, we must provide an additional means of ensuring that the applied external forces do not lead to a diverging simulation state. We address this concern with the introduction of dynamic previews (introduced in Chapter 5). Essentially since we know the simulation can become unstable due to the arbitrary large external forces applied to one of the objects within the simulation, we allow the artists to generate the next $n$ time-steps of the simulation to preview the results of the applied external forces. If the simulation becomes unstable due to the applied forces then we can inform the artist that their current edit violates the force boundary of the simulated object. While this does not eliminate the instability, it provides an effective means to inform the artist that the current behavior they would like to achieve is not obtainable from the current state of the simulation.

This is a critical aspect of developing high-level controls for deformable objects. When the simulation cannot achieve the desired result, providing feedback to the artist allows them to alter their editing parameters. From this we introduce an iterative method of controlling deformable-bodies with high level motion controls.

# CHAPTER IV

## HIGH-LEVEL DEFORMABLE-BODY MOTION CONTROL

The modification of an existing animation based on a physical simulation is a sub-domain within the field of simulated object motion control that has not been extensively addressed in prior research efforts. Most techniques that are introduced consider the initial setup of an animation then derive the resulting motions required to meet the artist's specifications for the animation as a whole. While these approaches provide physically plausible motions to reach the desired states, the level of interaction between the artist and the resulting animation is limited. This means that the artist must predetermine what motions they want to define to specify the behavior of all physically simulated objects before the animation exists. Typically this is not the case. If we can generate an animation that is relatively close to our intended result, then modifying the specific aspects of the animation that exists will be an easier process than redefining the entire animation. Furthermore there may be aspects of the current animation that we do not want to lose by redefining the entire animation. If we modify the entire animation we may lose these existing desire behaviors in our effort to modify only a small portion of the overall animation.

The core of this approach for providing high-level motion controls that modify localized behaviors of deformable objects, aims to succinctly address these deficiencies. We build upon the notion of modifying existing deformable-body animations with high-level controls and provide real-time motion feed-back to the artist. Additionally we provide the artist with the ability to dynamically modify the results of the recorded physically-based simulation. With this approach, an artist will be able to effectively

modify the localized deformations of simulated objects in an existing animation and incrementally adjust the animation to match their artistic vision.

**Control Metaphors**

The separation between the intended motion of a simulated object and the complex configuration of external forces required to achieve that result should be minimized from the perspective of an artist. Essentially, most common behaviors that are required to create a physically plausible animation can be identified as a set of high-level motion controls. A high-level control is simply defined as an overall motion enacted by an entire object or part of its surface as a local deformation. These are the controls that we aim to provide to an artist through the interactive editing process provided in this approach to deformable-body control.

The fundamental concept that we introduce with the proposed method of deformable object control is the notion of a high-level control metaphor. A high-level control metaphor represents a mapping between a control that specifies a motion or behavior and the physical implementation of that motion by a simulated object. This provides an artist with a generalized control methodology that can be used to direct the behavior of deformable-bodies.

Considering the set of fundamental control metaphors that can be defined for several abstract motions we define metaphors for bending, twisting, pushing, stretching, and compression. These are examples of high-level control metaphors that can be used to effectively modify the motion or localized deformation of a simulated deformable-body. The control metaphors specify a behavioral pattern of forces that derive some predefined motion. The reaction of the deformable-body to match the described behavior is

identified as the pattern adherence. The level of adherence to the prescribed motion illustrated by a deformable body depends on several characteristics of the control metaphors definition and how the external forces acting on the object are maintained while the object moves and deforms. Here we define these characteristics that are common to all control metaphors are required for the implementation of this approach.

At a technical level, control metaphors provide a mapping between these simple high-level motions and the sets of external forces required to impose these behaviors onto a deformable object. There are several components that contribute to the abstract representation of a motion that must be defined to develop a control metaphor. The collection of the parameters that define positions and orientations of the external forces that will be applied to the body to obtain some desired motion define the motion pattern to which the object must adhere. The main specification of a control metaphor is made through the pattern of external forces that it utilizes to achieve some desired motion. A control metaphor pattern $p$ defines the sets of node indices that are affected by the applied external forces. When considering a deformable-body with $n$ nodes we note that $|p| = n,$ where each node receives an independent external force. However, at this point the control provided by the metaphor has effectively been reduced and provides no additional utility towards obtaining an abstract motion. Therefore we minimize the total number of force sets that will be defined to achieve some intended motion. With the definition of the index sets that reference subsets of the nodes from the deformable-body, we can now consider the orientation of the external forces to derive some intended behavior.

As the second characteristic of a control metaphor that maps a high-level motion definition to the external forces applied to an object, the force orientations must be defined for each force set. Typically, simple behaviors can be derived from limited force sets with uniform force orientations. More complex behaviors can be derived by utilizing non-uniform force patterns for the selected sets of nodes; however the primitive motions we introduce can be obtained through uniform forces on a minimal number of force sets. Again the orientation of these forces must be consistent with the object it influences as the simulation progresses. This imposes a requirement that the force must be specified in the same reference frame as the deforming object. From this we can conclude that to define a control metaphor we must introduce a reference frame that tracks the movement and rotation of the object to properly define the orientation of the external forces.

The requirements of a control metaphor are dictated by the definition of the abstract motion pattern interface. These are the sets of nodes that will be affected by external forces and the orientations of these uniform force distributions. In this approach we assert that this is an abstract representation of a control metaphor because based on this simple premise, we are able to develop several controls that define unique deformation behaviors. To illustrate the flexibility of this technique we also permit multiple control metaphors to be applied to a single deformable object. This allows the artist to create complex behaviors based on the contributions made by simple independent motions.

With this approach we provide an efficient means for controlling the behavior of deformable objects within existing animations. Additionally we simplify this process by providing an artist with real-time feedback based on how they apply these generalized

motion control parameters to the existing animated objects. In this chapter we define the mapping between these high-level control metaphors and the external forces they must generate to modify the behaviors of the deformable-bodies in an existing animation.

**Local Coordinates**

Based on the notion of modifying the behavior of a single deformable-body, we must consider how high-level control metaphors should be applied to an object that can be translated, rotated, and deformed over time. When specifying a control metaphor to modify the particular motion of an object, the desired result is that the modification is made in the same location over time regardless of the translation and rotation of the object. Based on this simple requirement we must define a coordinate system that remains consistent with the geometric definition of the deformable-body. This will allow the control metaphor to be consistently applied to the object over time while also providing the expected deformation control.

Since the position of a deformable-body, its center of mass, is defined global coordinates, we define a secondary coordinate system at this position of the object. This newly introduced coordinate system defines the origin of the local coordinates of the deformable-body. Defining this position is trivial; however, to define a local coordinate system we must also define three orthogonal directional vectors that form the x, y, and z axes of the new coordinate system. Defining these axes for a deformable object is particularly challenging due to the deformations the of objects geometric definition. The required mapping from the geometric definition of the deformable-body to a set of rigid orthogonal axes cannot be derived from any direct single state of the object. As the object deforms over time, the geometric definition is modified and any direct correlation made

between the geometric definition and the set of orthogonal vectors that compose the local coordinate system will become invalid. In this section we introduce a robust method to determining this relationship between the deforming geometric definition and the set of orthogonal direction vectors that compose the local coordinate system. We build upon existing techniques to provide a stable and accurate representation of the rotation of a deformable-body.

Essentially the derivation of this set of orthogonal vectors is characterized by two criteria: (1) an appropriate estimation that matches the geometry of the object and (2) the stability in the representation of the objects rotation. The first criteria is critical for establishing a stable foundation for the second criteria, therefore we examine and define how each of these issues can be addressed to provide a robust local coordinate transformation.

In this approach to deriving the local coordinate system axes, we propose a two phase system. This system will initially define the best alignment of the coordinate system with the provided geometric representation of the deformable-body and then subsequently update the set of orthogonal vectors based on this alignment. We utilize a planar orthogonal regression to determine the best-fit plane of the data which defines the best fit alignment used to generate a basis for the local coordinate system. We then derive the set of axes by indirectly referencing the geometric definition of the deformable-body. Together, this two phase system provides an accurate and robust mapping between the deformable object and a rigid coordinate system. In the next section we define this process and how it is used to construct the local coordinate system utilized for our simulated objects.

**Robust Deformable Body Local Transformation Estimation**

The derivation of an accurate local coordinate system is essential to establishing an effective means of translating a high-level motion between a control metaphor and the sets of external forces that will act upon the nodes of the deformable-body to impose the intended motion. Previous research efforts [25] attempt to establish a basis for the axes of the local coordinates system; however they rely on the assumption that a large portion of the center of mass is defined on the vertical axes of the global coordinate system. To provide a robust method of determining the local coordinate system, this approach does not rely on this assumption. This approach generates a valid set of orthogonal axes for the geometric classes where the prior approach would fail to determine a valid coordinate system (ex, the class of planar geometric objects defined in the $x$-$z$ plane that have no vertical component).

The motivation of this approach is to define a local coordinate system that will best represent the geometric definition of the object and provide an accurate representation of the rotation of the deformable-body. The accuracy of the mapping between the rotation of the deformable-body and the rotation of the rigid coordinate system directly influences the ability to effectively modify the objects behavior to match the requirements of the control metaphor. Therefore maximizing the stability of the generated coordinate system is essential for modifying the behavior of simulated deformable-bodies.

The two-phase approach to generating the initial configuration and updating the local coordinate system that we propose provides a stable set of orthogonal axes for deformable object geometric definitions containing at least three nodes. The first phase of defining the local coordinate system is a static analysis process. When the geometric

definition of the deformable-body is loaded into the physical simulation as a mass-spring mesh, we analyze the configuration and the distribution of the vertices that compose the nodes of the system. This process of static analysis is performed before the simulation begins and therefore does not add to the cost of updating the physical state of the object.

The static analysis of this process defines a constant set of node references that are used to calculate and update the rotation of the local coordinate system. This update process introduces the second phase of this algorithm. When the simulated object is deforming over time, the positions of the nodes will be updated based on the forces enacted upon them. As the positions are updated we utilize the node references established in the initialization phase to update the geometric data upon which our local coordinate system is defined. The combination of this initialization and update process allows for an efficient generation of the local coordinate system based on the deforming objects geometric definition. In the next two sections we provide a detailed overview of each phase and the calculation of the local coordinate system.

**Orthogonal Regression and Static Analysis**

The static initialization process of this phase utilizes orthogonal regression to determine the best-fit plane for the provided geometric definition of the deformable-body. For this analysis we only consider the initial rest positions of the nodes that are defined by deformable-body. This is essentially a static set of points in three dimensions. Using this data-set we perform an orthogonal regression to determine the best-fit plane. Orthogonal regression is utilized due to the requirement of a robust solution to determining the local coordinate axes. Other statistically-based best-fit methods will not provide a robust best-fit plane for the given set of points for all geometric definitions.

Specifically if multiple-regression is used, then the definition of the regression plane is determined by the squared $y$-axis deviations within the data-set. This creates problems for instances of geometric objects such as a vertical cloth that lies within the $x$-$y$ or $z$-$y$ plane. Utilizing orthogonal regression we identify the best-fit plane $P$ of the data by minimizing the orthogonal distances to the all of the points in the provided geometric definition. From this process we derive a standard definition of the best-fit plane as a point and a normal. The point $P_0$ represents the center of mass of the deformable node set and the normal $P_n$ is the normal of the plane that best fits the data. We do not specify a preference for which handedness of $P_n$.

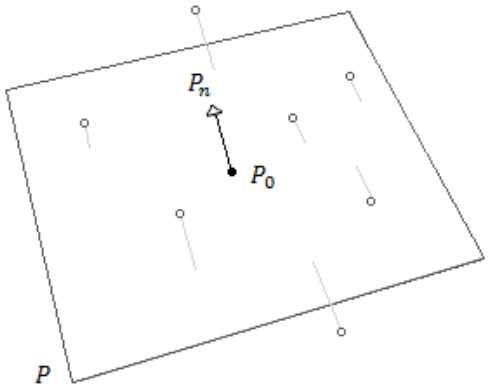



**Figure 4.1. Planar Orthogonal Regression**
Orthogonal regression of the rest state of the deformable object to determine the best-fit plane of the data. The regression provides the geometric centroid of the data, $P_0$ and the normal $P_n$ of the best-fit plane $P$.

Considering the best-fit plane $P$ we can proceed to generate the $x$-axis of our local coordinate system. Since the definition of the first axis of the coordinate system is unbound in its rotation, we utilize a basic heuristic to attempt to provide the direction that is most accurately aligned with the distribution of the mass of the node set and volume that the deformable body occupies. This basic heuristic is to select a node from the

deformable object that is furthest away from the center of mass after it has been projected onto $P$. The image in Figure 4.2 illustrates the best-fit plane $P$, the selected furthest node $f_i$ and the projection of $f_i$ onto $P$ noted as $f_i'$.

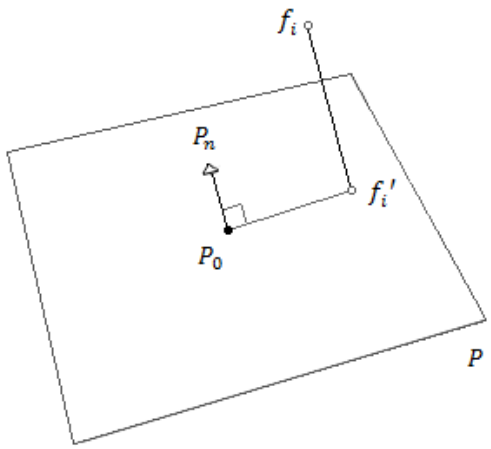



**Figure 4.2. Furthest-point Projection Heuristic for Local Coordinate Generation**
An illustration of the best-fit plane based on point $P_0$ and the normal $P_n$. The point $f_i'$ that is coplanar with $P$ is defined as the projection of some node $f$ at index $i$ such that the distance between this projected point and $P_0$ is maximized.

Since the initial configuration of the geometric data is completely arbitrary, we simply iterate through the node set, project the current node $f_i$ onto the plane $P$ and calculate the distance $d$ between $P_0$ and $f_i'$. At this point we note the index $i$ of this furthest projected node, as it will be used to generate the general direction of the $x$-axis we are creating.

Since we are unaware of the physical implications of the connectivity of the selected node, it may undergo drastic deformations throughout the duration of the simulation. This will negatively impact the stability of the coordinate system and jeopardize the accuracy of the rotation modeled by the local coordinate system. If the furthest selected node $f_i$ is included in a large local deformation, and the direction of $x$-axis of the local coordinate system is defined by the vector from the center of mass ($com$)

to the selected point, then the rotation of our local coordinate system will be drastically altered to match this individual nodes position. The image in Figure 4.3 illustrates that when this localized deformation occurs on the single selected node $f_i$, the direction of the $x$-axis (shown in red) defined by $f_i'$ will be drastically altered and will no longer provide an accurate representation of the rotation of the deformable-body.

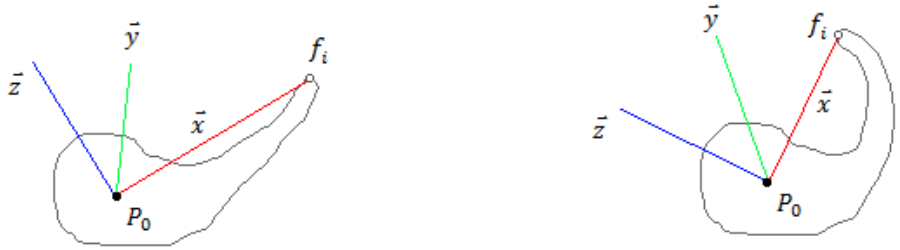



**Figure 4.3. Rotationally Instable Local Coordinate Derivation**
The (left) image illustrates the undeformed geometric structure of the deformable-body with the $x$-axis of the local coordinate system defined as a single node. The local coordinate system is well aligned with the object. The image on the (right) illustrates that this approach provides an unstable representation of the rotation of the deformable body. The local coordinate system indicates that the deformable body has undergone a large counter-clockwise rotation. This is a misrepresentation of the behavior exhibited by the larger portion of the object which has undergone minimal rotation.

To alleviate this problem we consider the selection of more than one node to define the initial direction of the $x$-axis in the local coordinate system. Selecting a larger group of nodes to derive the direction of this axis will provide stability in cases where large local deformations are introduced. To derive this set of selected nodes, let $X$ be the original set of all nodes and define the initial direction from the center of mass to the furthest selected node projected onto $P$ as $\overrightarrow{x_\iota} = (f_i' - com)$. This provides an initial direction that will be utilized to select a subset of points that will be used to create the $x$-axis of the coordinate system. With the direction $\overrightarrow{x_\iota}$ pointing towards the projected point $f_i'$ from the center of mass, we iterate through the remaining nodes, $X \backslash \{f_i\}$, to determine

the indices of the nodes closely surround the selected node. For each remaining node $r_i \in X \backslash \{f_i\}$, we define a direction from the center of mass to the node as $\vec{r_i} = (r_i - com)$. We now define a selection cone from at the center of mass that will be used to create a new set of nodes $Y$ that will be utilized to calculate the $x$-axis direction for our coordinate system. This selection cone is implicitly made by the following selection constraint: a remaining node is selected if and only if $(\overrightarrow{x'_i}) \cdot (\vec{r_i}) < t$ where $(0.0 \leq t < 1.0)$. This parameter is configurable, and represents the angle of the cone formed around the directional vector $\overrightarrow{x_i}$. Figure 4.4 illustrates the selection of the nodes that fall within the volume of the selection cone and will be placed into the set $Y$.

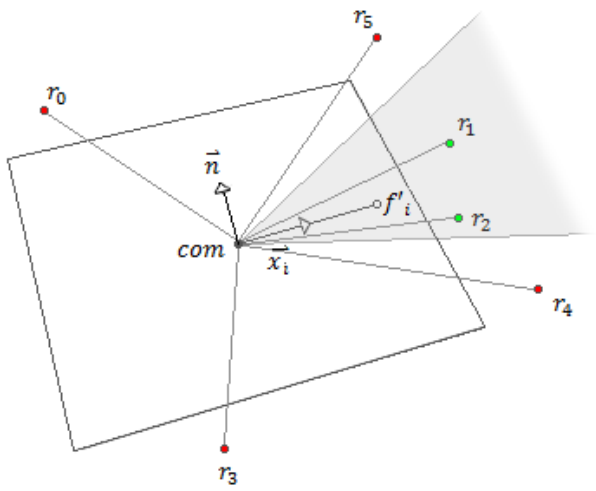



**Figure 4.4. Point-Cloud Subset Selection for Rotationally Stable Coordinates**
The image depicts the selection cone (shaded in light gray), and the nodes that reside within its volume. The green nodes $r_1$ and $r_2$ will contribute to the definition of the $x$-axis for the local coordinate system and are added to the set $Y$. The red nodes, $r_3$through $r_5$ do not contribute to the direction of the $x$-axis.

Once the set of nodes within the cone have been selected, the center of mass of the nodes within the selection, defined as $Y_{com}$ will be determined. From this we define the $x$-axis of our local coordinate system as $\vec{x} = (Y_{com} - com)$. The normalization of the

directional vector $\vec{x}$ completes the derivation of the direction of the $x$-axis. To update the $Y_{com}$ during the simulation, we note the indices of nodes within the set $Y$.

We now consider the derivation of the direction of the $y$-axis for the local coordinate system. Unlike the process of selecting the $x$-aixs of the local coordinate system, the process of determining the $y$-axis is subject to an additional constraint: the direction of the $y$-axis that we calculate must be orthogonal to the previously selected $x$-axis. Since the local coordinate system must be locked to the rotation of the object, we must provide a mapping between the deformable body and the $y$-axis. However, due to the required orthogonality of the coordinate system, we cannot guarantee that there will always be a node or center of mass of a collection of nodes that will be orthogonal to the previously defined $x$-axis. Therefore in this approach we determine the index $i$ of the node $y$ that is most orthogonal to our $x$-axis and store the index. The direction from the center of mass to this selected node may not be orthogonal to direction of the $x$-axis; however we can derive a rotation axis based on these two vectors. Figure 4.5 illustrates the previously defined $x$-axis, the most orthogonal direction defined by some node $\overrightarrow{y_i} = (y_i - com)$, and the rotation-axis: $\overrightarrow{rot} = \vec{x} \times \overrightarrow{y_i}$.

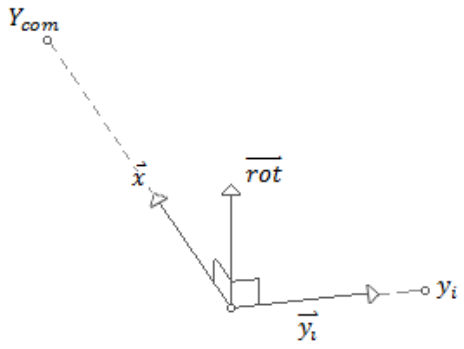



**Figure 4.5. Local Coordinate Rotation Axis**

Rotation axis $\overrightarrow{rot}$ defined as the cross product between the $x$-axis and the directional vector $\overrightarrow{y_i}$. This rotation axis will be used to determine the orthogonal $y$-axis of the local coordinate system.

The process of obtaining the $y$-axis of the local coordinate system is based on the rotation axis and the previously determined $x$-axis. We can obtain an orthogonal $y$-axis by simply rotating a copy of the $x$-axis about the $\overrightarrow{rot}$ axis by exactly 90[deg]. This will produce a set of vectors that are orthogonal in two dimensions. With the definition of the $x$-axis and the $y$-axis that are orthogonal, the derivation of the $z$-axis of the local coordinate system is trivial. We simply define the $z$-axis as the cross product between the $x$-axis and the $y$-axis.

**Dynamic Local Coordinate Updates**

During the static analysis phase that determines the initial set of nodes that define the local coordinate system, the calculated axes are based on the static representation of the deformable-body. From the static analysis we stored the indices of several nodes within the geometric definition of the simulated object. Specifically we note the set $Y$ that contains all of the nodes that reside within the selection cone illustrated in Figure 4.4. Additionally, we note the index of the node that was most orthogonal to the $x$-axis when the object was at rest. These indices will be utilized to determine the positions of the nodes as they are updated by the simulation. Since the nodes are continuously updated, the local coordinate system must be recalculated for every simulation time-step. This represents the second phase of our local coordinate generation algorithm, and closely matches the static analysis phase with the exception that we do not generate new node sets, but rather use the existing indices to reference the updated node data. As the simulation progresses, the change in the node positions will result in the local coordinate system being updated to represent the movement and rotation of the deformable-body.

The resulting local coordinate system that is calculated using this method provides a stable orthogonal axes set that accurately depicts the rotation of the deformable-body. The images in Figure 4.6 illustrate the result of this technique on two different deformable objects. The initial rotation of this generated local coordinate system does not need to exactly match the geometry of the object, rather the stability of the rotations that the coordinate system represent is the critical factor in how effective the system will be when impose the motions defined by control metaphors. The local coordinate systems in Figure 4.6 have been emphasized to clearly show their orientations.

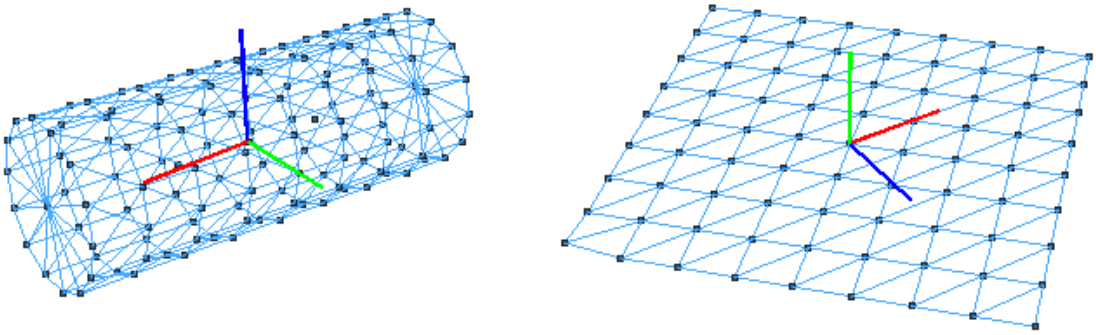


**Figure 4.6. Accurate Alignment of the Generated Local Coordinate Systems**
The image (left) shows a deformable cylinder at rest. The local coordinate system is illustrated as follows: red ($x$-axis), green ($y$-axis), and blue ($z$-axis). The image (right) shows a deformable cloth with the local coordinate system specific to its geometric definition (axis colors same as left). The calculated local coordinate system is accurately aligned for primitive types shown in both images. Also note here that the relative rotations of the local coordinate systems shown in these images are independent.

In addition to the required stability, an accurate initial alignment of the generated local coordinate system is illustrated in Figure 4.6; we also note the visibility of stability of this approach when the object undergoes local deformations. To view how the stability can be illustrated, we consider the pre and post deformation states of a deformable-body and show that the local coordinate system maintains an accurate representation of the objects position and rotation. Specifically the stability of the coordinate system is

characterized as its ability to resist rotations that do not represent the rotation of the entire deformable structure. When local regions of the deformable-body are stretched and deformed, they do not represent the rotation of the body as a whole. For our intended purpose, we only want the local coordinate system to define the rotation of the entire deformable object. The pair of images shown in Figure 4.7 illustrates the state of a deformable cylinder before and it has been deformed with a bend operation. The stability that we require is shown by the relatively unmodified rotation of the local coordinate system between these two states. Since the deformable body has undergone a substantial deformation, but has not introduced an incorrect rotation to the local coordinate system, we assert that this approach provides a highly stable representation of the rotation of the deformable-body.

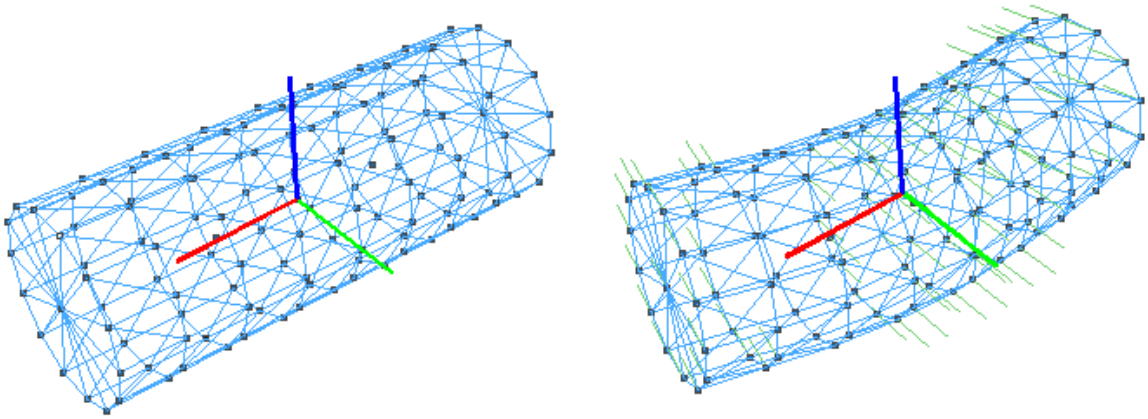


**Figure 4.7. A Rotationally Stable Local Coordinate System**
The state of a deformable cylinder before and after an bend operation has been applied. The local coordinate system remains stable and does not reflect a false rotation created by this deformation of the object. These images have been taken from the same perspective to illustrate that local coordinate axes ($x$-axis:red, $y$-axis:blue, and $z$-axis:blue) have not rotated due to the deformation.

The stability of the local coordinate's representation of the rotation of the object is critical to implementing control metaphors that impose a specific behavior on the

simulated object. The ability of this technique to calculate the local coordinate system and resist false rotations introduced by deformations of the object provides the basis upon which we develop the application of external forces to the object from this consistent frame of reference.

**Control Coordinates**

The introduction of a local coordinate system provides the necessary consistent reference frame that is required to apply constant external forces to a specified region of a deformable-body throughout a physical simulation. The generation of the external forces used to modify the motion or behavior of a deformable-body must be consistent in their position and orientation in relation to the object to create localized deformations, otherwise the objet would simply be pushed away from the external force. The local coordinate system that we introduced in the previous section provides an accurate representation of the objects center of mass and its rotation. However to facilitate localized deformations we must be able to define other locations within the local coordinate system. Therefore within the local coordinate system we introduce another coordinate system identified as the control coordinate system.

The control coordinate system defines the location and orientation of a localized region that we would like to perform a deformation. From the direct definition of a Cartesian coordinate system, we must provide both an origin and set of orthogonal axes for this control system. Here we note that the origin of the control coordinate system is specified in local coordinate system. Additionally the control coordinate system does not define a rotation. The user will provide the custom rotation through interacting with the interface of the implemented animation editing application.

The position of the control coordinate system is defined only within the local coordinate system. For a simplistic level of interaction we provide the ability to the artist to select the existing nodes of the deformable body. The origin of the control coordinates can be modified to match the position of one of these existing nodes. This sets the origin of the control coordinate system to be equal to the position of the selected node. This introduces a flexible interface for applying targeted local deformations to the selected object at the selected nodes position. Therefore the process of applying a control metaphor directly to a localized region of a deformable-body is enabled by simply selecting a node within the region.

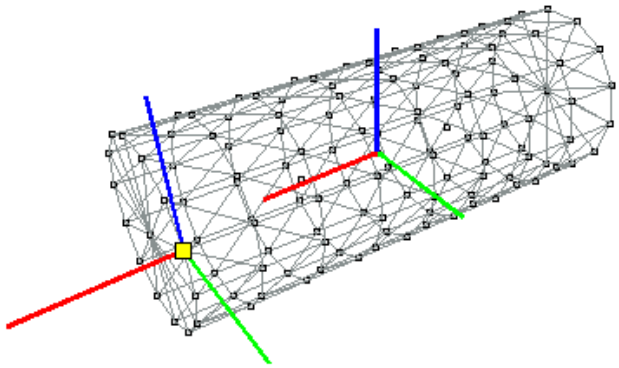


**Figure 4.8. Control Coordinate System**
The point (highlighted in yellow) is the node that the artist has selected as the location to apply the localized deformation. The control coordinate axis is also shown.

Given that the origin of the control coordinate system has been defined, we must also provide the orthogonal set of axes that define its rotation. When a control metaphor is initialized the control coordinates that it utilizes are initialized to have no rotation in reference to the local coordinate system. The rotation of the control coordinate system is only modified when the artist intentionally modifies the orientation of the selected control metaphor. This control coordinate system provides the artist with the freedom of directly

manipulating how the control metaphors external forces will be applied to the geometric definition of the deformable object.

**Compound Motions with Control Metaphors**

The application of this approach provides the flexibility to not only provide control metaphors to an object at any location, but also includes the ability to assign multiple control metaphors to the same object. The control metaphors that are assigned to the same deformable-body may or may not affect the same locations of the mesh. Due to the ability to target the control metaphors to create local deformations they may not interfere with one another. However we consider the case where there are multiple control metaphors that target the same localized area of the deformable object, or that target the global behavior of all nodes within the object. When this case is presented we calculate the net force acting upon each node to obtain its final external force.

$$NetForce(n) = \sum_{i}^{c} extForce(i, n)$$

**Equation 4.1. Net Force Imposed by Multiple Control Metaphors**
The net force acting upon node $n$ is the vector sum of all external forces provided by $c$ control metaphors.

This defines the net force on node $n$ as vector sum of all external forces provided by $c$ control metaphors. This allows for the generation of complex behaviors based on the development of simple control metaphors. Even if the provided set of control metaphors is relatively primitive, the external forces from each metaphor can be combined to create complex motions. We utilize this fact to not limit the possible deformations that can be generated with this approach. In the effort to incorporate artistic interaction in the development of controlled physically generated animations we look at how this essentially provides the artist with full control over the result of their work.

# CHAPTER V

## PRIMITIVE CONTROL METAPHOR IMPLEMENATIONS

Based on the abstract definition of a control metaphor that has been provided in Chapter 4, we define the design and implementations of four primitive control metaphors. Each of these implemented control metaphors provide a unique mapping between the force pattern defined in the controls coordinate space required to obtain the described motion and the resulting global forces that are applied to the simulated object. For each control metaphor there are several components that must be defined: (1) the unique force map that is used to obtain some primitive motion, (2) a geometric definition of how affected nodes are selected, and (3) a definition of how the artist will interact with visual representation of the control metaphor to alter the motions it generates.

When the operations that characterize a control metaphor are defined, they always operate in the control coordinate system introduced in Chapter 4. This simplifies the process of transforming the global positions of the nodes of the deformable object to the control coordinate space for all implemented control metaphors. To perform this operation on the global positions of the nodes from the deformable body we simply apply the inverse of the local coordinate transformation, and then apply the inverse of the control coordinate transformation. This will efficiently transform the global node position into the control coordinate system of the selected metaphor where it will become a selection candidate that may to receive an external force generated by the metaphor.

Since we translate the global node positions to the control coordinate system to derive the desired external forces, we must apply the appropriate rotations to these derived forces to convert them into equivalent forces in the global coordinate system.

This is achieved by defining the orientation of the external force in control coordinates and then directly applying the control coordinate transformation to get the orientation of the force in local coordinates. We then directly apply the local coordinate transformation to get the orientation of the force in the global coordinate system. Once the direction of this force is defined in global coordinates and the magnitude is provided by the force curve editor, it can be applied in the next simulation time-step.

**Stretch Control Metaphor Implementation**

This control metaphor introduces one of the simplest motions that can be applied to a deformable body. With this control, the intended motion we aim to achieve is the elongation of the deformable body in one dimension. Simply stated, we intend to stretch the object along the $x$-axis of the control coordinate system.

The overview of this process is defined by the following: select the set of affected nodes based on their $x$-coorindate in the control coordinate system. There are three artist defined parameters: (1) the minimum $x$ value that a node can have to be considered for selection, (2) a maximum $x$ value that a node can have to be selected and (3) an effect radius. The diagram in Figure 5.1 illustrates these selection parameters.

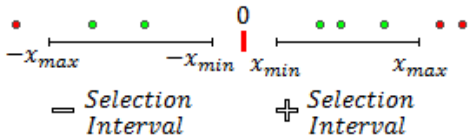



**Figure 5.1. Stretch Control Metaphor Node Selection**

Stretch control metaphor node selection. The nodes with the $x$-coordinate in the two defined selection intervals will have external forces applied to them if they are within the affect radius (the orthogonal distance from the $x$-axis in the control coordinate space). The selected nodes are highlighted in (green) and the rejected nodes are highlighted in (red).

Once the two selection sets have been populated with the nodes that reside in the selection interval, the force pattern unique to the stretch control metaphor can be applied. The simple force diagram illustrated in Figure 5.2 shows the direction of the forces that will be uniformly applied to the nodes of the two selection sets. The force that points in the left direction will be applied to the nodes within the negative selection interval and the force that points to the right will be applied to the nodes within the positive selection interval.

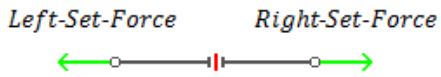



**Figure 5.2. Stretch Control Metaphor Unique Force Pattern**
Direction of the forces that define the motion behavior of the stretch control metaphor.

The stretch control metaphor can also be easily transformed to provide the forces that will compress a deformable object. Using the existing definition of the control metaphor we only have to simply provide a negative force magnitude to flip the directions of the applied external forces. When this is the case, the force acting upon the right set of nodes will point to the left and the force acting upon the left set of nodes will point to the right. This pushes the intermediate nodes closer, thus compressing the body.

The visual representation of this control metaphor is relatively simple. As with all control metaphor representations, we define a control widget, which defines the visual representation of the interactive 3D controls that are provided to the artist to manipulate the parameters of the control metaphor. All control metaphors are provided with a widget that provides a unit sphere that can be rotated to define the orientation of the control coordinates of the selected metaphor. The image in Figure 5.3 shows the control widget that has been implemented for the stretch control metaphor. The similarities between the

control widget and the set of configurable parameters defined in Figure 5.1 should be easily identified. The nodes of the deformable object that have been selected are highlighted to show that there are two distinct sets that will receive opposing external forces.

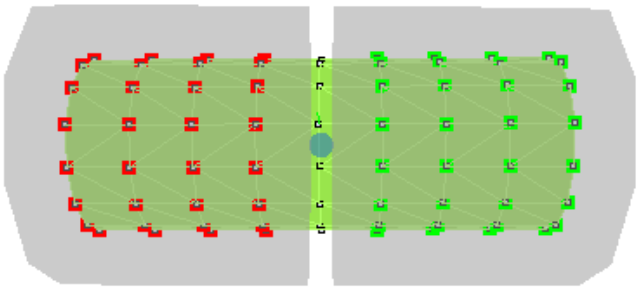


**Figure 5.3. Cage-based Control Widget for the Stretch Control Metaphor**
The control widget that allows an artist to select the nodes affected by this metaphor is displayed over the geometry of the deformable object. The selected nodes highlighted in (red) and (green) represent the nodes that will receive an external force from this metaphor. The shaded cylinders represent the three dimensional implementation of the selection intervals and the effect radius specific to this metaphor.

The introduction of the control widget that bounds the structure of the deformable-body is adaptation of the cage control structure introduced [18] to provide direct control of deformable objects. In this instance we can utilize a similar concept to quickly identify the nodes that we would like to control with the selected metaphor.

### Bend Control Metaphor Implementation

The bend metaphor produces a simple motion of an object being deformed in one dimension where the external forces try to pull different regions of the deformable object apart. With this control the motion we intend to achieve is a simple one-pivot bend. This is accomplished by identifying three node sets and then applying a uniform force to two of these sets and then applying the inverse of this force to the third joint set.

We define the joint set as the set of nodes that will provide the focal point for the bend and the left and right node sets as the bend sets. Typically the ideal position for the joint set is directly in between the two bend sets. This is because this will result in an equal amount of force being applied to both sides of the pressure point. This will provide a uniform bend that matches the intent of the control metaphor

To define the properties of the bend control metaphor and the motion it will produce there are five variables that can be controlled by the artist: (1) the imposed bend angle which will modify the direction of the bend forces, (2) the effect radius of each bend set and (3) the distance between the bend sets. The diagram in Figure 5.4 illustrates the definition of these three sets.

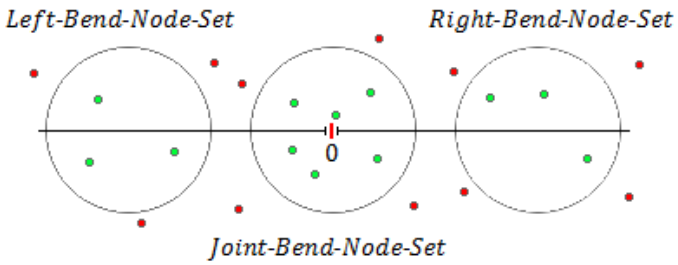



**Figure 5.4. Bend Control Metaphor Node Selection**
Bend control metaphor node selection with a bend angle of 0.0

The derivation of the bend forces is more complex than the process used for the stretch control metaphor. This is due to the geometric definition of the bend control metaphor. Based at the origin of the control coordinate system, we define two vectors that will act as bend arms. These originally start out as negative and positive unit $x$ vectors. The definition of the bend angle $\theta$ determines how these vectors are rotation about the $z$-axis of the control coordinate system. The result of this rotation on both bend arms is illustrated in Figure 5.5.

The length of these arms depends on the artist specified setting; however they are initially calculated using the center of mass of the nodes selected by a effect radius of $|CP_{cross}|/3$. Additionally we define the ends of the bend arms as the left bend node $L_{bn}$ and the right bend node $R_{bn}$ to form two directional vectors $\overrightarrow{L_{bn}} = (L_{bn} - O)$ and $\overrightarrow{R_{bn}} = (R_{bn} - O)$ where $O$ represents the origin of the control coordinate system. Then we define the bend left force direction as $\overrightarrow{L_{bn}} \times \hat{z}$ and the right bend force direction as $\overrightarrow{R_{bn}} \times \hat{z}$ (where $\hat{z}$ is a unit directional vector collinear with the $z$-axis of the control coordinate system).

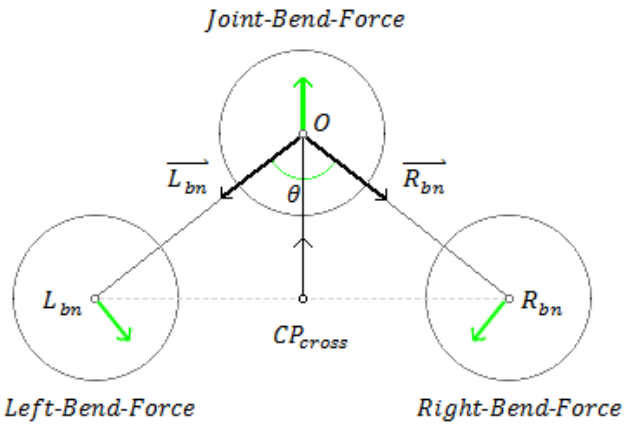



**Figure 5.5. Bend Control Metaphor Unique Force Pattern**

The force derivation diagram for the bend control metaphor. Three individual force directions are determined based on the set of artist provided parameters: bend angle, bend arm length, and the affect radius of each selection sphere.

The direction of the joint bend force is based on the closest point on the line that intersects $L_{bn}$ and $R_{bn}$. This point is noted in Figure 4 as $CP_{cross}$. Then the directional vector defined in the direction $CP_{cross} - O$ is the direction of the joint force. This force opposes the left and right bend forces because it must be provide the focal pivot for the bend to occur.

The visual representation of the control widget that is used to control the configuration of the bending control metaphor closely follows the geometric definition provided in Figure 5.6. Each of the effect spheres and bend arms can be selected and modified to alter the resulting behavior of this metaphor. The bend arms are represented by cylindrical volumes so that they can be easily selected and modified. This will change the bend angle and the length of the bend arms. If one of the effect spheres is selected, its radius can be modified to adjust the set of selected nodes.

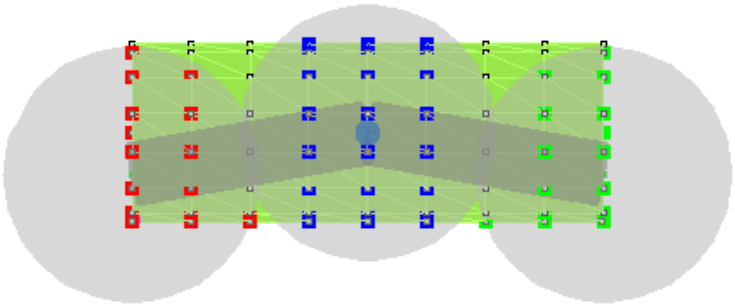


**Figure 5.6. Bend Metaphor Control Widget and Effected Node Sets**
The nodes of the selected deformable object that will receive external forces from the bend control metaphor. The three sets of nodes that have been selected are highlighted (left:red, joint:blue, and right:green). The set of nodes highlighted in red will receive the left bend force, the nodes highlighted in green will receive the right bend force, and the joint force will be applied to the set of nodes highlighted in blue.

After the external forces of the bend control metaphor have been derived it is common that the left and right forces will affect a larger number of nodes than are contained within the joint set. This will result in an imbalance of the applied external forces. This will lead to the object drifting as the bend motion is imposed. We counteract this problem by normalizing the magnitude sums of the sets. This is achieved by evenly distributing the excess force from one or more of the bend sets. This minimizes the net force on the object and prevents unintended drifting.

**Twist Control Metaphor Implementation**

As a simple example, the twist control metaphor defines a set of forces that rotate half of an object around a rotation axis in one direction and then rotates the other half of the object around the same axis but in the opposite direction. The separation of these two halves is performed at the origin of the control coordinate system and is identified as the separation plane. For the definition of the twist metaphor we derive a simple set of vector operations that inherently provide the correct orientations of the forces that produce the intended motion of this control metaphor.

Considering the $x$-axis of the control coordinate system, we make it the constant axis of ration for this metaphor. What must be defined is the orientations of the forces that will that rotate the nodes about this rotation axis. To achieve this we consider the points that are selected with a given effect radius and some finite interval on the $x$-axis. This selection process effectively replicates the approach defined for the stretch control metaphor. Once the two sets of affected nodes have been identified on each side of the splitting plane (the splitting plane is mathematically the $y$-$z$ plane), we iterate through each node to the left of the splitting plane and project it onto the $x$-axis of the control coordinate system. From this we define the following directional vector $\vec{p} = (n - n_{projected})$ where $n$ is the current node, $n_{projeted}$ is the projection of that node onto the $x$-axis and $\vec{p}$ points in the direction of the node from the $x$-axis. We can determine the external force that will rotate this point about the $x$-axis by simply using the cross product between $\vec{p}$ and the axis of rotation. The resulting force direction is depicted by the cross sectional view of the axis of rotation in Figure 5.7.

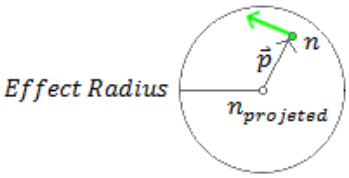



**Figure 5.7. Unique Force Pattern for the Twist Control Metaphor**

A cross-sectional view of the axis of rotation used to determine the force direction for the twist metaphor. The node $n$ and its projection $n_{project}$ onto the $x$-axis provides an orthogonal directional vector $\vec{p}$. The direction of the force, highlighted in green, is then defined as $\vec{p} \times \hat{x}$.

To define the directions of the forces that must be provided to the remaining set of selected nodes (those to the right of the splitting plane), we utilize the same method and then inverse the orientation of the resulting force vectors. This provides two sets of external forces that will force the influenced nodes to rotate about the $x$-aixs in opposite directions. This defines the basic requirement for a twist motion and will provide rotational forces as the object twists over time.

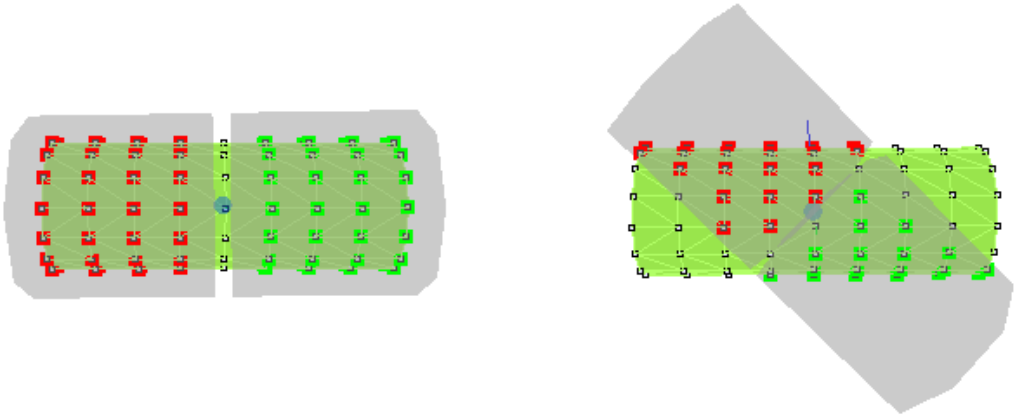


**Figure 5.8. Twist Control Widget with Modified Orientation**

The twist control widget (left) that the artist can interact with closely resembles the control widget of the stretch metaphor. This is because the configuration parameters are similar. The splitting plane is represented by the gap between the two selections. The nodes highlighted in red will be rotated clockwise while the green highlighted nodes will be rotated counter-clockwise. The image (right) illustrates a different selection pattern with the same widget.

Since the parameters that are used to determine the selected nodes (the orthogonal distance to $x$-axis, and a selection interval), the visual representation of the twist metaphor is similar to that provided for the stretch control metaphor. The image in Figure 5.8 shows the control widget for the twist metaphor with two different configurations and sets of affected nodes.

This set of control metaphors that provide primitive motions may initially seem limited in their utility for general animations; however with the ability to combine these metaphors on a single deformable-body, interactions between these forces can lead to complex behaviors. These control metaphors just provide the building blocks upon which a more extensive set can be built. The generalization of this approach allows for any number of additional control metaphors to be implemented and used with this existing set. With the controllable parameters provided for these metaphors and the implementation of control widgets, an artist can interactively modify the motions that these metaphors generate and fine tune them to provide refined object behaviors. The definition of these control metaphors provide a base upon which we can develop a set of intuitive controls that will allow us to effectively control the localized behaviors of deformable objects in existing animations.

# CHAPTER VI

## DYNAMIC RECORDING OF PHYSICALLY-BASED ANIMATIONS

The fundamental purpose behind utilizing a physically-based system to generate an animation is simple: manually recreating the physical interactions between moving objects is incredibly difficult and requires a keen perception of natural behaviors that result from colliding objects. Utilizing this approach to derive physically plausible interactions between deformable-bodies is practically intangible from this perspective. To circumvent this problem we provide a physics simulation that drives the movement of deformable-bodies from the standard equations of motion, appropriately resolves collisions and then records the states of each simulated object. For the foundation of this approach we introduce a dynamic simulation recording technique that allows for real-time animations to be generated as the time-step of the simulation is incremented. This technique stores the exact state of the physically-based system for each update, thus we can accurately reproduce the behaviors illustrated by the simulated objects during the recorded time-steps.

The foundation of the approach to controlling deformable objects is based on the modification of an existing physically-based animation. This initial animation includes deformable-bodies undergoing continuous deformations due to the interactions between these bodies and other static objects that exist in the physically simulated environment. The existing animation dictates how the objects deform over time and are typically subject to a default external gravitational force. Since this approach to deformable object control is based off of an existing animation, we provide a means to effectively generate an animation from a loaded set of deformable objects. This allows for this approach to

create and modify animations and is critical to the iterative process that artists use to refine their artistic vision.

Given an initial set of simulation parameters, we store the initial rest positions of the deformable objects and define this as the rest state, or beginning of our animation. Upon initializing the simulation we begin recording the state of all objects and store this information in an animation frame. This process is continued for each step in the simulation, effectively producing a smooth animation of the resulting object movements and interactions. We utilize this collection of simulation states as the main input to this deformable-body motion control technique. This allows us to record and modify the motion and behavior of the deformable-bodies within our physically-based simulation.

The basis of this work is directed at the modifications of an existing animation which will introduce new motions and localized deformations based on a set of high-level motion controls. In this chapter we introduce the simulation recording technique that provides the foundation of the the animation editing framework and defines the techniques that are used to provide real-time feedback from our high-level motion controls. In the next section we define how the simulated object states are compiled into animation frame data that will be used to reproduce the state of the physical system during an animation.

**Real-Time Simulation Recording**

The process of recording a physical simulation is based on the discretization of both an animation into distinct animation frames and physical simulation into fixed-interval time-steps. Based on the parallelism of the discrete representations, we define a one-to-one mapping between a given simulation time-step and its corresponding

animation frame. From this mapping we define a simulation recording as a set of animation frames that define the unique state of the physical simulation for each recorded time-step. This provides a complete definition of all object states and their resulting interactions of some finite duration of time. This provides the basic definition of a simulation recording that we develop to be used to provide real-time feedback in the deformable-body motion editing process.

The development of this approach to recording a physically-based system is characterized by the real-time constraint of the proposed deformable-body editing technique. The proposed technique allows an artist to interactively modify the resulting behavior of an object within a simulation. Specifically to maintain an acceptable level of interaction we assert that a recording should ideally be generated at a rate of at least $60[Hz]$ (or at least 60 frames per-second). Therefore we provide a recording architecture that will efficiently compile and store the physical state of all simulated objects for each recorded time-step. This is applied to the mass-spring system that we utilize to drive the deformable behavior of our animations.

Let the physical simulation $S$ represent a simulated system with $m$ mass-spring meshes. Each mass-spring mesh contains $n$ nodes that are updated every simulation time-step ($dt$). Let the ordered set of recorded animation frames $R$ contain $r$ recorded simulation states that correspond to any $r$ sequential time-steps of the simulation. The recording notation $R_i$ $(i \geq 0)$ represents the first recorded frame at time-step $i$ in the simulation. Similarly the notation $R_f$ $(f > i)$ represents the last recorded frame at time-step $f$. For each time-step between $i$ and $f$, a recorded frame $R_j$ is populated with the node data (position $p$, velocity $v$, acceleration $a$, and external force $e$) for each mass-

spring mesh. We note the process of compiling the node data for each mesh to a single animation frame $R_j$ as a $recordFrame$ function.

```
while simulating S {
     i = current simulation time-step
     if ( i ≥ R_i and i ≤ R_f ) recordFrame(i)->R_j
     increment simulation time-step
}
```

**Figure 6.1. Dynamic Recording Algorithm for Physical Simulations**

Overview of the dynamic recording algorithm used to capture the state of the physical simulation over time. The state of every object included within the simulation is recorded and stored into an animation frame. At a technical level we introduce an auto reference array that provides a large portion of pre-allocated memory to reduce the frequency of dynamic memory allocations. This allows us to efficiently record a physical simulation in real-time while maintaining a consistent level of interaction.

The result of this recording process is the generation of $r$ recorded frames within the recording $R$. Each frame $R_j$ contains the states of all mass-spring meshes within the simulation, therefore providing the playback of the recording is a straight-forward process. The state of each mesh is loaded from the recorded frame state and then loaded into the physics engine driving the simulation. This process is then simply repeated at the desired frame rate to reproduce the original simulation.

The ability to selectively load a simulation state provides additional functionality that is critical to process of analyzing the motion of simulated deformable objects. Specifically we look at how the simulation can be reversed or analyzed on a per time-step basis to interpret how the provided motion can be modified to match our intended motion or desired deformation.

Typically the animation playback speed, or the number of frames presented per-second, of the recording does not allow for precise adjustments of the resulting motion. Therefore this approach allows for the recording to be stepped on a per-frame basis, both

forward and backward in time. This provides the artist with the ability to closely analyze the state of all objects at any given time-step. Furthermore, this provides the highest possible resolution for introducing motion edits. Since each time-step of the simulation is completely recorded, edits that modify global trajectories and local deformations can be introduced at any time-step. This provides a high level of control over when an object can be modified in an animation while allowing previously recorded frames to be unmodified. In the next section we present the transition between loading the object states from a recorded frame and returning control to the physics engine to generate new object states that can be recorded.

**Recording Invalidation**

The recording of a simulation has a finite length of $r$ frames where each contains the node states of all mass-spring meshes in the simulation. When the recording is played at some constant frame-rate to create an animation, the state of each mesh is loaded from the current frame into the physics engine. This is simply a process of loading the position, velocity, acceleration, and external force for all nodes of each mesh and using that data to produce the next recorded state. However when the recording playback is halted due to the last frame being loaded, the generation of this node data must be returned to the physics simulation to generate the next time-step. This defines the point at which the recording playback has become invalid.

When a recording playback has become invalid it represents the point at which the physics simulation must regain control of the simulation to produce the state of the objects at the next time-step. This naturally occurs when the end of the recording has been reached. At this point we introduce the concept of automated recording. When the

end of the existing recording has been reached, the playback can be halted or the new state generated by the physical simulation can be compiled into a new animation frame and appended to the existing recording. This effectively allows future motion modifications to be automatically stored in the existing recording. This provides a seamless interface for introducing new motions and automatically recording the result. Similarly we must also consider when the motion of an object in the middle of a recording must be modified.

When the motion within a recording must be modified, the recording must be invalidated at the frame where the motion edit is made. Therefore when the state of the simulation is updated it will be generated by the physics simulation rather than being loaded from the recording. At this point, all recorded frames past the time-step at which the edit is made will be overwritten. This new frame data that contains the modified object motion state will replace the existing invalid frames. Therefore when the animation is replayed from the beginning, the motion will transition at the invalidation point and illustrate the newly introduced motion. We allow the artist to perform this operation as many times as they need to reach their target motions of all objects within the simulation. We utilize this feedback extensively while modifying and creating new deformations of the simulated objects. This introduces an iterative process that provides the artist with an effective feedback loop that allows the resulting animation to be refined until the artistic intent has been reached.

# CHAPTER VII

## CURVE-BASED FORCE MAGNITUDE AND DURATION

Through the abstract definition of a control metaphor that defines the pattern of external forces, their orientation, and the sets of affected nodes required to impose a motion on a deformable body, we are provided with a set of unit force vector that describe the physical implementation of the defined motion. This generalized pattern of forces and orientations however do not address all of the parameters that are required to implement this motion in a physical simulation. Two additional parameters must be defined to fully specify the influence of a control metaphor on a deformable-body. Specifically we must provide an effective mans to define the magnitude of the provided external forces and the time-steps within the simulation they are applied. Both of these parameters have the ability to drastically alter the resulting motion of the controlled deformable object and therefore provide an unlimited number of alternative animations that can be produced.

To provide an artist with precise control over these additional parameters we introduce an intuitive curve-based control. The definition of this control provides an effective means to defining both the magnitude and the duration of the external forces supplied by a control metaphor. This is done through a simplistic and interactive interface that allows for immediate feedback based on the provided curve settings. An artist can simply define the characteristics of the curve that will be interpreted as the simulation progresses to apply the external forces of the selected control metaphor with the provided magnitude, at the appropriate time, for the requested duration.

For this approach we define the curve in two dimensions and provide the following semantic meaning to the curve: the $x$-coordinate of the curve represents the current time-step in the simulation. This starts from 0 and progresses to the end, or past the end, of the existing animation. The $y$-coordinate represents a force magnitude scalar that should be provided to the selected control metaphor. Together this curve represents the magnitude of the external forces over time. Therefore with the definition of this curve we can define the two additional parameters that we required to complete the definition of the motion we would like to apply to the selected deformable object. The image in Figure 7.1 illustrates the basic concept of this newly introduced force curve $C$. The position of the first end point $C_{begin}$ represents the simulation time-step at which the external force magnitude will be greater than zero. The $y$-coordinate of the curve defines the magnitude of the external force at the provided time-step. When the second end point of the force curve $C_{end}$ is reached, the external force magnitude will be assigned to zero, thus eliminating the contribution of the selected control metaphor from the resulting motion of the controlled deformable body.

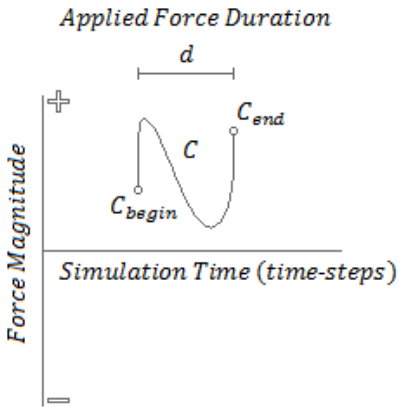



**Figure 7.1. Conceptual Force-Curve Editor**
The diagram in this image provides the conceptual outline of the force curve editor.

To provide an intuitive interface for controlling the duration and magnitude of the applied external forces, we must be able to define several characteristics of the force curve. Specifically the most critical aspect that must be addressed is the ability to modify the end points of the curve. Since the $x$-coordinate of the curves coordinate system represents the time-steps of the simulation, we must be able to define when the external forces are applied to the selected object and when they will be removed or scaled to zero.

These events are represented by the end points of the discrete force curve. Therefore in the implementation we simply allow the artist to select and move the end points to the desired simulation time-steps. This additionally provides an implicit definition of the duration of the applied force. Since the begin and end times of the application of the external forces are supplied, the number of time-steps the external forces will be applied is $C_{end} - C_{begin}$. This is important information that must also be tied to a unit that the artist is familiar with.

Since we note that the simulation time-step is incremented by one-sixth of a second each time-step, sixty time-steps with represent one second in the simulation. Therefore if the artist wishes to apply the external force for exactly one second they can simply define $C_{begin}$ and $C_{end}$ on any interval where $C_{end} - C_{begin} = 60$. This provides an intuitive connection between when and how long the forces should be applied and how this information is translated into a discrete set of simulation time-steps.

The second critical characteristic of the force curve that we must define is the form of the curve itself. To achieve a flexible and simplistic control of the curves form, we introduce the definition of a Bezier curve. This simple definition of a curve based on two end points and two control points provides the simplicity required for our intended

purpose. Referencing the requirements of our curve outlined in this section, we note that the only difference with the Bezier curve is the introduction of two control points. These additional points provide a flexible and intuitive means to defining the form of the curve. The resulting form of the curve defined by the Bezier control points provides a smoothly interpolated set of discrete points that can be extracted to define the magnitude of the external forces.

The application of a Bezier curve to this function provides a perfect mapping between force duration, force magnitude and the discrete time-steps of the simulation with only slight modifications. The first required modification is that the end points of the Bezier curve must snap to valid time-steps within the simulation. This effect is achieved with simple grid snapping and is provided with the implementation. The second requirement is that the number of points used to approximate the curve must match the number of time-steps between the beginning and end of the curve. This is simply because for each time-step in the simulation that will be provided a magnitude of an external force must have a corresponding point within the force curve. To meet this requirement, the definition of the Bezier curve and the number of points used to generate the discrete approximation of the curve is updated with every user-imposed modification. With these two requirements met, a simple and easy to modify Bezier-force curve editor can be implemented. In the implementation of this curve editor, the end points and control points can be easily selected and moved through the provided plot. The Bezier curve will automatically be generated based on the movements of these points. The image in Figure 7.2 shows the implementation of the force curve editor provided for our animation editing environment.

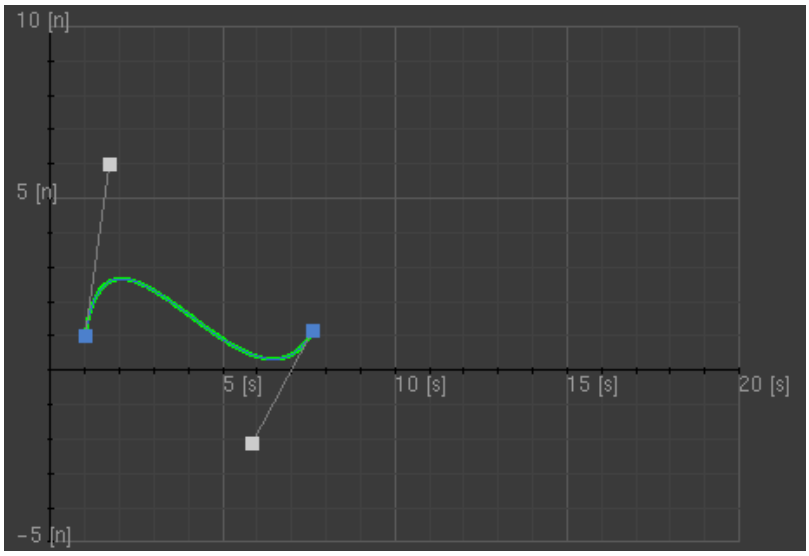



**Figure 7.2. Beizer Force-Curve Editor Implementation**

The image shows the implementation of the Beizer-based force curve editor that we provide to define the magnitude and duration of the external forces introduced by the applied control metaphor. The white points represent the control points of the Bezier curve, the blue points represent the start and end times of when the force is applied, and the green line shows the discrete approximation of the curve. From this we can see that there is a maximum external force magnitude of approximately 2.7[n] and the force is applied for approximately 7.5[s].

Utilizing this flexible definition of a force curve we can also extract more complex behaviors from a provided control metaphor. The $y$-coordinate of the force curve is not limited to positive force magnitudes. Simply defining the form of the curve to pass below the $y$-axis in the curve editor will specify a negative force magnitude for the associated time-steps. This introduces an additional level of flexibility to the motions that can be described with a control metaphor. Essentially the force directions provided by the control metaphor can be inversed by simply providing a negative force magnitude with the curve editor. The application of this principle that we illustrate with a control metaphor from the set we define is the stretch metaphor. When the provided force magnitude is positive the direction of the forces that are used to stretch the object will point in opposite directions. However if we utilize the curve editor to specify a negative

force magnitude, we will then see that the resulting external forces will be inverted. In the case of the stretch metaphor, this introduces another type of desirable deformation: compression. Since the stretch forces have been inversed, they will point towards each other, effectively defining the abstract motion metaphor required to compress an object.

# CHAPTER VIII

## PHYSICAL SIMULATION DYNAMIC PREVIEW GENERATION

Appropriately conveying the intent of an animator to a simulated system is a complex challenge that requires several methods of interaction between the animator, the graphical user interface, and the physical system driving the simulation. The challenge of properly addressing the interactions between the animator and these systems is that there is no distinct interface that defines how the intent of the animator is translated to user interface commands. Similarly, the interface that must exist between the graphical user interface and the physics system used to drive the simulation is difficult to define.

In order to fully illustrate the changes incurred by adding external forces to the manipulated object, an additional visualization must be introduced. Considering that the external forces introduced by the force curve will modify both local deformations and the global trajectory of the object, this visualization must effectively convey both of these changes over time. Utilizing the force curve to introduce external forces, the position in time that the simulation will be altered is known. At this point, whether at the beginning of a simulation or at the end of a recorded animation, we can generate the future states of all simulated objects. By recording and illustrating selected future states of all simulation objects we provide the artist with an exact outcome based on their introduced modifications. These illustrations provide a conceptual bridge between modifications to the force curve and the desired outcome of the simulation. This proposed visualization allows for the completion of a feed-back loop that allows the artist make appropriate changes based on the prior result. Thus the animator can effectively convey their original intent through the modification of the simulated result.

The process that we utilize to generate the dynamic preview of a physical simulation heavily relies on the real-time recording technique covered Chapter 6. Generating a preview of the simulation is a straight-forward process with our recording technique. We simply progress the simulation by $n$ time-steps and record the frame data. Since the generalized motion of the deformable objects in the simulation can be visualized from a reduced set of states between multiple frames, we introduce a modular selection key $m$. Therefore when the preview is generated for $n$ simualtion time-steps, we only record every $m^{th}$ time-step to an animation frame. Using this technique we approximate the benefits that dynamic keyframes provide the artist.

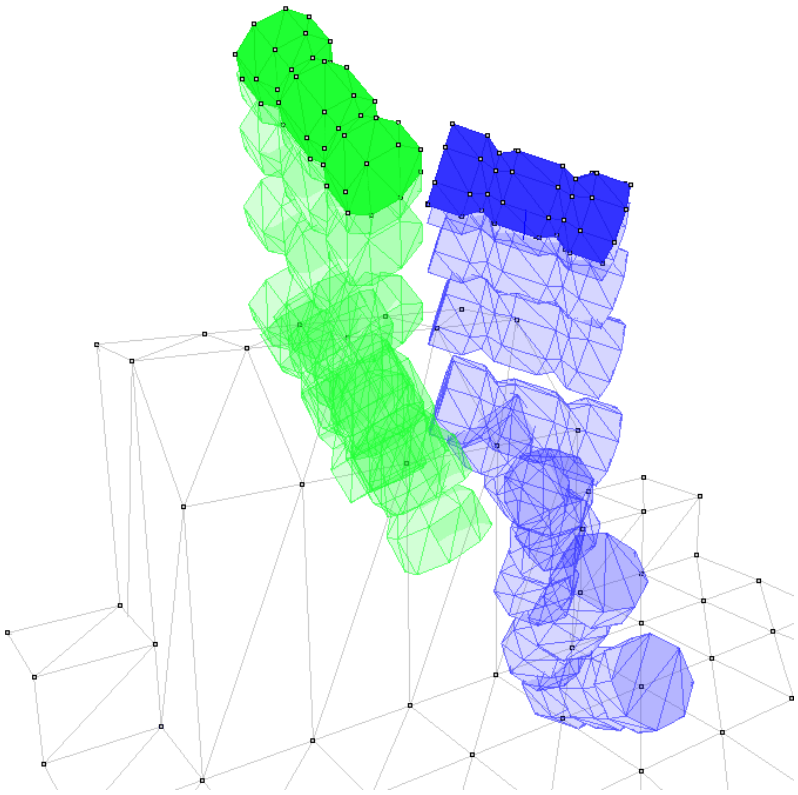


**Figure 8.1. Dynamic Simulation Preview Generated for 200 time-steps with $m = 20$**
This image shows the future states of the two deformable objects as they bounce off of the statically defined environment. The original objects are shown at the top of the image and are fully shaded. The future states of these objects are shown through the use of transparent future states. If the simulation is played from the rest position, the objects will follow the exact global trajectory and local deformations illustrated by this preview.

Since the artist has some intended motion in mind, they can tweak the control metaphor parameters to generate some motion that is close to what they ideally want. At this point they can use the simulation preview generator to view the next set of states that the objects will pass through. This provides a critical feedback mechanism that allows the artist evaluate the next motions that will be generated in the simulation without corrupting the currently recorded animation. The image in Figure 8.1 illustrates a direct application of the simulation preview generated for two deformable bodies interacting with a static environment and clearly shows their global trajectories. The image in Figure 8.2 shows the generation of a preview that contains a cloth model that undergoes a localized deformation provided by a bend control metaphor.

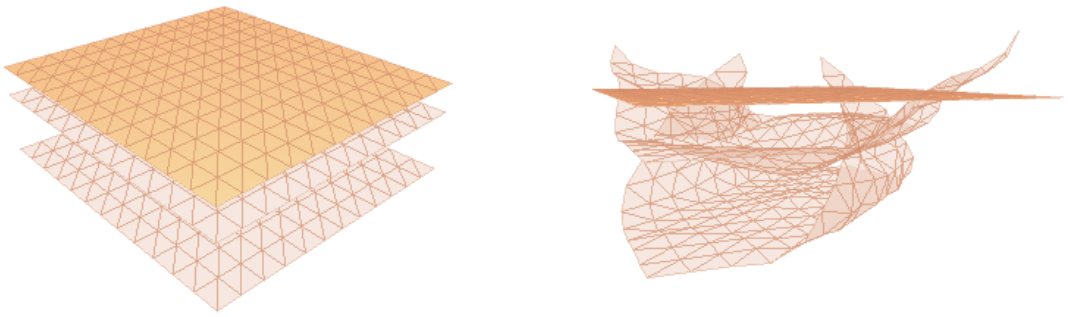


**Figure 8.2. Dynamic Preview of a Highly-Deformable Object**
The generated preview shows the original unmodified simulation of a cloth model falling (left). A bend metaphor is then applied to the cloth and the local deformation of the cloth model is displayed using the dynamic preview (right).

This provides the artist with an immense amount of feedback related to how the objects will interact with each other during the simulation, their global trajectories, and their deformations. These components all represent critical aspects of a physical simulation that define the overall behavior of the resulting animation. We provide this as part of the intuitive interface that aims to allow for the effective modification of existing

animations. This preview can be iteratively generated to focus on the current deformation states of the simulated objects. This technique allows the artist to easily see the effects of a modification imposed by a control metaphor immediately and provides additional information that can be used to tweak the result of the overall animation.

# CHAPTER IX

## INTUITIVE CONTROLS FOR EDITING ANIMATED SIMULATIONS

The highly interactive approach that we provide to dynamically modify existing physically-based animations requires an intuitive interface that provides easy access to these controls exposes the flexibility of this approach to the artist. Providing an effective interface that incorporates all of the editing and simulation features we have considered, is inherently complex. This complexity is mitigated by providing an ample amount of feedback to the artist and effectively displaying the current state of the animation.

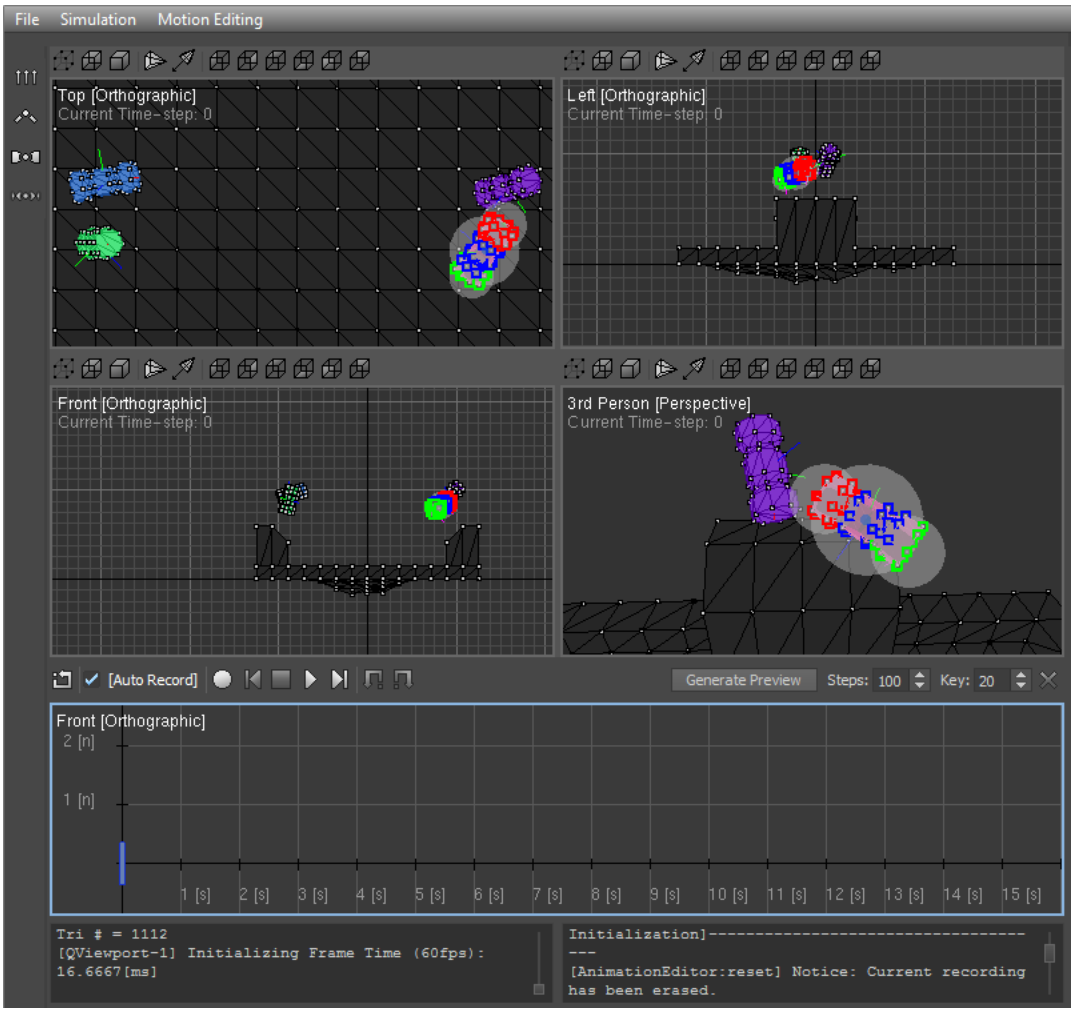



**Figure 9.1. Real-time Deformable Animation Studio**
This image shows the implementation of the real-time animation editor.

Since the main functionality of this approach revolves around the current state of the simulation and existing animation, the interface is designed around providing as much information about these systems as possible. The main view of this interface is dedicated to display the state of the physical simulation and the playback and recording of the animation. The most significant feature that the editor provides is the ability to view the current state of the simulation from four views simultaneously. These views are shown in Figure 9.1 as the viewports that contain the objects of the simulation. Along the bottom of the main editor interface the animation controls are implemented through the common media controls and the current time-step of the simulation is represented in the interactive time-line. As the simulation progresses, it can be recorded and the blue handle within the time-line will automatically display the current simulation time-step. Objects within the simulation can easily be selected and once a selection has been made, the toolbar in the upper left-hand corner can add control metaphors to the object. When an object within the simulation is selected, the different control metaphors that have been assigned to it can be configured through the use of the of the interactive curve editor.

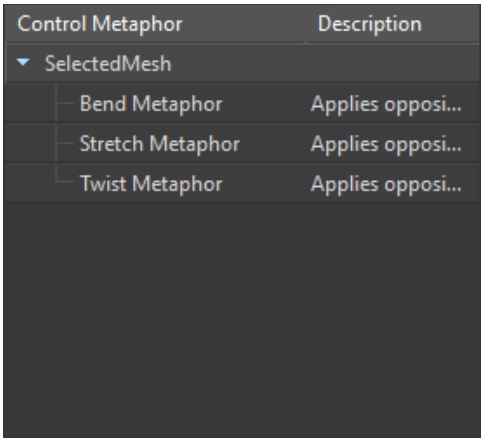



**Figure 9.2. List of Active Control Metaphors Acting upon the Selected Mesh**

The displayed list shows the current control metaphors that will modify the behavior of the selected object. In this instance there are three active control metaphors that will apply external forces to the selected mesh.

The artist must be able to explore and view the simulation both while it is halted and while the simulation is executing or being recorded. To achieve this we provide a set of four viewports that can be configured by the artist to provide the best perspective of the motion of the objects within the simulation. The ability to accurately view and judge the current motion of an object within an animation is a direct prerequisite of modifying that behavior to reach some intended new behavior. Therefore the viewports that are designed into the editor provide an accurate display of the simulations current state and can be used to closely view the deformations of the included objects. Figure 9.3 provides an example of the 3rd person view that is provided within a configurable viewport.

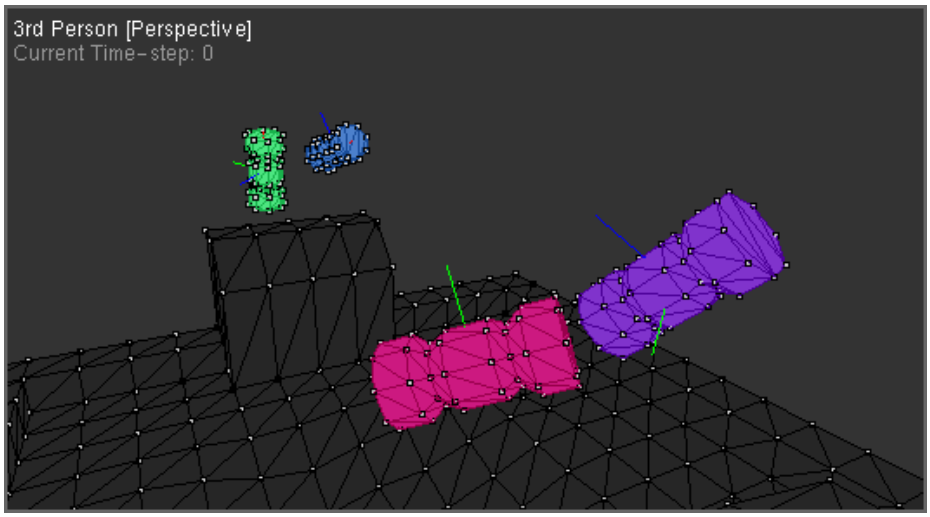



**Figure 9.3. Flexible Simulation Navigation**
The image provides a view of the simulation displayed by one of the four configurable viewports. The flexible controls that are used to allow the artist to navigate the simulation are critical to the animation editing process. The more visual information that the artist has about the state of the simulation and the recording, the better they will be able to address problems with the resulting object motions.

Since this approach focuses on the modification of an existing animation, the controls that are provided to the artist to edit and view the animation are also critical to the development of this approach. To provide the highest level of flexibility to view the

current animation, several useful controls have been developed and implemented into the interface of the provided editor.

Specifically the ability to record and playback the animation of the simulated objects represents the bare minimum requirement to achieve this proposed method, however we also provide several additional controls that can help the artist efficiently generate their target animation. The recording time-line can selected by the handle (shown in blue in Figure 9.4) and based on what simulation time-step the artist drags this handle to, the recorded state of the simulation at that time-step will be automatically loaded into the scene displayed by the set of viewports. Figure 9.4 provides a simulated look at the interactive simulation time-line control (as the user would drag the handle).

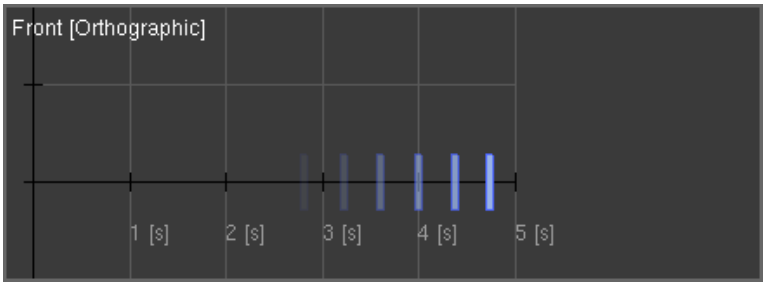



**Figure 9.4. Interactive Animation Time-line**
The time-line control handle progresses with the current time-step of the simulation. The image illustrates the behavior of the control handle as the simulation is updated. The artist has the ability to move this handle to any desired time-step to see that state of the simulation.

This allows the artist to effectively step through the animation at any speed based on where they drag this handle of the simulation time-line. This includes playing the simulation by individual steps and playing the recording in reverse. The image in Figure 9.5 shows the implementation of the media-player based simulation play-back controls that are provided in the main interface of the application. These additional controls in

combination with the interactive slider provide an extensive amount of visual information that can be used to analyze deformable object motion over time.

Therefore the interface that has been designed in cooperation with this proposed approach to modify the localized behaviors of physically simulated objects in existing animations provides an extensive amount of control to the artist to achieve this goal. The examples of the deformable objects presented throughout the description of this approach have been generated utilizing this interface. Therefore the utility of this application has been thoroughly demonstrated.


**Figure 9.5. Animation Media Controls**
These controls provide typical media-based simulation play-back, recording and individual frame stepping. The far (left) arrow provides a complete reset of the simulation and the far (right) arrow pair allows the artist to jump to the beginning or end of the recorded simulation. The remaining icons perform the standard operations implemented in most media-based players.

# CHAPTER X

## CONTROL METAPHOR IMPOSED MOTION VALIDATION

Objectively evaluating the effectiveness of an intuitive interactive interface is inherently challenging. With the proposed solution to deformable object control we provide an effective approach to modifying the behavior of deformable-bodies within a recorded simulation. The evaluation of the effectiveness of this solution is based on the ability to realize the desired motion of the simulated object in an animation modified with this approach.

Intuitively, the validation of the resulting motion introduced by the proposed control metaphors is easy to decipher by looking at the resulting behavior. Visually we can validate the effect of the control metaphors applied motion based on the deformation behavior of the modified object. When the effects of a control metaphor are considered independently, we can confirm that the result that this approach provides matches with the intended motion. This is due to the simplicity of the fundamental motions that are introduced by the provided control metaphors. We explore the primitive cases of motion that the control metaphors provide that can be effectively validated.

When considering the basic fundamental object motions we have introduced through the use of control metaphors, we can evaluate the effectiveness of each based on the pre and post modification deformations of the simulated object. For primitive objects these modifications can be evaluated by the post state deformation of the object and its correlation to the behavior defined within the control metaphor. With the simple geometry of the primitive object, we can validate the deformation as a function of the duration and magnitude of the external forces introduced. This chapter provides the

validation of the motions applied on simple deformable bodies from the set of implemented control metaphors. We also provide images produced from our physical simulation to show how the validation methods correspond to the resulting behavior illustrated by the simulation of our deformable objects.

**Objective Evaluation of the Twist Control Metaphor**

The defining characteristic of the twist control metaphor is the splitting plane that divides the geometric definition of the deformable body into two sets, the left twist set and the right twist set. The influence of the twist metaphor on these node sets will be provided by a set of external forces that rotate the sets in opposite directions about a common axis. The image in Figure 10.1 shows the shaded areas that are influenced by the twist forces, the axis of rotation (highlighted in red), and the splitting plane (highlighted in blue).

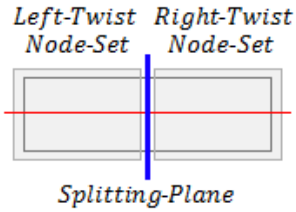



**Figure 10.1. Twist Control Node Sets**
The twist control metaphor defines two affected node sets: the left and right twist sets. From the cross-sectional view of a cylindrical object shown, we can identify the splitting plane (blue) of these two sets and the axis of rotation (red).

Given this pre-deformation state of the deformable-body, select two nodes $p$ and $q$ on opposite sides of the twist plane defined as the $x$-$y$ plane of the control coordinate system. Project $p$ and $q$ onto the $x$-axis of the control coordinate system of the control metaphor. Create the directional vectors $\vec{p} = (p - p_{projected})$ and $\vec{q} = (q - q_{projected})$ project them onto the splitting plane and then simply calculate the angle between them. This will represent the pre-deformation angle of the two vectors.

After the twist control metaphor has been applied for some duration $d \geq 1$ time-step with an external force magnitude scalar $s \geq 0.0$, we retrieve the current positions of the nodes $p$ and $q$ and note their new positions as $p'$ and $q'$. Again we derive two directional vectors from the projection of the points $p'$ and $q'$ onto the $x$-axis of the control coordinate system: $\overrightarrow{p'} = (p' - p'_{projected})$, $\overrightarrow{q'} = (q' - q'_{projected})$. The angle between these direction vectors defines the deformed relational angle between the two nodes in the deformable-body.

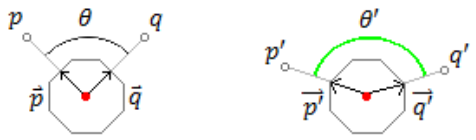



**Figure 10.2. Twist Control Metaphor Deformation Validation**
A rotated cross-sectional view of the cylindrical deformable body is illustrated before and after the twist metaphor has been applied. The image on the (left) illustrates the rest state of the deformable-body and the initial positions of the nodes $p$ and $q$ and the angle $\theta$ between them. The image on the (right) shows the state of the deformable object after it has been twisted. The updated node positions $p'$ and $q'$ are used to determine the new angle $\theta'$ between the two directional vectors $\vec{p}$ and $\vec{q}$ (highlighted in green) after the deformation.

Noting that as the deformation continues, this angle will increase, thus illustrating that the twisting motion is being imposed on the selected nodes. This process can be repeated for every unique pair of nodes within the deformable body split across the splitting plane of the twist control metaphor to conclude that this metaphor provides the intended behavior. The image in Figure 10.3 illustrates the state of the deformable body after approximately 40 time-steps of our recorded simulation.

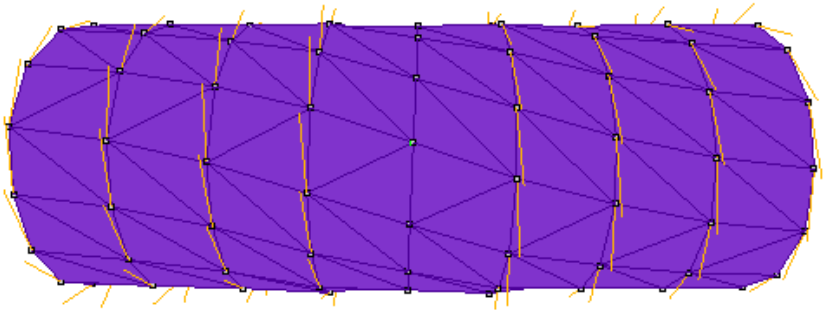


**Figure 10.3. Demonstration of Validated Twist Deformation**

The image shows the state of the deformable cylinder after the twist modifier has been applied. The external forces (highlighted in orange) show how the geometric definition of the object is deformed due to the rotation of the nodes on opposite sides of the splitting plane. This image has been enlarged to show the curvature of the geometry.

**Objective Evaluation of the Bend Control Metaphor**

The form of the pre-deformation state of a deformable body that will be bent using this metaphor is best illustrated with a cylindrical object. The principle remains the same for any deformable-body however this example is generally the easiest to depict visually. For the pre-deformation state of the bend metaphor we define three sets of nodes that correspond to the node sets that are affected by the bend control metaphor. The center of mass of each node set can easily be determined and defined as $lcom$, $rcom$, and $jcom$ for the left bend node set, right bend node set, and joint bend node set respectively. Figure 10.4 illustrates the spheres of influence that select the left, right and joint nodes for the bend metaphor on a cross-sectional view of the cylindrical object.

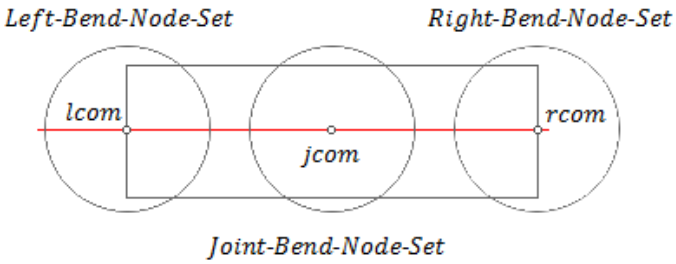



**Figure 10.4. Bend Control Metaphor Validation – Pre-Deformation**

Based on the pre-deformation state of the deformable cylinder example, the best-fit line between for $com$-points defines a splitting plane since the applied bend forces have opposite directions. The $x$-axis of the control coordinates is shown in red.

Considering the deformation of the cylindrical object after the bend control metaphor external forces have been applied, we again calculate the center of mass for each affected node set. The best-fit line of the post-deformation mass centers, $lcom'$, $rcom'$, and $jcom'$, again forms a splitting plane. To validate the behavior of the bend control metaphor the displacements of these points must increase as the simulation progress while the deformable-body is under the influence of the external forces provided by the bend metaphor. Figure 10.5 shows the deformation state of cylindrical object and the displacements from this splitting plane (highlighted in green).

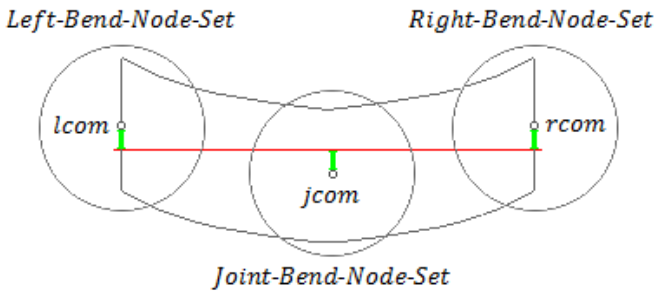



**Figure 10.5. Bend Control Metaphor Validation – Post-Deformation**
The post-deformation state of the object undergoing a bend modification illustrates that the displacements (green) from the splitting plane (red) increase as the external forces provided by the bend metaphor modifies the state of the deformable object. Thus a generalized behavior that matches the intent of the bend metaphor is achieved.

To validate this result visually we provide a simple demonstration of the bend control metaphor on a simple deformable cylinder. We select the ends of the cylinder to define the left and right bend node sets. The exact center of the cylinder is used as the joint position for the bend. The external forces are derived from these point sets and applied to the deformable object. The image in Figure 10.6 shows the state of the deformable-body after approximately 40 time-steps of our recorded simulation.

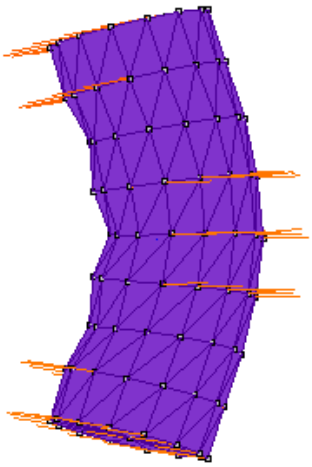


**Figure 10.6. Demonstration of the Validated Bend Control Metaphor**
The image shows the result of a simulated deformable-cylinder being bent along its major axis with this approach. The external forces (illustrated in yellow) show the direction of the forces imposed by the bend control metaphor. This behavior matches the intent in the design of the bend control metaphor.

**Objective Evaluation of the Stretch Control Metaphor**

The evaluation of the stretch control metaphor is a relatively straight-forward process. In the pre-deformation state the deformable-body will have an initial rest length defined on the $x$-coordinate of the control coordinate system as a closed interval. From this rest state we simply determine the length of this interval. After we apply the stretch control metaphor, the updated interval will be extended from the prior pre-deformation length. The illustration provide in Figure 10.7 provides a clear demonstration of how the deformable object will be stretched past its original rest length after the control metaphor has been applied.

A typical problem that is frequently encountered with applying this control metaphor is that generally the structural composition of the mass-spring system will resist the forces that try to stretch the object. Therefore it takes an arbitrarily large external force to properly stretch the object. In addition to this problem, if any torque is

introduced during the application of the stretch control metaphor it will leave an unintended secondary rotational motion.

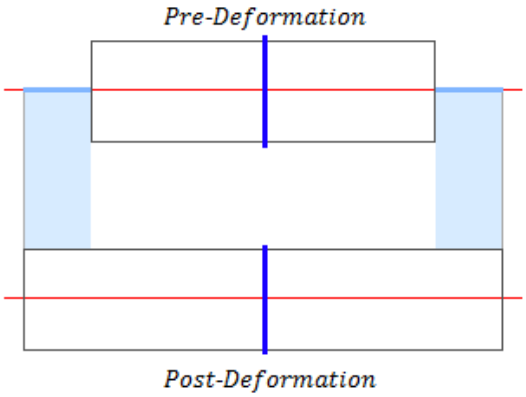



**Figure 10.7. Stretch Control Metaphor Validation**
This illustration shows both the pre- and post-deformation states of a cylindrical deformable-body. Based on the position of the splitting plane, the nodes on the left will receive a uniform external force to the left and the nodes on the right will receive a uniform external force in the right direction. This will stretch the object along the $x$-axis of the control coordinate system (shown in red).

The visual result of this stretch operation is fairly intuitive. The result of applying this control metaphor to an object within our recorded deformable simulation provides the result illustrated in Figure 10.7. It is evident that based on the visual inspection of the objects current state after the deformation, that the object has been stretched in the intended dimension. This is also illustrated in Figure 10.7 by the difference in the intervals of the objects defined length highlighted in light blue. This difference represents the total amount that the deformable object has been stretched in a single dimension. Here we note that some orientations of the stretch control metaphor may impose a torque on the simulated object. This is related to the asymmetry of this scenario and is not an adequate candidate for the application of this control metaphor.

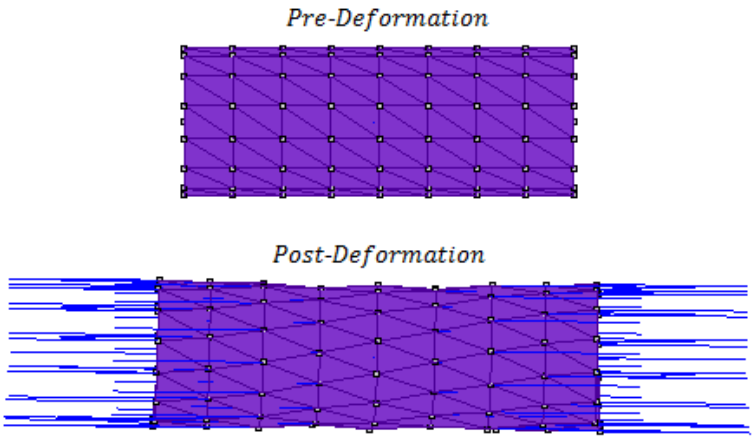



**Figure 10.8. Demonstration of the Validated Stretch Control Metaphor**

This image shows the result of the stretch control metaphor applied to a cylindrical deformable-body. The external forces that are provided from the stretch control metaphor are illustrated in blue for the post-deformation state.

Similarly, we utilize the same logic presented in this section to validate the resulting behavior of the compression control metaphor. The sets of nodes that are selected for the compression control metaphor are the same as those selected for the stretch behavior. The only modification that we must provide to define this behavior is a negative force magnitude. This will effectively inverse the direction of the external force.

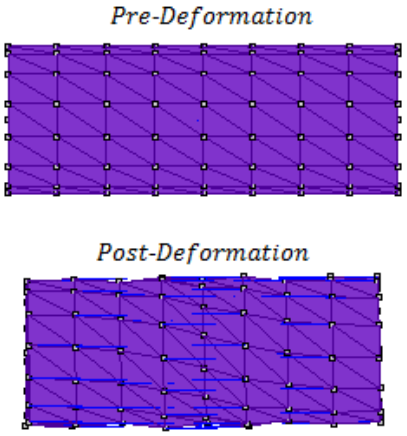



**Figure 10.9. Demonstration of the Stretch Metaphor utilized for Compression**

The image shows the pre- and post-deformation states of the cylindrical deformable-body through our recorded simulation. The applied metaphor is the stretch metaphor; however the magnitudes of the external forces have been defined in the curve editor as negative values. This essentially flips the direction of the stretch forces and creates the effect of compression.

Therefore we validate the motion imposed by this particular application of the stretch control metaphor. The directions of the external forces have been inverted by providing a negative force magnitude in our force curve editor. Therefore nodes of the deformable-body will be forced towards each other, achieving the desired compression.

**Evaluation of Localized Deformation Control**

The objective evaluation of the control metaphors provided by this approach illustrate that the intended motions can be applied to primitive deformable-bodies to achieve the correct result. The examples in the previous sections however are the result of global deformations of a simulated object. This approach is not limited to affecting only the global deformation of the simulated object. The unique control that this approach provides is the ability to target specific areas of a deformable-body and apply external forces provided by a control metaphor at that position to create a localized deformation. This represents an important part of this approach and is clearly demonstrated through the application of the provided set of control metaphors. The validity of this approach is demonstrated through a set of concrete examples that illustrate this robust and effective solution to introducing localized deformations using control metaphors.

The introduction of a control coordinate system within the local coordinate system of the simulated deformable object allows us to define both the position and orientation that the control metaphor will impose on the local geometry. This approach provides the functionality to select an existing node of the deformable-body to define the origin of the control coordinate system. The control metaphor that is applied at this position will affect the localized geometry. This illustrates the key concept behind this approach of introducing localized deformations to exiting animations.

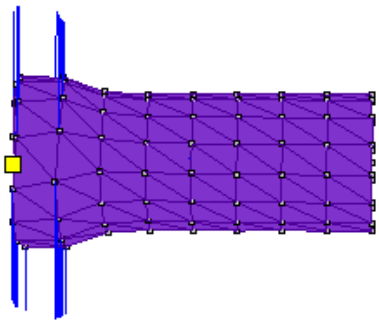


**Figure 10.10. Localized Stretch Deformation**

A stretch metaphor is applied to the local region shown by the selected yellow node. This illustrates the effectiveness of this approach to impose localized deformations on a deformable object.

This provides an additional level of control that most previously researched methodologies for deformable object control do not directly account for. This provides the artist with a much more powerful level of customization that can be used to modify existing animations. Effectively this allows us to introduce localized motion edits for existing animations. We demonstrate the flexibility of this approach by deforming any type of simulated object based on a localized position and orientation of the applied control metaphor. The image in Figure 10.11 illustrates the bend control metaphor and the localized deformation imposed by the external forces the metaphor introduces.

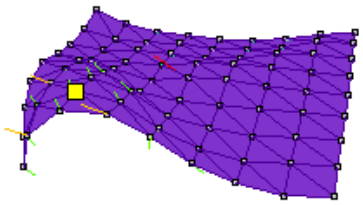


**Figure 10.11. Localized Bend Deformation**

Local deformation of a cloth model based on the application of the bend control metaphor. The yellow point represents the position of the control metaphor as it is applied to the object. It can be seen that the corner of the cloth model has been bent in relation to the position of the selected node. This represents an example where the intended motion was achieved using a control metaphor. This validates the assumption behind the proposed approach. This illustrates that control metaphors can be used to introduce targeted deformations to existing animations.

The flexibility provided by this set of targetable control metaphors illustrates that a large number of different behaviors can be imposed on a single deformable model. Using the same deformable cloth model as show in Figure 10.11, we illustrate that a local twist deformation can be applied to the same model and provide a completely different localized deformation. This allows for a varied set of animations that can be produced and modified using the same artistic assets. This means that the artist can edit and produce animations more efficiently. This efficiency however does not limit the control that the artist has over the physical simulation, but rather provides explicit control over all nodes within the simulated deformable-body. This is critical to expanding the amount of control the artist has when trying to modify an existing physically-based animation. The image in Figure 10.12 illustrates reuse of the cloth deformable model asset to produce a behavior that is completely different than that imposed by the bend metaphor shown in Figure 10.11. This alternatively imposed behavior illustrates that the provided set of control metaphors effectively match the intended behavior for various geometric definitions.

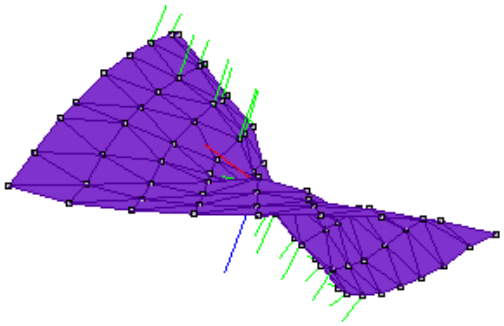


**Figure 10.12. Localized Twist Deformation**

The application of the twist control metaphor at the edge of the deformable cloth model. The external forces, highlighted in green, clearly show the imposed motion on the nodes within the influence range of the applied metaphor. The other half of the cloth remains mostly unmodified. This represents an effectively controlled local deformation.

Therefore we have illustrated that this approach provides a flexible and targetable local deformation control mechanism. The ability to modify existing simulations by controlling the movement of all regions of a deformable-body allows an artist to effectively express their intended motions to the physical simulation. This allows the artist to tailor the result of the physical simulation to adhere to the motion they want to visually depict. We provide this level of control through the implementation of the intuitive interface that allows an artist to dynamically modify the location, orientation, and parameters of the applied control metaphors. The resulting application of this approach through this interface defines a new level of control for the localized deformations of simulated objects.

# CHAPTER XI

# IMPLEMENTATION

The implementation of a comprehensive editing environment that facilitates the simulation of deformable objects and the interactive recording of physically-based animations requires an extensive number of components to create these described functionalities. In this chapter we look at the components that are required to develop a modular physics engine that supports real-time recording for physically-based animations. The implementation of this approach provides a fully functional animation generation and editing environment that is highly interactive and can be used to create complex animations of multi-object simulations. To provide all of the required functions of the proposed approach, several modular libraries have been created and incorporated into a centralized simulation framework. The result of this work culminates in the creation of a scene-based simulation application that allows the user to interactively view and generate animations through an intuitive graphical interface.

## Architectural Overview

The development of a real-time simulation library requires several modules that define the implementation of each component required to produce a physical simulation. In this section we introduce the main modules of the library developed to facilitate the interactive modification of deformable objects in physically-based animations. Just from this set of requirements we can identify several modular components that should be included in the definition of the architecture that will be used to create the interactive editing application. Based on the requirement of a physical simulation we can define three individual modules immediately: mathematics, physics, and simulation. Since the

simulation is a physical model driven by mathematics, we can easily visualize the dependencies of these three modules as shown in Figure 11.1.

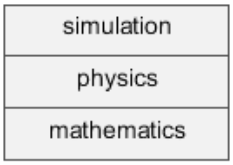



**Figure 11.1. Basic Simulation Architecture Overview**
Based simply on the requirements of the animation editing application, we can immediately define three of the modules that are required for the animation editing application.

These components provide the bare minimum requirement for developing an organized physical simulation package. Utilizing this definition, the requirements of physically-based animations are considered. These animations are produced by recording the states of the physical objects within the simulation; this process is outlined in detail in Chapter 6. With the additional components for recording and animation, the last module that defines how the artist will interact with all of these components is defined as the interface module. The interface module coordinates the input from the artist and dispatches the command to the appropriate sub-module. This interface module is abstract and is not tied to any external extension, therefore the graphical interface library (Qt), and graphics libraries (DirectX, OpenGL), can be replaced depending on the requirements of the application. The image in Figure 11.2 is an illustration of the modular architecture overview that is defined by the developed library: Scalable Abxtraction (Sx), named after the design principle of modular extensibility. The library architecture is composed of eight individual modules. The image in 11.2 additionally depicts the dependencies of the modules. The interface module depends on all other modules and the core, file, and mathematics modules have no dependencies.

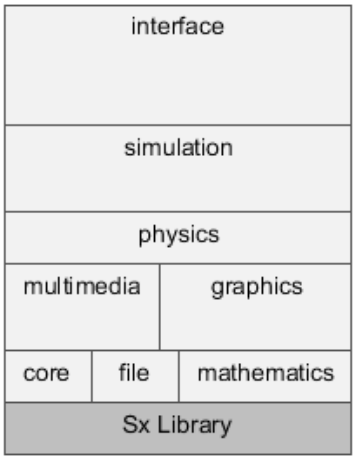



**Figure 11.2. Complete Sx Library Architecture**
The library provides the functionality for deformable-body simulation, real-time recording for animation, and interactive controls for modifying the behaviors of deformable objects via control metaphors.

This implementation provides all of the modules that facilitate the requirements of the proposed method of modifying deformable-body motion and deformation behaviors. Each of these modules are described in the next section. To provide the additional required components such as the interchangeable collision handlers and graphical user interface libraries a set of Sx Library extensions were developed. These provide the bridge between the abstract architecture provided in Figure 11.2 and the practical implementation of an animation editing application. The set of developed extensions are shown in Figure 11.3.

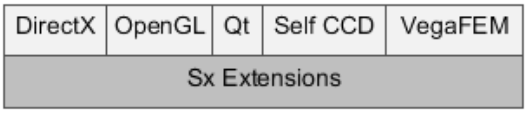



**Figure 11.3. Sx Library Extensions**
The modular design of the core architecture allows for completely interchangeable graphics libraries, external physics engines, and collision handlers. For the results illustrated in this work the libraries shown above were utilized.

**Sx Library Modules**

The development of this simulation framework is based on the requirements of recording a physical simulation to generate an animation. At a minimum this process requires that a set of objects can be loaded into an environment, simulated through the use of a dynamics engine, and recorded in real-time. Each of the included modules tackles one of these specific tasks. This section provides an outline of the major modules included in the Sx library and identifies the specific task the module is responsible for addressing.

The core module represents the fundamental building block upon which all other modules reside. This module provides the basic types and data structures that are utilized throughout the entire library. Standard types have been defined for all primitives and string representations to provide a consistent interface throughout the library. This is required due to the inclusion of several extension modules. Each library provides its own implementation of the basic types; therefore the Sx library provides a consistent set of internal data types for efficient communication within the library.

The file module is fairly simplistic. Essentially any parser or file reader that is required to load a graphics or simulated object (.obj, tetgen files, etc) is implemented in this module. Several modules such as the graphics and simulation modules include additional functionality for dynamically loading data from various file types into mesh-based object representations.

Developing a mature mathematics module is a requirement for any physical simulation and will typically require an extensive number of numerical functions. The mathematics library provides the basic mathematical functions that are required for physical simulation and interactive graphical environments. Interacting and manipulating

objects within a graphical scene requires an extensive number of basic mathematical operations. These include distances calculations between primitives, primitive-based intersection tests, and several numerical techniques. These functions are utilized extensively throughout the library to provide most of the functions that the artist will use to modify the state of the running simulation. The linear algebra library Eigen [37] is utilized for the standard vector and matrix types used throughout the entire Sx library.

The basic principles behind all collision libraries are derived from the mathematical process of using physics equations to update simulated objects. The physics module provides an abstract implementation of the architecture that facilitates these functions. Primarily this module provides a basis upon which a simulation module can be defined. Abstract implementations of physically-based models are provided and are implemented by any dynamics library provided as an extension.

The implementation of the interface module composes the core of the interaction between the artist and the physical simulation. The interface module is a representation of how user inputs are processed and interpreted by the dynamic editing environment and the physical simulation. This includes operations such as picking, the selection of objects within the simulation, and the direct manipulation of control metaphors. Additionally this module provides the functionality to decouple the graphics library from the controls that are used to modify the behaviors of the deformable objects. This module is also closely related to the multimedia library since the artist must be able to closely review and control the playback of the animation created by recording the physical simulation.

The process of recording the physical simulation is handled by the multimedia module. This module is particularly important for the development of the proposed

approach to deformable object control. This module provides all of the functionality related to recording specific types of frame data using an abstract recording architecture. Several different types of information have to be efficiently recorded in order to provide the artists requires. Specifically, the generation of the animation based on the physical simulation relies on storing the dynamic state information of the simulated objects. In contrast to this stored data type, the static states of the objects recorded for a preview of the simulation must also be supported. This module provides a flexible and extensible approach to developing customizable frame types that can all be used with an abstract recording definition. These features are utilized extensively to provide the implementation of the required animation recording process.

The most significant module in the library is the simulation module. This module defines the abstract requirements of a physical simulation and provides the extensions required for an external dynamics library to be connected to the physical simulation that allows an artist to apply the proposed method of deformable object control. In the provided implementation this module supports the creation of scenes where several objects can be loaded into the editing application as the rest state of the simulation. Once a simulation scene has been loaded, the artist can dynamically interact with the simulation to generate customized animations through the multimedia module features.

Through the modular and extensible design provided, the implementation provides support for any physical representation of a deformable object. Similarly the ability to interchange the collision detection and response handlers also illustrates the flexibility provided by the architecture. Based solely on the architecture these components can be neatly separated; however it does not account for the tight coupling between these

components that is required for a physical simulation. Each module contains components that are critical to the physical simulation and require direct communication with each other. The goal that was achieved through the development of this library was maintaining the extensible design presented by this architecture while allowing for efficient communication between loosely coupled components. This required the development of a novel approach to simulated object representation and graphics library independent viewport controls. The next two sections provide the introduction of graphics archetypes and viewport controllers. These constructs both effectively remove the tight coupling between the major components and libraries required for the implementation of a physical simulation.

**Archetypes**

Developing an abstract object type that defines the geometric representation of a simulated object requires that several modules of the architecture must be tightly coupled. Since the object is simulated, it must provide the dynamics engine with its current state within the environment. Additionally a static representation of the objects state must be provided to a graphics library to be rendered. Combining all of these components into a single object representation ultimately ties the object to a specific graphics library and dynamics engine. This completely disallows the ability to interchange either of these components. A novel approach to eliminating this tight coupling of unrelated components has been achieved through the development of graphics archetypes.

A graphics archetype represents an aggregate of three modular components: (1) an abstract representation of the simulated object, (2) a physical representation that is provided to the dynamics engine, and (3) a visual representation that can be provided to

any graphics library to render the object. Formally we define an archetype from these three components as the tuple as shown in Figure 11.4 where the object is represented by $\mathcal{O}$, the physical representation of the object is noted by $\mathcal{P}$, and the visual component $\mathcal{V}$.

$$\mathcal{A} = (\mathcal{O}, \mathcal{P}, \mathcal{V})$$

**Figure 11.4. Graphics Archetype Formal Definition**
Formal definition of the aggregate that forms a graphics archetype $\mathcal{A}$. $\mathcal{O}$ represents the abstract object type, $\mathcal{P}$ is the physical representation and $\mathcal{V}$ is the visual representation of the object.

The core of the simulation module is based off of the definition of a simulation archetype. As an example of the possible combinations of that can be used to define a simulation archetype, we present a brief list in Figure 11.5.

$$\mathcal{O} = \{cube,\ sphere,\ cylinder,\ prism, \ldots\}$$
$$\mathcal{P} = \{static,\ rigid,\ mass\text{-}spring\ system,\ FEM, \ldots\}$$
$$\mathcal{V} = \{rasterized,\ rendered,\ textual, \ldots\}$$

**Figure 11.5. Potential Archetype Definitions**
Potential graphics object definitions and physical representations that can be used to define an archetype.

**Viewport Controllers**

When an artist is interacting with a three dimensional scene, they must be presented with a graphical representation of the current state of the 3D world. Additionally, for an interactive application the artist must be able to provide input to the presented view to modify either the graphical scene or to switch between different viewing perspectives of the world. Following the standard model-view-controller (MVC) design principle, we introduce an architecture that allows for the modular definition of controls used to interact with a viewport that presents a three dimensional scene. The notion of a viewport controller provides the controller component of the standard MVC model, that is, it defines the controls that an artist can use to interact with the content of a viewport. This

represents an abstract type of controller that can then be extended to include the required functionalities of the interactive animation editing application. These functions include the dynamic selection (picking) of simulated objects, the ability to view the simulation from several different perspectives at once, orthographic views for precise object control and the direct manipulation of control metaphor widgets. These features are implemented through the introduction of a viewport controller hierarchy that is modularly designed to be utilized for any potential simulation or interactive control for the main viewport interface of the application. This hierarchy is illustrated in Figure 11.6.

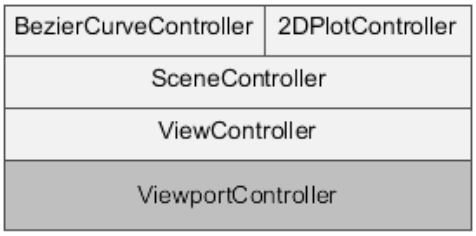



**Figure 11.6. Viewport Controller Hierarchy**

The viewport controller architecture provides interchangeable functionality between all viewports regardless of which graphics library is utilized (OpenGL, DirectX).

**Collision Models**

Physical simulations require efficient data structures for handling the data related to the simulated objects. For the simulation of a mass-spring system, the positions, velocities, accelerations, internal and external forces are stored as the state of the object. Similarly, an organized structure for storing collision information must also be provided. The implemented architecture considers the abstract representation of a collision model. The definition of the collision model is based on the generalized structure of a collision detection/resolution scheme required by any physical simulation. A collision model is defined as container that defines what interchangeable collision detection and resolution handlers are selected to resolve the collisions of the scene. Due to the modular behavior

of the collision detection and resolution handlers, the overall collision procedure can be dynamically changed at run-time.

The development of the collision model also requires that the collision information specific to the types of object colliding must be provided. The introduction of a collision aggregate addresses this requirement. A collision aggregate is a collection of collision events that are specific to the physical and visual representations of the object archetypes utilized in the simulation. For the introduced notion of a polygon-soup, the collision events are enumerated as follows: (1) Face-Face, (2) Face-Edge, (3) Face-Vertex, (4) Edge-Edge, (5) Edge-Vertex, and the least common (6) Vertex-Vertex. The collision aggregate is initially populated by the collision detection handler and then this collision information will then be processed by the collision resolution handler. This process is performed for each time-step of the simulation and completes the process of detecting and resolving collisions. Depending on the collision resolution approach, this architecture also allows for iterative resolution schemes.

**Performance**

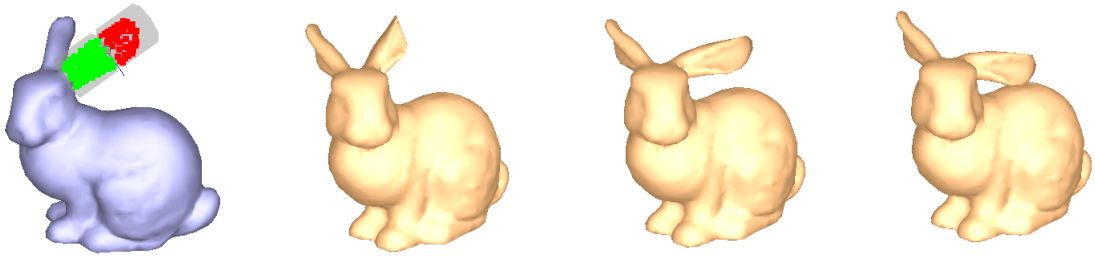
The implementation of an architecture that facilitates physical simulation must always consider the impact of the design on the performance of the resulting application. Computationally expensive operations like numerical analysis-based algorithms and dynamic allocations must be minimized to ensure that the performance will sustain a real-time interactive application. Specifically, the implementation of the provided collision resolution scheme does not provide a robust solution for all permutated deformable object collision possibilities; systems that provide a completely accurate collision resolution scheme are to computationally expensive to support highly interactive application. Rather

we utilize a collision resolution scheme that functions well in most common instances and use the extra performance to ensure that the artist using the program receives proper feedback about the simulation state in real-time.

Another performance concern with the implementation of the collision model architecture is the dynamic allocation of resources during a given time-step. The overload of redundantly allocating dynamic resources based on the number and types of collisions detected during a given time-step prevents the possibility of ensuring a stable real-time system. To counteract this problem we have introduced a memory scheme that utilizes pre-allocated memory blocks to hold the collision data. To provide an estimate for the initial size of this memory allocation we utilize heuristics that describe the geometric complexity of the simulated objects and the total number of objects within the simulation. The provided heuristic may not guarantee that a dynamic allocation will occur; however we can consider the worst-case scenario of all vertices, edges, and faces colliding during an individual time-step. The probability of this singular event is extremely low and therefore additional heuristics (such as analyzing the topology of the mesh as a graph utilizing Euler's formula for planar graphs) can be developed that provide a more intelligent estimate of the space required for the collision events. A similar technique of pre-allocated memory is also utilized within the multimedia module for recordings generated by storing the states of the objects within the simulation to create the resulting animation in real-time.

# CHAPTER XII

# RESULTS



Twist Control Metaphor | Frame [0] (Rest) | Frame [30] | Frame [60]

**Figure 12.1. Localized Twist Control Metaphor Deformation**

The application of the twist control metaphor applied to the left ear of the bunny model (left). The sequence of the next three images (right three images) illustrates the local deformation of the ear over the course of 60 frames due to this twist control metaphor. The image (left) is highlighted due to its selected status in the application. The images are all rendered using a Phong-based shader.

The results of utilizing the proposed methodology for deformable object control are dynamically generated from the interactive application introduced in Chapter 8. The process of applying a control metaphor to the deformable objects within a simulation follows a simple sequence of steps: (1) load the rest state of the simulation, (2) select any deformable object and apply one or more control metaphors, and (3) define the affected nodes and force curve for each applied control metaphor. This provides the information required for the simulation and animation generation. As the simulation progresses the frames are automatically recorded in real-time. To illustrate the deformation over time, we look at specific frames at a set interval. These depict the key moments where the deformations are distinctly visible.

Initially we consider the application of a control metaphor to illustrate a localized deformation of a complex model. The model is defined by an archetype that consists of a standard manifold mesh, a mass-spring system and a renderer that utilizes Phong-based

shading. Figure 12.1 provides an illustration of the twist control metaphor applied to the standard bunny model. This demonstrates the effect of the twist metaphor over the course of the animation as it is applied to the left ear of the model. This is illustrated by the red and green sets of effected nodes that belong to the ear region of the bunny model. The metaphors control widget is illustrated to provide the definition of edit used to obtain the resulting motion showed in the next three images. At frame zero in this figure the model remains unmodified and depicts the rest state of the simulated object. As the external forces are applied to the model by the definition of the twist metaphor, the ear begins to twist based on the provided force curve. This is illustrated by the remaining two images that show the model at frames 30 and 60 of the animation. The deformation is not only characterized by the definition of the twist control metaphor. Due to the connectivity of the effected region, the ear is not twisted directly in opposing directions. This is due to the fact that if the metaphor stiffly enforced the movement of the effected nodes, then the resulting deformation of the surface at the transition between the affected nodes and unaffected nodes would generate sever visual artifacts as the surface warps. The resulting motion softly twists the ear according to the accompanying force curve. When the end of the force curve is encountered, the external forces are removed and the object can return to its rest state.

This example provides a clear demonstration of a localized deformation imposed by the applied control metaphor. Applying different control metaphors at different locations result in different deformations. Additionally, the force curve provides an extremely flexible interface for providing detailed information about how the deformation should be imposed on the object.

**Compound Control Metaphors**

Demonstrating the impact, flexibility, and utility of force curve applications, we consider how a single control metaphor can be utilized to generate more than one form of deformation. Specifically, the sign of the discrete values of the force curve generate alternative deformations like those introduced in Chapter 8. Following the provided result of using the stretch control metaphor, we can demonstrate that using control metaphors with interchangeable force curves is a robust and effective method for introducing deformations that are independent of the geometry of which the metaphor is applied to.

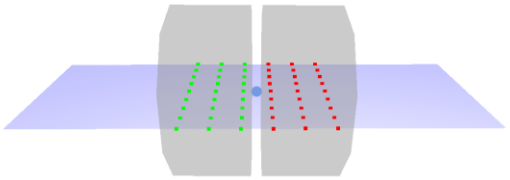
As an example of applying a control metaphor that can be used in several ways, we look at the application of the stretch control metaphor with a suspended cloth model. In this example, the process of utilizing a single control metaphor will be used to both stretch and fold the suspended cloth. The cloth is suspended through two sets of anchor nodes along the edge of the cloth. Anchor nodes are defined as point masses within the mass-spring system that are not affected by internal or external forces, therefore their positions remain constant during the course of the animation.

Looking at Figure 12.2, a sequence of images is presented that shows the initial configuration of the stretch control widget, the initial rest state of the suspended cloth, the effect of gravity on the cloth with no metaphor influence, and the result of applying two different force curves to the applied stretch metaphor. The applied force curves are illustrated in Figure 12.3.

The stretch control metaphor deformation behavior is defined by the process of separating two node sets with opposing forces. The selected sets of nodes defined at the rest state are highlighted in Figure 12.3 (a). If no force curve is provided or it defines a constant magnitude of 0.0, then the resulting motion of the suspended cloth is shown in

Figure 12.3 (c). When the force curve provides a positive value for the magnitude of the forces over time, the selected nodes are effectively separated. This can be seen in Figure 12.3 (d) where the center of the cloth has been stretched as it hangs. To provide a completely different behavior with this control metaphor we consider a force curve that provides a negative force magnitude. Applying a force curve that provides a negative force magnitude to the suspended cloth, we note the drastically modified behavior as shown in Figure 12.3 (e). The compression of the cloth results in a fold being formed at the center of the stretch metaphors location.

Considering the two examples (d) and (e) from Figure 12.2, we have demonstrated that the flexibility of this method allows for drastically different results of the deformations imposed on the object given that only the force curve has been modified. The effect of the negative force curve on the control metaphor implies that for every control metaphor that has been implemented, an additional operation (based on the inversed external force direction) has been provided. Briefly expanding on this notion we can see that each control metaphor applies a different deformation when the forces are inversed. The bend control metaphor will simply bend or deform the object in the opposite direction, while the twist metaphor will simply rotate the ends of the object in opposite directions. While these are simplistic examples, additional control metaphors can be developed that exploit these inverted forces to provide additional deformation behaviors. Figure 12 presents a simple example of inversed forces to achieve different behaviors. Through the extensible framework provided, it is trivial to apply these control metaphors to other objects and introduce new behaviors with slightly modified force directions. These new directions can also be inversed with a negative force magnitude.



(a) Stretch Control Metaphor



(b) Frame [0] (Rest)



(c) Frame [60] Uninfluenced result



(d) Frame[60] Result of the stretch metaphor

(e) Frame [60] Result of a compression via stretch metaphor

**Figure 12.2. Compound Control Metaphor Application**

The series of illustrations depict a cloth model with anchor nodes along two edges over the course of 60 frames (b, c, d, e) with the control metaphor used to derive the illustrated behavior (a).

The process of defining a negative force curve is incorporated in the intuitive interface of the implementation of the force curve editor. To define a force curve that provides negative magnitudes, the artist can simply define the control points of the Bezier curve to have a negative $y$-component. The result of defining positive and negative forces curves is show in Figure 12.3.

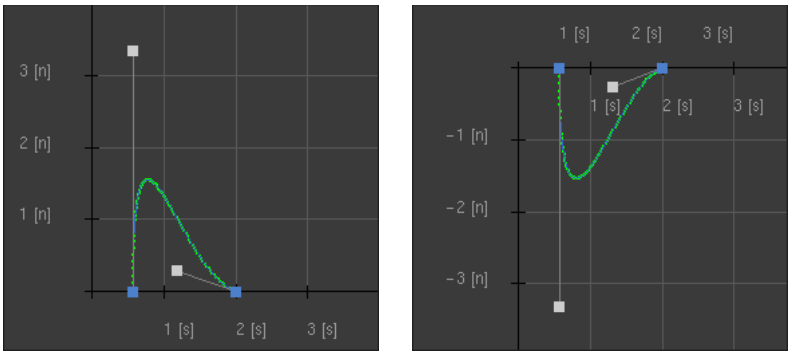



**Figure 12.3. Inverse Force Curves**

To generate different deformations within a simulated object utilizing a single control metaphor, an artist can provide various force curve definitions to achieve distinct localized behaviors.

The ability to derive multiple behaviors from the single definition of a control metaphor makes this approach extremely flexible for several reasons. The primary reason this introduces flexibility is because the amount of work required to fulfill a large enumeration of possible behaviors has been greatly reduced. The addition of a new control metaphor does not strictly define a single new behavior that can be imposed on a simulated object, but rather it maps to a larger set of possible deformations. This allows for ingenuity in how control metaphors are utilized and provides an open door for artistic experimentation. Another way that this method promotes flexibility is through the application of multiple instances of the same control metaphor on the same object to drive different localized behaviors. This introduces the concept of compound edits utilizing these control metaphors. The next section illustrates the process of creating complex deformations of the bunny model utilizing a compound edit.

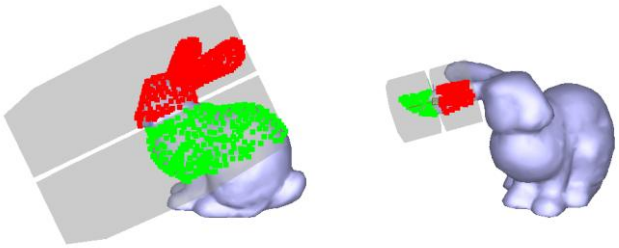
### Compound Deformation Edits

The set of control metaphors that have been introduced provide intuitive motion controls for localized deformations; however individually they are extremely simplistic and lead to simplified deformations. This aids with the interpretation of each control

metaphor, but independently each metaphor lacks the ability to express complex motions. The behaviors modeled by this set of control metaphors do not limit the range of deformation behaviors that can be achieved using this approach. The introduction of overlapping or concurrent control metaphor application provides the ability to derive complex behaviors from this simplified set of behaviors.

Given a deformable-body and two control metaphors that will act upon the object, the metaphors are overlapping if their associated force curves provide an external force at the same time-step of the simulation. The direction and magnitude of the resulting external force is defined by the compound motion equation presented in Chapter 4. This is simply stated as the vector sum of the external forces acting upon some given node. Therefore we present the result of applying two control metaphors to a single model to derive complex deformation behaviors.

For the presentation of a compound deformation edit we look at the application of the control metaphors and their configurations at two different instances in time. The artistic intent we are hoping to achieve in the animation is a bunny flapping its ears by rotating its head. This is a common naturally occurring behavior [31] in rabbits that we can demonstrate utilizing the standard bunny model. Initially we impose the required movement of the bunnies head in relation to its body. This is obtained using the twist control metaphor. This will rotate the head to promote the rotation of the right ear to achieve the desired “flapping” behavior. The configuration of the twist control widget used to obtain this result is illustrated in Figure 12.4 (a). This combination of applied control metaphors composes a compound edit. The resulting behavior will be determined based on the vector sum of external forces as defined by Equation 4.1

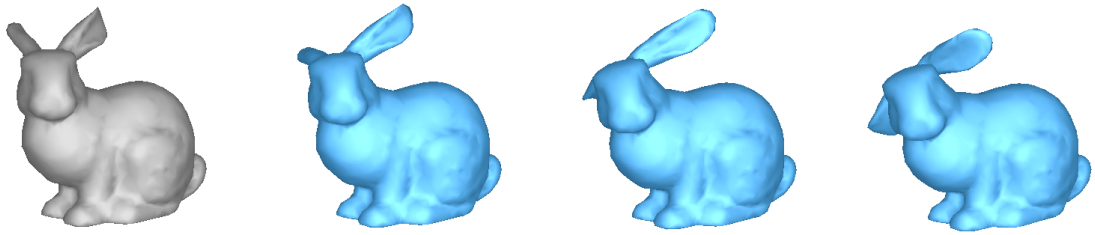


(a) Head Twist (b) Ear Rotation

**Figure 12.4. Compound Deformation Edit Configuration**

The head of the bunny model is initially rotated utilizing a twist metaphor (a). This is applied at the rest pose of the bunny model. To obtain the correct behavior of the ear as the head rotates, an additional control metaphor is applied during the rotation. This will provide a better separation of the ears to provide a more realistic flapping effect (b).

Since a control metaphor can be applied to any set of time-steps within the simulation, we illustrate the two deformations involving the same nodes can be imposed at the same time. In this example we look at how we can rotate the head of the bunny and at the same time, ensure that the ears will being to flap in the correct directions due to imposed rotation. This is achieved applying a second twist control metaphor to make the right ear of the bunny flap against the rabbit's cheek. The resulting motion of this compound edit is illustrated in Figure 12.5.



Frame [0] (Rest) Frame [20] Frame [40] Frame [60]

**Figure 12.5. Compound Deformation Behavior: Ear Flap**

Compound edit of a bunny model to simulate a compound edit that results in an ear "flapping" against the cheek of the rabbit's face. This result is achieved by applying two separate twist control metaphors.

# CHAPTER XIII

# EVALUATION AND DISCUSSION

The interactive methodology presented provides an extensive amount of control to the artist to modify the motions of deformable objects in existing animations generated from physical simulation. Several forms of feedback have been discussed, including the interactive replay of the recorded simulation, the visual representation of how a control metaphor will affect a deformable object, and the generation of preview states that illustrate how the current control metaphors will affect the outcome of the simulation. All of these factors contribute towards providing a highly interactive editing environment for deformable models. In this chapter we look through the contributions of this work in the domain of deformable object control and explore how the presented work can be applied towards a practical solution for simulated object control.

The effectiveness of introducing external forces to achieve a desired local deformation of a physically simulated object has been thoroughly explored by this approach and provides an effective means to modifying physically-based animations. Previous research efforts [1, 13] provide effective methods for controlling deformable objects in physical simulations; however they do not specifically address the introduction of local deformations for existing animations. Even with the proposed technique for editing localized deformations there are several considerations that must be accounted for. Since the artist has the freedom to modify the animation interactively throughout the simulation they incur an extensive amount of responsibility for correctly defining the parameters of the edits they impose. The artist is also responsible for ensuring that the parameters of the control metaphors they provide are valid for the effected object.

Additional considerations must be made for alternative approaches that utilize rest-pose parameterization [2], that is, the control of an object is limited to its direct interaction with the environment. If an object cannot make contact with another object within the environment the artistic control of the object is lost. The presented approach attempts to eliminate these problems to provide a higher level of interaction between an artist and the dynamic system. While this high level of interaction has been accomplished, the methodology of using control metaphors does have some inherent limitations.

**Control Metaphor Limitations**

When analyzing the effectiveness of imposing a high-level movement on an object we look at how an artist interacts with the implementation and how the resulting deformation represents the original intent. The control of the simulated deformable objects is provided through the set of defined control metaphors. Control metaphors however are subject to several conditions that must be met for their proper operation. This states that there are several factors that can contribute to an invalid configuration of the applied control metaphor which may lead to undesirable or unstable behaviors: (1) the geometric configuration of the control metaphor must match the requirements defined for the applied metaphor. That is, if a control metaphor (such as the bend metaphor) requires three distinct sets of nodes to properly impose the intended behavior, and is not configured by the artist to contain three sets of effected nodes, the resulting behavior will be incorrect. (2) The application of the external forces by the control metaphors imposes undesirable net forces and torque. This represents a large problem for closely refining the motion of an object over time. When a localized deformation is introduced, the net force and torque on the object will be affected.

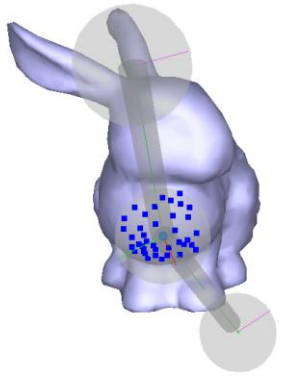


**Figure 13.1. Improper Bend Control Metaphor Application**
The set of points considered for the joint set of this bend metaphor is illustrated in blue. We note that this configuration does not illustrate any red or green node sets. The only nodes that will have an external force applied are those in blue. This configuration fails to meet the requirements of the bend control metaphor.

This proposed solution mitigates this effect by equalizing the forces based on the control metaphors force diagram; however the problem persists due to this approaches inability to remove the net torque on the object. (3) The magnitude of the external forces required for the control metaphor to properly impose the desired motion is completely arbitrary and hard to accurately define. Even with the inclusion of the force curve editor, it takes several experimental trials to derive an adequate force magnitude for the provided deformable object. This problem is further exacerbated by the stability of the physical simulation when extremely large external forces are applied to a deformable-body. When this occurs the simulation will diverge and the resulting animation will be lost with the invalid state. As a result of this problem the editing process is negatively influenced. The artist must provide an initially weak set of external forces to ensure that the simulation will remain stable. The iterative process does allow the artist to incrementally increase this external force, however this process is tedious.

With the consideration of the problems encountered with the proposed approach and the weaknesses of the prior approaches, we can attempt to identify the components that will contribute to a practical control solution. The important aspects of deformable-body control can be divided into two categories: the control of the behavior of the deformations over time and the final or desired state of the object. Taking the highly interactive component from this approach and its ability to define general movements and local deformations, we can derive intuitive motions that satisfy the first category. Looking at a goal-driven approach where the artist can define a desired static state of an object (control of the object's final state at some point in time), dynamic key-frames can be used to satisfy the second. In the next section we consider the implications of combining these approaches to provide a robust solution that addresses both of these requirements.

**Control Metaphors and Dynamic Keyframes**

Towards the goal of deriving a practical solution for deformable object control, the proposed method of altering existing animations provides a lot of flexibility to the artist and promotes an iterative development environment. This allows the artist to adjust and fine tune the influence of the control metaphors used to manipulate a deformable object. However, while this provides a highly interactive editing methodology, this process of refinement is not always desired. When the user is less concerned with the intermediate states of the deformable object and its apparent motion, the aim is to define the desired deformation state and position that they would like to impose on the object at some instance in the animation. In this section we describe the notion of combining the

proposed set of control metaphors with the existing approach of dynamic key-frames to achieve a dynamic, goal-oriented editing environment.

The method of providing this functionality is to provide dynamic key-frames that will identify the desired final state of a deformable-body. As previously mentioned in Chapter II, the derivation of these dynamic key-frame static states is incredibly challenging. To facilitate a complete editing solution we propose that our method can be used in combination with dynamic key-frames to generate goal-oriented movements of deformable bodies with relatively few modifications. With the provided method for manipulating the state of a deformable body based on intuitive motions, the static states of the object used for the key-frames can be derived for complex deformation states of the object. Therefore this combination of approaches provides the required means of creating dynamic key-frames and utilizing them to generate a final animation.

This provides a technique for efficiently generating the deformation states required for an inverse dynamics approach to physical animation. Specifically this allows an artist to iterate on the key component that will dictate the motion of the deformable-body in the animation created from a set of dynamic key-frames. The state of the object can be recorded an modified through the use of control metaphors and when the desired state has been reached it can simply stored as a dynamic key-frame. Since this methodology already provides an interactive recording of the deformable object, this information can simply be added to a dynamic key-frame set. Following the derivation of the dynamic key-frame approach, when the artist has added all desired states, the final animation can be generated.

This newly considered approach also introduces an extension to control metaphors. In the proposed method of introducing localized deformations, control metaphors are applied over time based on the associated force curve. Since the process of defining a dynamic key-frame can also be viewed as a static representation of the deformable-body at some instance in time, control metaphors can also be applied statically to adjust the desired state of the object. The only fundamental difference here is that the static control metaphor will not be applied through the dynamics engine; it will be directly controlled by the artist and will influence the static positions of the nodes within the effected object.

This combination of control metaphors used to modify the dynamic and static state of a deformable body and the application of inverse dynamics to generate a resulting animation provides the foundation of an intuitive editing environment that can be further explored through similar research. Independently these approaches have been shown to provide practical solutions to deformable object control. Extending this research to include a technique that utilizes the strengths of both of these approaches may yield a higher level of control in both desired motion and final state of deformable object.

# CHAPTER XIV

# CONCLUSION

The accurate control of deformable objects in simulated environments presents a challenging task. Prior to this work several existing methodologies that are commonly used to control physically simulated objects were extensively analyzed to try to provide a robust technique that would try to counter the problems that are commonly associated with the unwieldy motions of deformable bodies. Several of the beneficial aspects of theses previous attempts were adapted and incorporated into the presented approach to localized deformation control; yet while these aspects improved the level of interaction between the artist and the tools used to modify the an animation, several problems have been identified.

With this approach we allow the artist to explicitly define the force magnitudes that will used in the definition of the external forces that are directed by control metaphors. This provides a high-level of control to the artist; however this may shift from empowering the freedom of the artist to relying on their external knowledge to correctly modify the animation. While the implementation of this technique allows for artistic iterations, it also relies on the definition of completely arbitrary force magnitudes to perform the intended localized behaviors. An alternative means to determining the magnitudes of the external forces would probably alleviate the pressure applied on the artist to supply the appropriate input. This option is explored in Chapter 13.

With the bounds of correctly defining the magnitudes of the external forces, the presented approach accurately provides the motions that we intend. With the limited set of primitive control metaphors that were developed, several physically plausible and

visually appealing localized deformations were created. These results were generated from the use of the dynamic animation recording architecture that allows for the real-time generation of simulation-based animations. Therefore the motions that were intended to be applied to specific regions of deformable models at some point in an existing animation provided results that are aesthetically pleasing. This approach can be utilized to effectively impose local deformations on the geometric definitions of deformable objects based on prior animations.

# CHAPTER XV

# FUTURE WORK

In the development of this approach we noted the key elements that prior research efforts provided. We utilized some of these intuitive ideas presented in alternative ways to present similar feedback to the artist composing and editing the animation. While this provides an effective form of feedback to the artist, there are additional features that could be implemented to make the process even more intuitive.

Based on this approach that defines intuitive interface controls based on Bezier curves to alter force magnitudes and directions, we could extend this concept to include an inverse to this functionality. Specifically, a method of altering the position in the three-dimensional preview window could invoke an inverse dynamics function to determine the direction and magnitude of the force required to reach the users specified location of the deformable-body provided by the dynamic preview generated. This would allow the artist to control both the objects position in three-dimensional space as well as by the force curve that modifies the behavior of the object.

Additional research could be performed to determine alternative methods of defining the magnitudes of the external forces that are required for generalized motion control. A generalized form of inverse dynamics may provide a flexible method to determining these values. The complexity of this problem however is not simply solved by considering the motion of the object, additional research into how the external forces will generate completely different motions for similar but geometrically different physical representations. Addressing these problems would provide deformable body controls that are much more stable than the currently implemented techniques.

## REFERENCES


[1] Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier, Interactive spacetime control of deformable objects. ACM Trans. Graph. 31(4):71, 2012.

[2] Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, Markus Gross, Deformable objects alive!, ACM Transactions on Graphics (TOG), v.31 n.4, p.1-9, July 2012.

[3] Brian Mirtich, John Canny, Impulse-based simulation of rigid bodies, Proceedings of the 1995 symposium on Interactive 3D graphics, p.181-ff., April 09-12, 1995.

[4] D.J. Murray-Smith, The inverse simulation approach: a focused review of methods and applications. Journal of Mathematics and Computer in Simulation, pp. 239–247, 2000.

[5] Jovan Popovic, Steven M. Seitz, Michael Erdmann, Zoran Popovic, Andrew Witkin, Interactive manipulation of rigid body simulations, Proceedings of the 27th annual conference on Computer graphics and interactive techniques, p.209-217, July 2000.

[6] Jovan Popovic, Steven M. Seitz, Michael Erdmann, Motion sketching for control of rigid-body simulations, ACM Transactions on Graphics (TOG), v.22 n.4, p.1034-1054, October 2003.

[7] David Baraff, Andrew Witkin, Michael Kass, Untangling cloth, ACM Transactions on Graphics (TOG), v.22 n.3, July 2003.

[8] M. Teschner, S. Kimmerle, et al. Collision detection for deformable objects. In Proc. Eurographics State-of-the-Art Report. EG Association, 2004.

[9] David Baraff, Andrew Witkin, Large steps in cloth simulation, Proceedings of the 25th annual conference on Computer graphics and interactive techniques, p.43-54, July 1998.

[10] Carol O'Sullivan, John Dingliana, Thanh Giang, Mary K. Kaiser, Evaluating the visual fidelity of physically based animations, ACM Transactions on Graphics (TOG), v.22 n.3, July 2003.

[11] Teschner M, Kimmerle S, Heidelberger B, et al. Collision detection for deformable objects. Comput Graphics Forum 2005.

[12] Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus H. Example-based elastic materials. Gross. ACM Trans. Graph. 30(4):72, 2011.

[13] Jernej Barbic, Funshing Sin, Eitan Grinspun, Interactive editing of deformable simulations. ACM Transactions on Graphics (TOG), v.31 n.4, p.1-8, July 2012.

[14] Min Tang, Dinesh Manocha, Ruofeng Tong, Fast Continuous Collision Detection using Deforming Non-Penetration Filters, in Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2010), Washington DC, Feb. 19-21, 2010.

[15] Lynne Shapiro Brotman, Arun N. Netravali, Motion interpolation by optimal control, ACM SIGGRAPH Computer Graphics, v.22 n.4, p.309-315, Aug. 1988.

[16] Andrew Witkin, William Welch, Fast animation and control of nonrigid structures, Proceedings of the 17th annual conference on Computer graphics and interactive techniques, p.243-252, September 1990.

[17] Christopher D. Twigg, Doug L. James, Backward steps in rigid body simulation, ACM Transactions on Graphics (TOG), v.27 n.3, August 2008.

[18] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, Tom Sanocki, Harmonic coordinates for character articulation, ACM Transactions on Graphics (TOG), v.26 n.3, July 2007.

[19] Gain, J., & Bechmann, D., A survey of spatial deformation from a user-centered perspective. ACM Transactions on Graphics, 27(4), 1-21. 2008.

[20] Gibson, S. F. F., & Mirtich, B. A Survey of Deformable Modeling in Computer Graphics. October 1997.

[21] Kondo, R., Kanai, T., & Anjyo, K.-ichi, Directable animation of elastic objects. Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05, 127. New York, New York, USA: ACM Press. 2005.

[22] Müller, M., Heidelberger, Bruno, Teschner, Matthias, & Gross, Markus, Meshless deformations based on shape matching. ACM Transactions on Graphics. 2005.

[23] Jeon, H., & Choi, M.-hyung. (n.d.). Interactive Simulation of Motion Editing and Pattern-Based Control for Deformable Objects. Control, 1-10. 2007.

[24] Nealen, A., Müller, M., Keiser, Richard, Boxerman, E., & Carlson, M., Physically Based Deformable Models in Computer Graphics. Computer Graphics Forum, 25(4), 2006.

[25] Greway, R. Anthony, Localized Motion Control of Dynamically Simualted Elastic Models. (Masters Thesis). University of Colorado Denver. 2012.

[26] Terzopoulos, D., Platt, J., Barr, A., & Fleischer, K. Elastically deformable models. ACM SIGGRAPH Computer Graphics, *21*(4), 205-214. 1987.

[27] Teschner, M., Heidelberger, B., Muller, M., & Gross, M. (n.d.). A versatile and robust model for geometrically complex deformable solids. Proceedings Computer Graphics International, 312-319. IEEE. 2004.

[28] Martin, S., Thomaszewski, B., Grinspun, E., & Gross, Markus Example-based elastic materials. ACM SIGGRAPH 2011 papers on - SIGGRAPH '11 (Vol. 30, p. 1). New York, New York, USA: ACM Press. 2011.

[29] Popovic, J., Seitz, S. M., and Erdmann, M. "Motion sketching for control of rigid body simulations" ACM Transactions on Graphics. 2003.

[30] Wang, H., & Brien, J. F. O. Data-Driven Elastic Models for Cloth: Modeling and Measurement. *Measurement*, *30*(4). 2011.

[31] Shapiro, A. Reading Your Rabbit. House Rabbit Society: A national nonprofit corporation. 2010.

[32] Shewchuk, J R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Carnegie Mellon University. 1994.

[33] Sunil Hadap, Dave Eberle, Pascal Volino, Ming C. Lin, Stephane Redon, Christer Ericson, Collision detection and proximity queries, ACM SIGGRAPH 2004.

[34] Sung-eui Yoon, Young J. Kim, Takahiro Harada. Recent advances in real-time collision and proximity computations for games and simulations. ACM SIGGRAPH Asia. 2010.

[35] C. Ericson. Real-Time Collision Detection. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan and Kaufmann Publishers, 2005.

[36] Min Tang, Dinesh Manocha, Miguel A. Otaduy, Ruofeng Tong, Continuous penalty forces, ACM Transactions on Graphics. 2012.

[37] Andrew Witkin, Michael Kass. Spacetime Constraints. ACM Computer Graphics. Volume 22, Number 4, August 1988.

[38] James K. Hahn. Realistic Animation of Rigid Bodies. ACM Computer graphics. Volume 22, Number 4, August 1988.

[39] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," ACM SIGGRAPH Computer Graphics, 1987.

[40] Müller, M., Keiser, R., Nealen, a, Pauly, M., Gross, M., & Alexa, M. Point based animation of elastic, plastic and melting objects. Proceedings of the 2004 ACM SIGGRAPH. 2004.

[41] WOJTAN, C., MUCHA, P. J., AND TURK, G. 2006. Keyframe control of complex particle systems using the adjoint method. In Proc. of ACM SIGGRAPH/Eurographics Symp. on Computer Animation 2006.

[42] FROHLICH, S., AND BOTSCH, M. Example-driven deformations based on discrete shells. Comput. Graph. Forum. 2011.

[43] BARBIC, J., DA SILVA, M., AND POPOVI´C, J. 2009. Deformable object animation using reduced optimal control. In Proc. of ACM SIGGRAPH 2009.

[44] Eigen. A C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. http://eigen.tuxfamily.org/

[45] VEGA FEM Library. A Computationally efficient and stable C/C++ physics library. http://run.usc.edu/vega/

[46] Qt Project – A cross platform application and UI framework for developers using C++. http:// http://qt-project.org/