Procedural Methods Using Physically Based Modeling

by

Jong Won Lee

B.S. March 1986, Kyungpook National University
M.S. May 1991, Illinois Institute of Technology

A dissertation submitted to
The Faculty of
The Department of Computer Science
of The George Washington University in partial satisfaction
of the requirements for the degree of Doctor of Science

January 31, 2001

Dissertation directed by
James K. Hahn
Associate Professor of Engineering and Applied Science
Dissertation committee:
Professor John Sibert
Professor Shmuel Rotenstreich
Professor Simon Berkovich
Dr. James A. Ballas

Abstract

Physically based modeling can produce extremely realistic motions. However, it requires heavy computations, complicated numerical solvers and complicated control algorithms. On the other hand, procedural methods produce realistic textures and geometries by relatively simple procedures. However, the procedural methods do not include physical quantities such as mass and force and the final results are hard to predict. We propose *physically based procedural methods* that have advantages of physically based modeling and procedural methods.

Although physically based procedural method starts with all physical equations of the computational model, it subdivides and simplifies the equations according to their physical meanings. Finally the method utilizes the simplified equations to calculate motions of objects. Although it is not directly derived from Newtonian dynamics, our method provides an efficient and numerically stable way of generating visually plausible motions. The paradigm of the physically based procedural method can also be applied to sound generation. Sounds that are hard to simulate by vibration analysis were formulated by the physically procedural method. As an application example, a real time wind system was designed and implemented based on the physically based procedural method. In this virtual wind system, the user blows into the microphone, the system produces real time virtual wind according to the intensity of blowing, direction and distance of the virtual objects from the user. Physically based procedural method in Issac system was also proved that it is useful as a real time motion generation tool through simulating Alexander Calder’s mobile and Teri Elis’ clock.

CONTENTS

List of Figures

Chapter 1 Introduction

Since motions in real world obey physical principles, it is natural to apply physical principles to generating motions of objects. In computer graphics, physically based modeling (PBM) determines motions and shapes of objects by physical properties and laws. In the area of computer animation, we have many methods for generating realistic motions. Among them, physically based modeling is one of the most powerful methods to generate realistic motions.

Physically based modeling has its own pros and cons. It is based on Newtonian physics, and thus has the power of producing physically correct motions. However, it uses complex equations, and usually results in heavy computations. Additionally, physical quantities of objects including forces and accelerations are often directly used to control motions of objects, even though it is not an intuitive way of control. Thus, currently, physically based modeling is not widely used even though it is one of the best methods to generate realistic motions in computer animation and Virtual Reality (VR) [Barz97].

To solve problems in physically based modeling, we propose the paradigm of procedural methods based on physics. Procedural methods have tried to mimic physical phenomena rather than exactly simulating physical laws [Four82][Weil86]. These approaches achieved visual plausibility and also provided easy control of complex phenomena. Although the procedural method succeeded in generating visually plausible motion of natural phenomena, the method has nothing to do with physical properties. Therefore, if physical quantities of object are changed, the motion from the procedural method is difficult to correspond with the new physical quantities.

This dissertation presents procedural methods to solve the problems of physically based modeling. We called it physically based procedural methods. Unlike previous procedural methods, physically based procedural methods formulate equations of motions mainly based on physical quantities and physical meanings. Therefore, the motions from the physically based procedural methods better correspond with physical properties. Additionally, it provides an easy way of controlling the motions and sounds.

Generating sound is similar to generating motions. Generating motions means producing position and orientation of an object with time parameter. While generating sound means producing one-dimensional amplitude with time parameter. Although exact vibration analysis of an object produces realistic sound, analyzing vibration of a complicated object is difficult. Procedural methods can be applied to generate sounds easily like motion generation case.

The physically based procedural method for motion and sound is included in our *Isaac* system. Isaac is a motion and sound generation system that includes constrained dynamics, impact dynamics, collision detection, physically based procedural method, sound generation and user interactions. User interaction includes virtual mouth blowing system. This is a unique and powerful interactive way of controlling articulated objects modeled by physically based modeling. Since Isaac combined dynamics and procedural methods, users can select dynamics or procedural method depending on computational loads.

Section 1 presents motivation of this thesis. In Sections 2, we present our problem domain and research objectives. Section 3 explains proposed solutions of the problems. Section 4 describes original and significant contributions of this dissertation. Finally, Section 5 describes the organization of this dissertation.

1.1 Motivation

Although physically based modeling is one of the best methods to generate realistic motions, it has the following drawbacks:

(1) Physically based modeling requires heavy computations. Linear system solvers, ODE solvers and optimization methods are often required to solve equations from physically based modeling. Therefore, applying physically based modeling to real time applications is still difficult.

(2) Physically based modeling is usually accompanied with numerical stability problems. Since physically based modeling requires complicated numerical solvers, it is prone to have stability problems. In computer animation applications, stableness is more important than exactness. And the parameters of the numerical solvers are difficult to control especially for non-mathematicians.

(3) Motions from physically based modeling are difficult for animators to control, since objects in physically based modeling are moved by forces. For example, users find it difficult to place an object at a specific location by only controlling the forces applied on it.

To solve the problems of heavy computations and stability, we can try to develop stable and fast numeric solvers. To solve the control problems, better optimization methods can be developed. However, these efforts are so hard to expect any dramatic improvement.

In most of computer animations application, visually plausible motions are enough for the applications. Moreover, some applications require exaggeration motions. Therefore, we need motion generation methods which can provide better speed, stability and controllability than physically based modeling, even though it may sacrifice physical correctness of physically based modeling.

1.2 Problem Domain and Research Objectives

The dissertation is concerned with generating motions and sounds. Motion control methods can be categorized into key frame method, physically based modeling, behavior method and motion capture method. Each method has its own pros and cons. Among these motion control methods, the thesis tried to solve the problems of physically based modeling.

Key frame method generates motions through specifying key motions. Although key frame can control details of the motion, it requires long time experience to generate realistic motions of complicated objects. Behavior method analyzes motions of objects and sets up the rules based on the analysis. It is mostly used for motions of grouped animals. Recently, motion capture has become popular for animators. Motion capture samples real motions of living things. It can produce extremely realistic motions because the motions come from living things. However, editing and controlling of the captured motions is not easy.

Physically based modeling is the problem domain of the thesis. Physically based modeling concerns the following motions: motions of rigid bodies, articulated objects, deformable objects, fracture, fluid and aerodynamics. The approach proposed by this thesis can be applied to all motions of physically based modeling. However, this dissertation focuses on the motions of articulated objects and motions by aerodynamics.

Physical principles can be applied to generate sounds. Using physical analysis to generate sound has similar problems with motion generation. Exact analysis of complicated objects is very difficult. Manipulating sampling sounds requires large memory and complicated control methods.

The goal of this thesis is to design new motion generation methods that can be an alternative to physically based modeling. This new method would be faster, more stable and more controllable than physically based modeling.

1.3 Proposed Solution of This Dissertation

To solve the problems of physically based modeling, we applied the paradigms of procedural methods. In computer graphics, procedural methods have been used to mimic natural phenomena. For example, procedural methods can formulate equations of texture and geometry of mountains by analyzing shapes of the objects rather than its physical properties.

Unlike previous procedural methods, we propose physically based procedural method. Our physically based procedural method formulates equations of motions by analyzing physical principles and physical quantities of objects. The method subdivides equations of dynamics and constraints and simplifies/amplifies the divided equations according to user intentions. The followings are the basic steps in formulating physically based procedural method.

(1) Extract all equations from the computational model. For example, Newtonian second law and constraint equations are basic equations of constrained dynamics.

(2) Simplify the equations according to their physical meanings. When its control is important for the given application, transform the equations into easier control fields. For example, if direct positional control of objects is required, then transform equations of motions from functions of accelerations to functions of positions. This transformation may introduce some computational errors.

(3) Add missing parts of the equation. For example, when you transform functions of

accelerations to functions of positions, inertia term may disappear. You have to add the procedural equation of the inertia term.

Although this is not an exact solution for the physical models, this method can produce visually plausible motions and sounds.

1.4 Original and Significant Contribution

This dissertation describes works believed to be original and contributory in the following areas related to physically based modeling and sound generation:

● Applying procedural methods to the physically based modeling. To solve the problems of physically based modeling, we proposed procedural methods based on physics equations and physical quantities. The methods produce visually plausible motions that are faster and more stable than motions from physically based modeling methods.

● Designing real time wind model based on physically based procedural method. By applying physically based procedural method, we designed a system in which the user's breath can generate real time virtual wind. As the user blows on the microphone, the system produces virtual wind according to the current environment settings. The system was used to simulate Alexder Calder's mobiles.

● Applying procedural method to generate sounds. By applying the procedural method, we formulated equation of various sounds including the sounds due to collision, contact and winds.

The parameter of the equations can be mapped easily with equations of motions. Therefore, synchronizations with motions become easier.

● Constraint combining and breaking with respect to the time parameter. We propose a constrained dynamics system. It allows users to design new constraints by combining existing constraints. And attaching time parameters to the constraints enables us to break and morph the constraints.

● Implementing a motion sound generation system that combines constrained dynamics, procedural methods and sound generation into an integrated environment. In this system, user can choose procedural methods or dynamics based methods. Parameters of motions and sounds have close relations. Therefore, motion and sound synchronization is accomplished naturally.

The work described in this dissertation has already resulted in three peer-reviewed publications:

[Lee00] J.W. Lee, N. Baek, D. Kim, and J.K. Hahn. A procedural approach to solving constraints of articulated bodies, Eurographics 2000, (ISSN 1017-4656) pages 55-64 August 2000.

[Lee00] D.K. Lee, H.J. Bae, C.T. Kim, D.C. Lee, D.H. J, N.K. Lee, N.H. Baek, J.W.Lee, K.W. Ryu, and J.K. Hahn. Reproducing works of Calder, J. of Visualization and Computer Animation, (to be appeared)

[Hahn95] J.K. Hahn, J. Geigel, J.W. Lee, L. Gritz, T. Takala, and S. Mishra. An Integrated Approach to Sound and Motion. Journal of Visualization and Computer Animation, Volume 6, Issues No. 2, pages 109-123, 1995.

1.5 Document Organization

Chapter 2 reviews other researches about physically based modeling, procedural methods and sound generation. Various related approaches are described, their limitations are outlined, and they are compared to our proposed approach.

Chapter 3 explains the Issac system that is used to demonstrate the ideas presented in this dissertation. Basic features (constrained dynamics, impact dynamics and constrained combining and breaking) of the system are explained. The procedural method and sound generation features of the system are explained.

Chapter 3 describes our physically based procedural methods. First, the way of formulating the procedural method is explained. Second, we explain in detail the algorithms of the physically based procedural method for motions of articulated objects. The procedural algorithms for sphere-like objects and wind model are described.

Chapter 3 presents equations for sound generation. These equations were formulated by physically based procedural method. It describes physics-based collision sound of homogeneous objects and collision sounds for heterogeneous objects. Texture-based scraping sound and fractal based wind sound are described.

Finally Chapter 4 concludes the dissertation with a review of results and issues, and future work.

Chapter 2. Literature Review and Issues

This chapter describes related works of this thesis. The related works can be divided into physically based modeling, procedural method and sound generation. These three parts are primary features of Isaac system.

Section 2.1 describes works on physically based modeling including field of collision handling, articulated objects, control methods and deformable objects. Section 2.2 states previous works on texture generation, geometric modeling and motions. Section 2.3 describes previous works on sound modeling, sound synchronization and sound rendering.

2.1 Physically Based Modeling

Physically based modeling has been an important research topic since late 1980's. Physically based modeling has shown the possibility of automatic motion generations and realism of motions in real world. Since all objects in this world follows laws of physics, applying physics to motion generation is natural. Researches on physically based modeling started from motions of rigid bodies. Later, it moved to motions of articulated bodies and deformable objects and other computational models

2.1.1 Collision and Contact

Motions of collision and contact frequently occur in computer animation and VR. Motions of collision and contact can occur between rigid bodies, articulated bodies and deformable objects. Hahn and Moore introduced impact dynamics for collisions of rigid bodies [Hahn88][Moor88]. Before the introduction of impact dynamics to collision handling, the penalty method was commonly used. Since the penalty method requires small time steps, it demands heavy computations. In contrast, impact dynamics changes velocities of colliding objects immediately, and thus, larger time steps are applicable.

Constrained equations should be considered for collisions of articulated bodies [Moor88]. These equations are similar to equations of constrained dynamics. Mirtich introduced a formulation for articulated bodies based on reduced coordinate method [Mirt96]. This method is usually faster and more stable than Moore's simultaneous equations. However, the derivation of equations for the reduced coordinate method is much harder.

Although the penalty method can handle contact constraint to some extent, it can not calculate exact contact forces. Baraff introduced an analytic method for calculation of contact forces by using quadratic programming and Danzig's algorithm [Bara89][Bara93]. Mirtich used impulse forces to handle contact constraints [Mirt96]. Milenkovic introduced a position-based method to handle the contact situation of large number of objects. Although his method is stable, orientations of objects can not be handled properly [Mile96]

2.1.2 Articulated Objects

Since articulated body animation is one of the major topics in computer animation, there has been much research devoted to it. They can be classified into two categories: *kinematics-based methods* and *dynamics-based methods*. Both have their advantages and disadvantages.

Kinematics-based methods are relatively easy to implement and good for interactively controlling the motions. However, since they involve positions, orientations, and velocities, it is difficult to apply physical laws involving accelerations. Inverse kinematics method is one of kinematics-based method for articulated body animation. Girard and Maciejewski applied an inverse kinematics technique for motions of running and walking humans [Gira85]. Badler *et al.* developed an inverse kinematics-based algorithm for solving multiple constraints concurrently, and applied it for articulated bodies [Badl87]. Currently, several inverse kinematics systems are available and some of them achieved interactive speeds [Welm93]. Kinematics methods and inverse kinematics methods can be used to generate realistic active motions of articulated bodies. However, it is difficult to incorporate external and internal forces to generate realistic passive motions.

In the case of dynamics-based methods, the constrained dynamics method is widely used for articulated body animation. The constrained dynamics method uses a system of equations, which consists of equations of motions and constraint equations. The systems of equations are usually too complex to be solved efficiently, and many works are focused on the effective way of solving these systems of equations. Among them, Armstrong and Green presented a recursive formation for the constrained dynamics method and introduced a linear time algorithm for constrained dynamics equations of articulated bodies [Arms85]. Presently, the constrained dynamics methods are usually solved through one of the two numerical techniques: coordinate reduction technique and Lagrange multiplier technique. Currently a linear time solution for Lagrange multiplier technique is available [Bara96].

Isaacs and Cohen introduced the inverse dynamics method as a way of controlling the motions of articulated bodies [Isaa87]. In this method, inverse forces are calculated to satisfy user-specified accelerations. Westenhofer and Hahn presented a motion control system that integrates kinematics-based controls into a constrained dynamics system [West96]. Constrained dynamics method and inverse dynamics method are sufficient to generate realistic and physically correct motions of articulated bodies. However, solving the systems of constrained dynamics equations is hard to perform interactively, even though there are theoretically linear-time solutions. Thus the dynamics-based method is not widely used for real-time applications such as virtual reality applications and computer games.

2.1.3 Deformable Objects

Generating motions of deformable objects is one of major topics in physically based modeling. Power of physically based modeling is well exposed in animating motions of deformable objects. Soft objects, cloth and water are examples of deformable objects.

One of major issues in deformable objects is calculating motions with volume preserving constraint. Terzopoulos implemented motions of soft objects by employing elasticity theory [Terz87]. Platt and Barr used augmented Lagrangian constraints to animate soft objects [Plat88]. Pentland introduced modal analysis for fast calculation of deformable objects [Pent89]. Recently, Desbrun and Gascuel animated inelastic objects such as clay by using implicit surfaces [Desb95].

Motions of cloth are important in computer animation and computer games. Breen employed particles to simulate motions of woven cloth [Bree94]. Baraff used implicit integration method to have larger time step [Bara98].

The issue of water animation is how to simplify the calculation of Navier-Strokes equations. Kass used shallow water assumption to get fast and stable motions of water [Kass90]. Witting applied fluid dynamics to motions of traditional cell animation environment [Witt99].

2.2 Procedural Method

The Procedural method has been used to represent texture and geometries of objects and motions. In computer science, the adjective procedural is used to distinguish entities that are described by program codes rather than by data structures [Eber94]. Characteristics of objects or motions from procedural method is that it is synthetically, generated from a program or a model. Procedural techniques have been used for many years in computer graphics to produce realistic textures (marble, wood, stone, wallpaper, bricks, etc.) and shapes of objects or motions (water, smoke, steam, fire, etc.). They are a cost-effective alternative to physically-based simulation.

The Procedural method is easy to implement and can be easily parameterized. The main disadvantage of procedural method is that the result is not predictable and it does not have concrete theoretical background.

Perlin introduced a way of generating 3D textures by a procedural method [Perl85]. Fournier modeled geometries of mountains by a procedural method which uses stochastic functions. Fournier and Peaches both successfully expressed ocean waves through combining simple trigonometric equations rather than using fluid dynamics formulations [Peac86][Four86]. Weil also succeeded in presenting complex shapes of cloth objects using simple catenary functions [Weil86].

Currently, these procedural methods are being re-visited due to their simplicity and their visually plausible results. Milenkovic introduced a position-based formation for non-articulated objects [Mile96]. Using this formulation, he demonstrated that small sphere particles contained in an hour-glass shape can be animated in a way similar to traditional constrained dynamics simulations. Gascuel introduced the displacement constraint to solve constraints of articulated objects quickly. The idea of separating the constraint solving part from equation of motion is similar to our procedural method [Gasc94]. However, their method is a kind of iterative method unlike our method. Barzel introduced a *fake* dynamics technique, which can be classified as a kind of procedural kinematic method [Barz96][Barz97]. This technique was successfully used to mimic the dynamics behavior of ropes and springs in the animation film “Toy Story”. To our knowledge, there has not been any procedural method for articulated bodies.

2.3 Sound Generation and Synchronization

Although sound is not a major area in computer graphics, it is very important in VR, multimedia and computer games. Sound has been researched independently in computer music, acoustics and signal processing areas. The problem of generating effective sound in VR, multimedia and computer games can be divided into three sub problems: sound modeling, sound synchronization, and sound rendering.

2.3.1 Sound Modeling

Large body of work exists in various domains that relate to sound. The issues in sound modeling have long been studied in the field of computer music. Functional composition of sounds has been explored in a number of computer music systems, including Music V, Csound, cmusic and Fugue [Math69][Verc86][Moor90][Dann91]. There have also been a number of approaches used for modeling sounds, such as Fourier synthesis, signal multiplication, filtering (subtractive synthesis) and frequency modulation. These approaches in computer music concentrated on representing and manipulating sound signals. In computer music, they do not address the issue of representing general sounds so that their parameters correspond to the phenomena that caused them. This is essential in order to synchronize the sounds to the motion.

Researches in acoustics and physics have concentrated on vibration of object and sound waves. Fletcher and Rossing formulated many sound equations of musical instruments [Flet91]. Although formulating vibration equations of homogenous material is possible, formulating the equation of non-homogeneous material is very difficult.

Hahn et al. introduced timbre tree, a data structure for sound representation [Hahn95]. Since it is expressed by flexible tree structure, adding new components of sound generation is easy like shade tree [Cook84].

2.3.2 Sound Synchronization with Motions

The primary consideration of sounds related with motions is the effective parameterization of sound models so that the parameters being generated from the motion can be mapped to them. Effective sounds can mean realism or effective encoding of information, an area of interest to data sonification. Parameterization and synchronization of sound has been investigated in relation to user interface, data sonification, and computer animation [Scal91][Gave93] [Scal91] [Lewi90][Taka92].

To synchronize motion and sound, in addition to timing and intensity information, geometry and contact parameters should match with sound parameters. Therefore, the sound equation should have parameters that correspond with motions. Timbre tree is designed for easy mapping of sound parameters and motion parameters [Hahn95].

2.3.3 Sound Rendering

Sound rendering refers to the progress of generating sound signals from their models within a particular environment, very much like the process of generating images from their geometric models [Tapi92]. Problems associate with sound rendering have been studied in the field of acoustics. The primary concentration of sound rendering is realistic and fast calculation of environmental effects.

Tapio and Hahn introduced a concept of sound rendering to computer graphics community [Tapi92]. Tsingos calculated room acoustics based on radiosity methods [Tsin97].

Chapter 3. Isaac System

Although physically based modeling generates very realistic motions, it requires heavy computations and its control is difficult. To solve these problems, we introduce the Isaac system that includes procedural motion generation features.

Constrained dynamics, an example of physically based modeling, can be used to generate realistic motions of articulated bodies. It is based on Newtonian physics, and thus has the power of producing physically correct motions. However, it uses constraint equations along with the equations of motions, and usually results in heavy computations.

In addition to constrained dynamics, elasticity model for deformable object and fluid dynamics for water are common models of physically based modeling. And these models also require heavy computations. Therefore, applying these computation models to real time applications is difficult for current personal computers.

Control is another problem of physically based modeling. Physical quantities of objects including forces and accelerations are often directly used to control motions of articulated bodies, even though it is not an intuitive way of control. For example, users find it difficult to place an object at a specific location by controlling the forces applied on it. Because of above problems, physically based modeling is not so widely used even though it is one of the best methods to generate realistic motions.

Deciding levels of simulation is not a simple problem in physically based modeling. For example, we can have different levels of simulation for motions of cars. The simplest level is that in which a car is modeled by a rigid body. This level only requires rigid body dynamics. Second level may be that in which a car consists of rigid chassis and wheels connected by springs. Finally, we can assume that a deformable chassis and wheels are connected by constraint dynamics and damped springs. Choosing a proper model in many different models for a specific purpose is not an obvious problem.

Sounds in computer animation and VR should be integrated with motions. Isaac system introduces sound generation methods that are closely related with motion parameters. Therefore, sound synchronization in Isaac system is implemented naturally.

This dissertation proposes the Isaac system that can solve problems of physically based modeling. The Isaac system basically consists of a dynamics part, procedural motion part and a sound generation part. The dynamics part consists of constraint dynamics and impact dynamics. Constrained dynamics is used for motions of articulated bodies and impact dynamics is used for basic collision reaction. In the Isaac system, users can design new constraints by combining basic constraints provided in Isaac system. It allows users to design constraints easier and more flexible.

The Isaac system has procedural methods for motions and sounds to speed up simulation time and to allow easy control. Procedural method starts with equations from dynamics and try to divide the equations according to meanings. We formulate new procedural equations by simplifications and amplification of the divided parts of the equations.

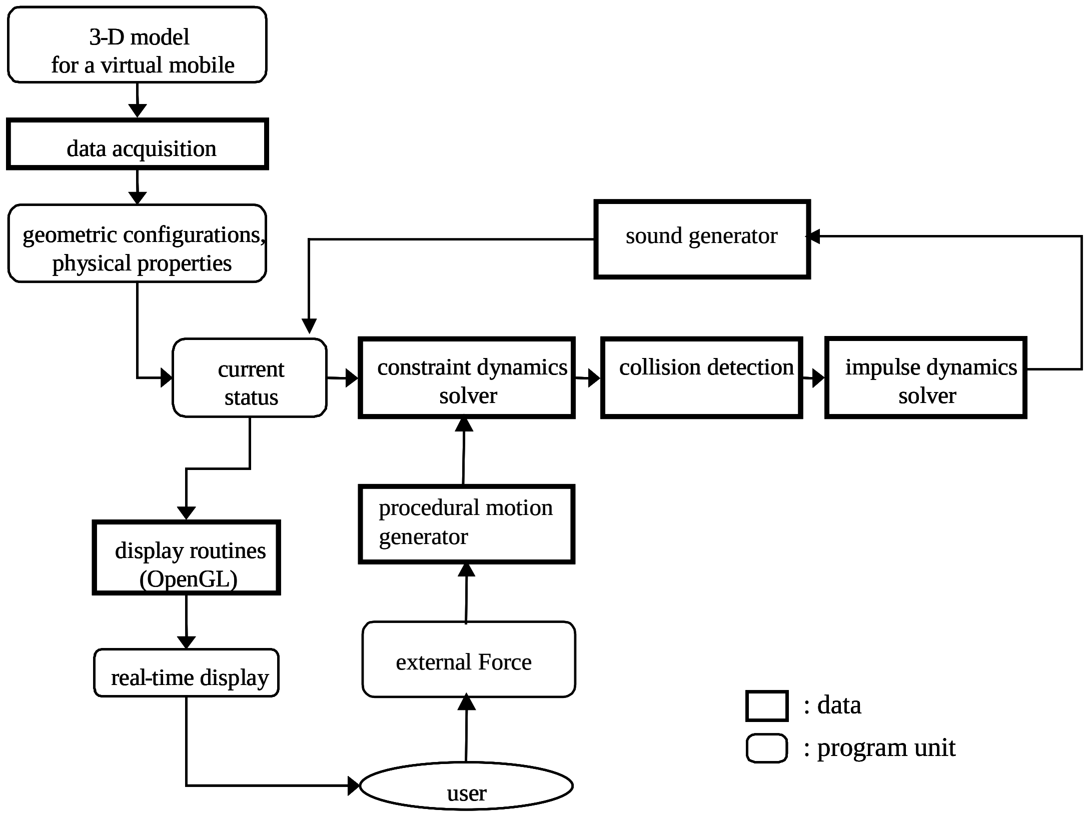

3-D model for a virtual mobile
data acquisition
geometric configurations, physical properties
sound generator
current status
constraint dynamics solver
collision detection
impulse dynamics solver
procedural motion generator
display routines (OpenGL)
external Force
real-time display
user
: data
: program unit

Figure 3.1 Block diagram of Isaac system

The sound generation part generates sounds according to physics analysis and procedural methods. Sound generation receives inputs from collision detection and reaction routines to synchronize with motions. These sound generation methods are closely related with parameters of motion generations. Therefore, sound synchronizations in the system occur easily

3.1 Overview of Isaac System

Isaac system consists of constrained dynamics solver, procedural motion generator, impulse dynamics solver, collision detection and sound generator, as shown in Figure 3.1. The user defines rigid bodies with their parameters (mass, moment of inertia, and friction coefficient), constraints, and external forces. With these inputs, this system generates motions based on dynamics and procedural motion generator. The user provides some of these parameters directly, while some parameters are calculated indirectly by the system. Possible external forces include wind, gravity and user interaction.

The constraint dynamics solver takes user-specified constraint description including dynamic constraint, velocity constraint and spring forces, and constructs a Jacobian matrix and uses it for constraint force calculation. Procedural motion generator takes the user-specified constraint description and solves the constraints and generates motions according to procedural methods. User can choose either constraint solver or procedural motion generator.

Sound generation part has sound producing modules based on physics and procedural method. It generates sounds according to the physical parameters and information from collision detection and reaction.

Isaac system was written in an object-oriented environment. Rigid body objects are inherited from kinematic objects by adding dynamics parameters. Therefore, every rigid body in this system can also be controlled by kinematic controls such as key frame. Camera and lighting objects are integrated into this system. Camera and lighting definition, rigid body definition, constraint

```
 // Simulation starting time and finish time
Starttime = 0.0;
Finishtime = 10.0;
 // Gravity
setGravity (Vector3 (0, -9.8, 0));

 // Camera and lighting definition
Camera cm (camera_pos, camera_interest, camera_up);
distantLight dl (Point(-1.0,-1.0,-1.0), Point(0,0,0), 0.7);

 // Rigid body definition
 // Geometry is defined at class definition
 Cube  cube(mass,MI,density, friction_coefficient, position,
            orientation);
 // applying wind force to the cube object
 cube.wind_force();

 // joint constraint definition
 Nail_Constraint nail (cb, Point(0,1,0), Point(0,1,0), 0.0,
                   Finishtime);
 simulate();
```

Figure 3.2. A sample code of the system that produces pendulum simulation for 10 seconds.

description and external forces are basic user inputs of the system. The system produces Open Inventor® or DirectX® output for real time interactions and RenderMan® RIB file for high quality rendering.

3.2 Constrained Dynamics

Although dynamics requires heavy computations, dynamics has been a major tool for generating realistic motions. Since the real world follows the law of dynamics, dynamics provides great realism. Constrained dynamics is a branch of dynamics. It restricts the motion of objects under dynamics environment. Constrained dynamics is widely used for articulated bodies and control of objects. This section explains constrained dynamics that is an important part of Isaac system.

Since constrained dynamics allows us to construct constraints for objects under Newtonian dynamics, it has been a useful tool for computer animation. The mechanics and Robotics community have researched constrained dynamics for a long time and computer graphics has employed their studies. Many realistic animations were done by constrained dynamics.

Constrained dynamics calculates constraint forces that act to satisfy user-defined constraints. There are many different names for the constrained dynamics formulations but all fall into two categories: coordinate reduction method and Lagrange multiplier method. The coordinate reduction method parameterizes maximal coordinates with generalized coordinates. Lagrange multiplier methods express the system by using simultaneous equations. Each method has its own pros and cons [Bara96].

If maximal coordinates are much bigger than generalized coordinates, the coordinate reduction method may be more efficient and tolerant of numerical drifting than the Lagrange multiplier method. For human-like structure of articulated bodies, linear-time algorithms of coordinate reduction methods are well known. However, formulation of general structure of articulated bodies by coordinate reduction method is not always easy.

3.2.1 Lagrange Multiplier

The Lagrange multiplier methods express constraints of objects by using simultaneous equations. Therefore, the Lagrange multiplier method bypasses the difficulty of parameterization of constraints. The Lagrange multiplier method has more software modularity and ability to handle velocity constraints. For globally deformable bodies, the Lagrange multiplier method has the advantage over the coordinate reduction method, since parameterization of the constraints that is required for the coordinate reduction method becomes very difficult for deformable bodies. Due to this modularity, an arbitrary set of constraints can be combined without difficult parameterization. Therefore, we employ the Lagrange Multiplier method for the basic constrained dynamics tool in Isaac system.

Lagrange Multiplier method has numerical disadvantages over coordinate reduction method. Since Lagrange multiplier method is prone to accumulate numerical errors, constraint stabilization methods that inhibit accumulation of numerical drifting error are often used along with the Lagrange multiplier method [Baum72].

The formulation of the Lagrange multiplier method starts with generalized Newton's law,

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\,\mathbf{F}, \tag{3.1}$$

where $\mathbf{x}$ is the generalized state vector of position, $\mathbf{M}^{-1}$ is the inverse of the mass matrix, and $\mathbf{F}$ is force vector.

In order to satisfy a positional constraint under dynamics environment, we must create an equation which express constraints. Let **C(x)** be a user-specified constraint, and **C(x) = 0** mean that the constraint is being satisfied. Since we are concerned with a dynamics environment, in addition to **C(x)** being zero, its first derivative, $\dot{\mathbf{C}}(\mathbf{x})$ and its second derivative, $\ddot{\mathbf{C}}(\mathbf{x})$ should be zero. Therefore, our goal is to find the constraint forces for which $\ddot{\mathbf{C}}(\mathbf{x})$ evaluates to zero.

First derivative of $\mathbf{C}(\mathbf{x})$ is

$$\dot{\mathbf{C}}(\mathbf{x}) = \frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}}$$

Second derivative of $\mathbf{C}(\mathbf{x})$ is $\ddot{\mathbf{C}}(\mathbf{x}) = \dot{\mathbf{J}}\dot{\mathbf{x}} + \mathbf{J}\ddot{\mathbf{x}}$, where $\mathbf{J} = \frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}}$. The matrix **J** is called Jacobian of **C.** Now we add unknown constraint force $\mathbf{F_c}$, and substitute equation (3.1)

$$\ddot{\mathbf{C}}(\mathbf{x}) = \dot{\mathbf{J}}\dot{\mathbf{x}} + \mathbf{J}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{F_c}) = \mathbf{0}$$

Rearranging terms,

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{F_c} = -\dot{\mathbf{J}}\dot{\mathbf{x}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{F}$$

By applying the principal of virtual work [Gold80], we can replace $\mathbf{F}_c$ with $\mathbf{J}^{\mathbf{T}}\boldsymbol{\lambda}$, where $\boldsymbol{\lambda}$ is a vector, called Lagrange multiplier;

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathbf{T}}\boldsymbol{\lambda} = -\dot{\mathbf{J}}\dot{\mathbf{x}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{F}$$

Now, all values are known except $\boldsymbol{\lambda}$. $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathbf{T}}$ is always a square matrix whose dimension is the same as the dimension of **C.** Many linear system solvers can be used to solve this equation. The conjugate gradient method has been popular because of it stability and speed. Baraff introduced a linear-time solution for this equation by using sparse matrix technique [Bara96]. Currently, this system uses Gaussian solver and Baraff's linear-time algorithm.

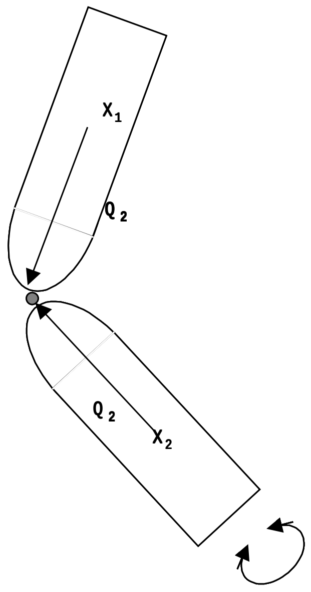


Figure 3. 3 Point-to-point constraint

The numerical drifting problem is one disadvantage of the Lagrange multiplier method. To solve this problem, we employ Baumgarte's stabilization method [Baum72]. $\ddot{\mathbf{C}} = -\mathbf{k}_s\mathbf{C} - \mathbf{k}_d\dot{\mathbf{C}}$ is used instead of $\ddot{\mathbf{C}} = \mathbf{0}$, Where $\mathbf{k_s}$ and $\mathbf{k_d}$ are coefficients. By choosing $\mathbf{k_s}$ and $\mathbf{k_d}$ to specify a critically damped system, the system can assembles parts of an articulated body automatically. The final equation of the motion, including the stabilization is:

$$\mathbf{JM^{-1}J^T\lambda} \;\; = -\dot{\mathbf{J}}\dot{\mathbf{x}} - \mathbf{JM^{-1}F} - \mathbf{k}_s\mathbf{C(x)} - \mathbf{k}_d\dot{\mathbf{C}}\mathbf{(x)}$$

3.2.2 Constraint Library

Isaac provides basic constraints as tools for designing and manipulating constraints. Therefore, the user can choose and combine constraints without having to specify their formulation. Due to the advantage of the Lagrange multiplier method, constraints can be combined without parameterization. The basic constraints in Isaac system are point-to-point, point-to-nail, orientation, point-on-plane, gear-to-gear and gear-to-weight constraints.

The constrains can be divided into one dimension and the constraints have time parameter. Therefore the user can also design his own constraints by combining these basic constraints. Dynamic constraints are calculated by applying second derivates of positional constraints or first derivates of velocity constraints.

Point-to-point constraint

Point-to-point constraint connects two bodies by the spherical joint. This is a common

joint constraint to construct articulated bodies. The equation of positional constraint is:

$$\mathbf{C} = \mathbf{Q_1} - \mathbf{Q_2} = \mathbf{R_1 P_1} + \mathbf{X_1} - \mathbf{R_2 P_2} - \mathbf{X_2},$$

where **R** is rotation matrix, **P** is point of constraint in body space and **X** is position of center of mass. First derivative of **C** is:

$$\dot{\mathbf{C}} = \omega_1 \times \mathbf{r_1} + \mathbf{v_1} - \omega_2 \times \mathbf{r_2} - \mathbf{v_2}.,$$

where **v** is linear velocity, **ω** is angular velocity and **r** is vector from center of mass to constraint point.

Second derivative of this equation is:

$$\ddot{\mathbf{C}} = \dot{\mathbf{v}}_1 + \dot{\omega}_1 \times \mathbf{r_1} + \omega_1 \times (\omega_1 \times \mathbf{r_1}) - \dot{\mathbf{v}}_2 - \dot{\omega}_2 \times \mathbf{r_2} - \omega_2 \times (\omega_2 \times \mathbf{r_2}),$$

Point-to-nail constraint

Point-to-nail is a spherical joint of rigid body to a fixed point in space. This constraint works

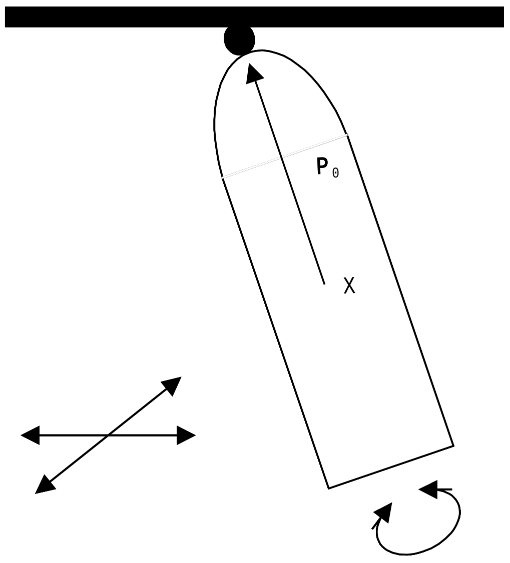


Figure 3.4 Point-to-nail constraint

like a nail in the space. Point-on-path constraint that a rigid body follows by given path can be represented by moving the nail position. The formulation of this constraint is:

$$\mathbf{C} = \mathbf{X} - \mathbf{P} = \mathbf{RP_o} + \mathbf{X} - \mathbf{P} = \mathbf{0},$$

where R is rotation matrix, P_0 point of constraint in body space, X is position of center of mass and P is position of nail point in world space. The first derivative of the equation is:

$$\dot{\mathbf{C}} = \omega \times \mathbf{r} + \mathbf{v}$$

Second derivative of the constraint equation is:

$$\ddot{\mathbf{C}} = \dot{\mathbf{v}} + \dot{\omega} \times \mathbf{r} + \omega \times (\omega \times \mathbf{r})$$

where **v** is linear velocity, **ω** is angular velocity and **r** is vector from center of mass to constraint point.

Orientation constraint

Orientation constraint restricts one degree of freedom of orientation. Orientation constraints are widely used to restrict orientations of rigid bodies. The constraint equation of the orientation is formulated by dot product of tangent vector of an object and constraint vector, as shown in Figure 3.5. Constraint equation and its first derivative is:

$$\mathbf{C} = \mathbf{T}(t) \cdot \mathbf{N}$$
$$\dot{\mathbf{C}} = (\omega \times \mathbf{T}) \cdot \mathbf{N},$$

where **T** and **N** are tangent vector and constraint vector , respectively.

The second derivative of this equation is

$$\ddot{\mathbf{C}} = (\dot{\omega} \times \mathbf{T}) \cdot \mathbf{N} + (\omega \times (\omega \times \mathbf{T})) \cdot \mathbf{N} = \mathbf{0}\ .$$

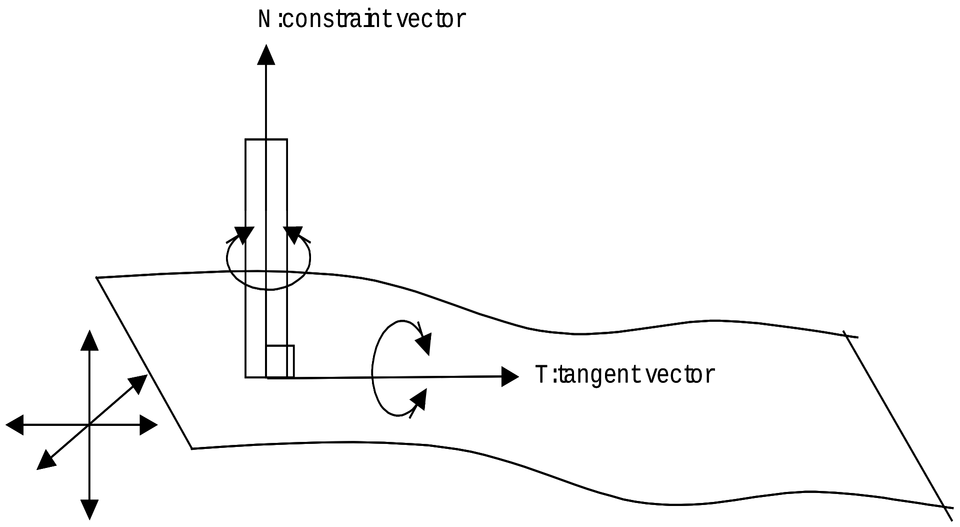


Figure 3.5 Orientation constraint

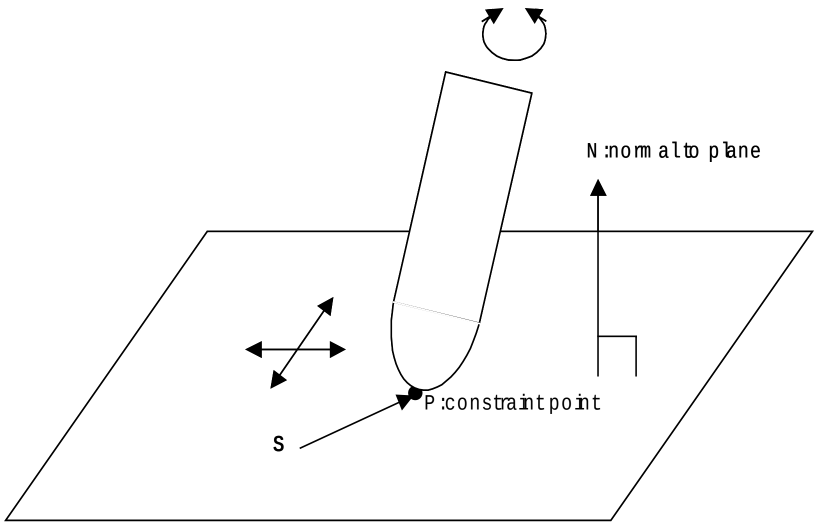


Figure 3.6 Point-on-plane constraint

Point-on-plane constraint

Point-on-plane constraint ensures that a point of rigid body remains on a given plane. Plugging the constraint point into the plane equation leads to the constraint equation. This constraint can be used for a simple contact force and line constraint that objects follow by given line. Combining two different point-on-plane constraints generate a line constraint. This shows design power of Isaac system.

Constraint equation for point-on-plane equation is:

$$\mathbf{C} = \mathbf{N} \cdot \mathbf{P} + \mathbf{d} = \mathbf{N} \cdot \mathbf{P} - \mathbf{N} \cdot \mathbf{S} = \mathbf{0},$$

where vector **N** and **d** construct given plane, and **P** is constraint point of the object, as shown in Figure 3.6. Second derivative is

$$\ddot{\mathbf{C}} = \ddot{\mathbf{N}}(\mathbf{P} - \mathbf{S}) + \mathbf{2}\dot{\mathbf{N}}(\dot{\mathbf{P}} - \dot{\mathbf{S}}) + \mathbf{N}(\ddot{\mathbf{P}} - \ddot{\mathbf{S}}) = \mathbf{0}.$$

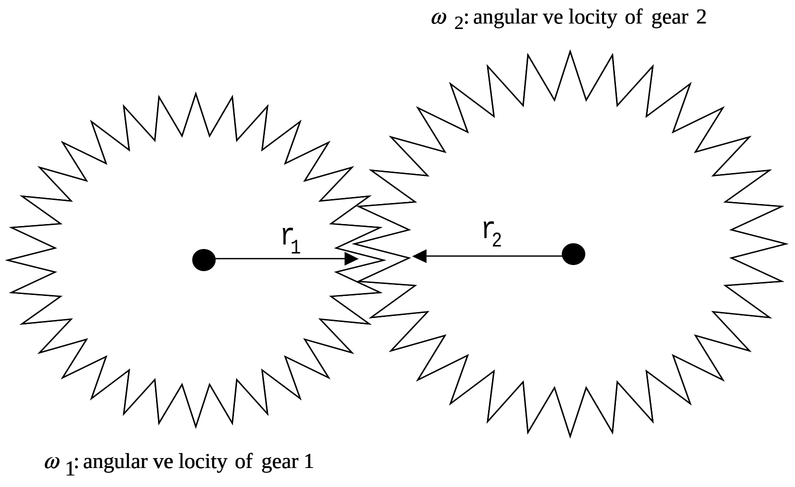

ω_2: angular ve locity of gear 2
r_1
r_2
ω_1: angular ve locity of gear 1

Figure 3.7 Gear-to-gear constraint

Gear-to-gear constraint

Gear-to-gear constraint in Isaac system was implemented by velocity constraint. Easy implementation and seamless combining of velocity constraint with dynamic constraints are advantages of Isaac system. Gear-to-gear constraint by velocity constraint is much faster than collision based constraint. Gear-to-gear constraints and gear-to-weight constraints are the basic examples of velocity constraints in Isaac system.

The angular velocities of connected gears are constrained by their radii. Velocity constraint and its derivative are

$$\dot{\mathbf{C}} = \mathbf{r}_1\ \omega_1 + \mathbf{r}_2\ \omega_2 = \mathbf{0}$$
$$\ddot{\mathbf{C}} = \mathbf{r}_1\ \dot{\omega}_1 + \mathbf{r}_2\ \dot{\omega}_2 = \mathbf{0},$$

where **r** and **ω** are radius and angular velocity of the gear, respectively.

Gear-to-weight constraint

Gear-to-weight constraint, shown in Figure 3.8, can be formulated by matching angular velocity of a gear and linear velocity of weight. The constraint represents connection between gear

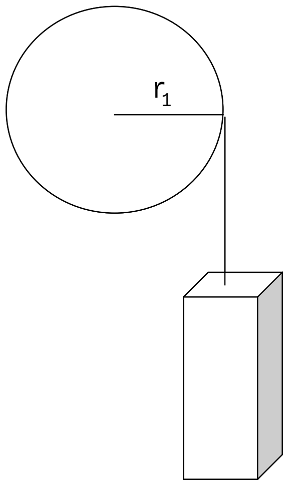


Figure 3.8 Gear-to-weight constraint

and weight. The constraint equation and its derivative are:

$$\dot{\mathbf{C}} = \omega_1 \times \mathbf{r}_1 - \mathbf{v}_2 = \mathbf{0}$$
$$\ddot{\mathbf{C}} = \omega_1 \times \mathbf{r}_1 - \dot{\mathbf{v}}_2 = \mathbf{0},$$

where **r** is radius of gear, **v** is angular velocity of the weight.

3.3 Constraint Combining and Breaking

To design new constraints, users should formulate positional or velocity constraints and

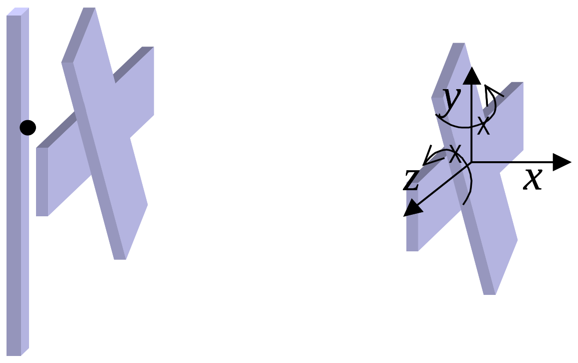


Figure 3.9 Constructing windmill-like constraint

differentiate the equations to get dynamic constraints. Usually this process is not easy for the end user. In Isaac system, basic constraints are implemented in the constraint library of the system. Users construct other constraints by combining these basic constraints in constraint library. In addition to basic constraints, Isaac system has *element constraints* that is one DOF constraint. The

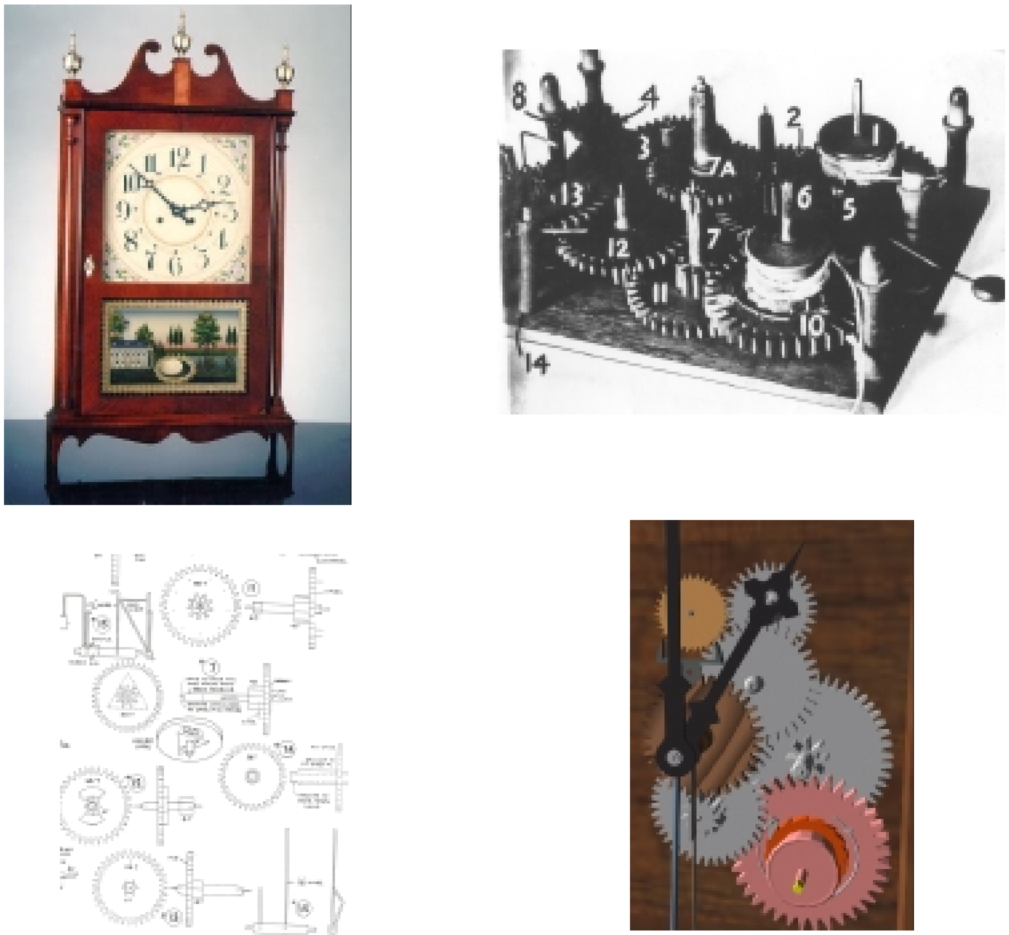


Figure 3.10 Terry Eli's clock

element constraint restricts one DOF among 6DOF of rigid body. This system can add or subtract the element constraint to the basic constraints. The element constraints are used to modify the basic constraints. The basic constraint can also be combined with each other for new constraints. The followings show examples of constraint combining. Windmill-like joint constraint, shown in Figure 3.9, results from combining a point-to-point constraint with two orientation constraints. In simulation of Terry Eli's clock[1], the retainer should be represented by line constraint. Because line equation of 3D can only be represented by parametric equation, formulating positional constraint equation of line constraint is not easy in Lagrange Multiplier method. In Isaac system, line constraint can be designed by combining two point-on-plane constraints. Figure 3.10 shows part of Terry Eli's clock.

Adding time parameters to the constraints contributes to an easy and flexible designing of time varying constraints. Since this approach allows the constraint to be parameterized, attaching the time parameters to each element constraint makes controllable time-varying effects. This time-varying constraint can be a useful tool to design constraint breaking and smooth transition from one constraint to another constraint.

[1] Terry Eli's clock was designed early 19th and it was first clock that was produced by mass production system in America.

```
//Transient to different axis rotation constraint

Pt-NailConstraint (wheel,Point(0,0,0),Point(0,0,0),0.0, 5.0);

OrinetationConstraint (wheel,Vector1(0,1,0) ,Vector2( 0,0,1), 0.0, 2.5)

OrinetationConstraint (wheel,Vector1(1,0,0) ,Vector2(0,0,1) 0.0, 2.5)

OrinetationConstraint (wheel ,Vector1( 1,0,0),Vector2( 0,1,0), 2.5, 5.0)

OrinetationConstraint (wheel,Vector1( 0,0,1),Vector2(0,1,0), 2.5, 5.0)
```

Figure 3.11 A sample code of constraint transition

Each constraint has time parameters which indicate starting and ending times. The constraints are alive during the indicated time. The default parameter is the starting and ending time of the simulation.

Constraint transition between two constraints like morphing can be achieved by controlling time parameters and Isaac system's self-assembling ability. Issac's self assembling ability is achieved by choosing critically damped spring in the stabilization routine. If users want to change y-axis orientation constraint to x-axis orientation constraint, users has to set the starting time and ending time of each constraint. Figure 3.11 shows sample codes for constraint transient.

Time-varying constraints are handled by managing the active constraint list which holds the currently valid constraints. Declaring initial constraints generates all constraint lists that keep all constraints regardless of the validity about current time. Checking the current time with the starting and ending time parameters generates the active constraint list which is actually used for current constraint-force calculation. More complicated time varying constraints can be generated by assigning functions of time to the parameters.

3.4 Constraint Impact

Modeling collision by force requires very small time step to avoid any penetration which is not desirable for computer animation and VR. Instead of using force for collision, we employ impact formulation for collision between rigid bodies. Since impact changes velocity instantaneously, we can keep large time step without penetration. If a constrained object collides with rigid bodies, the impulse violates dynamics constraints of the articulated object since impulse changes velocities without considering the constraints. For articulated objects, we need impulse calculations that consider the constraint condition. This system sends constraint information to get the constraint impulse that does not violate the constraints.

When simulating the motion of articulated bodies with constraint dynamics, collisions between its components will necessarily happen. Although there are a lot of general collision detection methods [Hubb96][Mirt96], collision detections in Isaac system can be achieved more efficiently through analyzing the characteristics of the articulated bodies. First, some pairs of the components can not collide with each other in any case. A preprocessing step detects all such pairs of components to finally be excluded from the collision detection process. Additionally, some components are simple geometric shapes such as cylinders and spheres. Thus, we can use cylinder-to-cylinder, cylinder-to-sphere and sphere-to-sphere collision detection methods, which are much faster than usual polyhedron-to-polyhedron collision detection methods.

After detecting collisions, we use the impulse-based collision response method [Hahn88][Moor88]. Since this method can calculate the new velocities instantaneously, it is suitable

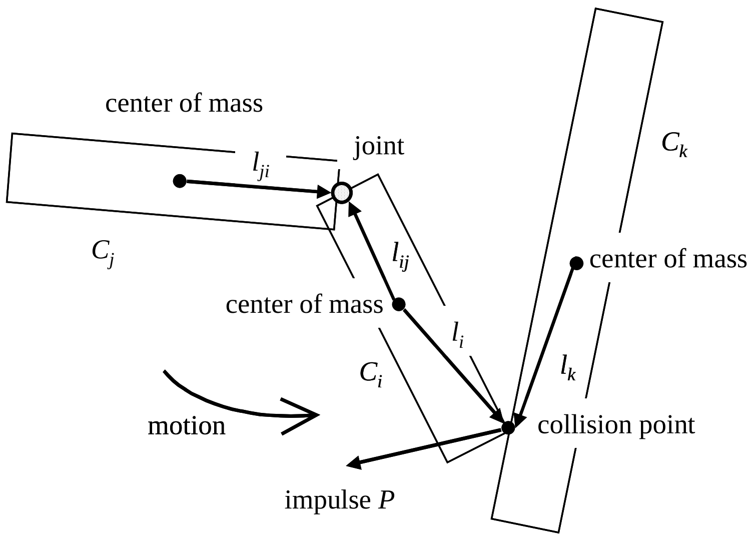


Figure 3.12 Impulse-based collision response

for real-time applications. The penalty method [Moor88], which is also widely used in the collision response, requires small time steps for accurate simulation, and thus it may cause to slow down a real-time application.

In the case of articulated bodies, the collision response method should cooperate with the constraint dynamics model. Thus, the constraints at the joints and frictions at the collision points should also be handled during the collision response. Moore formulated the impulse equations for articulated figures, and his equation can be used for joint constraints [Moor88].

Letting the components of an articulated body be O_i, $1 \le i \le n$, the mass and inertia tensor of O_i is denoted as m_i and $\mathbf{I}_i$, respectively. Due to the collision, the linear and angular velocity of O_i may be changed. Let $\mathbf{v}_i$ and $\boldsymbol{\omega}_i$ be the linear and angular velocity of O_i with respect to the center of mass of O_i, before the collision. The impulse-based collision response method aims to calculate the linear velocity $\overline{\mathbf{v}}_i$ and the angular velocity $\overline{\omega_i}$ of O_i after the collision.

The impulse-based collision response method starts from the law of momentum preservation. Since the change of momentum before and after the collision equals to the sum of impulses at the time of collision, the impulse equations for O_i can be expressed as follows:

$$m_i(\overline{\mathbf{v}}_i - \mathbf{v}_i) = \mathbf{P} + \underset{j}{\quad}\mathbf{P}_{ij}$$

and

$$\mathbf{I}_i(\overline{\omega_i} - \omega_i) = \mathbf{l}_i \times \mathbf{P} + \underset{j}{\quad}\mathbf{l}_{ij} \times \mathbf{P}_{ij} \ ,$$

where **P** is the impulse applied to O_i and $\mathbf{P}_{ij}$ is the attachment impulse on O_i due to O_j, which is connected to O_i using a joint constraint. When O_i and O_j are not directly connected to each other, $\mathbf{P}_{ij}$ is a null vector. Notice that **P** can be zero for non-colliding components. The vectors $\mathbf{l}_i$ and $\mathbf{l}_{ij}$ are the distance vectors from the center of mass of O_i to the collision point and to the joint connecting O_i and O_j, respectively, as shown in Figure 3.12.

Joint constraints also give additional equations. When a spherical joint connects O_i and O_j, their relative velocity at the contact point should equal each other:

$$\overline{\mathbf{v}}_i + \overline{\omega_i} \times \mathbf{l}_{ij} = \overline{\mathbf{v}}_j + \overline{\omega_j} \times \mathbf{l}_{ji} \, .$$

In the case of nail joints, a point of the component O_k has a fixed position. Thus, the linear velocity of the nailed point is zero:

$$\overline{\mathbf{v}}_k + \overline{\omega_k} \times \mathbf{l}_{kk} = 0 \, ,$$

where $\mathbf{l}_{kk}$ is the vector from the center of mass of O_k to the nailed point.

In this way, we can applied Moore's impulse-based collision response formulation to articulated bodies. Additionally, we improve his formulation to handle the frictions during collisions.

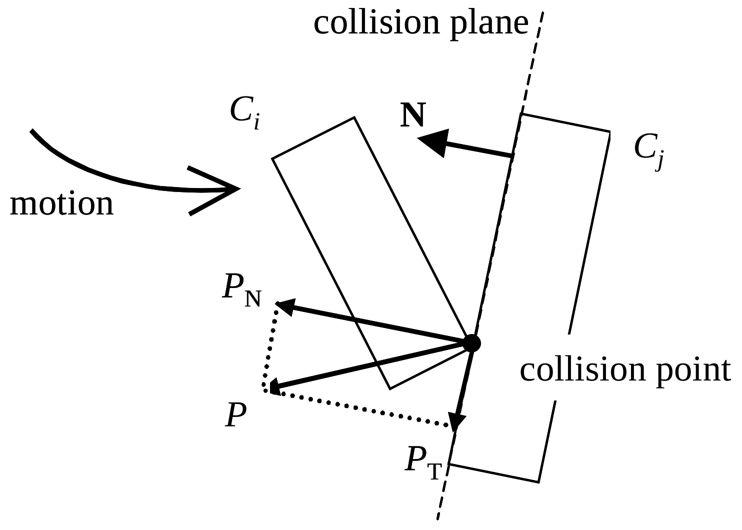


Figure 3.13 Collision plane

Although Moore also extended his formulation to the friction cases, it requires solving the whole systems of linear equations repeatedly to determine the friction status of each collision point. We improve this weak point of Moore's formulation through combining with Mirtich's conditional equation for friction [Mirt96].

When a point of O_i is colliding with a face of O_j, the plane containing that face is defined as the collision plane, as shown in Figure 3.13. With respect to the normal vector **N** of the collision plane, the impulse **P** for O_i can be divided into two components: the normal component $\mathbf{P}_\mathrm{N}$ and the tangential component $\mathbf{P}_\mathrm{T} = \mathbf{P} - \mathbf{P}_\mathrm{N}$. The state of friction can be classified into two cases: sticking case and sliding case. From the viewpoint of impulse, the sticking case means there is no slip at the collision point, which satisfies the condition of $|\mathbf{P}_\mathrm{T}| \leq \mu\, |\mathbf{P}_\mathrm{N}|$ with the friction coefficient μ. When

$|\mathbf{P}_T| > \mu\,|\mathbf{P}_N|$, the collision point will slip along the collision plane and it is called the sliding case.

Since the sliding and sticking cases result in different equations, it is important to identify whether a collision point is sticking or sliding. For sticking cases, the collision point does not move along the collision plane. In contrast, the friction force will act on the sliding collision points.

In Moore's formulation, it is impossible to decide whether a collision point is sliding or sticking without calculating the impulse. Mirtich introduces the decision equation to identify a collision state before calculating impulses [Mirt96]. We use the decision equation to speed up the impulse calculation. The friction at a collision point is sliding when it satisfies the following condition:

$$\left(\frac{1}{k_{13}}\right)^2 + \left(\frac{1}{k_{23}}\right)^2 > \mu^2 \left(\frac{1}{k_{33}}\right)^2,$$

where the 3-by-3 matrix $\mathbf{K}$ = [k_{ij}] equals to $(1/m_i + 1/m_j)\mathbf{E} - (l_i \times \mathbf{I}_i^{-1} \times l_i + l_j \times \mathbf{I}_j^{-1} \times l_j)$ with the 3-by-3 identity matrix $\mathbf{E}$.

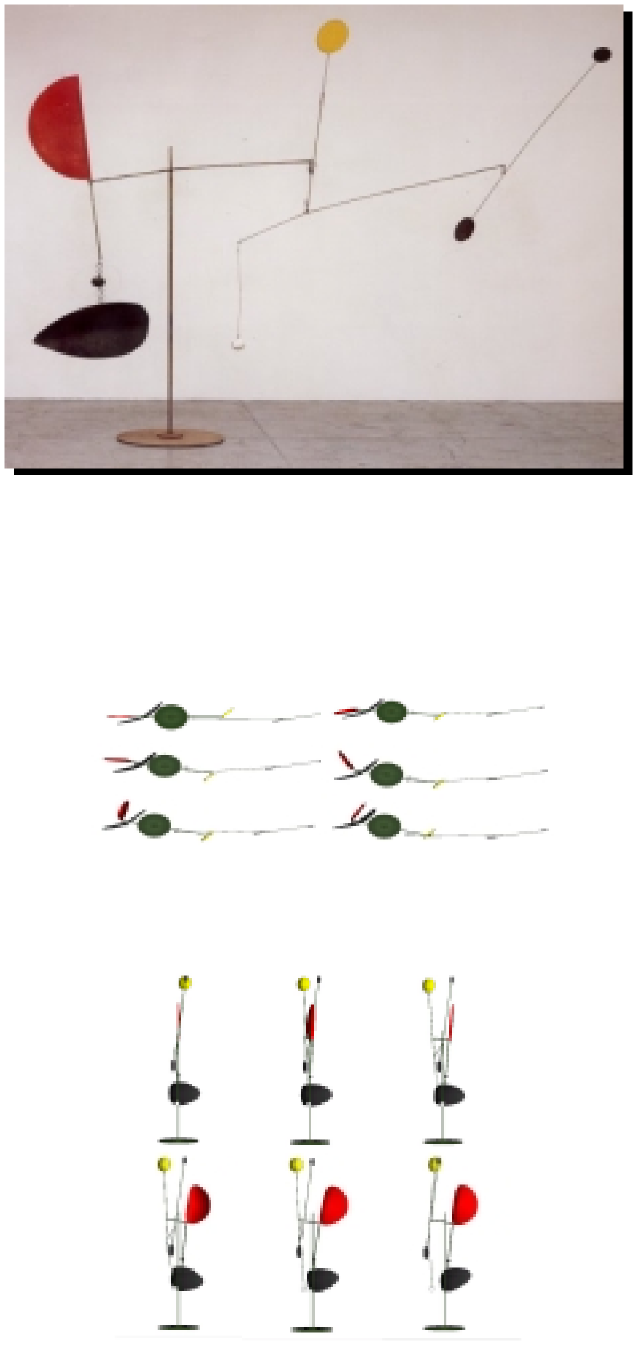


Figure 3.14 Collision motions of *Steel Fish*.

3.5. Procedural Motions based on Physically based Modeling

Physically-based modeling methods have their own pros and cons. For example, constrained dynamics method can be used to generate realistic motions of articulated bodies. It is based on Newtonian physics, and thus has the power of producing physically correct motions. However, it uses constraint equations along with the equations of motions, and usually results in heavy computations. Additionally, physical quantities of objects including forces and accelerations are often directly used to control motions of articulated bodies, even though it is not an intuitive way of control. For example, users find it difficult to place an object at a specific location by controlling the forces applied on it. Thus, currently, the constrained dynamics method is not so widely used even though it is one of the best methods for realistic articulated body animation [Barz97].

An alternative to physically-based modeling is the paradigm of *procedural methods* [Watt92]. In 1980's, some research have focused on mimicking physical phenomena rather than strictly simulating physical laws [Four82][Peac86][Weil86]. Although these approaches were motivated basically by the lack of sufficient computing power, they achieved visual plausibility and also provided easy control of complex phenomena. Even today, we still do not have sufficient computing power to simulate physically-based models of complicated phenmena in real time. In virtual reality environments and computer games, for example, it is a requirement to display the motion of objects in real time, even at the expense of displaying physically incorrect motions. This chapter introduces procedural methods for constrained dynamics, non-holonomic dynamics and constrained impact. These calculations require heavy computations in physically-based modeling.

3.5.1 Procedural Method to Solving Constraints of Articulated Bodies

We present a method to solve constraints procedurally to interactively calculate motions of articulated bodies. Our objective is not necessarily to generate physically correct motions but *visually plausible motions*. Although it is not directly derived from Newtonian dynamics, our method provides an efficient and numerically stable way of generating visually plausible motions. Its calculation procedure is based on the positions and orientations of the objects rather than dynamics properties such as forces and accelerations. Hence it additionally provides an easy way of controlling the motions via specifying the desired positions and orientations. Figure 3.15 shows an example of interactive control of an articulated figure. It is especially suitable for presenting dragging effects, which often occur when the user moves a selected portion of the articulated body.

In next sections, we present a detailed description of our procedural method and how to apply it to articulated bodies and various joint constraints. Some examples of image sequences generated by this procedural method are also explained.

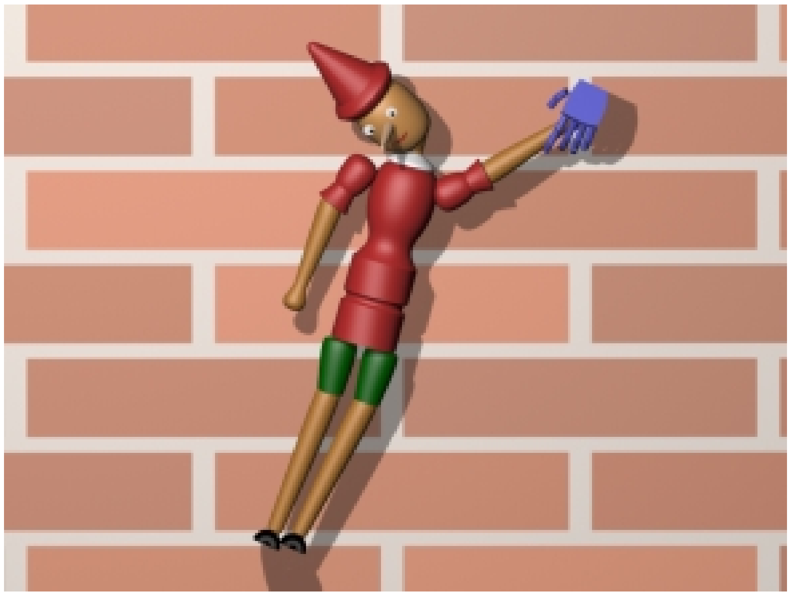


Figure 3.15 Interactive control of articulated body.

3.5.1.1 Overview

Generation of an articulated body motion means deciding the position and orientation of each object in the articulated body for each time instant. From the dynamics point of view, the current geometric configurations (positions, orientations, etc.) and physical parameters (velocities, accelerations, etc.) are calculated from the configurations of the previous time instant. This calculation process has two requirements:

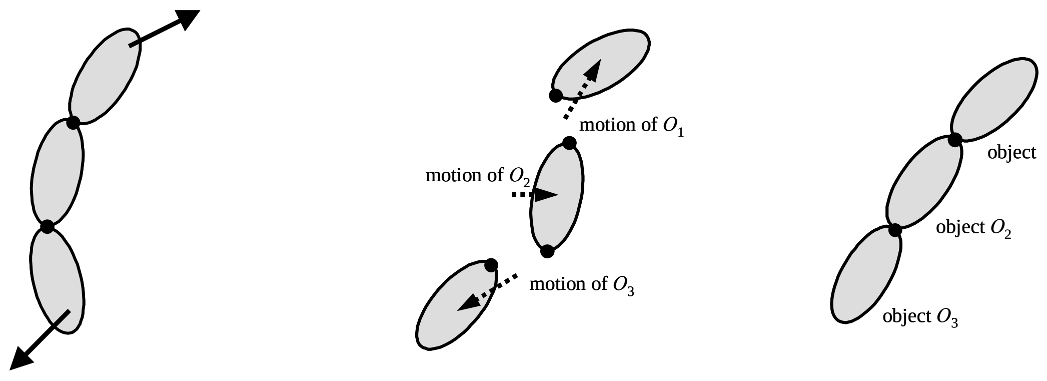
(1) The motion of each object should be generated according to the equations of motions, in which external forces are involved.

(2) The final geometric configurations of objects should satisfy constraints due to the joints of

the articulated body.

The constrained dynamics method starts from a system of equations, which explicitly express the above requirements. Equations of motions and constraint equations are usually integrated into a system of equations, which is usually solved by a relatively complex numerical method.

In contrast, the basic idea of our procedural method is separating the whole process into two stages each of which concentrates on one of the two above requirements. In the *update stage*, positions and orientations of objects making up the articulated body are updated by solving the



(a) original configuration (b) update stage (c) adjustment sta

Figure 3.16 Overview of the procedural constraint solving method.

equations of motions. The animator can specify the position and orientation of an object explicitly, if desired. Notice that any constraint equation is not considered at this time, as shown in Figure 3.16.(b).

In the *adjustment stage*, we adjust the positions and orientations of each object to satisfy the constraints, as shown in Figure 3.16.(c). During this adjustment process, a procedural calculation of required transform for each object is used rather than the original constraint equations. Notice that our goal is the visual plausibility rather than physically correct motion, and thus the constraint equations are not solved explicitly.

In comparison with traditional constrained dynamics methods, our procedural method has two advantages:

(1) It is faster than any constrained dynamics methods since our adjustment equations make it possible to satisfy the constraints without considering complex physical properties such as accelerations and velocities.

(2) It can be integrated into a direct manipulation system in which an object is selected to change its position and orientation interactively, since our method solves the constraints based on position and orientation. In constrained dynamics, inverse dynamics is required to control position or orientation.

The update stage is straightforward. We can use any dynamics methods to update positions and orientations of objects, since the constraint equations are excluded during this update step. In our implementation, we use Euler's integration method for this purpose, mainly due to its simplicity.

The strictly physically-based modeling often includes friction forces and drag equation from the fluid dynamics. In our implementation, we add a damping term to approximate them. Letting $\mathbf{x}^i$, $\mathbf{v}^i$ and $\mathbf{a}^i$ be the position, velocity and acceleration of an object at the i-th time step, Euler integration of Newton's law is expressed as follows:

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \mathbf{v}^i\,(\Delta t) \tag{3.2}$$

and

$$\mathbf{v}^{i+1} = \mathbf{v}^i + \mathbf{a}^i\,(\Delta t), \tag{3.3}$$

where Δt is the time interval between animation frames.

Combining Equations (3.2) and (3.3), the position can be evaluated with the following single equation:

$$\mathbf{x}^{i+1} = \mathbf{x}^{i} + (\mathbf{x}^{i} - \mathbf{x}^{i-1}) + \mathbf{a}^{i} (\Delta t)^2.$$

Since $(\mathbf{x}^{i} - \mathbf{x}^{i-1})$ corresponds to the velocity, we multiply a damping constant to it. Thus, the final equation for $\mathbf{x}^{i+1}$ is:

$$\mathbf{x}^{i+1} = \mathbf{x}^{i} + k\,(\mathbf{x}^{i} - \mathbf{x}^{i-1}) + \mathbf{a}^{i} (\Delta t)^2,$$

where $k \in [0, 1]$ is the damping constant. By controlling the value of k, we can control the visual illusions of frictions and/or motions in fluid such as water. Details of the adjustment process will be explained in the following sections.

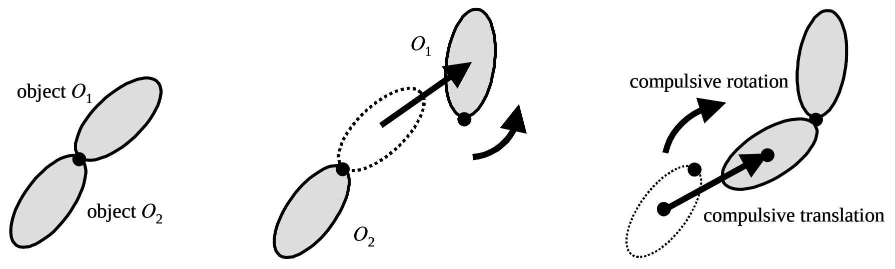
3.5.1.2 Fixed position solution for two-object articulated bodies

In the adjustment stage, positions and orientations of objects are adjusted to satisfy the joint constraints. We formulated this adjustment process to reflect the characteristics of constraint forces. In the case of constrained dynamics, constraint forces should satisfy the following two characteristics:

(1) The constraint forces applied to the objects connected by a joint have same magnitudes but opposite directions.

(2) Constraint forces should be workless.

Our adjustment process aims to mimic the constraint forces as much as possible. Especially, we hope to represent the motions of an articulated body when the user drags a point on the body.



(a) original configuration

(b) O_1 moves to a fixed position

(c) constraint solving

Figure 3.17. Compulsive translation and rotation.

In our approach, constraints are solved by translating and rotating the objects to satisfy the constraints. In the update stage, objects are moved due to the external forces and/or user inputs without considering any constraints. Thus the movements usually break the joint constraints of articulated bodies. The major role of the constraint solving is to decide the translational and rotational motions that satisfy the given constraints. We call these translations and rotations due to the constraints *compulsive translations* and *compulsive rotations*, respectively. *Compulsive motion* will be used to refer to both compulsive translation and compulsive rotation.

To formulate the equations of compulsive translation and rotation, we will start from the simplest case. Suppose that an articulated body has only two objects and the position and orientation of an object O_1 is f*ixed* after the update stage. This situation often occurs when the user drags O_1 to a specific location. Then another object O_2 should move closer to the dragged object, as

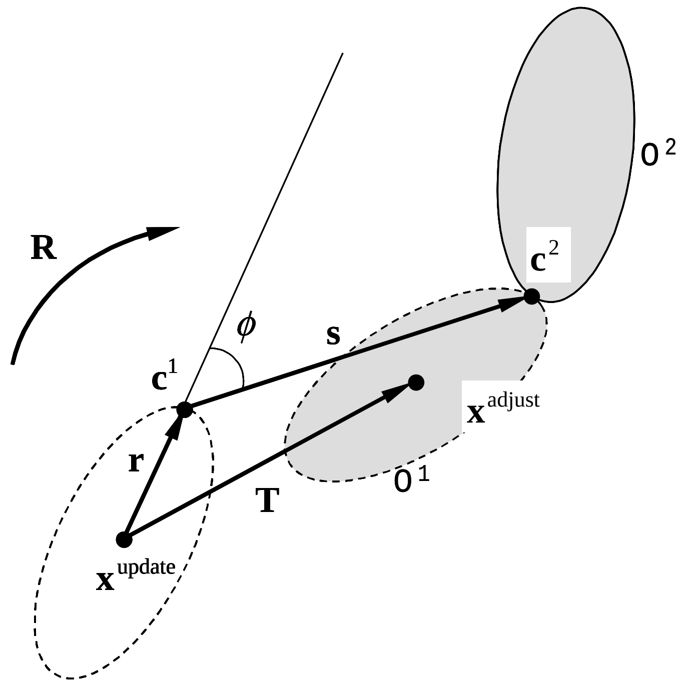


Figure 3.18 Solving the constraint through moving the object

shown in Figure 3.17

Let $\mathbf{c}^1$ and $\mathbf{c}^2$ be the position of the constraint point on O_1 and O_2, respectively. Our objective is moving $\mathbf{c}^1$ to $\mathbf{c}^2$ by applying compulsive translation and compulsive rotation onto O_2, as shown in Figure 3.18. The *arm vector* **r** for $\mathbf{c}^1$ is calculated as:

$$\mathbf{r} = \mathbf{c_2} - \mathbf{x}^{\text{update}},$$

where $\mathbf{x}^{\text{update}}$ is the position of O_2 after the update stage. Now the compulsive translation vector **T** and compulsive rotation matrix **R** should satisfy the following equality condition:

$$\mathbf{r} + \mathbf{s} = \mathbf{R}\,\mathbf{r} + \mathbf{T}, \qquad (3.4)$$

where the vector **s** equals to $\mathbf{c}_1 - \mathbf{c}_2$.

Notice that the constraint force should be workless. In other words, we should move O_2 along the shortest path to minimize the compulsive motion of O_2. In the case of rotation, **R** can be specified with the rotation axis **A** and the rotation angle θ. We can intuitively calculate the rotation axis **A** from the cross product of **r** and **s**:

$$\mathbf{A} = \frac{\mathbf{r} \times \mathbf{s}}{|\mathbf{r}||\mathbf{s}|}$$

Calculation of the rotation angle θ is indirectly derived from rotational dynamics. For the rotating object O_2, torque $\boldsymbol{\tau}$ with initial configuration can be calculated as follows:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{f},$$

where **f** is the linear force applied at $\mathbf{c}_2$. This imaginary force **f** will move $\mathbf{c}_2$ to $\mathbf{c}_1$. Assuming **f** generates constant acceleration **a**, it is possible to approximate **f** as follows:

$$\mathbf{f} = m_2 \mathbf{a} = \frac{2m_2}{(\Delta t)^2} \mathbf{s} ,$$

where m_2 is the mass of O_2 and Δt is the time interval between animation frames. Now, the magnitude of torque $\boldsymbol{\tau}$ can be expressed as:

$$\tau = \frac{2m_2}{(\Delta t)^2} rs \sin \phi , \tag{3.5}$$

where s is the length of **s** and ϕ is the initial angle between **r** and **s**. Letting α be the angular acceleration, τ also can be expressed as:

$$\tau = I_2 \alpha, \tag{3.6}$$

where I_2 is the moment of inertia for O_2. From Equations (3.4) and (3.5), the angular acceleration α can be approximated as follows:

$$\alpha = \frac{2m_2}{I_2 (\Delta t)^2} rs \sin \phi . \tag{3.7}$$

Theoretically, the rotation angle θ can also be approximated from the above angular acceleration. However, Equation (3.7) is available only for the initial configuration since the angle ϕ varies along with the rotation of O_2 due to the angular acceleration α. Thus we only use the characteristic physical parameters to build up our rotation angle calculation formula.

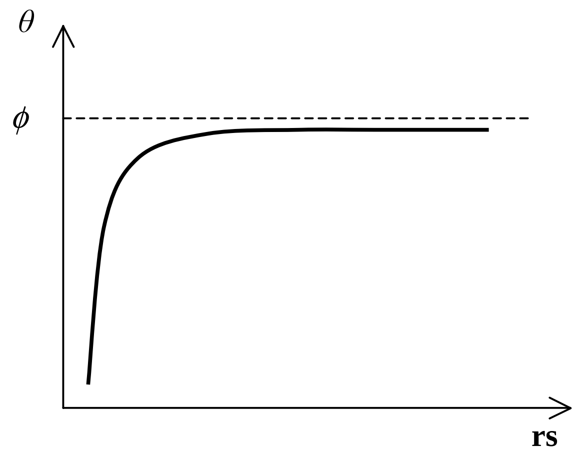


Figure 3.19. The graph of $\theta = \phi\ e^{-\frac{h}{rs}}$

Our starting point for approximating θ is the simple observation that θ is a value between 0 and ϕ. Additionally, the angle θ is influenced by parameters r and s. Thus it is natural to use exponential function as follows:

$$\theta = \phi\ e^{-\frac{h}{rs}}, \tag{3.8}$$

where the constant h is equivalent to $\frac{2m_2}{I_2(\Delta t)^2}$. As shown in Figure 3.19, this formulation shows that the rotation angle θ is nonlinearly proportional to r and s, while its value is bounded in [0, ϕ]. When r and/or s are increased, the rotation angle θ approaches to ϕ, while θ goes to near 0 with small r or s values. When r or s is 0, it means no rotation at all and thus the angle θ is trivially 0. The constant h is a user-controllable parameter, which decides the ratio of θ and ϕ for given r and s.

Now we have the formulations for the rotation axis and the rotation angle. Thus the rotation matrix **R** in Equation (3.4) can be calculated. The compulsive translation vector **T** is easy calculated from Equation (3.4) as follows:

$$\mathbf{T} = (\mathbf{I} - \mathbf{R})\,\mathbf{r} + \mathbf{s},$$

where **I** is the 3-by-3 identity matrix.

In this way, we showed that the compulsive translation vector and the compulsive rotation matrix can be calculated from the given geometric configuration. The object is then moved in order to satisfy the constraint. It is the final step of the adjust stage. In the next subsection, we will show another case in which neither of the objects has fixed location.

3.5.1.3 Moving objects solution for two-object articulated bodies

Suppose that an articulated body with two objects is moving freely. After the update stage, each object has its own position, and often does not satisfy the constraint. In the previous subsection, we presented a simpler example in which an object is fixed at a specific location. In contrast, this subsection focuses on the case in which the articulated body moves freely. Only the joint constraint restricts its motion.

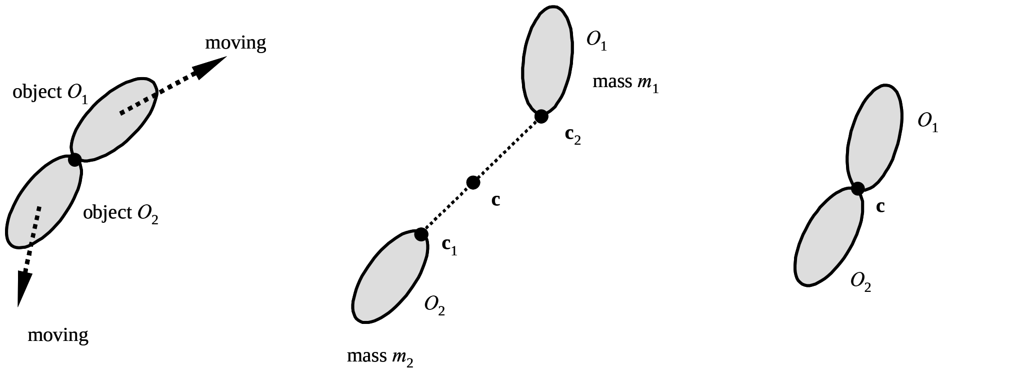
The central idea in this case is calculating the position of the constraint point. Then we solve two simple cases of fixed constraint point. In other words, the original problem of moving two-object articulated body is transformed to two separate problems of fixed object cases, as shown in Figure 3.20.

After the update stage, we have two positions of constraint points for each object. Our objective is calculating the position of the new coincident constraint point from these positions. Notice that the constraint force should be workless. Thus, it is natural to select the new coincident constraint point **c** to be located along the line segment connecting the two given positions $\mathbf{c}^1$ and $\mathbf{c}^2$.

Another characteristic of the constraint force is that it is applied to the objects with the same magnitude but opposite direction. When forces with the same magnitude are applied to objects, the linear movement of each object is inversely proportional to its mass. Letting the masses of O_1 and O_2 be m_1 and m_2, it is intuitive that

$$m_1 (\mathbf{c}_1 - \mathbf{c}) = m_2 (\mathbf{c} - \mathbf{c}_2).$$

Since the new coincident constraint point is located on the line segment $\overline{\mathbf{c}_1\mathbf{c}_2}$, we can easily derive that

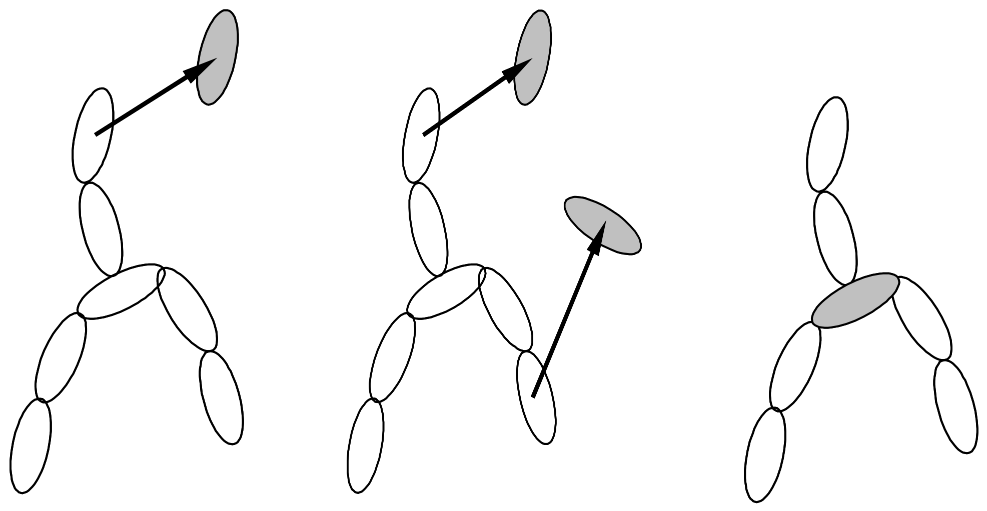


(a) original configuration (b) calculating **c** (c) applying fixed-point constraint solving

Figure 3.20. Constraint solving for a free-moving articulated body.

$$\mathbf{c} = \frac{m_1\mathbf{c}_1 + m_2\mathbf{c}_2}{m_1 + m_2}.$$

Now the two objects of the articulated body move to this coincident constraint point, as presented in the previous subsection.

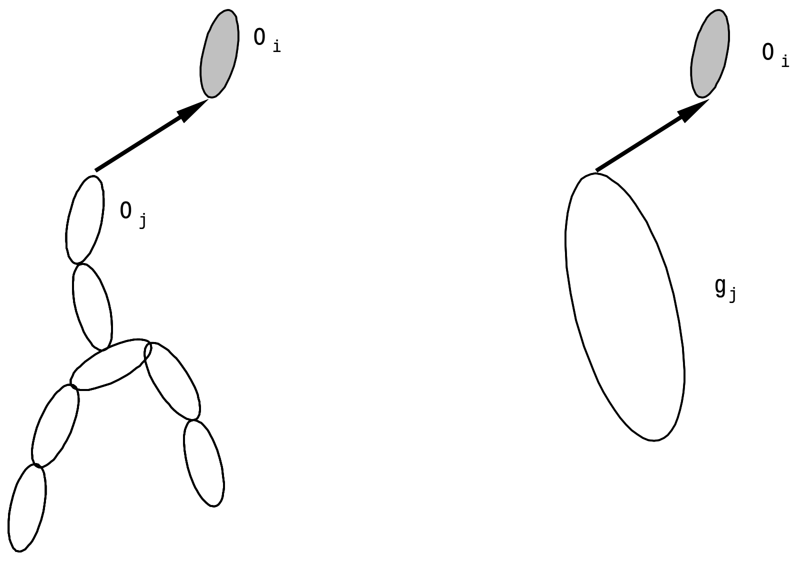


(a) dragging a single object (b)dragging multiple objects (c) free movement

Figure 3.21. Constraint solving for tree-like articulated bodies

3.5.1.4 Tree-like articulated bodies

Since an object of an articulated body is connected to its adjacent objects, propagation of forces from its neighbors affects itself. This propagation process makes the motions of articulated bodies realistic. Constraint dynamics methods usually achieve the propagation process by solving equations of all constraints simultaneously. Even though we have some linear time solutions for



(a) O_i and O_j (a) O_i and g_j

Figure 3.22. Object grouping

articulated bodies, formulations of such equations are complicated and their solutions usually require sophisticated numerical computations [Mirt96][Bara96].

In this subsection, we extend our position-based method to general tree-like articulated bodies, which consist of multiple objects. Constraint solving for tree-like articulated bodies can be classified into three categories, as shown in Figure 3.21. For the first case, the position of only a single object is fixed. This case is basically similar to the fixed single object case in Section 3.5.1.2, while the number of objects in the articulated body is more than two. The next case is multiple objects being dragged, where the positions of more than one object are specified. Finally, we also have free-moving case in which there are no external constraints. Each of these cases is presented in this subsection.

The basic idea of extending the position-based method to tree-like articulated bodies is grouping adjacent objects in order to regard them as a single object, as shown in Figure 3.22. Suppose that an articulated body has n objects, O_1, O_2, …, O_n. As an example, suppose that the position of O_i is fixed, and O_i is connected to O_j with a joint constraint. When the objects O_j, O_{j+1}, …, O_k are all connected together, we group these objects into G_j. Then we simplify this situation as a two-object articulated body whose objects are O_i and G_j. For efficient calculation, we simply assume that the geometric shape of G_j is identical to O_j, but the mass of G_j is the total mass of O_j, O_{j+1}, …, O_k. Then, we can calculate the compulsive motion required for O_j, and we apply the same idea for the next object O_{j+1} through grouping O_{j+1}, O_{j+2}, …, O_k.

When there are multiple objects whose positions are fixed, we cannot simply apply the same idea. A possible solution can be a relaxation process, which is similar to Weil's idea for the cloth modeling [Weil96]. For each fixed object, we apply the above grouping method to the entire articulated body while ignoring other fixed objects. Although objects may move from one place to another at each step, they will reach a stable state after a number of iterations. Of course, we should check impossible cases, which results in infinite loops.

The last case can occur when there is no fixed object of the tree-like articulated body after the update stage. The central idea in this case is fixing a pre-selected object. For example, we can select the torso of a human-like articulated body as its pre-selected one. At the adjustment stage, we first calculate the position of this pre-selected object. Suppose that the pre-selected object O_i has its neighbors O_j, O_{j+1}, …, O_{j+k}. We apply the two object articulated body solutions for each pair of O_i and its neighbor. Now we have k locations for each pair, and the final location of O_i is calculated as the weighted sum of these locations. After fixing the pre-selected object O_i, it is straightforward to calculate the locations of other objects.

3.5.1.5 Other Kinds of Constraints

In constraint dynamics, handling various kinds of joints is an important issue [Bara93]. Our position-based constraint method works well with various joint constraints, since the constraints can be explicitly expressed procedurally. To demonstrate the power of our method, we present approaches to handle joint-angle limit constraints, multiple positional constraints and contact constraints.

The joint-angle limit constraint is widely used in articulated body motions. It is a kind of an inequality constraint, and defines the acceptable range of angles between connected objects. In constrained dynamics, the inequality equations are usually solved by linear complementary method or quadratic programming, which require heavy computations [Bara93].

These difficulties are due to the fact that the constrained dynamics methods have to calculate accelerations even to limit an angle in a pre-defined range. In contrast, our procedural method does not calculate any acceleration, and we can express this kind of constraint in an explicit procedural form. For an articulated body, a pair of objects will be processed using two-object case solutions. After fixing the two objects, we check whether the angle between them violates the pre-specified joint-angle constraint. When it violates the constraint, we simply limit the angle to an extreme value of the given constraint. That is all that is required to satisfy the joint-angle limit constraint.

Multiple positional constraints provide useful tools for interactive control of articulated figures. For example, user may want to drag one hand of a human-like figure while its feet are fixed on the floor. In this case, we have three positional constraints: one for the hand and one for each foot. In inverse kinematics, an optimization method was already proposed [Bald87][Welm93].

However, using our position-based method, it is possible to speed up its calculation without using any optimization method. Notice that the multiple positional constraints are equivalent to the multiple fixed object case of tree-like articulated bodies. As explained in Section 4.1, we can satisfy the multiple positional constraints by a relaxation process.

In dynamics-based simulations, contact constraints are one of the hard-to-solve problems. They usually require complicate computations involving quadratic programming (QP) or Danzig's algorithm to solve the contact problem [Bara93]. Contact points are calculated using collision detection techniques and checking relative velocities of the objects. Since most implementations use discrete time steps, the objects are usually penetrating each other when the collision is detected. Thus, solving contact constraint is equivalent to removing the penetration, in most cases.

In Hahn's method, the penetration is eliminated by backing up the penetrating object with its relative velocity but along the opposite direction [Hahn88]. Our idea is similar to this backing up method. However, we directly move the position of the object while Hahn uses relative velocity for the same purpose.

Suppose that an object (a bouncing ball, for example) penetrates a stationary object (the floor). After detecting the intersection, we first search for a vertex that has penetrated the deepest among the vertices of the penetrating object. Then the penetrating object is compulsively translated along the surface normal direction of the penetrated object so that the two objects are just touching. When several objects are colliding simultaneously, the above translation is applied to each pair of objects.

3.5.1.6 Examples

We present examples of articulated figure motions to demonstrate the power of our procedural method.

Figure.3.23 demonstrate the interactive position control of articulated bodies. User selects the uppermost object of the chain-shape body and drags it to the desired location. Figure 3.24 is another example of the same chain-shape body, whose mass and damping terms are changed. It is easy to find the differences in motions due to the change of physical parameters.

Figure 3.25 shows the motion of a human-like articulated body. User can select an object and move the object, and then other objects follow the selected object. User can generate motions similar to that of marionette whose hand is dragged.

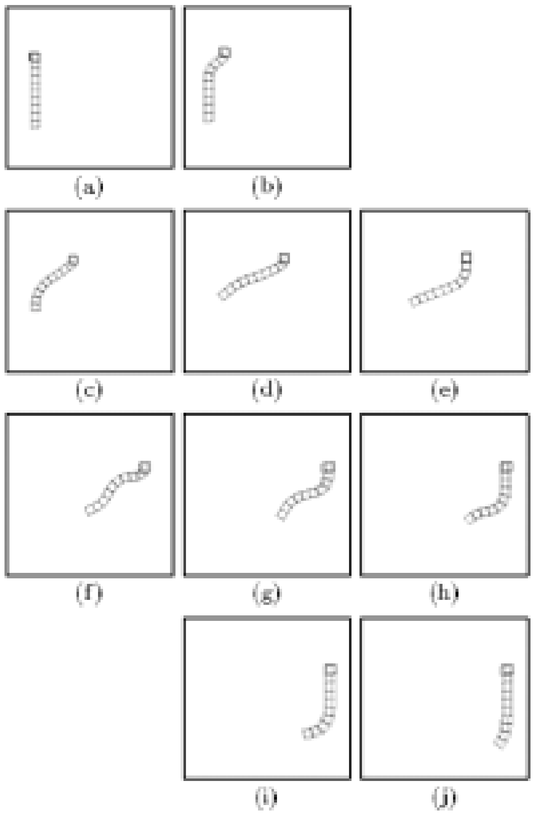


Figure 3.23 Example of dragging a chain: the uppermost object is dragged interactively.

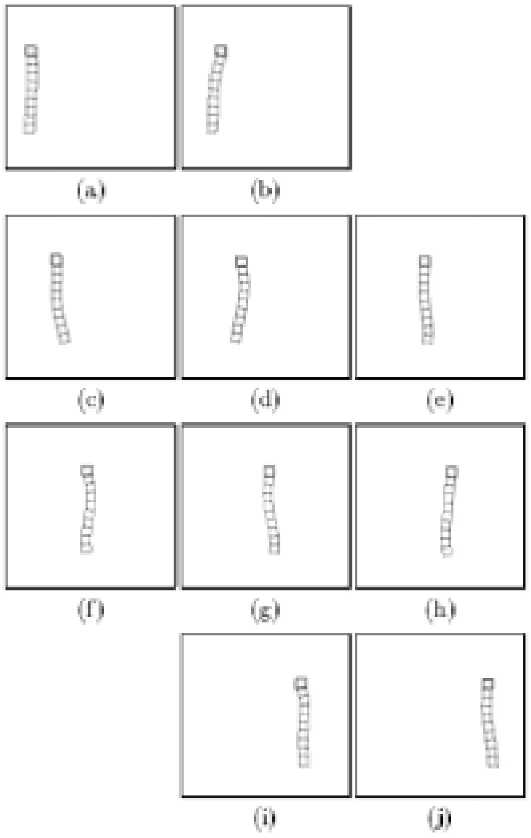


Figure 3.24 Another Example of dragging a heavy chain.

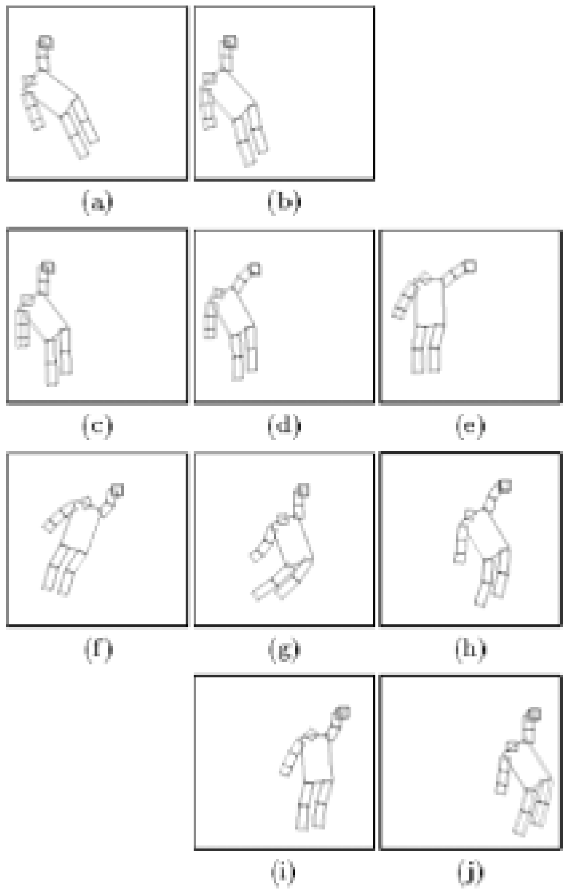


Figure 3.25. Example of dragging a human-like shape

3.5.2 Procedural Methods for Motions of Sphere-like Objects

In the field of computer animations and VR environment, motions of sphere-shaped objects are frequently used. Motions of billiard balls or bowling balls are good examples. To generate realistic animations, we need a physical model to process motions and collisions of these sphere-like objects. The motion of a sphere-like object can be decomposed into rolling and sliding. Additionally, the transient motions between rolling and sliding are also needed. For example, a billiard ball usually slides on the table with its own spin, and its spin will contribute to its linear velocity and vice versa.

Collisions can be easily modeled by impact dynamics techniques. In the case of rolling motions, it is needed to solve non-holonomic constraints [Deyo88]. Since the exact solutions for

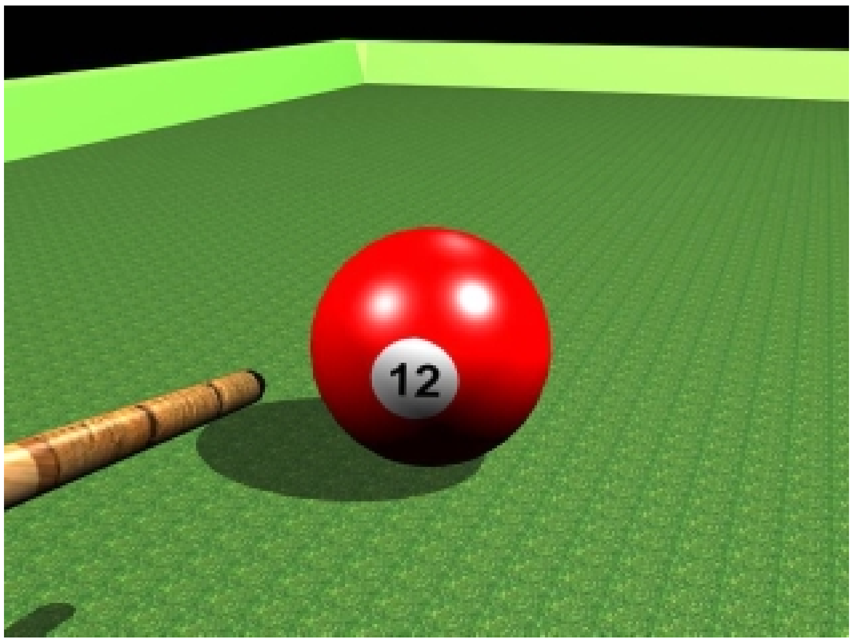


Figure 3.26 A motion of sphere

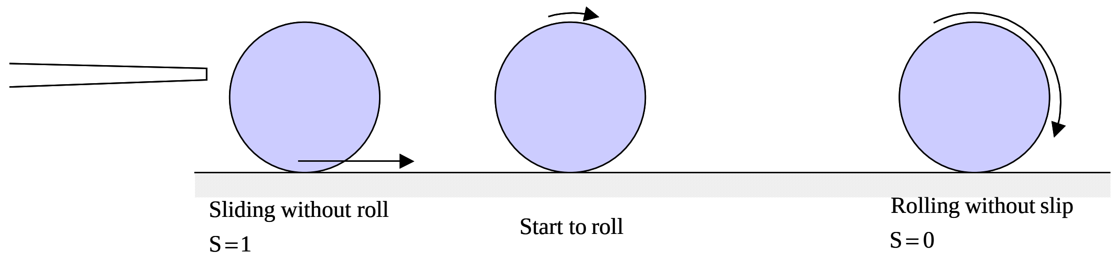


Figure 3.27 Sliding from rolling

non-holonomic constraints require complicated computation [Deyo88], we present a procedural method that can be used as an alternative way of solving non-holonomic constraints for rolling motion. Although our procedural method may produce physically-incorrect motions, its final result is visually plausible. Additionally, our method is easier to implement and faster than the traditional way of non-holonomic constraint solving.

A constraint is called holonomic when it can be represented as followings:

$$f(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., t) = 0,$$

where $\mathbf{x}_i$'s are positions of objects and t is the parameter for time. Point-to-point and point-to-nail constraints are examples of the holononic constraint. In contrast, non-holonomic constraints cannot be represented as a function of object coordinates, $\mathbf{x}_i$'s. A rolling ball without sliding is an example of this non-holonomic constraint.

To produce realistic motions of sphere-shaped objects, we should properly model its sliding, rolling, and transient motions between sliding and rolling. Instead of integrating these three cases into a single equation, we develop separated procedural equations. The system determines above three cases by comparing angular velocity and linear velocity. For the transient from sliding and rolling case, the decision equation is:

$$s = \frac{|v| - |r\omega|}{|v|},$$

where v and ω are linear and angular velocity, respectively. r is the radius of the sphere.

If s = 1, the sphere is slipping without rolling. If s = 0, the sphere rolls without slipping. Figure 3.27 shows motions from sliding without rolling to rolling without slipping. For rolling without slipping case (s = 0), only rolling resistance is applied to the rolling motions and velocity matching (v = rω) is applied. Instead of finding exact force and torque to satisfy the non-holonomic constraint, the system decides the angular velocity according to the linear velocity without considering angular acceleration. This is a kind of velocity-based physics mentioned by Milenkovic [Mile96]. Rolling resistance depends on surface roughness and decreases velocity of the sphere. For sliding without rolling case (s = 1), when someone hit middle of a ball hardly, only slipping without rolling occurs for a while. Friction force without rolling resistance should be applied for this case. For transient case, smooth transient should be modeled to get realistic motions. Friction force decreases when the spheres start to roll. When the sphere rolls without slipping, friction forces are zero. Smooth transient from initial sliding to rolling can be modeled by interpolating the friction force according to angular velocity. The equation of interpolated force, $\mathbf{F}_\mathrm{i}$ is,

$$\mathbf{F}_\mathrm{i} = \left(-\frac{|\omega\,\mathbf{r}|}{|\mathbf{v}|} + 1 \right) \mathbf{F}_{\mathbf{friction}}$$

We introduced procedural method for motions of sphere-like objects. This is easy to implement and fast. In addition to motions of balls, above equations can be applied to motions of tires in cars. Exact formulation of tire is very complicate.

3.6 Virtual Wind

Wind is an important external force and is widely used in computer animation and Virtual Reality applications. Modeling physically-correct wind is very difficult because it includes aero dynamics and turbulence problems. Isaac system has real time wind generation model. We will explain our virtual wind model with its direct application of virtual mobile system, which is also provided by Isaac system.

Recently, real world objects are successfully reproduced in the computer systems, using computer graphics and virtual reality techniques. A good application example is digital museums, which display reproductions of real world fine arts on the computer screens [McWh88]. For drawings, digital scanners and/or digital cameras can be used to generate the digital reproductions. In the case of sculptures, three-dimensional geometric and/or volume data can reproduce virtual sculptures on the screens[McGu93]. Image-based rendering techniques including MCOP(multiple center of projection)[Rade98] can also be used for this purpose.

Mobiles, which are also known as *moving sculptures*, however, can not be represented successfully using these techniques. As an example, a real world mobile, “Steel Fish” by A. Calder is shown in Figure 3.28. Typical mobiles are dynamic systems: their components are usually dangling from the stems and swing due to external forces such as winds. At least in the area of computer graphics, we have not seen any reproductions of mobiles.

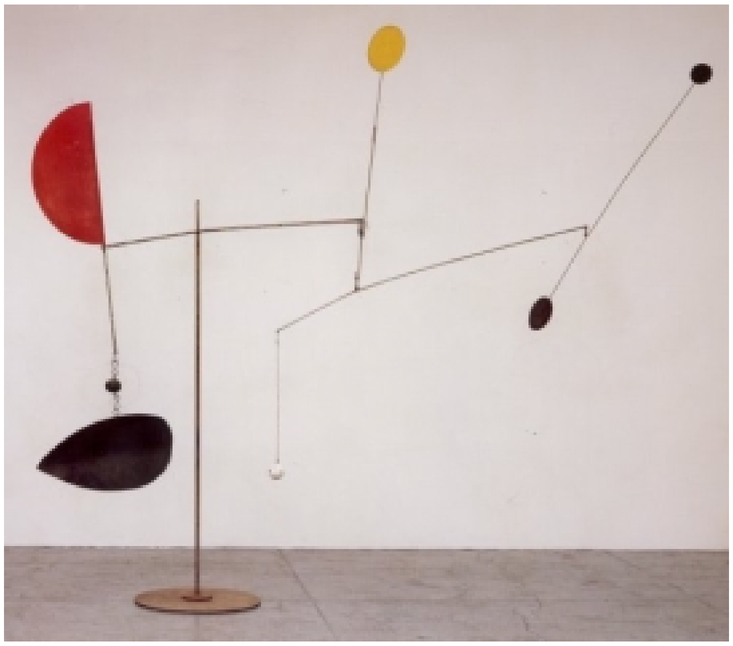


Figure 3.28. A real world mobile: *Steel Fish* by Alexander Calder.

Isaac system can be used to generate virtual mobile system. To simulate a real world mobile, we start from constructing its geometric shape. Then, some physical properties such as masses and inertia tensors are calculated. The mobile is simulated by a constraint dynamics system, based on these geometric data and physical properties. To invoke the motions of mobiles, users can generate virtual winds, and our constraint dynamics solver presents the motions and collisions of components. An impulse dynamics system is also used to response collisions among the components of the mobile. Using a simplified aerodynamics model for the virtual wind and other recent acceleration techniques, our system accomplished real time display of an example virtual mobile on Pentium chip-based personal computers.

While the natural wind causes the motions of real world mobiles, we need virtual wind to simulate the motions of our virtual mobile. To control the virtual wind, we may use traditional input

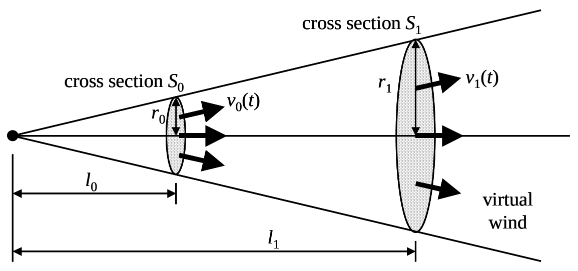


Figure 3.29 Virtual wind model.

devices such as keyboards and mice. Additionally, our system uses a microphone interface. The user blows at the microphone, and the speed of the virtual wind is proportional to the amplitude of the input sound. The direction of the wind is specified with the mouse.

Our microphone interface has some benefits. First, it is more intuitive for the user to generate the wind through the blowing action. Second, the microphone is more convenient to simulate the temporal variations of the wind speed. Another benefit is that the microphone is inexpensive and easily available even for personal computers.

Since it is generated from the human breath, the virtual wind is assumed to propagate in an infinite cone shape, as shown in Figure 3.29. A circular cross section S_0 with its radius r_0 plays the role of the source of wind. The vertex of the cone is located at the distance l_0 from the center of S_0. User can provide r_0 and l_0 to control the shape of the cone, and the wind propagates from S_0 in the direction opposite to the vertex. Using the microphone interface, user controls the wind speed $v_0(t)$ at S_0.

For simulating the wind, we need to calculate the speed of the wind at a distance $l_1 > l_0$ from the vertex. Although there are several results [Reev83][Wejc91][Shin92] for simulating aerodynamics in computer graphics applications, we use a simplified form to achieve real time display. We start from assuming that the fluid (in this case, air) is inviscid and incompressible, and no fluid can cross the boundary of the cone shape. This is a reasonable model for air at normal speed [Wejc91]. Then, the Equation of Continuity in fluid dynamics gives

$$A_0 v_0 = A_1 v_1, \tag{3.9}$$

where A and v represent the area of the cross-section and the fluid speed, respectively[Patt89]. The subscript 0 and 1 corresponds to the distance l_0 and l_1, respectively.

From Figure (3.29) and the geometric configurations of the cone, it is easily found that

$$\frac{v_1}{v_0} = \frac{A_0}{A_1} = \frac{\pi r_0^2}{\pi r_1^2} = \frac{\pi l_0^2}{\pi l_1^2} = \frac{l_0^2}{l_0^2} . \tag{3.10}$$

Using the Stoke drag equation [Feyn65][Patt89], the force acting on a face with its area A located on the cross section S_1 can be calculated as follows:

$$\mathbf{F}_{\text{stoke}} = \rho A v_1^2 (\mathbf{n}_v \cdot \mathbf{n}_a)\mathbf{n}_a, \tag{3.11}$$

where $\mathbf{n}_v$ and $\mathbf{n}_a$ are the unit directional vector of the wind and the face normal vector, respectively, as shown in Figure 3.30. The constant ρ is the density of fluid. Equations (3.10) and (3.11) give

$$\mathbf{F}_{\text{stoke}} = \alpha A \frac{v_0^2}{l_1^4} (\mathbf{n}_v \cdot \mathbf{n}_a)\mathbf{n}_a,$$

where α is a constant. For reflecting the turbulent behavior of wind, we add a random noise term and the final force can be expressed as

$$\mathbf{F}_{\text{wind}} = \mathbf{F}_{\text{stoke}} + \mathbf{F}_{\text{random}}, \tag{3.12}$$

where the direction of $\mathbf{F}_{\text{random}}$ is randomly selected, and $|\mathbf{F}_{\text{random}}| < \beta |\mathbf{F}_{\text{stoke}}|$ for a user-

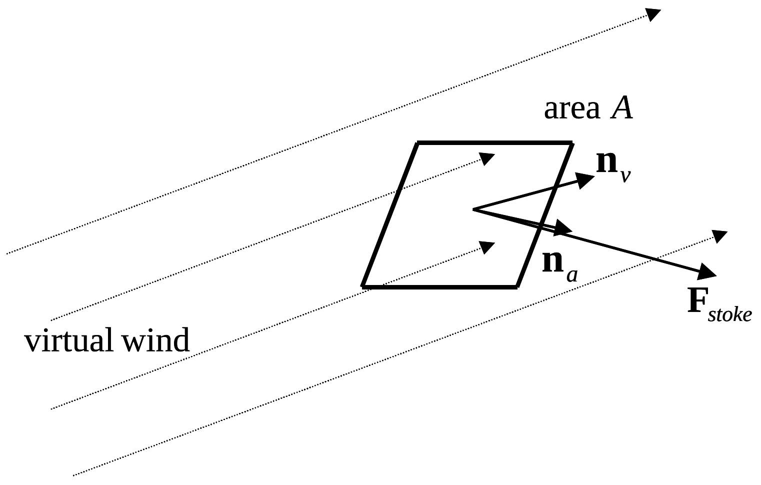


Figure 3.30. Force due to the virtual wind.

specifiable constant β.

Before applying the force calculated in Equation (3.12) to the face, we should check whether the wind is directly delivered to the face or not. When a face is occluded by another face in the air flow, we simply assume that the occluded face is not influenced by the wind. Without this assumption, it is hard to achieve the real-time display of the mobile. In this simplified wind model, the air flow can reach to the faces which are directly visible from the vertex of the cone and belong to the interior of the cone.

Due to this assumption, the influenced faces can be identified by a visible surface detection method [Fole90]. We use the depth-buffer method for more speed-up. To simulate the partially occluded cases, faces are first partitioned into small area on which a sampling point is assigned. Then, the graphics pipeline is used to capture the image containing the cross section S_0 using the synthetic camera located at the vertex of cone. Each sampling point has its own identification number, and it is also stored in the alpha buffer through the graphics pipeline. Scanning the alpha buffer, we can easily detect whether each sampling point is visible from the vertex of the cone or not. Since S_0 corresponds to a circle on the image plane, it is also easy to check the point belongs to the interior of the cone. When a sampling point is visible and also belongs to the interior of the cone, the force calculated in Equation (3.12) is applied to its corresponding area. In the next section, we will represent how the constraint dynamics techniques are used to apply these external forces due to the wind.

3.7 Sound Generation and Synchronization with Motion

Sound is a key factor in computer animation and virtual reality(VR) since hearing is as important as seeing to humans. Sound in the real world provides important information that cannot be acquired just through images. Therefore, realistic sounds along with realistic images are necessary for computer animation and VR applications. Since most sounds are generated due to object motions, the matching of generated sound with motion is very important. Accordingly, *synchronization* includes matching sound characteristics with the motion and material of an object as well as timing. To achieve a natural matching of sound and motion along with timing, the material properties and geometry of an object also need to be considered for realistic sound generation. For example, when the middle part of a drum is hit, a lower sound will be produced compared with the sound of hitting the edge part.

Most current sound systems for VR and computer animation merely consider timing and amplitude from motion, while the real world produces an array of distinct sound characteristics according to variety of collision and contact conditions. Sampling many sounds is one possible method of reproducing various conditions. However, sound sampling is not easy and requires a large memory. Therefore, an algorithmic sound generation method that considers the contact, collision, and material properties of object is an attractive one.

This thesis introduces a series of sound equations that produces various sounds from different motions. Through evaluating these equations, it is confirmed that they are fast enough for real time applications. Some of the equations are derived from the principles of physics whereas others are based on heuristic methods. Sounds of homogenous objects can be generated by physics

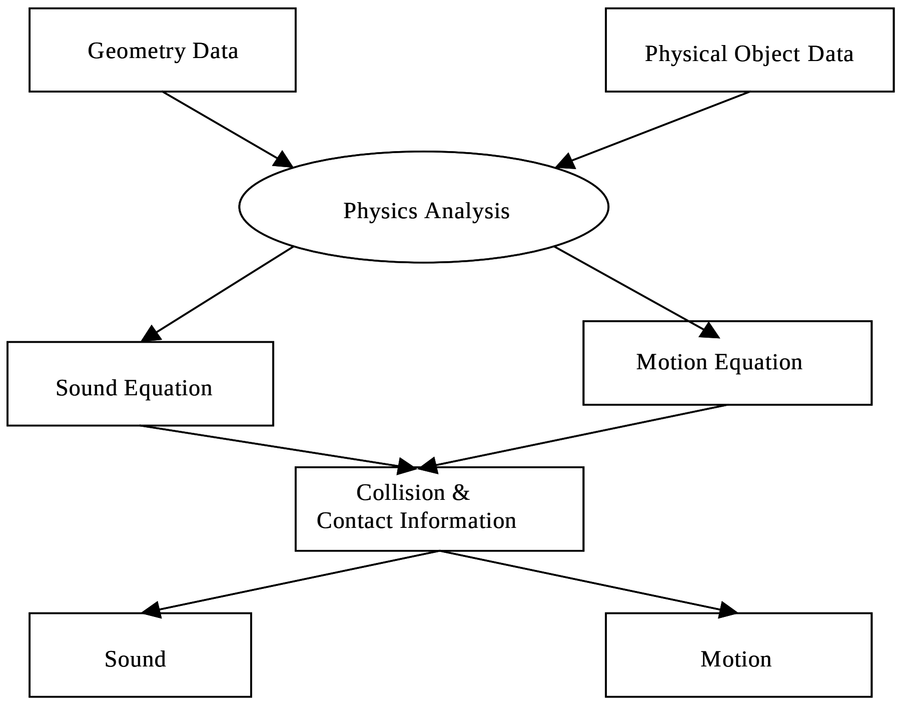


Figure 3.31 Sound synchronization with motion

equations. However, sounds generated by heterogeneous objects are very difficult to model by physical analysis. Therefore, heuristic based procedural methods are used to generate these kinds of sounds.

The parameters of the sound equations are connected with motion parameters (collision force, scratching velocity, etc.) and geometry information (collision/contact location, size etc). As a result, various sounds can be automatically produced according to the particular motion, which is infeasible using sampling-based methods.

There are many methods for representing and controlling sound signals. Among them, Additive synthesis and Frequency modulation (FM) are the primary methods in sound generation. Additive synthesis adds sound signals in the amplitude domain to get the desired sound, whereas

FM changes the signal frequencies in the frequency domain.

The following sections describe the details of sound generating methods. In section 3.7.1, we explain methods of generating and controlling sounds of homogenous material based on physical laws. Section 3.7.2 explains how heterogeneous material sounds are generated by procedural method. Section 3.7.3 present a method to generating scraping sound from textures. Finally, Section 3.7.4 explains a method of generating wind sound based on fractal.

3.7.1 Homogeneous Material Sound

In computer animation, physical laws are used to generate realistic motion. Likewise, realistic sounds can also be generated using the laws of physics. Sounds of homogenous material can be produced by the law of physics. Vibration analysis provides the basis for sound generation [Flet91]. A vibrating object generates sound through vibrating the air. Thus the sound equation is closely related to the object vibration equation. Material properties and geometry of an object are also key factors for these vibrations. Although vibration equations of simple objects such as spring and string are quite straightforward, the formulating vibration equations of complicated shapes or heterogeneous material objects is very difficult. Once a correct vibration equation is established, a variety of sounds can be produced using proper values of sound and motion parameters.

Since a vibration equation is often represented by frequencies, the sound of an object can be represented by adding up all the frequencies. This method is also known as Fourier synthesis. One of the simplest vibrating systems (the simplest vibration modes) is a mass-spring system. Its frequency can be represented as follows:

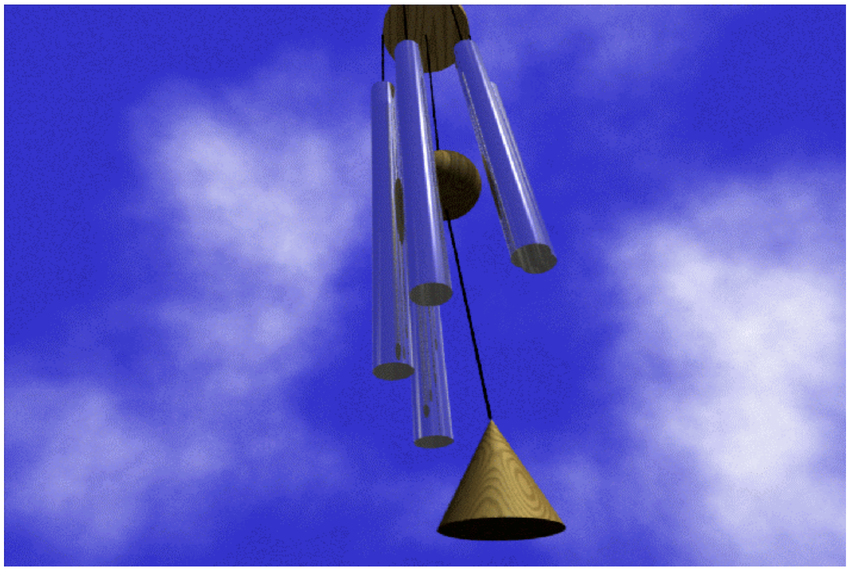


Figure 3.32 Blowing in the Wind

$$f = \frac{1}{2\pi}\sqrt{\frac{K}{m}},$$

where K is the spring constant and m is the mass.

Vibrating string can be considered as a spring-mass system that has many springs and masses. There are many modes of vibration and the modes are represented by multiples of the fundamental frequency (the lowest frequency). Longitudinal vibration frequencies can be represented as :

$$f_n = n\frac{v_L}{2L},$$

where n is natural number, v_L is speed of sound in the string, and L is the length of the string. The advantage of this formulation is that the equation represents sound as the motion of the string. Therefore, sound and motion can be combined seamlessly.

Chime sounds can also be represented by vibration analysis. Although the vibration analysis of a chime is similar to that of a string, the frequency dependence on tension is more complicated. A modal analysis of the frequency components of a chime is as follows:[Flet91]

$$f_n = \frac{\pi K}{8L^2}\sqrt{\left(\frac{E}{\rho}\right)}\left[3.011^2, 5^2, 7^2, \ldots, (2n+1)^2\right],$$

where L is the length of the chime, E the elasticity, ρ the density of material, and K the radius of gyration, which is the square of the chime's outer diameter over it's inner diameter. This equation can be used to generate wind chime sounds. The geometric modeling and material properties of a chime can then be mapped using *L, E, K,* ρ. The collision forces are mapped to the amplitude of the sound, whereas the collision timing can be found by the collision detection routines. As a result, the synchronization of motion and sound can be achieved naturally. Figure 32. shows an image from "Blowing in the Wind", the chime sounds were generated by above equations. Although the modal analysis of a complicated object is not easy, it is easy to map the physical attributes to the sound parameters, after establishing the vibration equations.

3.7.2 Collision Sounds for Non-Homogeneous Material

The creation of sound equations for simple shaped and homogeneous material objects is quite straightforward, however, equations for non-homogeneous objects are very complicated. Therefore, procedural equations without a complicated analysis are very useful in establishing sound equations for complicated objects. In computer graphics, similar methods are used to generate complicated images and motions. The modeling of gas motion, mountain geometry, and wood texture is very difficult using exact physical laws. Thus, instead of applying complicated physical laws, mathematical equations can be employed to generate these images and motions

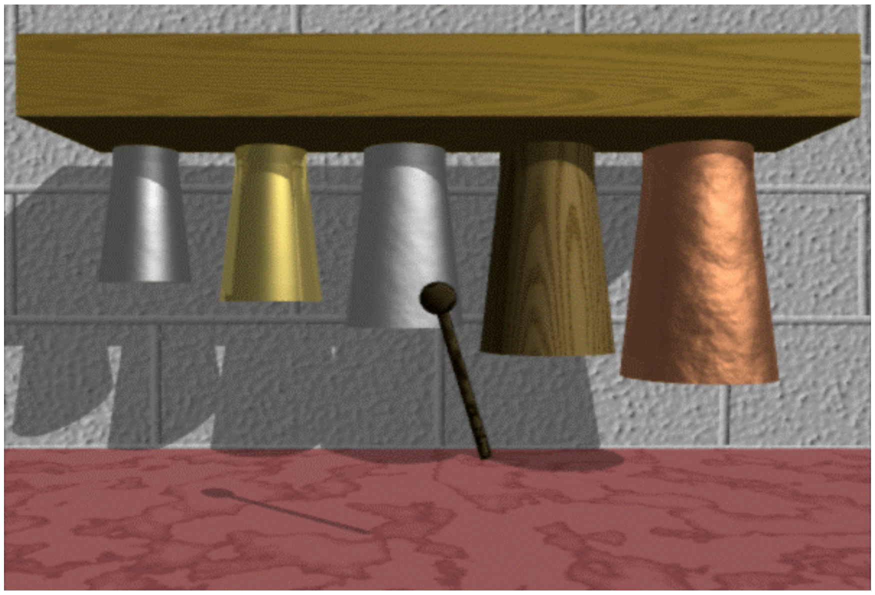


Figure 3.33 Bell and wood sounds

based on the analysis of their outward appearance. Users then control the parameters in the equations to gain the desired images and motions. Although the parameters are not directly related to the physical properties of the objects, visually plausible images and motions can still be generated by controlling the parameters. Similar methods can also be used to generate complicated sounds. Therefore, mathematical equations and the control of their parameters can generate aurally plausible sounds of complicated objects.

The proposed procedural sound generation method is based on the additive method. An advantage of Additive synthesis is the intuitive control of a sound examining its amplitude, timbre, and decay. The timbre of a sound is controlled through the number of added signals and decay parameters. For example, a bell sound has a small number of signals, whereas a plastic or wood sound has a large number of signals. The following equation represents a general collision sound,

$$\text{Signal(t)} = \alpha \sum_{i=1}^{n} e^{-c\omega^{t}} \sin(\omega_i t) ,$$

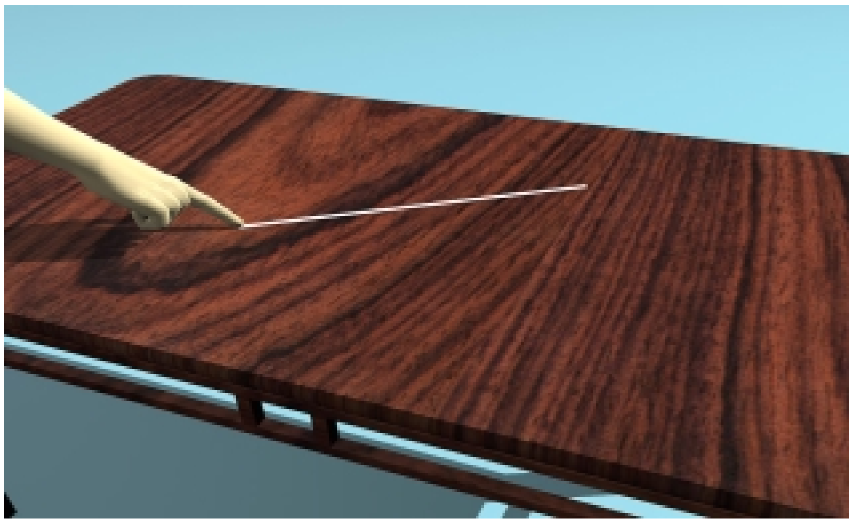
where ω_i corresponds to a set of random frequencies, c is the damping constant, and α represents the amplitude.

There is a relationship between the physical and geometric parameters and the sound waves. The collision/contact between objects has close relation with the amplitude, spectrum and bandwidth of the sound waves. The material parameters, including the density, damping, and homogeneity all have an effect on the signal frequency. The shape, size, and resonating cavities have effects on the frequency, spectral pattern, and bandwidth. All these parameters and their relationships must be considered when determining the sound equation parameters.

Many signals and small value of damping parameters generate wood-like sounds. Random number generators are then used to decide the frequencies and amplitudes of the adding signals. A small and limited frequency signal makes a bell sound. By controlling the main frequency of the signal, the frequency of the bell can also be controlled. Figure 3.33 shows different bell and wood sounds. The frequency number of the wood is 600, the drum number is 250, the bell number is 10, and the metal number is 25.

3.7.3 Texture-based Sound

Mapping between a user gesture and the sounds resulting from that gesture are very important in VR and real time animation. Many motions are involved in an interaction between the



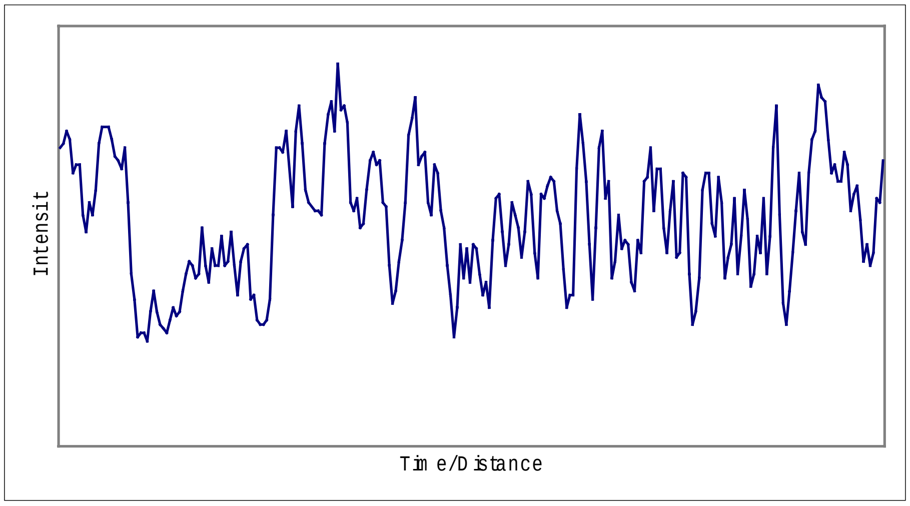


Figure 3.34 Wood texture and signal from scratch line

user and an object. Among these interactions, rubbing and scratching make sounds based on a combination of interaction and surface properties. When someone scratches a surface with his/her nail or pen, the sound should reflect the roughness of the surface. In computer graphics, the roughness of a surface is represented by bump mapping or a 3D texture, rather than by geometric modeling. In this thesis, it is assumed that the roughness of a surface is represented by texture information. Therefore, the synchronization of a scratching gesture and its sound can be generated using texture mapping. Perlin introduced 3D texture generation methods[Perl85]. Using his noise function, Perlin generated wood and marble textures where the patterns were neither random nor regular. Figure 3.34 shows the wood texture using a noise function and the resulting signal from the scratch line. The signal is generated from the intensity values of the texture. In fact, the roughness of a surface and intensity of the texture color has a close relationship in a real wood surface.

Various sounds can be generated by applying different filters to this signal. For example, sounds of soft or hollow objects can be generated by applying low-pass filters. The speed of the scraping object is then mapped to the scaling factor of the signal (on the time axis) to synchronize the motion and sound. The overall amplitude of the sound is scaled using the normal force with which the surface is scrapped to arrive at the final sound. The sounds generated should be a function of the texture, the scraping object, the speed, and the force with which the surface is scraped.

3.7.4 Fractal-based sound

There are so many factors in the real-world, therefore, specifying them in a deterministic way is impossible in some cases. Consequently, complicated natural phenomena, such as clouds and mountains, can be represented by the fractal geometry [Four82]. Fractals are specified by iterative procedures such as $f(x_i) = f(f(x_{i-1}))$. In computer graphics, random fractals are used to model natural shapes and motions. The fractal method can generate very complicated geometries with small codes and simple parameters by amplifying basic data.

A wind sound is also an example of a natural phenomenon. Therefore, it is reasonable to apply simplified fractal method to generate a wind sound is reasonable. The “Blowing in the Wind”(see Figure 3.32) animation demonstrates that a realistic wind sound can be generated using a simplified fractal function and the wind sound and wind force can be naturally mapped for synchronization.

Unlike a general fractal function, we employed Multifractal function [Eber94] that replaced the addition of frequency in the inner loop by multiplication. Multifractal represents heterogeneous (not same everywhere) signal than a general fractal. Since wind in nature has not same amplitude, Multifractal shows better wind effect. The Prelin noise-function is used as the basic function [Perl85]. As a result, the wind sound of the “Blowing in the Wind” animation is

$$\underset{i=1}{\overset{5}{}} \left\{ \frac{noise(t \cdot 50 \cdot i)}{2^{i-1}} \cdot \left(noise(0.5 \cdot t) - 0.2 \right) \right\}$$

We introduced algorithmic sound generation methods based on physical laws and procedural methods. Unlike sampling methods, our method has parameters that correspond with object’s motion parameters. Therefore, timbre of sounds is naturally synchronized with motions and object’s properties. Sound synchronization, one of most important problems in VR and animation, is naturally solved in Isaac system.

4. Conclusions and Future Works

This dissertation concludes by describing summary of solution, results, original contributions and future works.

4.1 Summary of Solution and Results.

We presented procedural methods for motions of articulated bodies, sphere-like objects, and real time wind generation. Our aim was to generate visually plausible animation sequences rather than physically correct motions. Since our method does not solve complicated dynamic equations, the method can achieve interactive control of the motions with numerical stability. This physically based procedural method can be an alternative to dynamics simulation, especially for real-time applications such as virtual reality environment and computer games. The physically based procedural methods are formulated by physical analysis, to finally use physical quantities as their input parameters.

We reproduced a real-world mobile on a computer system by Isaac system. The virtual mobile system interactively simulates motions of the mobiles according to user’s blowing action. Virtual wind model, constraint dynamics solver, and the impulse dynamics solver were combined seamlessly for the realistic simulation of the mobile.

As a real time virtual wind model, we suggested a simplified wind model, which is simple but sufficient to simulate directional winds. Additionally, the microphone interface is developed for easy control of the virtual wind. Since this wind model can generate directional winds, it is suitable for simulating artificial winds generated by electric fans, air-conditioners, etc.

Constraint combining was proved to be a useful tool for designing complicated constraints. Constraints of Terry Elis's clock were formulated by combining constraints in Isaac system. Constraint breaking and time varying parameters give more freedom to designing complicated constraints such as time varying constraints.

Procedural method was also proved to be a useful tool for generating sounds. Various realistic collision sounds including that of wind chimes, bell and wood were formulated by the procedural method. Texture-based scraping sound and fractal based wind sounds are also implemented. All sounds are naturally mapped with motion parameters such as collision and contact parameters.

4.2 Original Contributions

This thesis proposed physically based procedural methods. This was designed to solve the problems of physically based modeling. This is a new method to have advantages of procedural method and physically based modeling. The methods produce visually plausible motions of articulated bodies, sphere-objects and wind that are faster and more stable than motions from physically based modeling method.

We formulated sound generation equations by the physically based procedural method. By combining the procedural method and physically based modeling, we formulated equations of various sounds including collision, contact and wind. The parameters of the equations can be mapped easily with parameters of motion equations.

Real time wind system was designed and implemented based on the physically based procedural method. When user blows to the microphone, this system produces real time virtual wind according to the intensity of blowing, direction and distance between the user and the virtual objects.

We designed and implemented a motion generation system that combined with constrained dynamics, procedural method and sound. Constraints in the system can be combined and subtracted with time paramters for new constraints.

4.3 Future Works

We applied the physically based procedural method to articulated bodies, sphere-like objects and collision sounds. The concept can be applied to other heavy computational models such as soft objects and cloth. Since the structure of articulated bodies and deformable objects are similar, applying to the deformable objects may be possible. Previous procedural method can also be combined with the physically based procedural method. For example, noise functions can be added to the procedural method to express voluntary motions.

Automatic degradation would be an interesting future work. According to its computing power, a computer system may automatically choose the dynamics model or the procedural model.

Control algorithm can be added in our procedural method. Currently, our procedural method is controlled by user's interaction. Walking or running algorithm can immediately use our procedural model. Developing collision model by procedural method may be another future work.

Mathematical error measurement of our method can also be explored in the future. Currently, our visual perception is only the measurement of our algorithm like other procedural methods.

However, mathematical error measurement will provide more theoretical background of our work.

References

[Arms85]W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. *Visual Computer,* Springer-Verlag, pages 231-240, 1985.

[Badl87]N. Badler, K. Manoocherhri, and G. Walters. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, pages 28-38, June 1987.

[Bara89]D. Baraff, Analytic methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH'89*, pages 223-232

[Bara93]D. Baraff, Fast contact force computation for non-penetrating rigid bodies. *SIGGRAPH'93*, pages 23-34, 1993.

[Bara96]D. Baraff, Linear-time dynamics using Lagrange multipliers. *SIGGRAPH'96*, pages 43-54, 1996.

[Bara98]D. Baraff and A. Witkin. Large time steps in cloth simulation. *SIGGRAPH'98*, pages43-54, 1998.

[Barz88]R. Barzel and A.H. Barr. A modeling system based on dynamic constraint. *SIGGRAPH'88*, pages 179-188, 1988.

[Barz96]R. Barzel, John F. Hughes, and Daniel N.Wood. Plausible motion simulation for computer graphics. *in Computer Animation and Simulation '96* (*Proceedings of the Eurographics Workshop*), Springer Wien, New York, pages. 183-197, 1996.

[Barz97]R. Barzel. Faking dynamics of ropes and springs. *IEEE Computer Graphics and Applications*, 17(3), pages. 31-39, May-June 1997.

[Berc86]B. Bercoe. Csound: A manual for the Audio Processing System and Supporting programs, MIT Media Lab, MIT, MA, 1986.

[Bree94]D.E. Breen, D.H. House, and M.J. Wozny. Predicting the drape of woven cloth using interacting particles. *SIGGRAPH'94,* pages 365-372, 1994.

[Cook84]R. Cook. Shade Tree. *SIGGRAPH'84*, pages 195-206, 1984.

[Dann91]R. Dannenberg, C. Fraley and P. Veliknoj. Fugue: a functional language for sound synthesis, *IEEE Computer*, 24 (7), pages 36-42, 1991.

[Desb95]M. Desbrun, M. Gascuel. Animating soft substances with implicit surfaces. *SIGGRAPH'95*, pages 287-290, 1995.

[Deyo88] R. Deyo, J.A. Biggs and P. Doenges. Getting graphics in gear: graphics and dynamics in driving simulation, *SIGGRAPH'88*, pages 317-326, 1988.

[Eber94]D.S. Ebert, F.K. Musgrave, D. Peachy, K. Perlin and S. Woley. *Texturing and Modeling*. Academic Press, 1994

[Feyn65]R. Feynmann R. Leighton and M. Sands. *The Feynman Lectures on Physics*. Addison-Wesley, 1965.

[Fole90]J. Foley, A. van Dam, S. Feiner and J. Hughes. *Computer Graphics, Principles and Practice*. 2nd Ed., Addison-Wesley, 1990.

[Four82]A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *CACM*, 25(6), pages 371-384, June 1982.

[Four86]Alain Fournier and William T Reeves. A simple model of ocean waves. *SIGGRAPH'86*, pages 75 - 84, 1986.

[Flet91]N. Fletcher and T. Rossing. *The Physics of Musical Instruments*. Springer-Velag, 1991.

[Gasc94]J. D. Gascuel and M.P. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures, *Visual Computer*, pages 191-204, 1994.

[Gave93]W. Gaver. Synthesizing auditory icons, *Proceedings of INTERCHI*, 1993.

[Gira85]M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures, *SIGGRAPH'85*, pages 263-270, 1985.

[Glei94]M. Gleicher. A differential approach to graphical manipulation. PhD thesis, Carnegie Mellon University, 1994.

[Gold80]H. Goldstein, *Classical Mechanics*. 2nd Ed., Addison-Wesley, 1980.

[Grit95]L. Gritz and J. Hahn. Genetic programming for articulated figure motion. *Journal of Visualization and Computer Animation*, pages 129-142, July 1995.

[Hahn88]J. K. Hahn. Realistic animation of rigid bodies. *SIGGRAPH'88*, pages 299-308, 1988.

[Hahn95]J.K. Hahn, J. Geigel, J.W. Lee, L. Gritz, T. Takala, and S. Mishra. An Integrated Approach to Sound and Motion. *Journal of Visualization and Computer Animation*, Volume 6, Issues No. 2, pages 109-123, 1995.

[Hara95]M. Harada, A. Witkin and D. Baraff. Interactive physically - based manipulation of discrete/continuous models. *SIGGRAPH'95*, pages 199-208, 1995.

[Hodg95]J. K. Hodgins, W. L. Wooten, D.C. Brogan and J.F. O'Brien. Animating human athletics. *SIGGRAPH'95*, pages 71-78, 1995.

[Isaa84]P. M. Isaacs and M. F. Cohen, Controlling dynamics simulation with kinematic constraints, behavior functions, and inverse dynamics. *SIGGRAPH'87*, pages 215-224, 1987.

[Lee00]J.W. Lee, N. Baek, D. Kim, and J.K. Hahn. A procedural approach to solving constraints of articulated bodies, *Eurographics 2000*, (ISSN 1017-4656) pages 55-64 August 2000.

[Lee00]D.K. Lee, H.J. Bae, C.T. Kim, D.C. Lee, D.H. J, N.K. Lee, N.H. Baek, J.W.Lee, K.W. Ryu, and J.K. Hahn. Reproducing works of calder, *The Journal. of Visualization and Computer Animation*, (submitted on Dec 13 1999)

[Lewi90]J. Lewis. Automated Lip-Synch: background and techniques, *The Journal of Visualization and Computer Animation*, Vol 2, No.4, pages 118-122, 1990.

[Math69]M. Mathews. *The technology of computer music*, MIT Press, MA, 1969.

[McKe90]M. McKenna, and D. Zeltzer. Dynamics simulation of autonomous legger locomotion. *SIGGRAPH'90*, pages 29-38, 1990.

[Mile96]V. J. Milenkovic. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. *SIGGRAPH'96*, pages 129-136, 1996.

[Mirt95]B. Mirtich. Impulse-based simulation of rigid bodies. *1995 Symposium on Interactive 3D Graphics*, pages 181-188, 1995.

[Mirt96]B. Mirtich, Impulse-based dynamic simulations of rigid body systems, Ph.D. thesis, University of California, Berkeley, 1996.

[Moor88]M. Moore and J. Wilhelms. Collision detection and response for computer animation, *SIGGRAPH'88*, pages 289-298, 1988.

[Moor90]F. Moore, *Element of Computer Music*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[Ngo93]J. Ngo and J. Marks. Spacetime constraint revisited. *SIGGRAPH'93*, pages 343-350,1993.

[Patt89]A. Patterson. *A First Course in Fluid Dynamics*, Cambridge University Press, 1989.

[Peac86]D.R. Peachy. Modeling waves and surf. *SIGGRAPH'86*, pages 65-74, 1986.

[Pent89]A. Pentland and J. Williams. Good vibrations: model dynamics for graphics and animation. *SIGGRAPH'89*, Pages 207 – 214, 1989.

[Perl85]K. Perlin. Image synthesizer. *SIGGRAPH'85*, pages 287-296, 1985.

[Rade98]P. Rademacher and G. Bishop. Multiple-Center-of-Projection Images. *SIGGRAPH'98*, pages 199-206, 1998.

[Raib91]M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *SIGGRAPH'91*, pages 349-358, 1991.

[Reev83]W. Reeves, Particle systems – A technique for modeling a class of fuzzy objects, *ACM Trans. on Graphics*, 2(2):91-108, 1983.

[Reyn87]C. W. Reynolds, Flocks, herds and schools: A distributed behavior model, *SIGGRAPH'87*, pages 25-34, 1987.

[Rose96]C. Rose, B. Guenter, B. Bodenheimer and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. *SIGGRAPH'96*, pages 147-154, 1996.

[Scal91]C, Scaletti, The Kyma/Platypus computer music workstation in the well-tempered object: musical applications of object oriented software technology, Stephen Travis Pope, ed. MIT Press, 1991.

[Scho90]P. Schoroder and D.Zeltzer. The virtual erector set: Dynamic simulation with linear recursive constraint propagation. *In Proceedings 1990 Symposium on Interactive 3D Graphics*, volume 24, pages 23-31, March 1990.

[Shin92]M. Shinya and A. Fournier, Stochastic motion – motion under the influence of wind, *EUROGRAPHICS'92*, 11(3):119-128, 1992.

[Surl92]M. Surles, An algorithm with linear complexity for interactive, physically-based modeling of large proteins, *SIGGRAPH'92*, pages 221-230, 1992.

[Taka92]T. Takala and J. Hahn, Sound Rendering, *SIGGRPAH'92*, pages 211-220, 1992

[Tsin97]Nicolas Tsingos and Jean-Dominique Gascuel. A general model for the simulation of room acoustics based on hierarchical radiosity. *The art and interdisciplinary programs of SIGGRAPH '97*, page 149, 1997.

[Weil86]Jerry Weil. The synthesis of cloth objects, *SIGGRAPH'86*, pages 49-54, 1986.

[Wejc91]J. Wejchert and D. Haumann, Animation aerodynamics, *SIGGRAPH'91*, pages 19-22, 1991.

[Welm93]C. Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Fraser University, September 1993.

[West96]B. Westenhofer and J. K. Hahn. Using kinematics clones to control the dynamic simulation of articulated figures. *Proceeding of Pacific Graphics*, 1996.

[Witk90]A. Witkin, M. Gleicher and W. Welch. Interactive dynamics. *In proceedings 1990 Symposium on Interactive 3D Graphics*, volume 24, pages 11-21, March 1990.

[Witk96]A. Witkin and M. Kass. Spacetime constraints. *SIGGRAPH'96*, pages 159-168, 1996.

[Witt99]P. Witting, Computational fluid dynamics in a traditional animation environment. *SIGGRAPH'99*, pages 129-136, 1999.