The Virtual Piano Action: Design and Implementation

Brent Gillespie
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
brent@ccrma.stanford.edu

Abstract

The design of a virtual piano action composed of a keyboard of motorized keys and a real-time mechanical system simulation is presented. Using this simulator, we have re-created certain aspects of the feel of the grand piano by numerically integrating the equations of motion of a simplified piano action in real time in a human-in-the-loop simulation scheme. In this paper, the simulation of the release and catch of the hammer is used to introduce the simulator architecture. The structure of a software module which manages the simulation of models with changing kinematic constraints is discussed, including a finite state machine driver which allows for the simulation of rigid-body systems which may take on various constraint conditions in a sequence dependent on run-time interaction.

1 Introduction

Musical instruments are judged not only by how they sound, but also by what they feel like to play. Although modern synthesis algorithms have made it possible for synthetic instruments to very closely approximate the sounds of their acoustic counterparts, technology has not yet been able to adequately answer the musician's concerns about differences in feel. Moreover, in acoustic instruments there exists a tight correspondence between an instrument's acoustic response and its touch response. In fact, sometimes more information is available about an instrument's behavior through its feel than through its sound. For example, to get the fastest possible repetition on a piano, the performer detects the minimum key return necessary for reset of the jack under the hammer by feel rather than by sound. Re-establishment of a relationship between the mechanical behavior and the acoustic behavior in synthetic instruments will greatly increase the value of the information coming from the instrument over the haptic [1] channel and thus give musicians greater expressive control.

At CCRMA we are designing and building a virtual piano action–a keyboard action simulator (software) and a haptic display device (hardware) which together make it possible to emulate the feel of various keyboard instruments. A haptic display device is a motorized key or other manipulandum which, under computer control, makes it possible to interact with virtual objects through touch. Without a doubt, synthesizer controllers are an ideal application of haptic display technology. In fact, we believe that this technology will one day be a more viable means of creating the desired touch in commercial instruments than mechanical design with passive components. Of course, the programmability of the virtual piano action is its best selling point. The feel of a harpsichord, piano, forte piano or something altogether new would each be available at the push of a button.

Several robotics research labs are developing haptic display technology, most notably the labs at MIT, Northwestern, and North Carolina. Claude Cadoz' group, ACROE in Grenoble, has also developed a haptic display device for virtual musical instruments [Cadoz 93]. Their modeling and synthesis tools are designed to run on transputers, multiple parallel processing machines. By contrast, our tools are based on more common modeling techniques and designed to run on single processor platforms.

Our simulation testbed is capable of running real-time simulations of linear and non-linear mechanical systems, systems with and without memory [2] and systems with changing kinematic constraints. In this paper, we will address the construction of models and their incorporation into the simulator, paying particular attention to

[1]The word haptic is used to refer to the perceptual modality of touch which includes both taction –senses of the skin, and kinesthesia –senses of the muscles.

[2]A system with one or more degrees of freedom, or one which must be modeled with at least one independent state variable can be considered to have 'memory'. Its present state depends on past input history.

the accommodation of changing kinematic constraints. Discontinuous systems (i.e., those in which bodies may make and break contact with one another) are to be core members of this simulator's repertoire. Some of the most interesting haptic cues in our environment arise from changes in kinematic constraints, a fact which is also true for keyboard actions. Examples include the release of tension when the plectrum plucks the strings of the harpsichord, and the extra resistance during escapement in the piano action.

In the most general rigid-body mechanical system, the various constraint conditions may be taken on in any order, depending on how other systems (possibly a user) interact with it. Barzel [Barzel 92] and others have addressed the realization of discontinuous systems by simulation of a sequence of ordinary differential equations. In our work, we adopt their nomenclature and combine it with a Finite State Machine (FSM) simulator, which will allow a sequence of conditions or 'states' to be taken on in an order which is not known ahead of simulation-time.

Building a model suitable for simulation with haptic display which duplicates all aspects of the complex behavior of the piano action will necessarily be a step-by-step process. Section 2 to follow outlines the structure of a model suited for our simulator. Section 3 presents a (piece-wise) linear model of a bouncing ball and its simulation algorithm. In section 4, we construct a non-linear model which covers the lever action of the key and hammer and discuss its simulation algorithm. The incorporation of a finite state machine manager to handle the addition of a virtual keybed into the model is discussed in section 5. Experimental results are mentioned in 6. Finally, summaries are made and future work discussed in section 7.

2 Model Components

Of all the various forms in which mechanical system models are expressed, a set of reduced ordinary differential equations (ODEs) (with the constraints incorporated) is the simplest. We have chosen to base our simulator on models expressed as reduced ODEs. In so doing, we hope that we will be led to investigate some of the more intricate problems involved with real-time simulation.

The model itself is formulated as a piece-wise continuous ODE. Discontinuities are allowed at timepoints corresponding to changes in the kinematic constraints. The time periods between the discontinuities are each governed by one of a set of 'submodels', each of these being a continuous ODE constructed to describe the system in one of its constraint conditions.

For a more complete introduction to the nomenclature used to describe these piecewise continuous ODEs, see either [Barzel 92] or [Gillespie 93]. Here, we briefly summarize. Submodels are divided into three parts to facilitate decision making by the simulator as to which submodel is to govern the behavior at a given time. The three submodel components are: the equations of motion, a readout equation, and an indicator function. The equations of motion are solved numerically to maintain up-to-date state variables at all times. The readout equation is an expression for the output (in our case the response force) in terms of the input motion. The indicator function is tested once per servo cycle to indicate if it is time to switch to the next submodel.

3 The Bouncing Ball

Here we describe a very simple system, a ball which bounces on a vertically moving paddle. It is linear and has only two submodels: ball in the air, ball on the paddle. After introducing this system, we will claim that it is actually a good model of the piano action.

Figure 1: *The Bouncing Ball*

Figure 1 shows the two submodels which make up the bouncing ball: a) the ball is attached to the paddle through a spring, and b) the ball is flying vertically in the air, free from the paddle. For submodel a), the equation of motion, readout equation, and indicator function are, respectively:

$$\ddot{q} = \frac{k}{m_b}(q - d) - g \tag{1}$$

$$F = m_p\ddot{d} + k(q - d) + m_p g \tag{2}$$

$$k(q - d) < 0 \tag{3}$$

The indicator function (3) evaluates to TRUE when the ball/paddle interaction force is tensile, signalling the end of applicability of model a). For the ball in air submodel, the equation of motion, readout equation, and indicator function are:

$$\ddot{q} = -g \tag{4}$$

$$F = m_p\ddot{d} + m_p g \tag{5}$$

$$q - d < 0 \tag{6}$$

The indicator function (6) evaluates to TRUE when there is interference between the ball and paddle.

A linear differential equation such as we have here is always expressible in state space form as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u \tag{7}$$

This differential equation can be converted to a difference equation

$$\mathbf{x}_{n+1} = \mathbf{\Phi}\mathbf{x}_n + \mathbf{\Gamma}u_n \tag{8}$$

suitable for simulation on a digital computer. The discrete equivalent matrices $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are given in terms of the continuous matrices $\mathbf{A}$ and $\mathbf{b}$ and the time step T by

$$\mathbf{\Phi} = e^{\mathbf{F}T} \tag{9}$$

$$\mathbf{\Gamma} = \int_0^T e^{\mathbf{F}\eta} d\eta \mathbf{b} \tag{10}$$

Several common computer algorithms with good numerical properties are available to do the conversion. The simulation algorithm then simply involves a matrix multiply to advance the simulation by one time step T.

Because these equations are second order linear ODEs, analytical solutions exist. There is in fact no need to solve the differential equations numerically for this simple model. The state can be expressed as a function of the input and time. The force output is computed as a function of the motion input using the readout equation of the applicable submodel until such time that the indicator function evaluates to a negative number. The simulator then exchanges submodels, using the final conditions of the last as the initial conditions of the next submodel. Note that in this case, the 'next' submodel is just the *other* submodel. This rather simplistic model has created a very convincing virtual bouncing ball when implemented with the haptic display device. Interaction between the ball and user through a motorized key (in this case to be viewed as a paddle handle) includes all the properly timed power exchanges to suggest manipulation of a bouncing object. In summary, we have implemented a unilateral constraint (a gross non-linearity– a contact capable of supporting compressive but not tensile forces) by combining two linear submodels with some management routines for exchanging them in and out of the simulator.

4 A Simplified Piano Action

Figure 2 shows a simplified schematic diagram of the piano action. This model has only two bodies, the key and hammer. The letoff function of the whippen and jack are not modeled. The hammer and key are coupled with a unilateral constraint. A spring accounts for compliance in the action, most of which is due to softness of the hammer knuckle. The other submodel in which the hammer flies free of the key is not shown; it can be surmised. This model will behave like a piano action in which the regulation button is set too high, inactivating the letoff and repetition functions.

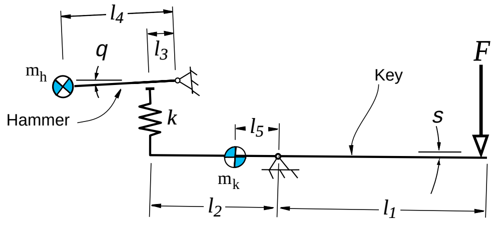

Figure 2: *Simplified Piano Action*

The non-linear equations of motion, readout equations and indicator functions are not presented here. The simulation is realized in this case with an ODE solver instead of the difference equation (8). A Runge-Kutta or other numerical ODE solution routine is responsible for advancing the state by each time step using the previous state and the input (key motion). As before, the force output is given by the readout equation, and the indicator function is tested each time step.

Because the angles through which both the key and the hammer move are rather small, several linearizing assumptions can be made in the construction of a piano action model. Specifically, we shall assume that all interaction forces and gravity forces act perpendicular to the bodies to which they are applied as seen in Figure 2. Also, the force of the key on the hammer is applied at a fixed position on the hammer determined by l_3.

Given these assumptions, the equations of motion, readout functions, and indicator functions respectively are as follows:

$$\ddot{q} = \frac{kl_3}{I_h}(l_2 s + l_3 q) - \frac{m_h l_4 g}{I_h} \tag{11}$$

$$F = \frac{I_k}{l_1}\ddot{s} - \frac{kl_2}{l_1}(l_2 s + l_3 q) + \frac{m_k l_5 g}{l_1} \tag{12}$$

$$k(l_2 s + l_3 q) < 0 \tag{13}$$

See [Topper 87] for an explicit derivation of the equations of motion for this model.

Note that the function of the action is very much like that of the ball and paddle in the model outlined above: to throw the hammer toward the string and then catch it again. The simple addition of a ceiling for the ball to bounce off of (a virtual string), and an inversion of the paddle's motion to reflect the fact that the hammer is actuated from the opposite side of the key fulcrum will turn the bouncing ball model into a good first approximation of the piano action. After further assuming that inertia forces dominate over gravity forces in the coupled hammer-key model, appropriate mass and spring values for an approximating ball and paddle model can be deduced by comparing equations (1), (2), (3) with (11), (12), (13).

5 Finite State Machine

A useful addition to our model is a virtual keybed. The key dip differs between a harpsichord, fortepiano and piano, and this is an aspect we would like to include in our keyboard simulator.

The method outlined so far only accommodates models in which the sequence of submodels is known: going back and forth between two submodels. Depending on the manner in which the key is depressed, either the hammer could fly free or the key could meet the keybed first. The other change in condition may not follow, again depending on how the key was depressed. [3] In order to manage the sequencing through various submodels, we employ a finite state machine simulator.

A finite state machine is a dynamical system capable of taking on a finite number of states in a possibly complex sequence of transitions from a particular state to certain others of the set of possible states. A finite state model is fully specified by its state transition graph, one of which is shown in Figure 3. This finite state model is for the simplified piano action with a virtual keybed. Coupling between bodies is noted in Figure 3 by spring icons. Only certain transitions are allowed. Associated with each transition path is an indicator function which, upon evaluation to a number less than zero, indicates that it is time to transition to the model pointed to by that path.

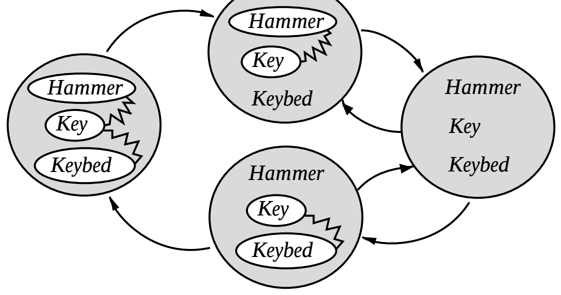

Figure 3: *State Transition Graph for the Piano Action*

[3]One can effect fast light playing on the piano by not completely bedding the keys.

6 Experimental Results

We have conducted several introductory experiments using this apparatus. Subjects have made side-by-side key-press comparisons between the above virtual action and a physical action with its regulation button removed with promising results. We have begun to address real-time simulation problems such as extra energy introduced into the simulation by model transition timing errors with compensating additions to the simulation algorithm.

7 Summary

We have presented a modeling and simulation algorithm which accommodates dynamical systems with changing kinematic constraints and provides for the re-creation of their mechanical impedance by simulation and haptic display. The method involves modeling the system in each of its constraint conditions. Readout equations expressing the force output in terms of the state variables as well as indicator functions which signal the end of applicability accompany each model. The sequence of models can be considered a piece-wise continuous ODE. If the model is linear, it can be discretized and then simulated with a difference equation. Otherwise, an ODE solver is used. The method is also useful for systems in which the sequence of constraint conditions is not known ahead of time with the addition of a submodel manager based on a finite state machine driver.

A model of a bouncing ball and a simplified piano action were presented.

References

[Barzel 92] R. Barzel. *Physically-Based Modeling for Computer Graphics*, Academic Press, Boston, 1992.

[Cadoz 93] C. Cadoz, A. Luciani, J-L. Florens. *CORDIS-ANIMA: A modeling and simulation system for sound and image synthesis–the general formalism*, Computer Music Journal, Vol 17, No. 1, Spring 1993, pp.19-29.

[Gillespie 93] B. Gillespie, M. Cutkosky. *Interactive dynamics with haptic display* In Proceedings of the 1993 ASME Winter Annual Meeting, New Orleans, pp. 65-72.

[Topper 87] T. Topper, B. Wills. *The computer simulation of piano mechanisms* International Journal of Modelling and Simulation, Vol. 7, No. 4, 1987.