Synthesizing Contact Sounds Between Textured Models

Zhimin Ren* Hengchin Yeh† Ming C. Lin‡

Department of Computer Science
University of North Carolina at Chapel Hill, USA

Abstract

We present a new interaction handling model for physics-based sound synthesis in virtual environments. A new three-level surface representation for describing object shapes, visible surface bumpiness, and microscopic roughness (e.g. friction) is proposed to model surface contacts at varying resolutions for automatically simulating rich, complex contact sounds. This new model can capture various types of surface interaction, including sliding, rolling, and impact with a combination of three levels of spatial resolutions. We demonstrate our method by synthesizing complex, varying sounds in several interactive scenarios and a game-like virtual environment. The three-level interaction model for sound synthesis enhances the perceived coherence between audio and visual cues in virtual reality applications.

Index Terms: H.5.5 [Sound and Music Computing]: Modeling—Systems; H.5.2 [User Interface]: Auditory feedback.

1 Introduction

In everyday life, sound generated by interactions among rigid bodies is ubiquitous. To realistically portray the real world in a virtual environment (VE), good audio feedback can augment and greatly enhance the visual display. Foley artists construct soundtracks manually to provide realistic audio for feature animations, computer games, and virtual environment applications in an off-line fashion. While such a practice is feasible and common, it is a time-consuming, tedious, and repetitive process. Moreover, synchronizing events and sound in VEs takes a considerable amount of training and experience.

Physics-based sound synthesis provides an automatic mechanism to drive the sound synthesis process using dynamic simulation of physical events in the scene, making it possible for the same simulations to automatically create both visual and auditory displays at the same time. Recently there has been much progress made on sound synthesis for solids using rigid-body dynamics simulation [13, 15, 19]. However, existing synthesis models for sound due to frictional contacts have been limited to mostly parametric methods [19] or physics-based simulations that compute the non-linear friction [3, 4]. The more accurate physics-based methods reproduce some complex friction phenomena (e.g. stick-slip), but these interaction models require manipulating many physical parameters, which is a non-trivial task. Traditionally vision has been considered the dominant modality in the human multi-sensory perception. Frequently real-time performance requirement necessitates the use of various forms of texture representations, such as bump maps, normal maps, displacement maps, etc., to augment simpler polygonal models by accelerating graphical rendering while enhancing visual realism. Off-line recording and existing synthesis methods do not capture all sounds due to real-time interaction commonly encountered in a game-like scene or VR environments that heavily rely on textured models for real-time rendering. As a result, visual-auditory disparity can arise – causing cross-sensory conflicts and disrupting the user's sense of immersion.

*e-mail: zren@cs.unc.edu
†e-mail:hyeh@cs.unc.edu
‡e-mail:lin@cs.unc.edu

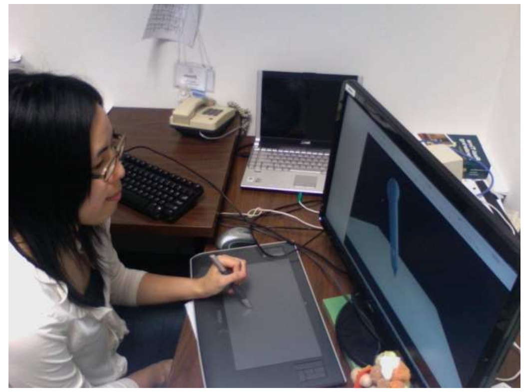


Figure 1: **The System Setup.** A user is synthesizing sound using a tablet connected to our sound rendering system by moving the stylus to interact with the virtual environment.

Main Results: We present a new algorithm to display *contact sounds* between *textured models* using a novel **three-level surface representation**:

- **Macro level**: the polygonal geometry;
- **Meso level**: the texture information (e.g. bump maps, normal maps, displacement maps, etc);
- **Micro level**: material roughness (e.g. friction).

The integration of the three levels influences both visual and auditory feedback *simultaneously*, reinforcing the crossmodal perception that may otherwise be interfering with each another due to visual-aural disparity caused by texturing, lighting, and shading. The resulting system (see Figure 1) is able to handle complex, varying interaction among rigid bodies and achieve real-time sound synthesis.

2 Related Work

Design of auditory display involves audio hardware technology, software rendering pipeline, modeling and simulation of acoustic spaces, signal processing, sonification techniques, perceptual evaluation, and application development. The two compute-intensive components of auditory displays are *sound synthesis* and *sound propagation*.

Sound propagation deals with computational modeling of acoustic spaces that takes into account the knowledge of sound sources, listener locations, 3D models of the environments, and material absorption datato compute impulse response (IR) of the space. These IRs are convolved with recorded or synthetically produced sound signals to generate spatialized sound effects, i.e. *auralization*. In general, sound propagation is complimentary to the synthesis process. We refer the interested readers to a recent survey [11] on this topic. For simplicity and other practical concerns (cost and portability across heterogeneous desktop PCs, etc), we ignored sound propagation in our current system, and all sounding objects are considered to be a point source, which generates sound to all directions. Such a simplification can be remediated by using an available room filter to cheaply and efficiently emulate room acoustics – a common practice for some interactive games.

Sound synthesis deals with how sound is generated using physical principles [11] and is the focus of our work. Next, we will briefly describe some related work on sound synthesis between rigid objects.

- **Sound simulation framework**: Takala and Hahn [18] proposed a general sound rendering pipeline to computer graphics community. The pipeline includes sound synthesis, synchronizing sound sources with objects and animation, and sound propagation simulation. To physically-based synthesize sound, Van den Doel and Pai [21] describes a general framework, using the well-known vibration mode analysis, i.e. *modal analysis*, in computational mechanics for sound synthesis. This approach generates sound dependent on the materials, shapes, and struck positions of the simulated sounding objects. Their paper also presented analytical solutions to the vibration analysis for simple shapes like rectangular membrane, strings, and bars. However, this analytical model cannot handle more complex and arbitrary shapes. O'Brien et al. [12] used finite element method (FEM) to more accurately model the surface vibration of virtual objects.

  In the area of computer music, there is also abundant research on sound synthesis for digital instruments. Most methods concentrate on simulating digital instruments through various types of digital signal processing, modal synthesis applied to specialized setting (e.g. strings, tubes, membranes, and more), etc. Cook [10] provided an excellent review on techniques related to this topic. Bruyns [8] introduced the creation of digital software synthesizer for arbitrary shapes using modal analysis. Chadwick et al. [9] proposed coupling modes generated from linear modal analysis with nonlinear thin-shell force to produce more accurate thin-shell sounds. We target at real-time applications, so decoupled *modal analysis* is used in this paper.

- **Modal parameter extraction**: A number of methods have been proposed for analyzing objects' vibration modes and obtaining modal parameters used later in *modal synthesis*. As mentioned previously, there are analytical solutions for simple shapes. For arbitrary shapes, Van den Doel et al. [14] built a robotic device that can measure the impulse response of an object when being hit at various locations and fit the parameters to the recorded and scanned data. While this approach is a feasible solution to synthesizing sound for arbitrary objects, the hardware constraint limits its application. Later, approximation algorithms were proposed for handling arbitrary shapes. A finite element method (FEM) was proposed for modal analysis by O'Brien et al. [13], while Raghuvanshi and Lin [15] suggested converting arbitrary triangle meshes into spring-mass systems for calculating the vibration modes. As we are targeting toward VR applications, our system adopted the spring-mass approximation for its simplicity, generality, and most of all its runtime efficiency.

- **Interaction models for modal synthesis**: Existing physics engines (e.g. NVIDIA's PhysX [1]) for rigid-body simulation can be used for reporting impacts and directly converting impacts into excitation to modal synthesis. [15] uses this scheme for generating impact sound. Rolling sound can also be treated as a series of impact sounds triggered by the tessellated geometry colliding against each other. O'Brien et al. [13] retrieve the collision information from physics engine, convolve it with a Gaussian kernel, and feed this result into the modal model to create a soft collision sound. However, neither method can generate sound from continuous contacts (i.e. sliding sound), because the physics engine cannot run fast enough to calculate contact force at audio sampling rates. Van den Doel et al. [19] generates a fractal-noise based force profile that is sampled at audio rates to simulate the variation of friction force. This model reflects the contact velocity, contact normal force, and the roughness of the material by changing the playback parameters of the force profile. The method is fast but produces repetitive sounds, and only works well when surface roughness is homogeneous in a large area. Avanzini et al. [3] proposed a more accurate friction simulation model for rubbed dry surfaces, but such calculations require tweaking a large number of control parameters in order to achieve convincing sliding sound effects. This is not practical for complex virtual environments that involve many sounding objects with different physical properties.

- **Acceleration techniques for modal synthesis**: *Modal synthesis* methods are suitable for real-time sound synthesis. However, when the number of sound sources is large, this technique does not scale well. Raghuvanshi and Lin [15] introduce perceptually-driven acceleration methods like mode compression, mode truncation, and quality scaling. They can interactively simulate complex scenes with hundreds of sounding objects. We adopted their techniques to improve the performance of our system. Similar improvement can also be achieved with frequency domain processing proposed by Bonneel et al. [7].

3 Overview

Next, we provide an overview of our system and the sound rendering pipeline. In a virtual environment (VE), objects move and interact with one another. As these objects come into contact, different sound should be generated depending on where the contact is taking place, the duration of the interaction, and the type of surface interaction. Our sound rendering system is designed to synthesize these different sounds, and a high-level system overview is shown in Figure 2. We describe the *sound synthesis* and *interaction handling* modules in more details below.

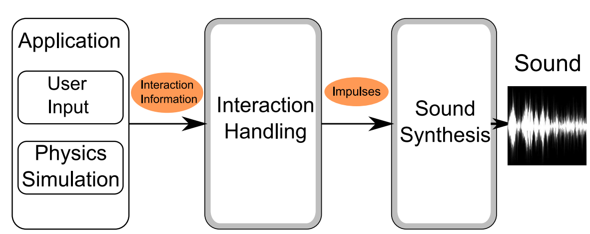


Figure 2: **System Overview:** Given the user input to manipulate the virtual objects in the VE and physics engine to simulate various types of physical interaction due to the user manipulation, our sound synthesis system can take the interaction data, classify the contact events and transform the interaction forces into a sequence of impulses that drive our sound synthesis module to generate the corresponding sound automatically.

During the pre-processing, modal analysis module takes any arbitrary triangle mesh of each model in the VE as input, converts each mesh into a spring-mass system, and approximates various vibration modes of the original mesh as a bank of damped oscillators (as shown in Figure 3) that can be excited by external forces to generate sounds.

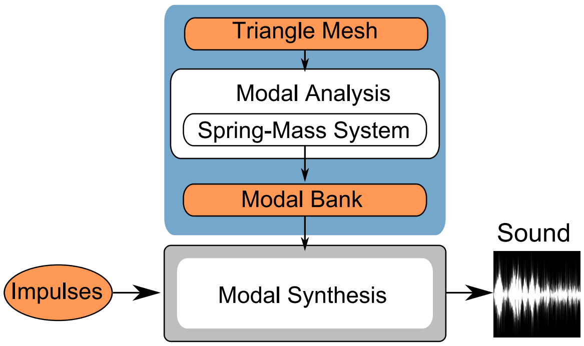


Figure 3: **Sound Synthesis Module:** Given a triangle mesh of each object in a VE as input to the sound synthesis module, modal analysis transforms every mesh into a spring-mass network in our current implementation during the pre-processing (the box above Modal Synthesis). It also computes a bank of frequency modes and represents them as a bank of damped oscillators, or 'modal bank', for each given geometric mesh. At runtime, the modal synthesis uses the train of impulses due to collisions to excite damped oscillators to generate rich sounds corresponding to various types of surface contacts arising from object interaction.

At runtime, the VE application provides and updates the information (e.g. geometry, location, and velocity) for all objects and their interaction information, including their current contact status (separated or colliding), when, where, and how they come into contact (i.e. bouncing off each other, or sliding/rolling over each other). The interaction handling part of our system is to analyze the interaction data due to contact and transform them into a series of collision impulses, shown as the interaction handling module in Figure 4.

The external excitation to the sound synthesis module is generated by the interaction handling module, which detects collisions, classifies them into lasting or transient contacts, and generates impulses to approximate these contacts. These impulses act as the external excitation to the modal synthesis module. The modal synthesis module in turn takes these impulses as input, approximates the oscillating responses of objects when the impulses are applied, and then generates sound from the approximated vibration (as shown in Fig. 2).

The interaction handling and sound synthesis are the core components of our system. We will first explain how the sound synthesis module works in Section 4, and then present the interaction handling in Section 5.

4 Modal Analysis for Sound Synthesis

Sound is a traveling wave produced by the variation of medium pressure, which is caused by the vibration of objects. The pressure oscillation at frequencies between about 20 and 20K Hz can be heard by human auditory systems. To simulate the physical process of sound generation, we need to model the mechanical vibration of sounding objects. This vibration may not be visually noticeable, but can make a considerable difference to human ears.

Many recent real-time physics-based sound synthesis methods adopt the modal synthesis approach for discretely approximating the vibration of sounding objects [13, 15, 19, 21]. The complete process is composed of two stages: modal analysis (in preprocessing) and modal synthesis (during run-time). Modal analysis represents the vibration of an arbitrarily shaped object with a bank of damped harmonic oscillators. In this process, the amplitude, damping, and decay coefficients are extracted from the mesh geometry for each sounding object. The next process, modal synthesis, approximates the vibration caused by external force applied on the object using a linear combination of the damped oscillators determined by modal analysis. Next, we briefly introduces modal analysis and synthesis, as well as its application in our sound synthesis system.

4.1 Modal Analysis

Each sounding object can be viewed as a continuous system. To represent its vibration for sound synthesis, model. Different discretization approaches can be adopted for models of different shapes to obtain the parameters for the modal representation. Modal analysis [16] is a well-known technique in computational mechanics for modeling the structural vibration of objects and we adopt this technique to model the surface vibration leading to sound generation. For some simple shapes, first principles can be used to solve for the parameters [21]. For an arbitrary shape, finite element methods (FEM) can be used to discretize the objects [13]. The physics properties of this geometry can also be modeled with a spring-mass system [15]. Finally, the parameters can be fitted to recordings of real objects [14].

In our sound synthesis system, we adopt the mass-spring representation for modal analysis. This representation is less accurate compared to FEM, but it is much faster. Therefore, it is more suitable for real-time VR applications, because the materials and shapes of the objects can be changed on the fly. In the mass-spring representation, each vertex of the input triangle mesh is considered as a particle mass, and each edge between two vertices is considered as a damped spring. Different parameters used in the mass-spring system construction creates different modal models (i.e. frequencies, damping, and mode shapes) that sound like different materials. We refer the readers to [15] for more details on the input processing.

The mass-spring system created from the input mesh forms an ordinary equation (ODE) system as below:

$$M\frac{d^2r}{dt^2} + C\frac{dr}{dt} + Kr = f \tag{1}$$

where M, C, and K are respectively the mass, damping, and stiffness matrix. If there are N vertices in the triangle mesh, r in Eqn. 1 is a vector of dimension N, and it represents the displacement of each mass particle from its rest position. Each diagonal element in M represents the mass of each particle. In our implementation, C adopts Rayleigh damping approximation, so it is a linear combination of M and K. The element at row i and column j in K represents the spring constant between particle i and particle j. f is the external force vector. The resulting ODE system turns into:

$$M\frac{d^2r}{dt^2} + (\gamma M + \eta K)\frac{dr}{dt} + Kr = f \tag{2}$$

where M is diagonal, and K is real symmetric. Therefore, Eqn. 2 can be simplified into a decoupled system after diagonalizing K with $K = GDG^{-1}$, where D is a diagonal matrix containing the eigenvalues of K. The diagonal ODE system that we eventually need to solve is:

$$M\frac{d^2z}{dt^2} + (\gamma M + \eta D)\frac{dz}{dt} + Dz = G^{-1}f \tag{3}$$

where $z = G^{-1}r$, a linear combination of the original vertex displacement. The general solution to Eqn. 3 is:

$$z_i(t) = c_i e^{\omega_i^+ t} + \bar{c}_i e^{\omega_i^- t}$$

$$\omega_i^{\pm} = \frac{-(\gamma\lambda_i + \eta) \pm \sqrt{(\gamma\lambda_i + \eta)^2 - 4\lambda_i}}{2} \tag{4}$$

where λ_i is the i'th eigenvalue of D. With particular initial conditions, we can solve for the coefficient c_i and its complex conjugate,

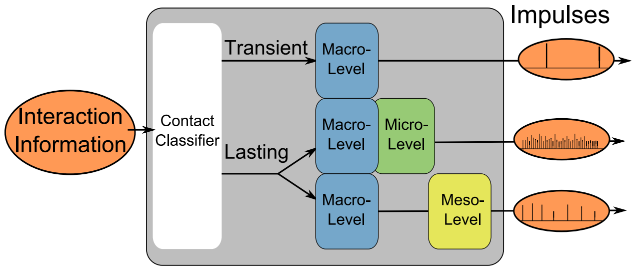


Figure 4: **Interaction Handling:** Given contact information, this module will classify the type of contacts based on velocity and contact normals. It then uses the three-level surface representation for contact handling to generate impulses that drive the sound synthesis module.

$\bar{c}_i$. Therefore, the vibration of the original triangle mesh is now approximated with the linear combination of the mode shapes z_i.

4.2 Impulse Response and Modal Synthesis

When an object experiences a sudden external force f that lasts for a duration of time, Δt, we say that there is an *impulse* $f\Delta t$ applied to the object. This impulse either causes a resting object to oscillate, or changes the way it oscillates, we say that the impulse *excites* the oscillation. Mathematically, since the right-hand side of Eqn. 3 changes, the solution of coefficients c_i and $\bar{c}_i$ also changes in response, which is called the *impulse response* of the model.

The impulse response, or the update of c_i and $\bar{c}_i$, for an impulse $f\Delta t$ follows the rule[15]:

$$c_{i,t_0+\Delta t} = c_{i,t_0} e^{\omega^+ t_0} + \frac{g_i}{m_i(\omega_i^+ + \omega_i^-)}$$
$$\bar{c}_{i,t_0+\Delta t} = \bar{c}_{i,t_0} e^{\omega^+ t_0} - \frac{g_i}{m_i(\omega_i^+ + \omega_i^-)} \quad (5)$$

where g_i is the i'th element in vector $G^{-1}f$. Whenever there is an impulse acting on an object, we can quickly compute the approximated displacement of the mesh representing the object at any time instance onwards by plugging Eqn. 5 to Eqn. 4.

5 Interaction Handling

In the previous section we have discussed how to generate sounds, once the impulses applied to the object are given. In this section we will explain how to actually produce these impulses from the complex interactions that take place in the VE application. Due to performance constraints of real-time sound synthesis, these impulses approximate the complex interactions but still retain the characteristics.

We present a novel *three-level interaction handling* approach that models various interactions. The pipeline of this approach is shown in Figure 4, which provides a more detailed view of the second block of Figure 2. The approach requires first categorizing the interaction among objects into *lasting contact* and *transient contact*. These contacts are then handled by three-level surface representation for contact handling to generate sound. Sounds generated using this representation have contributions from different levels of surface details for different types of contacts: transient contacts can be sufficiently handled using the macro-level geometric representation alone, while lasting contacts are handled using all three-levels of surface representation. The micro-level geometry aims at simulating the friction interaction at audio sampling rate and provides the overall roughness of the contacting material. The meso-level representation provides the variation of sound caused by the bumpiness of the material that is typically encoded in some forms of texture maps for visual rendering. The ridges and troughs at this level

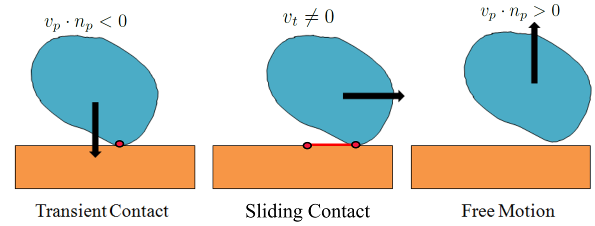


Figure 5: **Different Contact States.** The arrows indicates the linear velocity of the object. The dots indicate the contact point, and the line between them indicates the contact area.

are both visible from the screen and perceivable from the synthesized sound using our new representation for contact-handling. The macro-level simulation is updated at the physics engine's time step, so it can provide the shape and contact information on the scale that the rigid-body simulator can handle. The three-level representation for simulating contact sounds is illustrated in Figure 6 and we will elaborate it next.

5.1 Contact Categorization

We adopt the state and event computation from the event-based approach developed by Sreng et al. [17] to identify and categorize contacts, using the position, velocity and geometry information of the objects.

Two objects are said to be *contacting* if their models overlap in space at a certain point p, and if $\mathbf{v_p} \cdot \mathbf{n_p} < 0$, where $\mathbf{v_p}$ and $\mathbf{n_p}$ are their relative velocity and contact normal at point p.

Two contacting objects are said to be in *lasting contact* if $\mathbf{v_t} \neq 0$, where v_t is their relative tangential velocity. Otherwise they are in *transient contact*. The process is illustrated in Figure 5.

Lasting Contacts: Sliding contacts are ubiquitous. When any two solid objects scrub against each other, there is a sliding contact. However, it is a very difficult task to simulate the micro-level collision of objects, which is essential for modeling the friction forces that actually excite the vibration of surface resonators during a sliding contact.

There are mainly two different approaches to simulate the friction interaction between two objects: physics-based and parametric models. Each has its strength and issues. The physics engine normally has a simulation rate on the order of 100Hz, which is much lower than the audio sampling rate (i.e. 44100Hz). If we choose to simulate the physics of friction faithfully, it would be impossible to achieve real-time simulation rate. In addition, it is infeasible to obtain the roughness geometry at such a micro level. The fractal noise friction model introduced by Van den Doel et al. [19] is a good approximation of the friction force at the micro level. However, the method only emulates the micro-level interaction between materials that are visually smooth. Some intermediate-level details of the object cannot be simulated with only fractal noise excitation.

Transient (Impact and Rolling) Contacts: For simulating the impact and rolling sound, we adopt the method of Raghuvanshi and Lin [15]. When the interaction handling module detects a transient contact, an impulse is added to the sound synthesis module. The magnitude of the impulse is modulated by the magnitude of the relative velocity between the two colliding objects. Rolling sound is generated by adding a sequence of impulses to the sound synthesis engine. This is feasible due to the geometry tessellations of models used in graphics applications. Normally, a number of discrete geometries are used to approximate the smooth curvature of objects. The rigid-body simulator automatically reports contacts between the tessellated geometries, and corresponding impulses are added to the sound synthesis module.

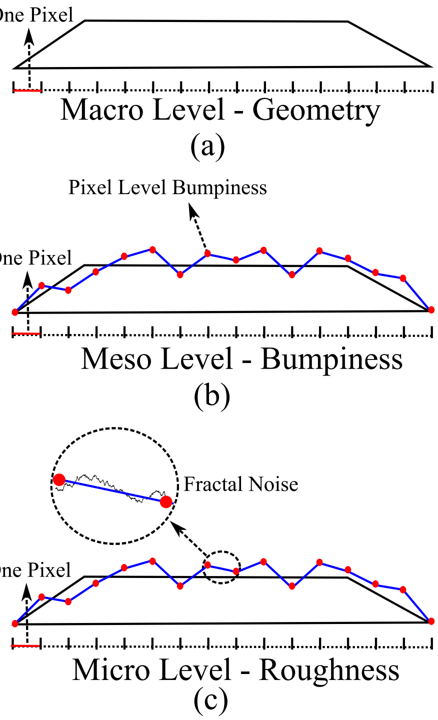


Figure 6: **The Three-level Contact Surface Representation.** (a) The trapezoid conceptualizes the geometry of the object. (b) The wiggly curve represents the surface of the geometry after the surface normals being changed by a normal map. (c) Within one pixel, the roughness of the surface is represented by a fractal noise. The geometry, bumpiness, and roughness models all contribute to various levels of frictional interaction.

5.2 Three-Level Surface Representation

In this section, we describe our novel three-level representation for contact handling to synthesize sounds.

The Macro Level: Geometry

The macro shapes are represented by the input triangle meshes of objects. These macro-level geometries are used for handling collision and computing forces in the rigid body simulator.

The Micro Level: Friction

The roughness of the contacting material is reflected by the micro-level simulation of friction sound. We use the method proposed by Van den Doel et al. [19] to generate an approximated force profile at this fine level. A fractal noise is used as the force profile, and the spectrum of the fractal noise varies with the auditory roughness of the material. The force profile is stored in a wave-table and played back to give users the sound that varies at audio sampling rate. The wave-table play-back speed is governed by the contact speed to give users the feeling of scratching through the grainy material fast or slowly. The magnitude of the impulse added also linearly varies with the normal force between the two objects scrubbing against each other. In summary, this parametric model reflects the contact speed, contact normal force, and the roughness of the material at the micro level.

The Meso Level: Bumpiness

Solely using the micro-level force profile generated by a fractal noise to excite the resonators does not render any information for the bumpiness or *heterogeneous* variation of the contacting geometry at the meso level. Many graphics applications use bump mapping, normal mapping, and height mapping for rendering the complicated bumpiness of materials, using image-based representations. This level of details is clearly visible to the users but transparent to the rigid-body simulator; in contrast, the micro-level details are neither seen by the users nor by the physics engine.

Barrass and Adcock considered using bumpiness as a single surface-level granular synthesis to generate sound due to granular interaction [5]. In contrast, our synthesis method takes the normal map from the visual rendering and considers this pixel-level information as small geometries.

Imagine an object in sliding contact with another object, whose surface F are shown in Figure 7(a), the contact point traverses the path P within a time step. We look up the normal map associated to F and collect those normals around P. The normals suggest that the high resolution surface looks like f in Figure 7(b), and that the contact point is expected to traverse a path P' on f. Therefore, besides the momentum along the tangential direction of F, the object must also have a time-varying momentum along the normal direction of F, namely, $\mathbf{p_N}$, where $\mathbf{N}$ is the normal vector of F. From simple geometry (Figure 7(c)), we compute its value

$$\mathbf{p_N} = m\mathbf{v_N} = mv_N\mathbf{N} = m\left(-\frac{\mathbf{v_T}\cdot\mathbf{n}}{\mathbf{N}\cdot\mathbf{n}}\right)\mathbf{N},$$

where m is the object's mass, $\mathbf{v_T}$ is the tangential velocity of the object relative to F. The impulse along the normal direction $\mathbf{J_N}$ that applies on the object is just the change of its normal momentum:

$$\mathbf{J_N} = \mathbf{p_N}(i) - \mathbf{p_N}(j),$$

when the object moves from pixel i to pixel j on the normal map. With this formulation, the impulses actually models the force applied by the bumps on the surface of one object to another, generating sound naturally correlated with the visual appearance of bumps from textures.

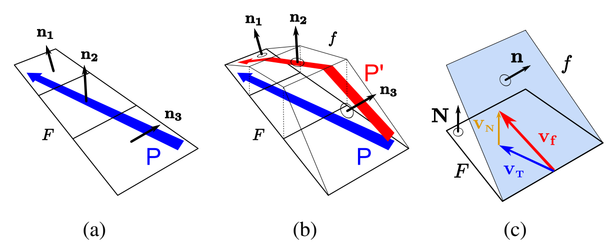


Figure 7: **Impulse Computation.** (a) The path P traced by an object sliding against another object within a time step, and the normals stored in the normal map around the path. The path lies on the surface F, which is represented coarsely with a low-resolution mesh (here a flat plane). (b) The normal map suggests that the high-resolution surface looks like f, and the object is expected to traverse the path P'. (c) The impulse along the normal direction can be recovered from the geometry configuration of $\mathbf{n}$, $\mathbf{N}$, and $\mathbf{V_T}$.

6 Implementation and Results

We have implemented the method described in this paper using C++ and integrated it with OGRE3D, an open-source graphics rendering engine [2].

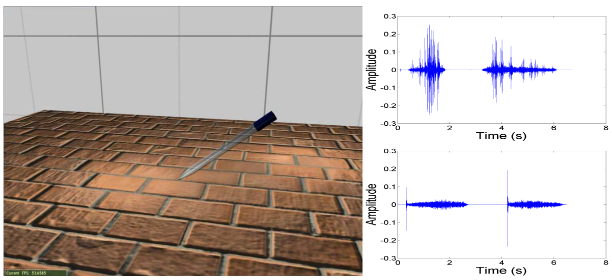
6.1 User Interface

In designing the user interface to our sound synthesis system, we attempt to minimize the need for key-presses, mouse input, and any complex control that are required from non-technical users. Inspired by the intuitive user interface provided by the virtual painting system [6], our system also takes user input from a Wacom Intuos tablet. Users can create sounds by simply moving the stylus on the tablet with very minimal keyboard input. Figure 1 shows an user using the system to synthesize sound of a pen scrubbing against a surface. This simple interface allows users to intuitively interact with the virtual objects in the synthetic environment.

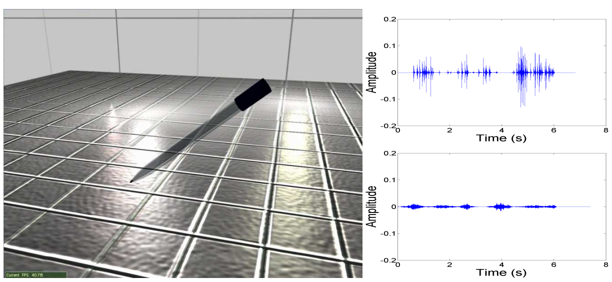
Users also have the flexibility to change the material parameters to design and synthesize the sounds that they desire to closely match the graphics rendering. By giving users the freedom to choose material parameters, we also introduce some difficulty in how to select the right parameters for some inexperienced users. We reduced this difficulty by providing the users a repository of materials. The sound synthesis parameters for many representative and normal materials in everyday life are given to the users. Based on these preselected material parameters, it should be much easier for users to *design* the material that sounds right to them. For now, we use trial-and-error method to find the parameters that generate the modal models that corresponds to the materials in our repository.

6.2 Results

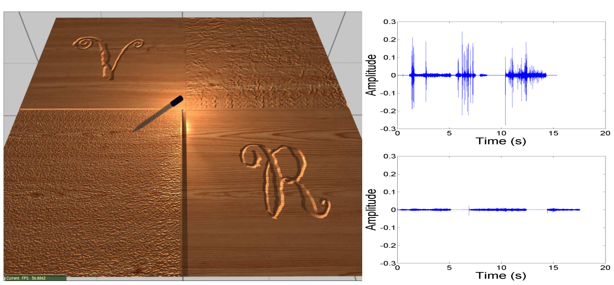
In this section, we demonstrate some of the results produced by our sound synthesis system and enumerate its possible applications.



(a) Brick



(b) Ceramic tiles



(c) Wood

Figure 8: **Comparison:** Snapshot images of a pen scrapping on three surface textures with different normal maps. The wave plots to the right show the sounds generated by our method (upper) and those generated from previous methods with only contact and friction sounds (lower).

Surface Scrapping: This scene shows a user scrapping a pen on surfaces textured with normal maps, generating contact sounds that highly correlates with the visual cues. It also shows the ability to handle different materials. Since impulses are universally handled in our sound module, if we change the material property of the object, the change is automatically reflected in the resulting scrapping and impact sound.

In Figure 8, our method successfully captures the characteristics of the bumpiness. Scrapping on various surfaces using only fractal noises approximating frictional contact sounds is distinctively different (can also been seen in the wave plots) from scrapping textured surfaces using our sound synthesis method (please also view the accompanying video).

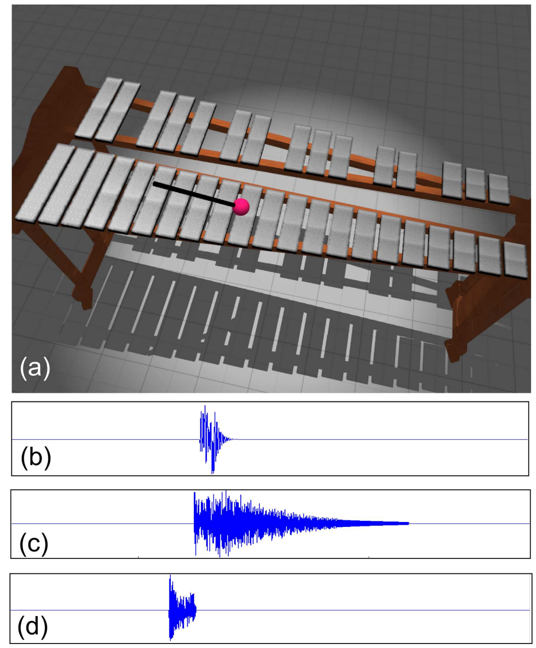


Figure 9: An example of a contact sound generated from the virtual marimba-like instrument. The bars are set to have different material parameters. In the three wave files shown above, sound waves correspond to marimba (b: wood), xylophone (c: metal), and a user designed material (d).

Virtual Instruments: Our system can be used to construct virtual instruments for education and entertainments. Users can build virtual instruments out of their designed sound by changing the material properties, and play them with our tablet user interface. With our interaction model, users are allowed to have complicated interaction with the instrument like scraping at various speed and tapping with different forces. Figure 9 shows a marimba-like virtual instrument with a user controlled mallet. Figure 9(b)-(d) show the different wave patterns generated by hitting the same bar made of different materials.

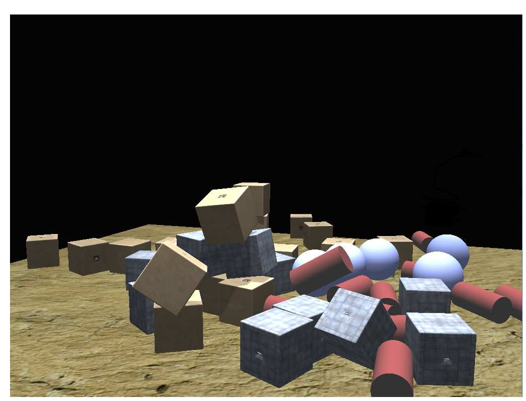
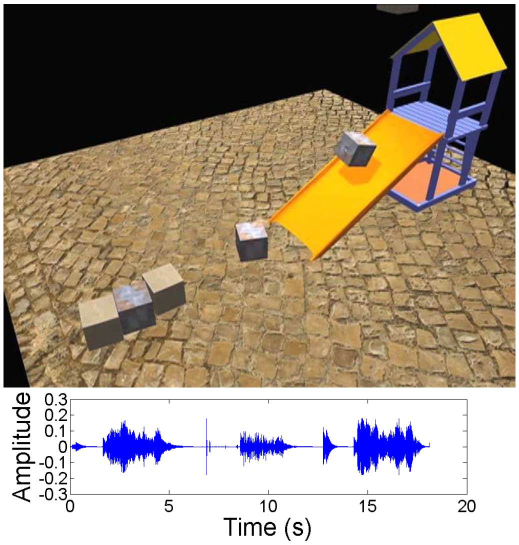
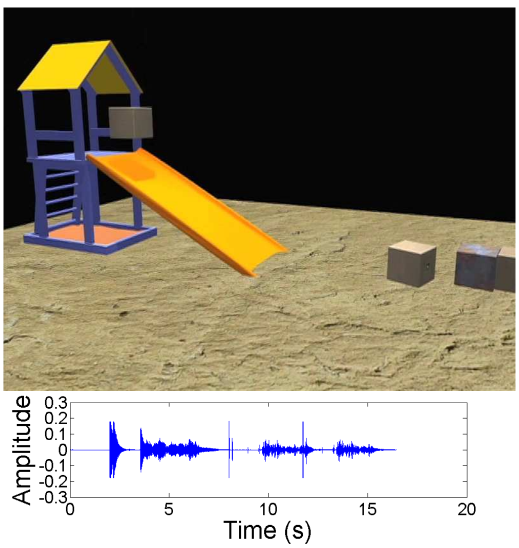


Figure 10: Many objects interacting with each other, making colliding, rolling and sliding sounds.

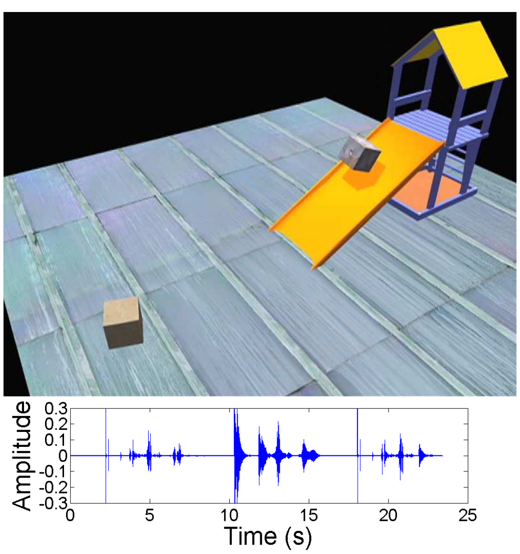
Add-on to Game Engines: Our sound synthesis system is able to



(a) Cobblestone



(b) Rough Mud Terrain



(c) Gridded Floor

Figure 11: Contact sounds (shown in wave plots below each image) generated by our method by the objects moving in a game-like environment, where boxes slide through the same surface with three different textures.

synthesize sound from physics-based simulation in real time. This capability makes it a great add-on to applications like games, virtual environment and simulators. We integrated our sound synthesis system with a general graphics engine: Open Source 3D Graphics Engine (OGRE) [2] and with a physics engine: NVidia's PhysX [1]. In the scenes shown in Figure 10 and Figure 11, we are able to easily achieve real-time performance with graphical rendering, physics simulation, and sound synthesis all running at the same time, which makes our approach a good candidate for sound synthesis in games.

Performance: In all the benchmarks mentioned above, impulses are generated by our method at faster than real-time rates: micro-level at about 5000 samples per second, meso-level at 1000 samples per second, and macro-level at 100 samples per second or higher. For all the scenes, the sound synthesis module runs at about 100 frames per second (fps) or higher; while the entire system, including visual rendering, sound synthesis, and physics simulation, typically runs at 30 to 60 fps, depending on the events in the scene.

7 Preliminary User Study

To assess the effectiveness of our approach, we have designed a set of simple experiments to solicit user feedback on the perceived difference of the auditory experiences accompanying a series of video clips. We have focused on two key aspects: (a) Does the addition of sound synthesized by our approach offer a more immersive experience than the visual simulation alone; (b) Does the sound synthesized by our approach offer a more immersive experience than the sound generated by the existing technique [19] that simulates sliding sounds with only the micro-level information?

7.1 Procedure

The participants consist of 19 volunteers: 6 women and 13 men, in the age of 8 to 43. For each subject, six sets of video clips were presented. For each set of video clips, all video clips have the same visual simulation but with different sound effects.

In the first three sets of video clips, we show several boxes falling down a ramp and sliding down to the same surface with three different textures: (1) cobblestone, (2) rough, mud terrain, and (3) gridded floor (see Figure 11). For each set of video clips, one video is completely silent and the other has impact and sliding sounds generated by our method. For each set of two clips, we asked the user study participants which one offers a more immersive experience over the other.

In the last three sets of video clips, we hope to in addition compare the sense of immersion between our method and an existing method for simulating sliding contacts. The video clips show a pen scraping (4) a brick surface, (5) a ceramic tiled surface, and (6) a wooden, textured surface (see Figure 8). In each set, there are three videos. One video has no sound, one video has sound generated using existing technique (i.e. the parametric method for sliding contact sounds [19]), and one video with sound generated by our technique (i.e. three-level simulation). The modal basis in [19] was constructed based on measurements using a robotic arm which is not available commercially. For a fair comparison, we used the same mass-spring formulation for constructing the modal models and the same transient contact handling [15] in both our method and the parametric method [19]. So, the only difference in the two methods in our user study is how each method handle lasting contacts, i.e. sliding contacts, which is the only variable factor our study focuses on. For each set, these video clips were presented in random orders and we asked the participants which one offers a more immersive experience over the other two.

Some of the video clips used in this preliminary user study are included in the supplementary video accompanying this paper and the entire study can be found at: `http://gamma.cs.unc.edu/SlidingSound/UserStudy`.

7.2 Statistics

In Table 1 we summarize the results for the experiment using the set of video clips as described in (1), (2) and (3). In Table 2 we summarize the results for the experiment using the set of video clips as described in (4), (5), and (6).

It is well known that good auditory display reinforcing the visual experience can enhance the sense of immersion; similarly unrealistic sound effect that is poorly synchronized with visual cues can disrupt the sense of presence in a VE. Therefore, the addition of auditory cues would not automatically improve the sense of immersion in a VE, unless the added sound effects are realistic and correlate with the visual events well. In all six sets of our experiments, the participants clearly prefer the same video clip with sound over without, indicating that the sounds generated by our method has achieved a satisfactory level of realism to reinforce the visual experience of nearly all subjects.

It has been reported in [20] that individual's ability to perceive sound may vary significantly from subject to subject. However, they overwhelmingly and consistently found the sliding sounds generated by our method offer more immersive experiences than the sounds synthesized by only using the parametric technique.

Experiment	No Sound	Our Method
(1) Cobblestone	1	18
(2) Mud Terrain	0	19
(3) Gridded Floor	2	17

Table 1: **Results of User Study:** the number of subjects who feel either no audio or the addition of contact and sliding sounds generated by our method make the video more immersive for each scenario shown.

Experiment	No Sound	Parametric Method	Our Method
(4) Brick	0	0	19
(5) Ceramic	0	0	19
(6) Wood	0	0	19

Table 2: **Results of User Study:** the number of subjects who feel no audio, or the addition of sliding sounds using only the parametric method, or using our method offers more immersive experiences.

8 Conclusion and Future Work

We have presented a new contact model for sound synthesis using a novel three-level representation to capture various surface characteristics at different spatial resolutions. This method is simple to implement and effective to minimize visual-auditory sensory conflicts, thus introducing much richer and varying sounds. We plan to extend the current triangle mesh spring-mass approximation by incorporating tetrahedralized models to simulate sounds.

Other possible enhancements include adding 3D auralization and room acoustic filters to the sound synthesis pipeline and conducting more extensive user study (similar in style to [20]) on both the sound synthesis and user interface components.

Acknowledgements

The authors wish to thank Nikunj Raghuvanshi for providing his code on sound synthesis between rigid bodies. This work was supported in part by the Army Research Office, Intel Corporation, National Science Foundation, and RDECOM.

References

[1] NVidia PhysX. http://www.nvidia.com/object/physx_new.html.

[2] OGRE: Object-oriented graphics rendering engine. http://www.ogre3d.org/.

[3] F. Avanzini, S. Serafin, and D. Rocchesso. Modeling interactions between rubbed dry surfaces using an elasto-plastic friction model. In *Proc. COST-G6 Conf. Digital Audio Effects (DAFx-02)*, pages 111–116, 2002.

[4] F. Avanzini, S. Serafin, and D. Rocchesso. Interactive simulation of rigid body interaction with friction-induced sound generation. *IEEE Transactions on Speech and Audio Processing*, 13(5):1073–1081, Sept. 2005.

[5] S. Barrass and M. Adcock. Interactive granular synthesis of haptic contact sounds. In *AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio, Helsinki University of Technology, Espoo, Finland, 15th-17th June*, 2002.

[6] B. Baxter, V. Scheib, M. C. Lin, and D. Manocha. Dab: interactive haptic painting with 3d virtual brushes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 461–468. ACM, 2001.

[7] N. Bonneel, G. Drettakis, N. Tsingos, I. Viaud-Delmon, and D. James. Fast modal sounds with scalable frequency-domain synthesis. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–9. ACM, 2008.

[8] C. Bruyns. Modal synthesis for arbitrarily shaped objects. *Computer Music Journal*, 30(3):22–37, 2006.

[9] J. N. Chadwick, S. S. An, and D. L. James. Harmonic shells: a practical nonlinear sound model for near-rigid thin shells. *ACM Trans. Graph.*, 28(5):1–10, 2009.

[10] P. Cook. *Real sound synthesis for interactive applications*. AK Peters, Ltd., 2002.

[11] D. Manocha et al. *Interactive Sound Rendering*. ACM SIGGRAPH Course Notes, 2009.

[12] J. F. O'Brien, P. R. Cook, and G. Essl. Synthesizing sounds from physically based motion. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 529–536. ACM, 2001.

[13] J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 175–181. ACM, 2002.

[14] D. Pai, K. Van Den Doel, D. James, J. Lang, J. Lloyd, J. Richmond, and S. Yau. Scanning physical interaction behavior of 3D objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 87–96, 2001.

[15] N. Raghuvanshi and M. Lin. Interactive sound synthesis for large-scale environments. *ACM Symposium on Interactive 3D Graphics and Games*, pages 101–108, 2006.

[16] A. Shabana. *Vibration of discrete and continuous systems*. Springer Verlag, 1997.

[17] J. Sreng, F. Bergez, J. Legarrec, A. Lécuyer, and C. Andriot. Using an event-based approach to improve the multimodal rendering of 6dof virtual contact. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 165–173. ACM, 2007.

[18] T. Takala and J. Hahn. Sound rendering. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 211–220. ACM, 1992.

[19] K. van den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 537–544. ACM, 2001.

[20] K. van den Doel, D. Pai, T. Adam, L. Kortchmar, and K. Pichora-Fuller. Measurements of perceptual quality of contact sound models. In *Proc. of the International Conference on Auditory Display (ICAD 2002), Kyoto, Japan*, pages 345–349, 2002.

[21] K. van den Doel and D. K. Pai. The sounds of physical shapes. *Presence*, 7(4):382–395, 1998.