Event-driven Interactive Solid Sound Synthesis

C. Verron, M. Aramaki, A. Gonot, T. Scotti, C.-E. Rakovec, A. Mingasson, and R. Kronland-Martinet *

LMA, CNRS, UPR 7051, Aix-Marseille Univ, Centrale Marseille
name@lma.cnrs-mrs.fr

Abstract. This paper describes a framework for interactive solid sounds in virtual environments. In an offline stage, the finite element method is used to pre-compute the modal responses of objects impacted at several locations. At runtime, contact/collision events from the game engine are used to drive excitation functions simulating impacts, rolling or friction interactions at audio rate. These functions are filtered by modal filterbanks simulating object resonances. Collision location is taken into account to modify the amplitude of resonant filters in real-time. This approach is appropriate for reproducing impact, rolling and friction interactions between objects. Results are illustrated by a real-time demonstration using the Blender Game Engine and Max/MSP.

1 Introduction

Procedural sound synthesis is an attractive alternative to pre-recorded sound samples for increasing the sense of realism in games and interactive audio/-graphics scenes [4, 5, 13]. In this paper we focus on interactive solid sounds and their possible integration with a physics engine for rendering impact, rolling and friction sounds in virtual worlds. Previous works in the computer graphics community provided methods for modeling modal properties of objects with arbitrary shapes [7, 3, 11] and controlling modal synthesis by game engine collision events [12, 9, 14]. This paper provides an overview of existing techniques, and introduces a demonstration framework for event-driven interactive sound synthesis in games and virtual environments. The framework is illustrated on Figure 1. Focusing on solid sounds, we use the action-object paradigm [2], recalled in Section 2, to generate complex interactions (impact, rolling, friction) between modal rigid-bodies. Section 3 describes the computation of modal resonances for arbitrary 3D objects with the finite element method. In Section 4, methods are proposed to compute an audio rate excitation function, controlled by game physics events to drive the sound synthesis. Finally, the framework is illustrated by an interactive rigid-body demonstration using Blender and Max/MSP.

* This work is supported by the French National Research Agency (ANR) under the Physis Project - CONTINT 2012 (ANR-12-CORD-0006). The authors thank I. Rosu, E. Debieu and C. Gondre for their help.

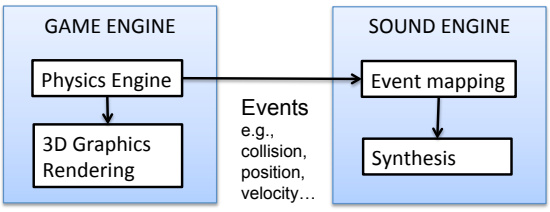

Fig. 1. Framework for event-driven interactive sound synthesis.

2 Action-object paradigm

Our framework is based on the action-object paradigm [2] that models interactive resonating objects by two components: an excitation function modeling the action (e.g., impact, rolling, friction) and a set of resonant filters modeling modal resonances of the object.

The excitation function $e(t)$ is computed at audio rate so that the sound $x(t)$ produced by the object in response to $e(t)$ is given by :

$$x(t) = (e * r)(t) \tag{1}$$

with:

$$r(t) = \sum_{m=1}^{M} a_m \sin\left(2\pi f_m t\right) e^{-\alpha_m t} \tag{2}$$

where the frequencies f_m are the modal frequencies of the excited object, while the amplitudes a_m depend on the excitation point, and the decay factors α_m are characteristic of the material [1]. Using this approach, interactions with arbitrary objects can be realized in two stages: first modal responses of the object are determined for several impulse locations. Second, an audio rate excitation function is computed and convolved by impulse responses to simulate complex interactions at different locations on the object. These two stages are described in more detail in the following parts.

3 Modal parameters for arbitrary shapes

Several approaches have been proposed in the computer graphics community to model modal responses of rigid-bodies, based on finite elements [7] and spring-mass systems [10]. [7] first introduced the finite element method (FEM) with thetraedral meshes to extract modal resonances from arbitrary shaped objects. [3] extended the approach using the boundary element method (BEM) to compute sound radiation of each mode, leading to complex directivity patterns. [8] proposed a multi-scale voxelisation process to alleviate FEM computational complexity when dealing with large mesh objects. Also, an hybrid physics/example-guided approach was proposed in [11] for extracting FEM material parameters

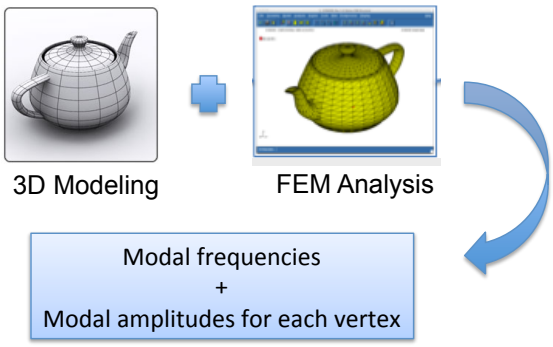

Fig. 2. Modal Analysis of arbitrary 3D objects. The finite element method using thin shell elements is performed in Sysnoise to compute modal frequencies of the object and their amplitude for impulses on each vertex.

from recorded sound samples, allowing independent shape/material transformations.

In this study, we used the FEM approach [7] with thin shell elements to model arbitrary shaped objects. Analysis was performed using the Sysnoise vibro-acoustic software[1]. A Matlab toolbox was designed to import generic 3DS models into Sysnoise and automatize the modal analysis, as illustrated on Figure 2. Sysnoise allows to compute modal responses to excitations using both FEM and BEM models. In practice FEM analysis requires elements sizes approximately ten times smaller than the analysed wavelength. This imposes high resolution meshes, sometimes leading to high computation time. To reduce the complexity, a first approximation consists in assuming omnidirectional directvity (i.e., no BEM calculation) and use the movement (or velocity) of a single vertex on the object (typically the impacted vertex) to compute the modal response. Such simplification is crude but convenient when a high number of impact locations are pre-computed for a single object. Note that since calculations are performed offline, the full FEM/BEM chain provided by Sysnoise may still be used for better accuracy at a few impact locations.

Considering the synthesis model of Eq. 1, modal analysis provides the frequencies f_m of resonant filters and their amplitudes a_m for several impact locations on the object (potentially for each vertex). In our synthesizer, the dampings are given to the sound designer as an efficient control to interpolate between different materials, as proposed in [1]. In the next section we present our approach to compute the audio rate excitation function $e(t)$ (see Eq. 1) from game engine events, to sonify complex interactions between rigid-bodies.

[1] http://www.lmsintl.com/SYSNOISE

4 Event-based control

In [12] rigid-body physics events from the game engine were used to drive the sound generation, leading to tightly couple audio/graphics. Since the physics engine typically run at low frame rate (e.g., 60 frames per second) a mapping stage is necessary to control synthesis from physics events (see Figure 1). This is especially true when using the action-object paradigm which requires an excitation function at audio rate to feed the resonant filterbank. Physically inspired mapping strategies were proposed in the Phya library [6] to control synthesis process from rigid-body movements. Additionally, bump-map information was used for computing the excitation function from image textures [9]. This was extended in [14] where a three layers excitation function was computed from object geometry, bump-map, and additional fine grain audio texture.

Our approach for computing the audio rate excitation function is based on the model proposed in [2]. In this study the authors proposed a generic excitation model, allowing users to navigate continuously between impact, rolling and friction (rubbing, scrapping). We use this model along with collision/contact events provided by the game engine to simulate real-time rigid-body interactions. Physics engines typically provide developers with enter/stay/exit collision events. Here, these events are used to switch between impact, rolling and friction excitation models. At each event, object velocity is sent to the sound engine to parameterize the excitation function. For more details on the excitation function parameters, the reader is referred to [2].

5 Implementation

A demonstration was designed to illustrate the event-driven sound synthesis framework. We use the Blender Game Engine[2] for physics and graphics rendering, while sound synthesis is performed in Max/MSP[3].

The Blender scene is based on the excellent Studio Physics Demo[4] by Phymec. Here it involves two objects, a ball and a cube, that can be manipulated by the user and interact with a square plate. Figure 3 illustrates the scene, and the sound synthesis implementation scheme. Modal responses of the plate were computed offline using Sysnoise for 38×38 impact locations evenly spaced on a square grid. The excitation function $e(t)$ and resonant filters $F_i(Z)$ are implemented using a set of Max/MSP objects provided by the Metason project[5]. At run-time, computed modal amplitudes are linearly interpolated to reflect the position of the objects on the plate.

Appropriate logic is implemented in the Blender Logic Editor to send information from Blender to Max/MSP at each collision event (see Figure 4). Event

[2] http://www.blender.org/
[3] www.cycling74.com
[4] http://youtu.be/hM3wke1mVgE
[5] http://metason.cnrs-mrs.fr/

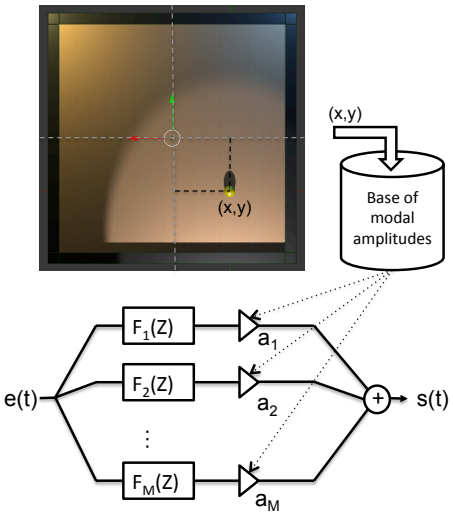

Fig. 3. Interactive plate/ball demonstration. Excitation function $e(t)$ (for impact, rolling...) is convolved by the plate modal filterbank $[F_1(Z), ...F_M(Z)]$. Pre-computed amplitudes $[a_1, ..., a_M]$ are applied to simulate the ball/plate contact point.

communication between both softwares is realized with the Python implementation of Open Sound Control (OSC) provided by Labomedia[6]. Additionally, the script given below shows an example for sending the velocity of Blender objects via OSC, to control the excitation function of modal synthesis in Max/MSP.

Fig. 4. Logic bricks for detecting object collisions in the Blender Logic Editor

Python script for sending OSC messages from Blender to Max/MSP

```
# OSC Blender Collision Demo
#
# send object velocity via OSC
# when enter/stay collision is detected
```

[6] http://wiki.labomedia.org/index.php/Communication_entre_Pure-data_et_Blender_en_OSC

```
from bge import logic
import OSC

ip_send = 'localhost'
port_send = 11000
client = OSC.OSCClient()
msg = OSC.OSCMessage()
print("osc client has been created !")

def sendOSC(controller):

    # get object velocity
    owner = controller.owner
    v = owner.getLinearVelocity()
    v = v.length # keep magnitude

    # get collision sensor
    touch_sensor = controller.sensors[0]

    if (touch_sensor.positive): # object is touching Ground

        if (owner['touchingGround']==0): # enter collision
            print('Enter collision')
            owner['touchingGround']=1
            address = "/blender/" + owner.name + "/groundEnterCollision"

        else: # stay on collision
            print('Stay on collision')
            address = "/blender/" + owner.name + "/groundStayOnCollision"

        msg.setAddress(address)
        msg.append(v)
        client.sendto(msg, (ip_send, port_send))
        msg.clear()

    else:
        owner['touchingGround']=0
```

6 Conclusion

We described a framework for event-driven synthesis of impact, rolling and friction sounds in virtual environments. Modal responses of rigid-bodies are computed offline for a set of impact locations using the Sysnoise acoustic software. At runtime, an excitation function is computed from the game engine collision events. This excitation function is filtered by the modal responses of colliding objects. Impact location is taken into account by modulating the energy of modal components in real-time. A demonstration illustrates these functionalities for simulating the interactions between

a ball, a cube and a plate. This approach for linking sound synthesis processes to the game physics engine provides promising results for tightly-coupled audio/graphics integration. In future works, perceptual evaluations should be carried out for validation and calibration of the system.

References

1. M. Aramaki, M. Besson, R. Kronland-Martinet, and S. Ystad. Controlling the perceived material in an impact sound synthesizer. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(2):301–314, 2011.
2. S. Conan, E. Thoret, M. Aramaki, O. Derrien, C. Gondre, R. Kronland-Martinet, and S. Ystad. Navigating in a space of synthesized interaction-sounds: Rubbing, scratching and rolling sounds. In *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, 2013.
3. D. L. James, J. Barbič, and D. K. Pai. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):987–995, 2006.
4. C. Picard Limpens. *Expressive Sound Synthesis for Animations*. PhD thesis, Université de Nice, Sophia Antipolis, 2009.
5. D. B. Lloyd, N. Raghuvanshi, and N. K. Govindaraju. Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 55–62, New York, NY, USA, 2011. ACM.
6. D. Menzies. Physically motivated environmental sound synthesis for virtual worlds. *Springer EURASIP Journal on Audio, Speech, and Music Processing*, 2010, 2010.
7. J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 175–181, 2002.
8. C. Picard, C.Frisson, F. Faure, G. Drettakis, and P. G. Kry. Advances in modal analysis using a robust and multiscale method. *EURASIP J. Adv. Signal Process*, 2010:7:1–7:12, 2010.
9. C. Picard, N. Tsingos, and F. Faure. Audio texture synthesis for complex contact interactions. In *proceedings of the 8th Workshop in Virtual Reality Interactions and Physical Simulation*, 2008.
10. N. Raghuvanshi and M. C. Lin. Interactive sound synthesis for large scale environments. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 101–108, 2006.
11. Z. Ren, H. Yeh, and M. C. Lin. Example-guided physically based modal sound synthesis. *ACM Trans. Graph.*, 32(1):1:1–1:16, 2013.
12. K. van den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 537–544, 2001.
13. C. Verron and G. Drettakis. Procedural audio modeling for particle-based environmental effects. In *Proceedings of the 133rd AES Convention*. AES, October 2012.
14. R. Zhimin, Y. Hengchin, and M. C. Lin. Synthesizing contact sounds between textured models. In *Proceedings of the 2010 IEEE Virtual Reality Conference*, VR '10, pages 139–146. IEEE Computer Society, 2010.