Non-Linear Contact Sound Synthesis for Real-Time Audio-Visual Applications using Modal Textures

Maunsbach, Martin Lennart Wendelboe; Serafin, Stefania

Published in:
16th Sound and Music Computing Conference (SMC2019), Málaga, Spain, 28-31 May

DOI (link to publication from Publisher):
10.5281/zenodo.3249410

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Non-Linear Contact Sound Synthesis for Real-Time Audio-Visual Applications using Modal Textures

Martin Maunsbach
Aalborg University
mmauns17@student.aau.dk

Stefania Serafin
Aalborg University
sts@create.aau.dk

ABSTRACT

Sound design is an integral part of making a virtual environment come to life. Spatialization is important to the perceptual localization of sounds, while the quality determines how well virtual objects come to life. The implementation of pre-recorded audio for physical interactions in virtual environments often requires a vast library of audio files to distinguish each interaction from the other.

This paper explains the implementation of a modal synthesis toolkit for the Unity game engine to automatically add impact and rolling sounds to interacting objects. Position-dependent sounds are achieved using a custom shader that can contain textures with modal weighting parameters.

The two types of contact sounds are synthesized using a mechanical oscillator describing a mass-spring system. We describe the discretization methods adopted, the solution of the nonlinear interaction and an implementation in the Unity game engine.

1. INTRODUCTION

High quality audio effects for virtual environments, as seen in video games, are important to the user's feeling of presence and overall experience. Specifically, impact sounds of colliding objects are of great importance to games [1], but sound of friction and rolling are also in demand. Sound effects are, however, slow and difficult to create and require specialized talents to implement correctly [2]. Furthermore, lacking realism in the sensory experience of any virtual environment leads to a break in perceived presence. For example, if a table is struck with a hammer and only produces a slight tapping sound, the experience is sending conflicting information to its user. To counter this problem, the field of physical modelling of sound effects is of particular interest.

Through this field of knowledge, the automatic generation of contact sounds can be synthesized without the need for large libraries of pre-recorded samples.

One popular synthesis technique adopted to simulate several material properties is modal synthesis [3, 4]. In modal synthesis, the frequencies of vibration in a material are simulated considering the normal modes of real objects. The normal modes describe the peaks in the spectral contents of a sound. This form of physical modelling of sound is also easily scaled in its level of detail, since the number of frequencies that are modelled at any time can be altered as necessary, and is computationally efficient. This makes it particularly well-suited for real-time implementation [2]. The modes are not only different because of material, but also the shape of the object in question and the position of impact. To accommodate for the position-dependency, a weighting ratio of each mode changes depending on the position of impact. The physical properties of simulated materials are of great importance. For example, materials that are more stiff will produce inherently higher pitched sounds when struck.

In this paper, the design and implementation of a modal synthesis toolbox for the game engine Unity is described. The simulation is produced to free developers using the game engine from the time-consuming process of capturing libraries of impact and rolling sound events, and instead have these sounds rendered through a mechanical oscillator, based on events within their virtual environments.

2. RELATED WORK

The fields of computer graphics and physical modelling of sound synthesis often overlap. The finite element method, though computationally heavy, can calculate the modes of 3D models, as seen in a program like mesh2faust, than can compute the modes from a mesh [5]. "Example-guided" automation to modal synthesis has been done, where recordings estimate the physical parameters across an example object. For other objects using the same material - but different geometric shape, the parameters of the example object can be used to automatically transfer the modes for object of differently shaped object [6]. A method modeling the wave propagation of a mesh is the digital waveguide mesh, that uses bidirectional delay units to simulate the reflections and transmissions through connected digital waveguides at a wave impedance [7].

Impact sounds have been synthesized in various ways. One example uses banded digital waveguide synthesis, where dynamically filtered white noise is passed though bandpass filters to add the characteristics of a material [1]. The mechanical oscillator used in this paper is not only useful for impact sounds, but can also other interactions like friction for a rolling wheel, finger rubbing on glass and squeaking doors [8]. The rolling and rubbing interac-

tion was further explored by Conan et al., where the same equation for the force that explains the sphere-plane interaction in this project also was used [9]. By utilizing the sparsity of modal sounds in the frequency domain, some modal synthesis was found to be 5-8 times faster compared to the time-domain equivalent [10].

When combined with computer graphics, sound synthesis has used shaders to design physical models of massive digital instruments [11], where the GPU is used to help with the computational load. Sound signals have also been stored as RGBA sound signals to use GPU processing [12]. Similar work storing model parameters in textures was done in 2001 by scanning objects using a highly automated robotic facility [13].

Both proprietary and publicly available game engines have begun incorporating synthesis into their sound design. Rockstar's RAGE engine, that was used to develop Grand Theft Auto V, has a real-time synthesis toolkit for assets "you can't necessarily create with samples alone" [14].The motivation for real-time synthesis was based on dynamic assets, fidelity and memory usage. The publicly available game engine Unreal Engine has also begun adding real-time synthesis to their game engine, and is looking into computing it on the GPU in a way that can be referred to as audio "shaders" [15].

3. SYSTEM MODELLING

A mechanical oscillator can take the form [16]:

$$\ddot{x} + g\dot{x} + \omega^2 x = \frac{1}{m} f \tag{1}$$

where the oscillator displacement x is used to produce the audio output. The frequency is set in ω and the damping constant g is determined by a quality factor q by $g = \omega/q$. The oscillator velocity and acceleration are determined by $\dot{x}$ and $\ddot{x}$ respectively. The modal weighting is determined by $1/m$. Using the K method to eliminate the delay-free [17] loop that will arise from the force equation and the trapezoidal rule, the output can be discretized take the form

$$w[n] = H(Cy[n] + y[n-1]) + H(\alpha I + A)w[n-1]) \tag{2}$$

where w is the vector $[x; \dot{x}]$, y is the force f and A, C and H are transformation matrices found from Equation (1)

$$\begin{aligned} H &= \frac{1}{\alpha^2 + \alpha g + \omega^2} \cdot \begin{bmatrix} \alpha + g & 1 \\ -\omega^2 & \alpha \end{bmatrix}; \\ A &= \begin{bmatrix} 0 & 1 \\ -\omega^2 & -g \end{bmatrix}; \quad C = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \end{aligned} \tag{3}$$

If the frequency, quality factor and modal weight remains constant throughout the interaction, these matrices need only be computed once.

The contact force is given by [18]

$$f(x, \dot{x}) = x^{\alpha}(k + \lambda \dot{x}) \tag{4}$$

where x is the same oscillator displacement as in Equation (1). This mutual dependency creates the delay-free loop, where the K method can be used. The velocity at the moment of impact is $\dot{x}$, while the constant k is the elastic coefficient and α is a value between 1.5 and 3.5 that describes the surface geometry [16]. A value of 1.5 is used in this paper, which leads to the non-linearity, that is solved by approximating the value using Newton-Rhapson.

The modal weight at specific positions can be changed with the weighting factor $1/m$ in Equation (1). The position-dependency means the matrices that previously only had to be computed once on impact have to be computed every time the modal weight changes. This adds additional constant multiplication for each buffer window, that is negligible in the overall computation cost, as the modal weight only appears in the C transformation matrix as seen in Equation (3). Since a single mode is not enough for a realistic sound, multiple instances of the mechanical oscillator can be coupled together through matrix generalization.

4. IMPLEMENTATION

The synthesis is implemented directly into Unity using C# utilizing Unity's base class MonoBehaviour to access the function OnAudioFilterRead that can insert data directly into the audio buffer at sample rate.

4.1 Exciters and Resonators

The implementation allows the user to choose which objects produce sound and select between three interaction types for those objects. Each sound-emitting object has its own individual audio source, which allows the output to be spatialized based on its position in the 3D space. The three interactions are based on the exciter-resonator relation, where one object acts as a hammer and adds the excitation to the resonating object. The first two interactions are exclusively as an exciter or resonator while the third is both of them combined.

The first interaction type is where the object only is an exciter. The exciter only emits sound when colliding with a resonator. This is useful for moving objects hitting non-moving objects, like a ball hitting a wall or rolling on the floor.

Having an object set to the second interaction type of a a resonator is especially useful for static objects. These are objects that should not act as exciters themselves. In a virtual environment this can be walls, floors or other non-moving objects.

The third interaction type is an object being both an exciter and a resonator, which gives the full effect of the system. Realistically, all objects are both exciters and resonators, but this does not necessarily have to be the case in virtual environments. This is useful for moving objects that create a sound, like a falling plate or a rolling glass.

4.2 Material Properties

As described in Equation (1) and modal synthesis, the sound of the mechanical oscillator is a result of the normal modes of the resonator, their weighting ratios and quality factors. The normal modes can be found by analysing a sound or

computing it from object meshes alongside some of its material parameters. Any number of modes can be used for the material. More detail is obtained with the use of more modes at the cost of additional computation power. A post-gain is applied to the output amplify the signal that is otherwise a very low value. In some cases this could affect the sound, but working with the C# type floats that stores 32-bit floating-point values gives a high bit depth.

Beside the modes and its corresponding parameters, an object can also be set to be ”rollable”, adding the rolling sound to the object. There should be distinguished between sliding objects that require a friction sound and rolling objects.

4.3 Micro-Impact Rolling

The sound of rolling can in many instances be described as many small impacts [16]. How rough a surface is will have an effect on the micro-impacts. Consider rolling sound in dirt, cobblestones or smooth, new asphalt. On most surfaces, where the roughness is not entirely visible like it is with cobblestones, the rolling sound can be modeled with randomly spaced impacts. The micro-impacts must be within a maximum and minimum margin, as too far spaced apart impacts sound like a repeated knocking, while impacts too close to each other makes it sound like a continuous sound and not rolling. The random factor is important to avoiding the rolling keeping a constant frequency and evolving into what resembles a tone. Rolling can be modelled by having the impacts occurring in between durations of time where the force is set to 0.

4.4 Object Interaction

The triggering of sound is achieved by utilizing MonoBehaviour’s collision system in the physics module. These function are called when objects enter, stay or exit a collision. Upon entering a collision, the relative velocity between the objects can be found, but the magnitude of this is not precise enough. If only this value is found, moving from one surface to another without altitude change or impacting at a narrow angle can have the same velocity as a direct impact, orthogonal to the surface. The velocity for the impact is found by taking the dot-product of the velocity and the normal vector of the contact point. If an object rolls from one surface to another without moving the object vertically, the velocity excitation to control the impact sound is zero.

If an object is set to being rollable, the velocity is found as the objects stay in a state of collision. The velocity is mapped to control the time between each micro-impact. The more micro-impacts are heard, the faster the rolling is heard to be. Once the objects exit the collision, there is no more force added to the system and the audio fades out naturally as the impacts would.

4.5 Modal Weight Texture

As seen in Fig. 1, a custom shader is built that can contain multiple textures. The special modal textures do not affect the visual aspect of the object as they are never rendered,

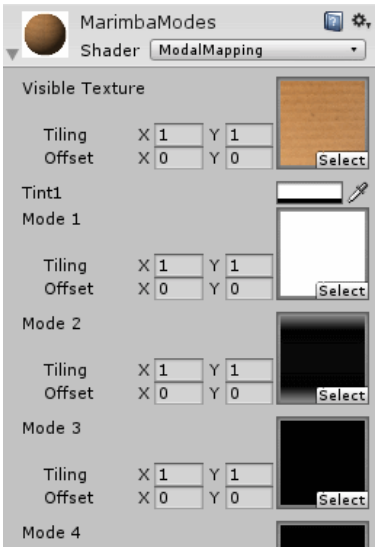


Figure 1. View of the custom shader from the inspector.

but their modal weighting values can still be accessed. The precision of the model weighting obtained depends on the resolution and interpolation of the created texture as well as how the image file of the texture is imported with or without compression. The contact point of that returns the pixel and its modal weight can either be found using the contact point provided by the collision system or as a raycast when the mouse is used.

5. OBJECT SIMULATIONS

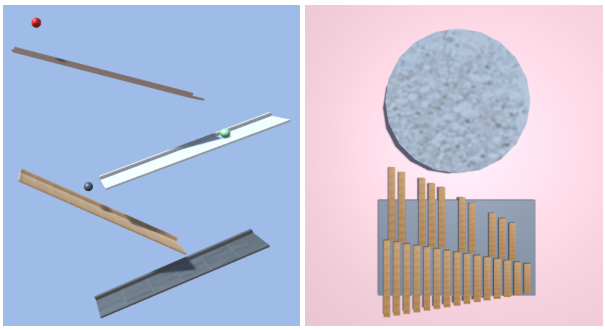
The implementation described has been applied to three different acoustic system with impact-specific interactions. Fig. 2 shows a visual representation of the scenarios using the Unity engine. The first is marbles, that only are affected by gravity and objects in its way, while the second is a marimba instrument and the third is the surface of a glass table. Combined, these interactions show the contact synthesis in instances with no modal texture, a one-dimensional modal texture and a two-dimensional modal texture.

5.1 Rolling Sphere

The spring-mass system can be used for a free-falling object impacting at a single point. The total force impacting on such an exciter is given by

$$f^{(h)} = f - m_h \cdot g \tag{5}$$

where f is the impact force approximated using Newton-Rhapson and m_h is the mass and g is gravity. This results in multiple impacts like a bouncing ball, but the system determining the force can be set to zero after a single impact, thus only emitting the sound of a single impact. Similarly to Equations (2) and (3), the force of the exciter is also discretized focusing on the displacement and velocity, that is used to approximate the total force using Newton-Rhapson.



(a) Marbles rolling. (b) Glass table and marimba.

Figure 2. The interactive simulations built in the Unity game engine. Marbles are rolling down only affected by gravity on the left while specific positions can be pressed on the right hand side using the mouse.

An example using marble-like objects rolling down a track is shown on the left hand side of Fig. 2. The marbles are on "rollable" planes, as described in Section 4.4, where the material of each plane determines the characteristics of the impact and rolling. More accurate simulations can be achieved by taking the incline into account.

5.2 Marimba Instrument

The second physical model is the marimba instrument. The marimba is described in detail by La Favre [19]. A marimba is an idiophone made up of bars that vibrate in complex patterns resulting in the unique sound. It has been shown that up to 25 modes of vibrations ranging from 0 to 8,000 Hz can be found in a struck C3. Simple marimbas need only be tuned for the fundamental frequency, while concert marimbas will be at least "triple tuned". The marimba can be modelled by having a fundamental frequency with the second and third modes at 4.0 and 9.2 times the fundamental [20], though some calculations put the third mode above 10.08 times that [19]. A fourth mode can be found at 19.6 times the fundamental. The modes are transverse modes, and it is shown that the amplitude of the modes change depending struck position on the bar by the mallet [21].

For this implementation, the modes and their respective amplitudes at four positions are found by La Favre [21]. The bar is struck at 4 positions; center, off-center, off-edge and at the edge of the bar. These values are one-dimensional, as the position only changes in one dimension from center to edge. A two-dimensional modal texture could be obtained by striking the bar across the shorter edge at the same positions.

The instrument is controlled using the mouse. On a click, a raycast traces a line to the hit object and obtains the UV coordinates of the raycast hit position. The UV is transformed to pixel coordinate by multiplying by the texture's width and height and the color (and therefore modal weighting) of the pixel at the coordinate is obtained. The mechanical oscillator is excited by an initial velocity at impact.

Figure 3. Textures used for the marimba bar, with intensified brightness. The grayscale value of the color is the modal weight. The leftmost depicts the fundamental frequency while the rightmost in the connected field depicts the fourth mode. In this instance, all other position's weighting values are scaled compared to the fundamental frequency, the modal texture of the fundamental frequency is is a single value across all positions.

Fig. 3 shows the four mode textures with intensified lighting, as it otherwise would be too hard to distinguish between the dark colors. The four textures are applied to every marimba bar in Fig. 2 and once hit, the value at the point is passed on to script controlling the synthesis. To scale each position to approximately the same overall amplitude, the first mode is set to the same for each position and the rest are scaled after it. The values are set between 0 (not inclusive) and 1, with 0 being completely black and 1 being white. It is not possible to have a value of 0, as the mechanical oscillator model at one point divides by the modal weighting. To ensure a value of 0 isn't read due to import and compression settings, a check is done while getting the pixel, setting it no less than a minimum value.

5.3 Circular Glass Table

The surface of a glass table is a simple everyday object with a circular plane as its top. Since the diameter is equal all around table, the amplitude of each mode is equal for all positions at equal direct length to the edge or the center. Five recordings were obtained from the glass table and used to create the modal textures. They can be described from the edge to the center as the edge, off-edge, middle, off-center and center. By analysing the recordings to see which modes were prominent across all of them, 12 modes were selected to create 12 modal textures.

The surface plot in Fig. 4 shows the values of one modal texture as height data. A circular texture of the modal weight of the recordings is created by rotating and interpolated line of the data around the center. The texture used is 800 times 800 pixels for the surface plot, but it can be as low as 9 times 9 pixels (five for the radius including the center and 9 for the whole diameter) if Unity's own image stretching is used for the interpolation, though higher

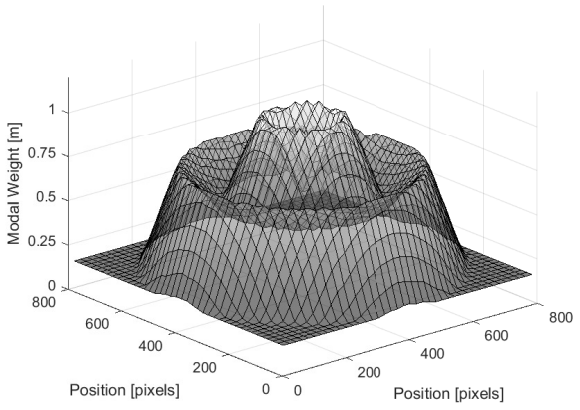


Figure 4. A 3D surface plot of the texture for the third mode of the glass table. The recordings of five positions from the center to the edge are interpolated around in a circle with the center as the anchor to create the 2-dimensional texture.

resolution textures are recommended to achieve smoother transitions between the weighting of each position.

6. CONCLUSION

In this paper an approach to using physical modelling for impact and rollings sounds in a virtual environment is proposed. This approach also includes the use of graphical textures for modal weightings to simulate position-dependent impacts.

The implementation can be extended to other interactions than an impact. Friction can use the same mechanical oscillator albeit with a different force excitation. The friction interaction of rubbing on a glass could use the modal texturing.

There are clear advantages to this combination of computer graphics and physical modelling of sound synthesis using the mechanical oscillator. Theoretically, if an object is modelled in 3D and a visual texture already is created, modal weights from recordings of its real-world counterpart can be specified to a point on the visual texture, and a modal texture can be computed for the whole model.

Since the modal textures are computed offline, which saves computational cost at runtime, the end result can be described as conveniently placed lookup-tables.

7. REFERENCES

[1] M. Aramaki and R. Kronland-Martinet, "Analysis-synthesis of impact sounds by real-time dynamic filtering," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 2, pp. 695–705, March 2006.

[2] 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*, 2001, pp. 537–544. [Online]. Available: http://doi.acm.org/10.1145/383259.383322

[3] J.-M. Adrien, "The missing link: Modal synthesis," in *Representations of musical signals*, 1991, pp. 269–298.

[4] K. Van Den Doel and D. K. Pai, "Modal synthesis for vibrating objects," *Audio Anectodes. AK Peter, Natick, MA*, pp. 1–8, 2003.

[5] R. Michon and S. Martin, "Mesh2faust: a modal physical model generator for the faust programming language application to bell modeling," *in Proceedings of the International Computer Music Conference (ICMC-17)*, 2017.

[6] Z. Ren, H. Yeh, and M. C. Lin, "Example-guided physically based modal sound synthesis," *ACM Trans. Graph.*, vol. 32. [Online]. Available: http://doi.acm.org/10.1145/2421636.2421637

[7] S. Van Duyne and J. Smith III, "The 2-D digital waveguide mesh," in *Proceedings of the International Computer Music Conference*, 11 1993, pp. 177 – 180.

[8] 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*, vol. 13, no. 5, pp. 1073–1081, Sep. 2005.

[9] 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 *16th International Conference on Digital Audio Effects (DAFx)*, Sep. 2013, pp. 202 – 209. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00877652

[10] N. Bonneel, G. Drettakis, N. Tsingos, I. Viaud-Delmon, and D. James, "Fast modal sounds with scalable frequency-domain synthesis," *ACM Trans. Graph.*, vol. 27, 08 2008.

[11] V. Zappi, A. Allen, and S. Fels, "Shader-based physical modelling for the design of massive digital musical instruments," in *NIME*, 2017.

[12] E. Gallo and N. Tsingos, "Efficient 3D Audio Processing on the GPU," ACM Workshop on General Purpose Computing on Graphics Processors, ACM, Aug. 2004, poster. [Online]. Available: https://hal.inria.fr/inria-00606754

[13] D. K. Pai, K. v. d. Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau, "Scanning physical interaction behavior of 3D objects," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. ACM, 2001, pp. 87–96. [Online]. Available: http://doi.acm.org/10.1145/383259.383268

[14] A. MacGregor, “The sound of Grand Theft Auto V,” in *Game Developers Conference*. Rockstar North, 2014, accessed: 2019-02-14. [Online]. Available: https://www.gdcvault.com/play/1020587/The-Sound-of-Grand-Theft

[15] A. McLeran, “The future of audio in unreal engine,” in *Game Developers Conference*. Epic Games, 2017, accessed: 2019-02-14. [Online]. Available: https://www.unrealengine.com/en-US/events/gdc-2017-the-future-of-audio-in-unreal-engine

[16] D. Rocchesso and F. Fontana, “The sounding object,” *IEEE Multimedia - IEEEMM*, 01 2003.

[17] G. Borin, G. De Poli, and D. Rocchesso, “Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems,” *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 597–605, Sep. 2000.

[18] D. W. Marhefka and D. E. Orin, “A compliant contact model with nonlinear damping for simulation of robotic systems,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 29, no. 6, pp. 566–572, Nov 1999.

[19] J. La Favre, “Tuning the marimba bar and resonator,” http://www.lafavre.us/tuning-marimba.htm, 2007, accessed: 2019-02-10.

[20] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer, 1998.

[21] J. La Favre, “Position of mallet blow on bar - effect on bar timbre (sound quality),” http://www.lafavre.us/FFT-mallet-position.htm, 2007, accessed: 2019-02-10.

ISBN 978-84-09-08518-7 ISSN 2518-3672