COMPUTATIONAL REAL-TIME SOUND SYNTHESIS OF RAIN

Stanley J. Miklavcic, Andreas Zita and Per Arvidsson

Department of Science and Technology
University of Linköping,
S-601 74, Norrköping Sweden
stami@itn.liu.se

ABSTRACT

Real time sound synthesis in computer games using physical modeling is an area of great potential. To date, most sounds are prerecorded to match a certain event. Instead, by using a model to describe the sound producing event, a number of problems encountered when using pre-recorded sounds can be avoided. This paper deals with the application of physical modeling to the sound synthesis of rainfall. The implementation of a real-time simulation and a graphics interface allowing an interactive control of rainfall sound are discussed.

1. INTRODUCTION

The goal of sound synthesis is the artificial production of acoustic information mimicking actual sound events. One area of application is the real-time sound synthesis for computer game environments. To date, most computer game sounds are pre-recorded for use in certain events, engines, collisions and other interactions, weapons fire, weather effects, etc. However, as computer games are becoming increasingly more realistic, allowing the user to interact with the virtual world in many concievable ways, sounds that match any possible situation are more difficult to produce using sampling methods. At best, pre-recorded sounds can be filtered or conformed in some way to match a given situation, but this is often an inefficient and time-consuming task. Furthermore, a given sound source can behave in a vast variety of ways depending on the variables specific to each occurence. For example, a physical event such as a collision can be very different depending on a large number of factors, such as the amount of energy released, the velocities, masses and angles involved. Obviously, too great a scope of possible scenarios exist to be treated with pre-sampled sounds. Sounds generated procedurally require precisely the type of input dictated by physical parameters. In such cases, real-time generated sound based on a physical model can represent a more desirable alternative. This is our point of departure.

In this paper we report on our efforts concerning the synthetic generation of the sound of rainfall using physical modeling of the sound generation process [1]. Emphasis is placed on rainfall onto solid surfaces. Based on this model, a real-time simulation of large clusters of raindrops has been implemented, relying on the superposition principle and a random time and space distribution of impact points and raindrop sizes. Important physical parameters such as rain intensity, drop volume and impact speed contribute to the overall audio simulation.

We have also been involved with the modeling of sound generated by drops falling specifically onto a liquid surface [2]. The overall sound event is schematically represented in Figure 1 [3, 4]. Naturally, to synthetically produce a realistic drop sound as heard above the surface of the water using a physical model requires a correct understanding of the mechanism(s) behind the acoustic process. Although a consensus in the literature on this subject is doubtful, what is generally accepted is that the familiar, characteristic "*plopp*" or "*plunk*" sound is strongly correlated with the appearance of an entrained bubble under the water surface [3, 4]. Furthermore, it is generally accepted that not all falling raindrops generate entrained bubbles nor produce the characteristic sound. Clearly, these facts too must also be taken into account in a computational sound simulation model. We outline how these features can be incorporated and give an expression for the sound function to be implemented in a simulation.

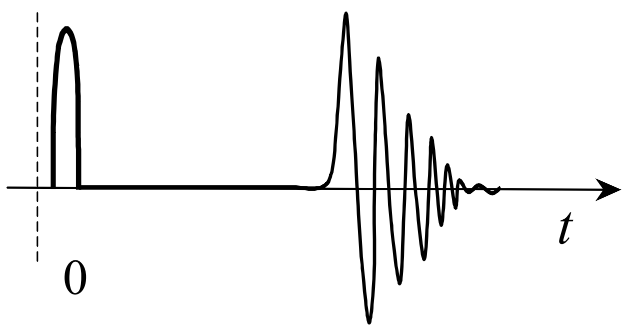


Figure 1. *Schematic of the sound produced by a water droplet falling onto a liquid surface. For a drop onto a solid, the second oscillatory contribution is absent.*

2. SOUNDS OF IMPACTING RAINDROPS

2.1. Raindrop impacting on a hard surface

A mathematical model for sound produced by a falling drop onto an arbitrary flat surface can be specified in terms of the acoustic pressure, $p = p\left(\overrightarrow{x}, t\right)$. This function satisfies the wave equation

$$\nabla^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0,$$

subject to appropriate boundary conditions. For a rigid surface these are the no fluid penetration condition, $\overrightarrow{u} \cdot \overrightarrow{n} = u_z = 0$, and, sufficiently far away, the Sommerfeld radiation condition [5, 6]. With help of the Green's function for a rigid surface (response from a point source in the form of a direct wave and a surface reflection), the acoustic pressure registered at position $\overrightarrow{x}$ at time t, can be expressed as the boundary integral

$$p\left(\overrightarrow{x},t\right)=\frac{\rho_0}{2\pi}\iint_{\Gamma_s}\frac{\dot{v}\left(x_s,y_s,t-R/c\right)}{R}dx_sdy_s, \qquad (1)$$

over a source distribution found on the horizontal water surface, $z=0$, denoted Γ. Here, $R=\sqrt{\left(x-x_s\right)^2+\left(y-y_s\right)^2+z^2}$ is the geometrical distance from the observer position $\overrightarrow{x}=(x,y,z)$ to the source position $\overrightarrow{x}_s=(x_s,y_s,0)$. ρ_0 is the air density and c is the air speed of sound. The pressure is completely determined once the surface velocity is specified. For simplicity, it is sufficient to assume that a raindrop imparts an impulse velocity

$$v\left(x_s,y_s,t\right)=A\left(v_{term}\right)H\left(t\right)$$

uniformly over a circular area symmetrically positioned around the impact center. The factor $A\left(v_{term}\right)$ is related to the kinetic energy of the drop with terminal velocity, v_{term}, just prior to impact. $H\left(t\right)$ is the Heaviside step function. Under these assumptions $\overrightarrow{x}_s$ takes values within a circle of radius a (the drop radius) centered at the impact center $(x_0,0,0)$ (see Figure 2). Taking the listener position to be $\overrightarrow{x}=(0,0,H)$, the integration in (1) can be performed [1, 2] giving

$$p\left(0,0,H,t\right)=\left[\frac{\rho_0 c}{\pi}A\left(v_{term}\right)\cos^{-1}\left(\frac{c^2t^2-H^2+x_0^2-a^2}{2x_0\sqrt{c^2t^2-H^2}}\right)\right] \qquad (2)$$

for $t_s=R_s/c\leq t\leq R_l/c=t_l$, and $p=0$ for all other t. Here $R_{s(l)}=\sqrt{\left(x_0\mp a\right)^2+H^2}$ is the shortest (longest) distance from the impact zone to the listener position. This analytic result is an accurate representation of the initial contribution to the sound schematically represented in Figure 1. In terms of sound simulation, two advantages are evident in (2). Firstly, the result is analytic, which can make computation rendering very efficient. Secondly, physical characteristics of the drop such as size, terminal velocity and impact location are naturally incorporated.

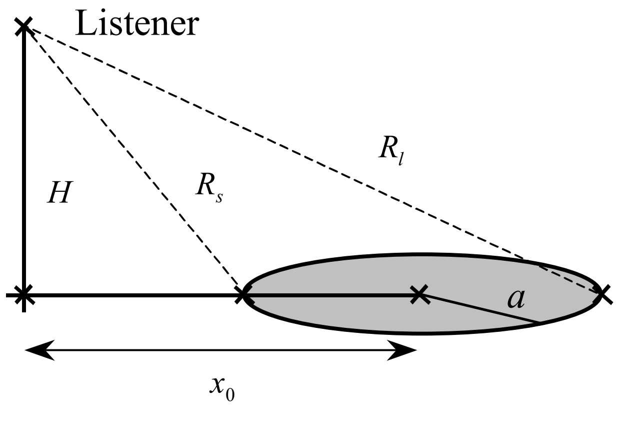


Figure 2. *Schematic showing the geometric relation between the circular source region $(x_s,y_s,0)$ and the listener position $(0,0,H)$, and distances involved.*

2.2. Raindrop impacting on a water surface

As rain continues to fall onto an impermeable surface one expects pools of water to form. The sound of the rainfall changes character which must be included for realistic effects. Elsewhere we have presented a description of the physical processes involved and a mathematical model based on these. However, the main feature is that the impact drives the fluid interface to form a crater in the shape of a cone [7]. At the apex of this cone, an entrained bubble is pinched off. The pinching-off gives rise to oscillations in the cone tip which lead to sound generation at the cone/crater opening [2].

As mentioned, not all drops satisfy conditions on size and terminal speed to create an entrained bubble and thus produce the characteristically familiar sound effect. However, assuming that these conditions are met and an impact crater in the shape of a truncated cone of length, l, splay constant, λ, and area, A, of opening at $z=0$ is formed, the sound produced can be represented by the fluid particle velocity at the surface, $z=0$ for $t\geq l/c$ is [2]

$$u\left(l,t\right)=v_0\left(t-\frac{l}{c}\right)\left(1-\frac{l}{s}\right)-c\frac{l}{s^2}$$
$$\times\int_0^{t-l/c}v_0\left(t-\frac{l}{c}-\tau\right)\exp\left(-\frac{c}{s-l}\tau\right)d\tau. \qquad (3)$$

Here, $s=\sqrt{A/\pi}/\lambda>l$ and $v_0\left(t\right)$ for $t\geq 0$ (zero otherwise) is a velocity representing the pinch-off effect triggering acoustic vibrations in the cone.

To pursue the issue of the sound heard above the level of the water it is sufficient to insert this expression into the boundary integral expression, (1), for the acoustic pressure. As in Section 2.1, the integration is over the $z=0$ plane, effectively restricted to the cone opening. The end result depends very much on the nature of the apex motion, v_0. Simple examples are considered elsewhere [2].

3. IMPLEMENTATION AND SOUND REPRODUCTION

3.1. Rendering of the computational sound

Using superposition, the above models of single impact sounds have been implemented in an algorithm based on clusters of drops falling in an area bounded by two circles centered at the listener position. The drop source positions are distributed randomly and uniformly in this area, with each source receiving a direction, ϕ, and a distance, x_0, relative to the listener. Naturally, the radius, a, of each drop source must either be user-specified or randomly allocated a value. We have taken the latter approach.

Some care must be exercised when distributing each drop source in the simulated time interval. The most straightforward approach of randomly producing, for each drop, a value of a global impact time $t_0=r\times L$ where r is a random value between 0 and 1 and L is the simulation length, can lead to problems. Randomizing t_0 would only define the *impact time* of each drop, not the arrival time. Since each drop also receives a random impact *distance*, the arriving sound impulses are distributed quite differently in the simulation period. In a real-time algorithm this is not a suitable solution. For example, assuming a continual rate of drop impact, since the sound registered initially comes from the drops lying closest to the listener, the intensity will artificially grow until the sound from drops lying furthest away reach the listener. From this point on the intensity will stay at a constant level. This effect prevails each time a change in conditions is implemented. The problem arises from the difference between the impact time and the initial arrival time, and this depends on the distance between the impact point and the listener.

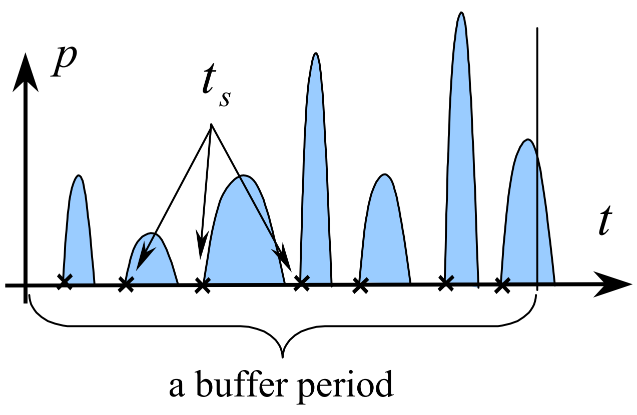


Figure 3. *Distributing drops by randomizing t_s.*

This problem can be solved if one instead judiciously randomizes the arrival time, t_s. This results in a direct response to user implemented changes. Further considerations are necessary to take into account due to the coupling between distance and sound arrival time [1].

In a real-time simulation we restrict the length of the simulation rendering stage described above and loop over this interval during the entire simulation. This iteration loop has a length of only a few hundredths of a second[1]. Hence, the algorithm prepares these so-called sound *buffers* (Figure 3*)*, which are handed over to the playback device upon request. This technique allows for continuously playing sound. As fast as a buffer has been played, a new buffer will be taken on. Note also that this process is not sequential; the buffer production stage does not need to wait for the playback of a previous buffer to finish. It can safely continue preparing the upcoming buffer while the current buffer is being played.

This non sequential, semi-parallel process has many benefits. It is naturally faster and also allows for the possibility of user-interaction. Since new buffers are prepared many times a second, one can change the properties defining the rain system smoothly during the simulation. This is precisely what is desired in a real-time algorithm. Naturally, it is more complex and cumbersome to implement. Firstly, it is important to have an algorithm which produces sound pressure buffers at a fast enough rate. If the algorithm is too slow, it will not be able to produce buffers at the same rate as they are being played and there could be a delay in the process in which the playback routine waits for follow up buffers. When this happens repeated buffers will be played, which is undesirable.

[1]The algorithm uses a simulation period, or buffer, of 512 samples. With a sample rate of 44100 smp/sec we have a buffer size $\approx$ 0.0116 seconds. Longer or shorter buffer sizes can be experimented with. However, 512 was found suitable in this case.

Secondly, to design a system that handles simultaneous production and playback[2] is non-trivial. However, as our algorithm is implemented in C++, we are able to take advantage of available C++ routines for playback. The Sound Toolkit (STK) [8] is a free set of pre-implemented C++ routines to handle various useful sound synthesizing and playback tasks. Among other things, it handles real-time playback of synthesized sound information on a wide variety of platforms. In this project, STK was used for these tasks with very good results. The code is very easy to use and well documented.

3.2. Multi-speaker sound reproduction

As the listener is modeled as a single point, binaural effects are not incorporated in the present version of the audio simulation. However, equivalent stereophonic or, more generally, multiphonic sound can be produced using a simple alternative method. The impact zone symmetrically located around the listener is divided up into circle sectors defined according to the number of output speakers at one's disposal. For example, two half-discs in the case of stereo, or quadrants in the case of quadraphonic sound, etc. Note that these need not be equal area sectors, as in Figure 4.

When a drop impact with the ground is simulated using the random distribution function its sector is determined by its relation to the placement of the two nearest speakers. The sound pressure calculated using either of the two methods described in previous Sections is distributed between the pair of speakers, weighted according to a simple linear function of the angle between the drop direction and the speaker directions, relative to a zero-angle in the listener's direction. A drop falling in the direction of one of the speakers results in a weighting of 100% distributed to this speaker. This method not only allows for a stereo speaker setup, it also allows for any arrangement and number of speakers, as in Figure 4.

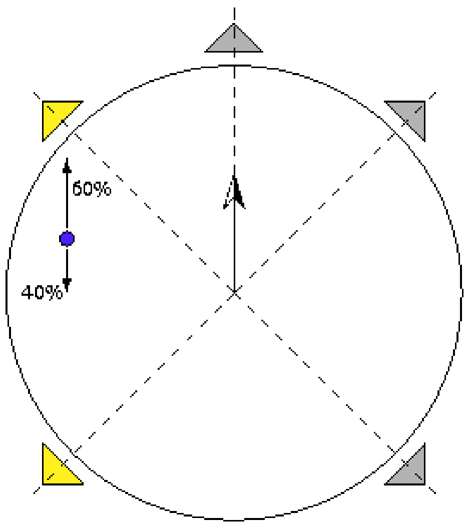
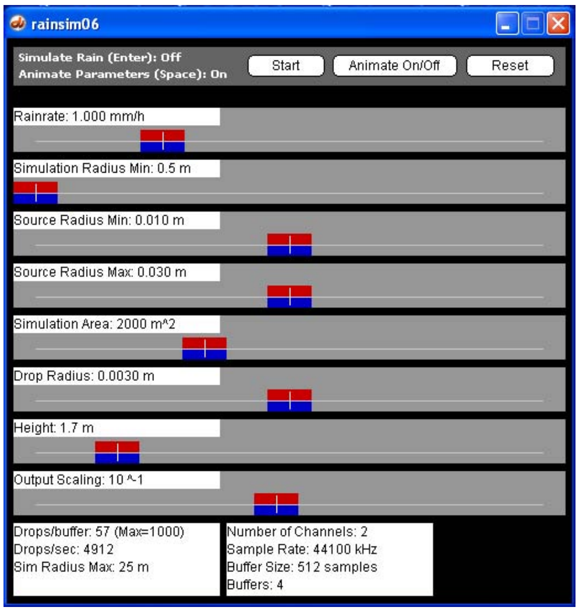


Figure 4. *Multi-phonic setup indicating circle sectors defined by number and positions of speakers.*

Using this method we have an easy and fast method of reproducing a sense of direction for every drop in the simulation. It has

[2]A system such as this is called a *multi-threaded* system. In it, simultaneous tasks can be calculated or processed.

been found to greatly enhance the surround sound effect as well as the depth of produced sound. This method also fits for further extensions of the simulation, using localized source types and sound events in a 3D environment.



Simulate Rain (Enter): Off
Animate Parameters (Space): On

Rainrate: 1.000 mm/h
Simulation Radius Min: 0.5 m
Source Radius Min: 0.010 m
Source Radius Max: 0.030 m
Simulation Area: 2000 m^2
Drop Radius: 0.0030 m
Height: 1.7 m
Output Scaling: 10^-1

Drops/buffer: 57 (Max=1000)
Drops/sec: 4912
Sim Radius Max: 25 m

Number of Channels: 2
Sample Rate: 44100 kHz
Buffer Size: 512 samples
Buffers: 4

Figure 5. *Our user-interface wrapper, "Rainsim".*

3.3. User interface *Rainsim*

A simple user interface application was implemented in Macromedia Director to wrap the functionality of the real-time algorithm. This was made possible by producing a so-called scripting Xtra[3] of the real-time algorithm and playback system (not described here). Using the interface, as shown in Figure 6, one can easily vary physical parameters interactively and produce near instantaneous real-time response. A number of simulation variables selectively control the real time output sound. As seen in Figure 5 there are several buttons and controls or "sliders" available. The "Animation On/Off" button basically controls the dynamic state of the simulation as set by the sliders. The red sliders denote the actual implemented value of various property and thus directly determine the sound produced. The blue sliders are subject to user control and indicate the desired or destination values of properties. The kinematics of the animation (the red sliders move to the blue slider positions) are set by the user to simulate different rain conditions.

4. SUMMARY AND CONCLUSIONS

When compared with the sound produced by other methods, *e.g.*, a granular synthesis model[4], the present physical approach is percieved to be much more accurate and flexible in terms of user control. Our model describes rain sounds valid for a much wider class of rain situations, automatically parameterized by real physical properties that can be varied in real-time. Methods such as granular synthesis rely on extensive psycho-acoustic surveys in order to establish parameters that affect the sound in the desired way [10]. Such a process is time consuming and often results in parameters that do not have the flexibility needed for simulation of real rain.

Compared with real rain sound, the physical model produced sound that was very similar in nature to the bare sound of rain, *i.e.*, under ideal circumstances. In practice it is very unusual to hear only the rain sound we have modeled. More often than not a number of other events produce sounds we are accustomed to hearing and associate with rainfall: thunder, water flowing and splashing, rain impacting on vibrating surfaces (rooftops, etc.), rustling leaves, howling winds, etc. Only when a combination of the present simulated sound and some of these other sound events is arranged will it be possible to fully appreciate the end result of our synthesized rain sound.

An important consideration is the computational effort involved in the simulation. Depending on simulated rain intensity (number of drops) and rain source radius (larger sources result in more samples to evaluate), the programme proved to run smoothly on an ordinary PC setup (>1Ghz) with all ranges of simulated rain sound.

In summary, we find that a physical model is a good option for use in virtual environments such as game worlds. It is likely that as more physical scenarios are successfully modeled, the computer games industry will be offering more complete solutions of real-time sound engines based on physical models for use in real-time game environments.

[3]Xtras are based on the Macromedia Open Architecture (MOA) and are used to extend the functionality of for example a Macromedia Director application. Refer to the macromedia homepage for more information. [Macromedia]

[4]A granular synthesis model uses similar sound producing methods [9]. However, it is not parametrized with real physical properties as in a true physical model. It is commonly used for synthesizing rain sound [9].

5. REFERENCES

[1] Zita, A. 2003 *Computational Real-Time Sound Synthesis of Rain* MSc. Thesis Linköping University (ISRN: LITH-ITN-MT-EX-03/01-SE, LiU Press)

[2] Miklavcic, S. J. 2004 *J. Phys. A: Math. General*, (submitted April 2004).

[3] Pumphrey, H.C. and Crum, L.A. 1990 *J. Acoust. Soc. Am.* **87**, 142–148

[4] Leighton, T. G. 1994 *The Acoustic Bubble* (San Diego: Academic Press)

[5] Morse, P.M. and Ingard, K.U. 1968 *Theoretical Acoustics* (Princeton: Princeton University Press.)

[6] Kinsler, L.E., Frey, A.R., Coppens, A.B. and Sanders, J.V. 2000 *Fundamentals of Acoustics*, 4th Ed. (New York, Wiley)

[7] Longuet-Higgens, M. S. 1990 *J. Fluid. Mech.* **214**, 395–410

[8] Cook, P. R. and Scavone, G. P. *The Synthesis Toolkit in C++ (STK)* (`http://ccrma-www.stanford.edu/software/stk/index.html`, 1995-2002)

[9] Cook, P. R. *Physically Inspired Stochastic Event Modeling (PhiSem)* (`http://www.csounds.com`, Feb. 1997); Lakatos, S., Cook, P. R. and Scavone, G. P. 2000 *J. Acoustic Soc. Am. (Letters Online)* **107**, L31–L36

[10] Miner, N. E. and Caudell, T. P. *Presence: Teleoperators & Virtual Environments* **11**, 493–507; Miner, N. E., Goldsmith, T. E. and Caudell, T. P. *ibid* 508-524