Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn


# Schematic interface of sound creation for computer animators

TONG Kam-pang Maya, WONG Kam-wah
(*School of Creative Media, City University of Hong Kong, Hong Kong, China*)
E-mail: 50346861@student.cityu.edu.hk; smkam@cityu.edu.hk
Received Apr. 14, 2006; revision accepted Apr. 29, 2006

**Abstract:** As an audiovisual medium, computer animations require superior image quality and professional soundtrack to lead audiences into their fascinating virtual world. Without sound, the impact of storytelling is reduced or the story is even not understandable. Despite the importance of sound, most animators are unfamiliar with sound editing software. Limited budget projects such as independent or student works have difficulty hiring sound professionals to create tailor-made soundtrack. Therefore, we need a suitable sound tool to express their individual ideas. In this paper, we propose an approach using schematic for both computer animation and sound. Our approach provides (1) physical simulation on sound through animation parameters and (2) a new channel for animators to add aesthetic values in sound through their experience of using schematics in existing animation software. We demonstrate our idea through several examples, such as Doppler shift, obstacle effect, importance, energetic, and sputtering effect.

**Key words:** Computer animation, Multimedia, Digital media, Schematics, Physical simulation, Aesthetic, Sound effects, Soundtrack
**doi:**10.1631/jzus.2006.A1141 **Document code:** A **CLC number:** TP319

## INTRODUCTION

In the competitive field of animation, sound is an essential element for one to stand out from the others. With sound, we perceive a unique meaning from the frame. Imagine a simple bouncing ball scene—we can apply the sound of "a pingpong hitting on a table" or "a basketball hitting on a gymnasium floor". It tells us a different narrative of the ball. Without sound, we lose an effective portrayal on its storytelling. Besides, sound also affects our perception of an animation. Previous research (Mastoropoulou *et al*., 2005) proposed that sound maintains a motion smoothness even at low frame rate. Without sound, we lose its advantage of using fewer frames to maintain the smoothness, which can be inefficient in interactive, multimedia or Web animation.

However, many professional audio editing tools are unsuitable for CG professionals, as they are designed for sound specialists. Animators with finite knowledge of digital audio would find it difficult to use for creating a soundtrack. Therefore, to overcome the difficulties, they usually "drag and drop" sound to their animations from sound effects libraries without any consideration of the scene suitability. The result of a poor soundtrack can ruin their animations, even though the images are of good quality.

The approach we present here utilizes 3D animation and sound effects in one schematic environment. Our idea comes from two facts: First, most animators are already familiar with schematic approach in their favorite animation tools. Maya (Alias, 2005), 3D Studio MAX (Discreet, 2005), LightWave (Newtek, 2005) and Softimage (Avid, 2005) have schematic interfaces. Second, the power of schematic is to synthesize contents with visual blocks and connection lines, which gives an unlimited possibility and great flexibility to animation and sound creation, and provides a clearer logic compared with track-based approach in the traditional audio editing software.

## RELATED WORK

Previous work showed that audio application researches in the computer graphics community concentrate on the synchronization of sound and motion as well as automatic sound generation (Cardle *et al*., 2003; Hahn *et al*., 1995; O'Brien *et al*., 2001; 2002; Takala and Hahn, 1992; van den Doel *et al*., 2001). Most of them proposed methodologies for generating physically correct sound by object deformation (O'Brien *et al*., 2001; 2002), dynamic simulation (van den Doel *et al*., 2001) and object interaction (O'Brien *et al*., 2001; 2002; Takala and Hahn, 1992; van den Doel *et al*., 2001). Methods of motion-driven sound were also introduced (Cardle *et al*., 2003; Hahn *et al*., 1995; Takala and Hahn, 1992).

Without much effort, animators with minimal sound knowledge could benefit from these systems that provide a convenient and automatic means to create their "physically correct" realistic soundtrack. However, none of them considered elements other than physical quality such as aesthetics and creativity from animators. Quality soundtrack, like those created by sound designers, should accompany these elements so as to succeed in making unique and admirable animation. Our system addresses this subject by showing corresponding aesthetic examples in the later part.

Until recently, there were only a few sound-editing tools attempting to create a working environment suitable for animators. The first type is a third party add-on tool implemented into existing 3D software. Foley Studio MAX (Boomer Labs, 2005) and Thunder (Digimation, 2005) are typical examples. They are implemented into 3D Studio MAX by utilizing its built-in interfaces, functions and objects of 3D software. Another example is AN-Sound 3D (Anilogix, 2005), an API merged with 3D Studio MAX which allows audio information outputted with 3D geometry.

The second type is a native sound-editing tool featuring an animation software interface. For instance, in Unreal game engine (Unreal Technology, 2006), users can call a "SoundCue Editor" in the animation viewport to edit the sound(s) for a selected geometry. It uses a schematic interface to represent the relationship of each sound with respect to its geometry.

The advantage of these tools is that it makes animators comfortable as they work in the same environment without switching to another unfamiliar tool for sound editing. However, again the soundtrack created in these tools is actually based on simulating the physical phenomenon of sound. No aesthetic issue is considered. It means that other than producing a realistic sound, users have lesser or no control of creating a personal soundtrack style.

## STRUCTURE OF SCHEMATIC

Our schematic approach (Fig.1) is quite similar to those in other animation software—it contains nodes and connection lines. For each node, there exist left inlets to receive data and right outlets to send out data. Therefore, the flow of data in a schematic is from left to right. Connection lines must be drawn from an outlet of a node to one or more inlets of another node.

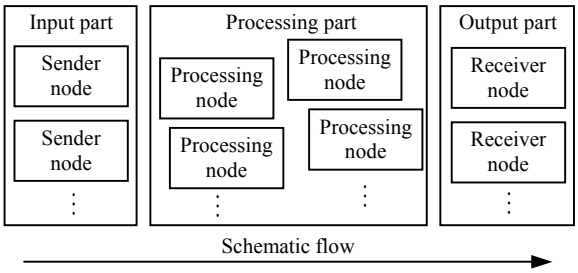



**Fig.1 The flow of the schematic. From left to right: input part with sender nodes, processing part with processing nodes and output part with receiver nodes. These nodes are discussed below**

### Sender node

Sender node passes stored or received data from its inlets to another node. These nodes can (1) have both inlets and outlets or (2) have outlets only. They do not modify their data. Type (1) node is constructed within the schematic. It stores data by receiving it from another, or is entered by the users. It lets users control some simple operations, such as on-off switch and user-defined data, and shows the status of received data for debugging. Node examples are Audio Mute, Audio Solo, Number, Name and Array. Type (2) node receives data from a 3D file or digital wave file, retrieves necessary contents such as geometry transformation, material and texture, file setting and wave

sample content. Node examples are Transform, Material Color, Key, Normal, Geometry Group, Frame Rate and Wave File.

**Processing node**

Processing node implements a specific computation on the audio sample or data received from its inlets. And then it sends the result to its outlet. These node types include (1) basic processing and (2) combined processing.

Basic processing node performs common DSP (digital signal processing) effects. It provides the same functionalities as other sound-editing software, such as Gain, Normalize, Pan, Equalization, Time Stretch, Pitch Shift, Chorus, Delay and Reverberation. They enable users to construct their custom sound effects from these basic elements.

Combined processing node performs a series of processing to achieve a desired effect. Through schematic, such node can be constructed to achieve more complicated effects, such as Distance, Doppler Shift, Occlusion, Exclusion and Obstruction. These are physical simulations of 3D sound. Unlike ordinary sound-editing software that only provides basic DSP effects, these nodes aim for aesthetics in sound expression that fits into the animation story. Examples of aesthetic usage include Importance, Energetic and Sputtering. For instances, how can I give this character to sound more important than the others? How can I make a sound tell the audience a car is going to be broken down? We will demonstrate these ideas in the later examples.

**Receiver node**

Receiver node collects information from other nodes and performs a final processing on the wave sample. It contains no outlets for further connection. It is designed for the ultimate process on a sound file. Examples are Wave Out, Audio Output Port and Audio Level Meter.

## EXAMPLES

**Physical simulation**

Although our proposed system aims at non-physical quality simulation, we show here the use of constructing sound simulation in 3D environment as well. Before we go through our aesthetic examples, we first look at two examples of 3D aural simulation.

**Doppler shift**

The first schematic example demonstrates a physical phenomenon called Doppler shift (Fig.2), which is driven by animation parameters and a digital sound wave.

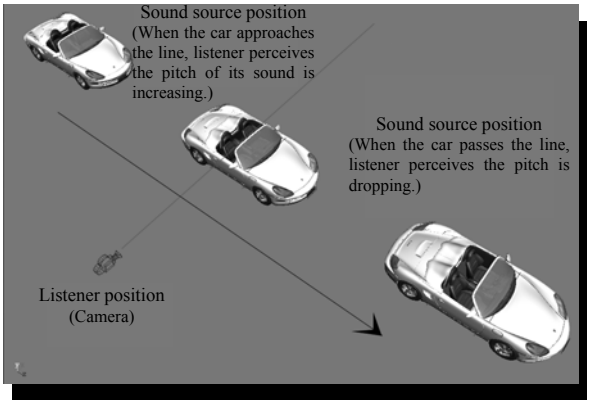



**Fig.2 Doppler shift produced when a car is passing through the listener (camera at the left lower corner)**

To construct a Doppler shift mentioned above, a schematic is proposed in Fig.3.

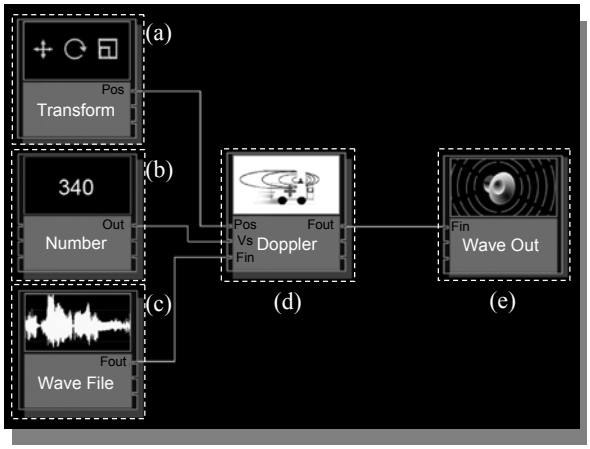



**Fig.3 A schematic for Doppler shift. (a) Transform node; (b) Number node; (c) Wave File node; (d) Doppler node; (e) Wave Out node**

Transform node (Fig.3a) reads required data from a 3D scene to provide animation parameters, such as transformation, scale and rotation of a geometry. In this example, the sole outlet "Pos" gives a position

attribute in $(x, y, z)$ and is connected to the inlet "Pos" of the Doppler node (Fig.3d) and provides location data for the Doppler shift computation that we will discuss later.

Number node (Fig.3b) stores numerical values. In this example, the sound velocity, 340 m/s, is defined. We assume that user will enter a number, but it can also receive a value from its inlet. The value is then sent to "Vs" (sound velocity) inlet of the "Doppler" node (Fig.3d).

Wave File node (Fig.3c) stores digital waveform buffer and sends out the samples in sequence to the "Fin" inlet of Doppler node (Fig.3d).

Doppler node (Fig.3d) receives geometry position, sound velocity and wave content from the nodes we mentioned above. The mathematical representation of a Doppler shift effect is given by:

$$f_{\text{observer}} = f_{\text{source}}\left(\frac{1+v_{\text{observer}}/v_{\text{sound}}}{1-v_{\text{observer}}/v_{\text{sound}}}\right),$$

where $v_{\text{observer}}$ and $v_{\text{source}}<v_{\text{sound}}$.

$f_{\text{observer}}$ is the frequency perceived by the observer. $f_{\text{source}}$ is the original frequency emitted from the sound source. $v_{\text{observer}}$ is the observer's velocity and $v_{\text{source}}$ is the velocity of the sound source. $v_{\text{sound}}$, the speed of sound in dry air, is approximately 340 m/s in standard room temperature (22 °C).

In a virtual 3D world, since the observer is always attached to a virtual camera, its velocity is zero. Therefore, we can simplify the Doppler shift equation to the following:

$$f_{\text{observer}} = f_{\text{source}}\left(\frac{v_{\text{sound}}}{v_{\text{sound}}-v_{\text{source}}}\right).$$

To simulate Doppler shift, we use a technique called fractional delay line (Zolzer, 2002) or variable delay line, which can be found in (Eigenfeldt, 2005). The basic concept is that we first read a wave signal, copy a portion of it with a definite length, and then push the portion back to the copying inlet repeatedly according to the delay time $t_{\text{delay}}$. In order to produce a frequency shifting effect over time, we vary $t_{\text{delay}}$ continuously. When $t_{\text{delay}}$ is decreasing, the sound pitch increases which simulates a sound source approaching the observer. When sound pitch decreases, (i.e. $t_{\text{delay}}$ is increasing) it simulates the source retreating from the observer.

The major step here is to find $t_{\text{delay}}$. To do this, express $t_{\text{delay}}$ in milliseconds in terms of the number of delay audio samples $s$ and the audio sampling rate $r$. We get:

$$t_{\text{delay}} = \frac{s}{0.001r}. \tag{1}$$

Also:

$$s = \frac{d_{\text{source}}r}{v_{\text{sound}}}, \tag{2}$$

where $d_{\text{source}}$ is a distance that can be obtained by calculating the length of the displacement vector between the moving geometry and the camera position. As every frame, $d_{\text{source}}$ is varying (it is decreasing when the geometry moves towards the camera, and vice versa). Finally, substitute Eq.(2) into Eq.(1) to obtain:

$$t_{\text{delay}} = \frac{d_{\text{source}}}{0.001v_{\text{sound}}}.$$

Note that our proposed schematic (Fig.3) provides all necessary values for this formula: $d_{\text{source}}$ comes from Transform node, $v_{\text{sound}}$ comes from Number node and Wave File node provides sample data.

Wave Out node (Fig.3e) receives in its inlet "Fin" the processed signal from the outlet "Fout" of Doppler node. It then writes the information a wave file in a memory location.

**Occlusion, exclusion and obstruction**

We show another example of 3D sound simulation here (Fig.4). Environmental acoustic effects, such as occlusion, exclusion and obstruction, can be found in recent 3D sound simulation like EAX (SoundBlaster, 2006; Krebs, 2002). Suppose, in our 3D world, we have a sound source, a listener (virtual camera) and two different rooms separated by a wall. Occlusion occurs if the source and the listener are in different rooms. The direct sound emitted from the source cannot reach the listener, but a muffled sound (filtered by the wall) is heard instead. If the listener can see the source in front of an open window con-

necting both rooms, exclusion occurs as the direct sound reaches the listener through the window but most reflected sound cannot. If they are in the same room but there is a cabinet inserted between them, obstruction occurs as the direct sound cannot reach the listener, but the reflected sound could take another path (e.g. reflected from the wall) to reach him.

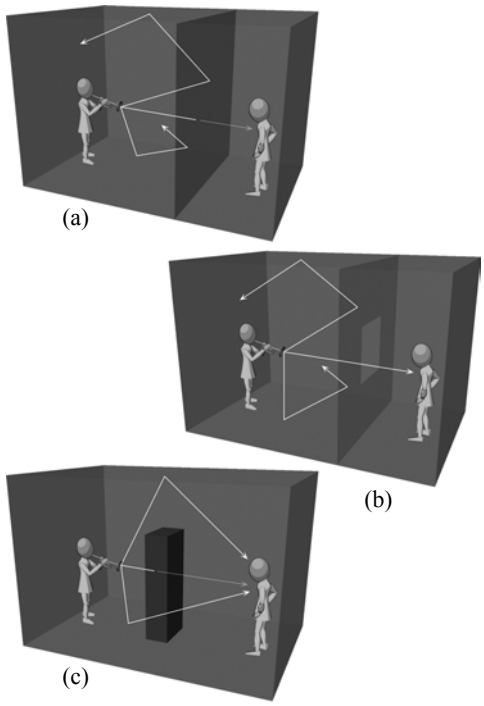



**Fig.4 Three conditions for producing occlusion (a), exclusion (b) and obstruction (c) effect in an enclosed area. A trumpeter is playing his instrument in front of an audience within the space. Brighter arrow lines are unblocked paths of direct and reflected trumpet sounds. Darker lines are the paths of the direct sound after being blocked by obstacles (wall and cabinet)**

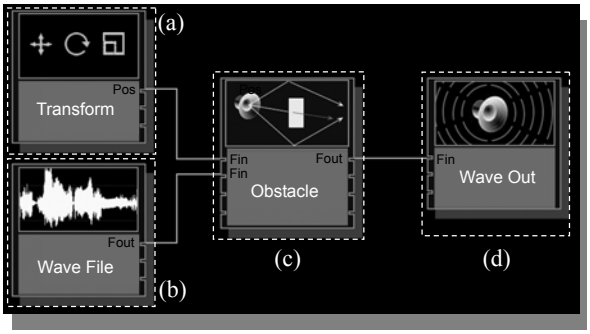



**Fig.5 A proposed schematic for "obstacle effect". (a) Transform node; (b) Wave File node; (c) Obstacle node; (d) Wave Out node**

The following schematic shows a proposed connection of each node:

Obstacle node (Fig.5c) provides acoustic effects caused by an obstacle with respect to the listener. That is, occlusion, exclusion and obstruction. The major mechanism of this node is that it first collects, from Transform node (Fig.5a), the location data of scene geometries such as virtual camera (listener), sound source and "obstacle" objects. A bounding volume of an obstacle is calculated, and then tested for intersection with a ray (direct sound path) constructed from the listener (camera position) to the sound source. The node applies a corresponding effect according to the conditions shown in Table 1.

**Table 1 Conditions for checking different types of obstacle effect**

| Condition | Obstacle effect |
|---|---|
| Intersect, reflected sound can reach the listener | No effect |
| Ray blocked by a scene object and no reflected sound can reach the listener | Occlusion |
| Ray blocked by a scene object but reflected sound can reach the listener | Obstruction |
| Intersect, no reflected sound can reach the listener | Exclusion |

Reflected sound can be generated from the original sound source (Wave File node, Fig.5b) as discussed in (Zolzer, 2002). Basically, a reflected sound is a delay signal of an incoming digital wave with filter applied to simulate the absorption of frequencies from reflecting materials such as wall or cabinet. In addition, if the ray is blocked in an intersection test, filter should be applied to reduce the level of frequency component in a direct sound. Object surface absorbs high frequencies of the sound, which makes it muffled. The overall volume of the sound is thus attenuated, yet not completely muted.

## Aesthetic usage

The following four examples demonstrate aesthetic usage through our proposed system.

## Important sound

In contrast to the last examples of physical simulation, this example demonstrates an aesthetic usage, territory sound, which was discussed in (Chion, 1994; Cooley, 1998). Territory sound is a sound effect

that attracts audience attention to a specific location from all the surroundings other than background ambient sound.

In this example, our scene portrays a musical concert (Fig.6). The audience is talking about the musicians' performance (so the ambient sound). Among all musicians, the violinist plays the most emotional melody to attract the audience. At the same time, music from the trumpeter and drummer covers the audience sound.

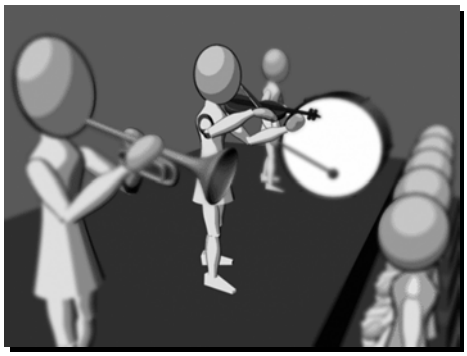


**Fig.6 3D scene used in the territory sound example. On the stage there is (from front to back) a trumpeter, a violinist and a drummer. Below the stage is a line of audience**

Therefore, from the above description, we can arrange the order of importance of these sounds as follows:

$$\text{Violinst sound} > \begin{matrix}\text{Trumpeter sound}\\ \text{Drummer sound}\end{matrix} > \text{Audience sound},$$

where '>' means 'is more important than'.

To arrange the order of importance, we introduce a node called Importance node. A proposed schematic is shown in the following:

"Sound importance" here is the ability of a sound to draw audience attention. For an importance node, we can construct an importance mode in more than one way. For example, we can use volume to distinguish between important and unimportant sound. This concept comes from volume control dynamics mentioned in (Gibson, 1997). A balance mix of all sound effects produces no attraction to us on a specific part. However, when a sound suddenly changes to a higher volume among others, it draws people attention. In other words, it becomes more important.

In addition, we can also use equalization to make a sound becomes closer to a listener, which gives an intimate feeling. For instance, for an important character voice we may boost the high frequency around 3 kHz range by equalization. Or otherwise we reduce 3 kHz range of other background sounds. The result is the dialogue sounds closer and clearer to draw audience attention.

The Importance node (Fig.7e) in our example can also be constructed through the concept of volume control dynamics (Gibson, 1997). We take the idea of volume effect from here (Micea *et al*., 2001). Basically, a scaling factor controls each sound volume. We can sum them up to obtain a final volume:

$$v_{\text{final}} = \sum_{i=0}^{n} s_i v_i, \tag{3}$$

where

$$\sum_{i=0}^{n} s_i = 1, \tag{4}$$

where $v_{\text{final}}$ is the final volume, $s_0,\ldots,s_n$ are the scaling factors and $v_0,\ldots,v_n$ are the original volume of each sound effect respectively. The range of all variables is from 0 to 1. Note that we should prevent the summed volume goes beyond the maximum peak, 0 dBFS (dB Full Scale) level, which causes a clipping sound. We can use "normalize" or "gain" audio functions to reduce the source volume before summing up. In our

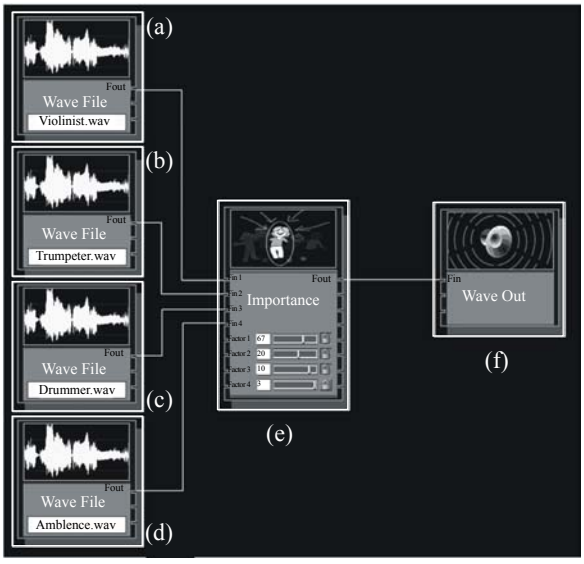



**Fig.7 A schematic for "importance effect". (a)~(d) Wave File; (e) Importance; (f) Wave Out**

example, we used sound sources around −13 dB to −9 dB peak level without any clipping problem. And if necessary, we can apply a hard limiter to cut down any clipped levels, or normalize the final volume (e.g. reduce to −3 dB peak level).

To control the node, we can have two methods. The first one is animated importance. For example, when we decrease the violinist importance over the shot, the other sounds relatively increase their importance. The second method, fixed importance, is that a user locks the most important sound after setting its value and redistributes the remaining importance to the second important sound, and the process is repeated until all sounds have their own importance.

To demonstrate the idea, Fig.8 show the steps of using fixed importance.

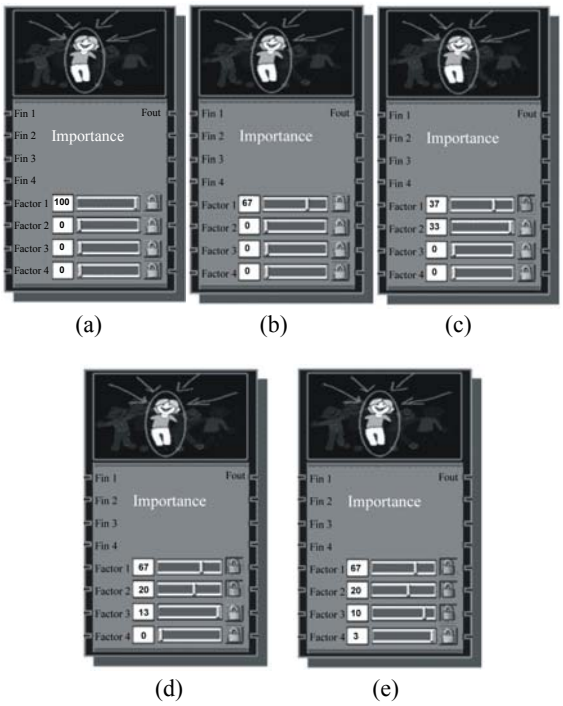



**Fig.8 Five steps of using Importance node for fixed importance setting**

Starting from the first sound (violinist sound), 100% importance is assigned to it before any modification (Fig.8a). The factors 1 to 4 are the scaling factors for each sound:

$$factor1=s_{\text{violinist}},$$
$$factor2=s_{\text{trumpeter}},$$
$$factor3=s_{\text{drummer}},$$
$$factor4=s_{\text{ambience}}.$$

According to Eq.(4), the sum of all scaling factors equals 1 (100%). Therefore, when the *factor*1 holds 100%, all other factors do not have any importance.

Now, the user sets the *factor*1 slider to two-third of it (Fig.8b). It means that we assign 67% importance to the violinist sound. The remaining 33% will be stored in the node.

When the user locks the *factor*1 value by clicking the lock icon beside the slider, it fixes the importance of the first sound to 67% from any further modifications (Fig.8c). Also, the stored 33% importance is sent to the second sound effect of the trumpeter sound.

If the lock icon of the second sound is not pressed, the value is still editable. Note that the second slider is at maximum level of 1.0. This level also represents the *factor*2 is at its maximum value of 33. Therefore, the slider is actually controlling the relative scale of the factor's value (i.e. 1.0=33%, 0.66=22%, …), but not the actual scale. This rule also applies to all sliders we discussed in here.

To continue, the user has set 20% importance of the trumpeter sound and locked it (Fig.8d). The remaining 13% importance is sent to the third sound effect (drummer sound) when the last sound does not hold any importance.

Finally, the user has set the third sound 10% importance and locked it (Fig.8e). The remaining 3% importance is sent to the last sound (ambient sound). Since we do not have more sound effects, that 3% importance may not be modified anymore even though it is unlocked.

At this point, the user has set all importance for each sound effect. All scaling factors will be sent to multiply with the input volume according to Eq.(3) to yield the final volume. The result will then be sent to other connected node by its Fout outlet.

**Inverse importance**

We present this example using the same 3D scene. Consider the following case: at the start, the audience is silent and the music is loud. Thus, the importance of musicians is larger than the audience. A few seconds later, they receive a burst of applause from the audience, which cover the volume of music. Therefore, the importance of audience (their claps) becomes larger than all musicians.

The above example shows a usage of an inverse importance. At the beginning, musician importance is larger than the audience. Afterward, the situation is reversed when the audience claps the musicians' performance.

To define an inverse importance, we compute from an importance node. Refer to the previous example (important sound), our four scaling factors (factor 1~4) are 67%, 20%, 10% and 3% respectively before an inversion. After an inversion, they become 3%, 10%, 20% and 67%. Therefore, in a general case, the scaling factors in an inverse importance will be:

$$\begin{aligned} factor_{ii}[1]&=factor[n]\\ factor_{ii}[2]&=factor[n-1]\\ factor_{ii}[3]&=factor[n-2]\\ \cdots&=\cdots\\ factor_{ii}[n-1]&=factor[2]\\ factor_{ii}[n]&=factor[1] \end{aligned}$$

where $factor_{ii}$ is a scaling factor after an inverse importance.

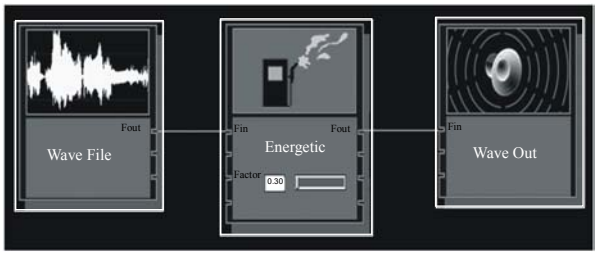
Note that for each factor, the change from a normal importance to an inverse importance should have a short transition. For instance, the scaling factor $s_{violinist}$ changes from 67% to 3% in 500 ms transition. It ensures no sudden discontinuity in volume especially from a very small to a large importance.

**Energetic effect**

Most animators who deal with animation sound do not realize that characterized sound is very effective for pronouncing a trait of an animated object. Rather than using a sound to describe merely a physical phenomenon (for examples, a robot walks with metallic footsteps; or a goblet hits on the ground which has a glass-broken sound), characterized sound adds more meaning to the object. For instance, we can portray the energetic characters of a car (1) with a roaring engine sound to give a feeling of ready for pouncing, or (2) with a sputtering engine sound to give a feeling of breakdown.

To characterize a sound effect, sound manipulation involves modifying the rhythm, temporal attribute (time stretching), and timbre (pitch variation).

In this example, we have a car starting its engine to run. We begin with a natural engine sound without any manipulation. Then we use a node called energetic effect to manipulate the engine sound, which gives different meaning to the car (Fig.9).




(a) (b) (c)

**Fig.9 A schematic for "energetic effect". (a) Wave File; (b) Energetic; (c) Wave Out**

For the Energetic node, we use a method called variable speed replay (Zolzer, 2002) to change the speed of a sound. It sounds like running an analogue audiotape in forward mode with a slower or faster rate than normal playback. It leads to time stretching of a sound, and compression or expansion of the whole frequency spectrum. We can draw a relationship between the input sampling frequency, $f_{s,input}$, and the playback sampling frequency, $f_{s,playback}$, which is given by:

$$f_{s,playback}=sf_{s,input}, \tag{5}$$

where $s$ is a speed factor and is larger than 0.

Fig.10 shows, using the car engine sound in our example, what is happening with different values of $s$ in terms of the time-amplitude graphs (left) and the frequency-audiolevel graphs (right).

Fig.10a shows the temporal length and the frequency spectrum when the speed factor $s$ is in normal ($s$=1.0). Note that, for this case, the spectra of an input sampling frequency and a corresponding playback sampling frequency are consistent. In Fig.10b we see that if $s$ is larger than 1 (e.g. $s$=2.0), temporal length is compressed and the frequency spectrum is expanded (as frequency is inversely proportional to time period, that is, $f$=1/$T$). The spectrum expands towards the right (high frequencies) so a raised pitch occurred. In Fig.10c, we see that if $s$ is smaller than 1 (e.g. $s$=0.5), the time is expanded and the spectrum is compressed. The spectrum constricts towards left (low frequencies) so a lower pitch occurred.

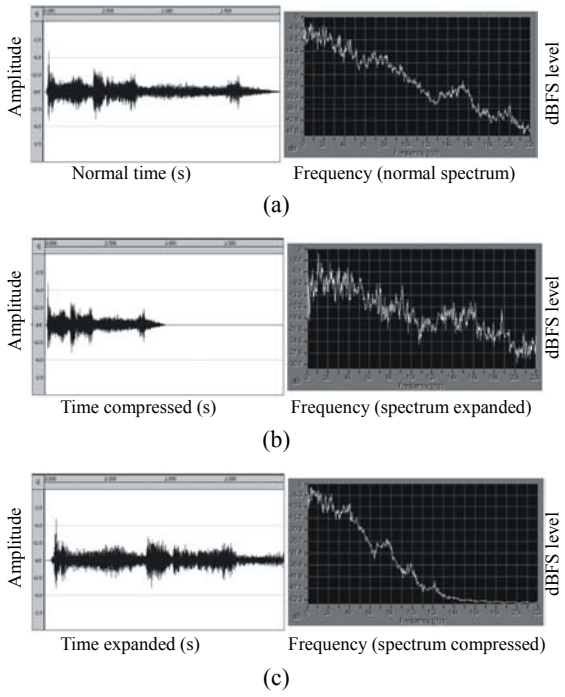



**Fig.10 Three pairs of time-amplitude graphs (left) and frequency-audiolevel graphs (right) with respect to different values of the speed factor, *s*. (a) *s*=1.0 (normal speed); (b) *s*=2.0 (double speed); (c) *s*=0.5 (half speed)**

The phenomenon mentioned above, from the aesthetic point of view, can be a kind of manipulation for creating "energetic" meaning of a sound. As mentioned in (Mott, 1990)[1], the playback speed of a sound can change its property. For a sound with rhythmic characteristic, such as an ignition sound of a car engine, the lowering of its pitch and slowing its time will result in a feeling of losing energy. On the other hand, to heighten its pitch and increase its speed will yield a vigorous impression.

Another clue of the above phenomenon is to playback videotape with different speeds to listen to its sound effects. For instance, with playback mode is enabled, forward in slower than normal rate will make the stepping sound a tyrannosaurus becomes dull in the Jurassic Park. In contrast, fast forward will give its sound like a vivid cartoon chipmunk walking.

From the above examples, speed and energy interfere with the sound property. We attempt to express such relationship with the technique of variable speed replay (Zolzer, 2002). Think about if we treat the speed factor *s* as a factor to indicate energy level of a sound; a higher value gives it more energy so faster in speed, and vice versa. Here, we use this technique as the node basis as describe below:

Digital wave signal feeds the energetic node (Fig.9b) with an input sampling frequency. Users can control a factor, called energy factor, from the slider as shown in Fig.9b, to adjust what kind of energetic level they want. The resulting factor is sent to a variable speed replay unit inside the node. Based on Eq.(5), it calculates the desired output sampling frequency of the input signal.

**Sputtering effect**

In this example, a major factor deciding an engine is "energetic" or not is the rhythm produced by the mechanism inside—a faster rhythm seems to have more energy than a slower rhythm. In general, for the energetic effect we described, a rhythmic sound source, such as a sound of machine, wind blow, seawater, running footstep and vocal, can possibly produce a better result.

Consider this example: a slowed-down engine sound gives us a feeling the car is leaking its gas, which tells us the car is going to break down. However, to emphasize the effect, besides slowing down the sound, we reinforce the rhythm by varying the envelope shape of the sound. Imagine the car is going to break down, and its engine gives a series of fast rhythmic sputters and gradually slows the rhythm down, and finally dies out without sound.

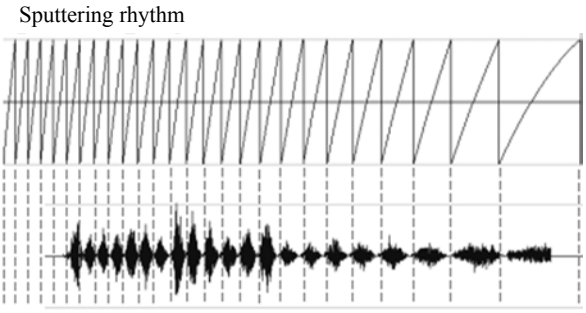
A fast-to-slow rhythm gives a feeling of something is leaking its energy and eventually dies out. The rhythmic impression can be generated through a frequency-alternating saw-tooth wave. The wave starts at a fast frequency, for instance, 12 Hz, and ends with a slow frequency, such as 1 Hz. The frequencies change from fast to slow can give us a feeling of the rhythm that the engine is slowing down.

We name this rhythm as a sputtering rhythm. What we can do with the rhythm is to multiply it with the sound effect signal to yield a new sound that gives a sputtering feel, as shown in Fig.11.

[1] Robert mentioned the speed component, in one of the nine components of sound, varies the properties of a sound effect (Mott, 1990)


This is the original sound with energy factor=0.5 (half speed of the original). It makes the rhythm of the engine sound slower and the timbre duller

(a)




At the top, it is a saw-tooth wave with the start frequency 12 Hz and the end frequency 1 Hz within 4 s. It multiplies with the engine sound to yield a sputtering sound. The result is shown just below the saw-tooth wave.

The saw-tooth wave acts as an envelope for the engine sound after signal multiplication. Two successive dot vertical lines (1 wave cycle) give the shape of a discrete "sputter" unit. As shown in the resulting wave, there are 21 sputters and the sputtering rhythm created by the saw-tooth envelope gradually changes from fast to slow

(b)

**Fig.11 Creation of sputtering engine sound. (a) Original engine sound with energy factor=0.5; (b) Top: saw-tooth wave; Bottom: resulting wave after multiplication**

The sputtering effect can be a node consisting of a saw-tooth wave generator with alternating frequency. Its inlet receives processed wave data from the energetic node. Therefore, in our schematic (Fig.9), we connect it just after the outlet of the energetic node but before the inlet of the wave output node.

## EXPERIMENTAL RESULTS

To create a schematic proposed here, we are implementing our prototype through Max/MSP (Cycling'74, 2005b), a real-time MIDI and audio processing software using primitive flowchart objects. The examples presented in the previous sessions are some test results through our in-progress prototype by using a machine with Pentium IV 2 GHz CPU. At this moment, we still have many interface issues to deal with in Max/MSP. In the next step we will use Jitter (Cycling'74, 2005a), a set of external 2D and 3D graphic objects for Max/MSP environment, to implement an importer for 3DS format that is popular in the game industry. Note that Jitter provides basic graphical functionalities through OpenGL commands, and is suitable for presenting visual output.

## CONCLUSION AND FUTURE WORK

In this paper, we proposed a schematic approach for making sound for computer animation. As animators are usually familiar with schematic interface, our approach helps them to facilitate the job that is traditionally done in sound-editing software with which they are unfamiliar. We believe that such interface assists animators to create and understand a sound network more easily.

Moreover, the schematic has unlimited potential to create any style of sound effects through their imagination of animation parameters. Besides conventional concept of a sound with physical simulation in 3D space, aesthetic usage can also be constructed through the schematic logic. Examples such as importance, inverse importance, energetic and sputtering effect, are shown in this paper.

Since thousands of aesthetic nodes can be custom-built, in the future, we want to find a way to generalize these nodes with well differentiated categories.

## References

Alias, 2005. Retrieved 22 Dec. 2005. Http://www.alias.com/eng/index.shtml

Anilogix, 2005. Sound Technology—Anilogix Pvt. Ltd. Retrieved 25 Dec. 2005. Http://www.anilogix.com/sound-technology.html

Avid, 2005. Retrieved 22 Dec. 2005. Http://www.softimage.com/home/

Boomer Labs, 2005. Retrieved 25 Dec. 2005. Http://www.boomerlabs.com/

Cardle, M., Brooks, S., Bar-Joseph, Z., Robinson, P., 2003. Sound-by-Numbers: Motion-Driven Sound Synthesis. Eurographics/SIGGRAPH Symposium on Computer Animation 2003, p.349-356.

Chion, M., 1994. Audio-Vision: Sound on Screen. Columbia University Press, New York.

Cooley, M., 1998. Sound + Image in Computer-Based Design: Learning from Sound in the Arts. International Conference on Auditory Display 98, Retrieved 2 Dec. 2005. Http://www.icad.org/websiteV2.0/Conferences/ICAD98/default.html

Cycling'74, 2005a. Jitter. Retrieved 9 Oct. 2005. Http://www.cycling74.com/products/jitter.html

Cycling'74, 2005b. Max/MSP. Retrieved 18 Jun. 2005. Http://www. cycling74.com/products/maxmsp.html

Digimation, 2005. Thunder by Digimation, Inc. Retrieved 25 Dec. 2005. Http://www.pluginz.com/product/10413

Discreet, 2005. Retrieved 22 Dec. 2005. Http://www.discreet.com/

Eigenfeldt, A., 2005. MSP-Sample Playback and Processing. Retrieved 27 Dec. 2005. Http://www.sfu.ca/sca/Manuals/247/MSP/MSP-Sample.html

Gibson, D., 1997. Session A, Volume Control Dynamics, The Art of Mixing. MixBooks, Auburn Hills, Michigan, p.79-95.

Hahn, J., Geigel, J., Lee, J.W., Gritz, L., Takala, T., Mishra, S., 1995. An integrated approach to sound and motion. *Journal of Visualization and Computer Animation*, **6**(2):109-123.

Krebs, E.M., 2002. An Audio Architecture Integrating Sound and Live Voice for Virtual Environments. Master's Thesis, Naval Postgraduate School, Monterey, CA. Retrieved 2 Feb. 2006. Https://www.movesinstitute.org/Theses/Krebs.pdf

Mastoropoulou, G., Debattista, K., Chalmers, A., Troscianko, T., 2005. The Influence of Sound Effects on the Perceived Smoothness of Rendered Animations. Proceeding of the 2nd Symposium on Applied Perception in Graphics and Visualization, A Corona, Spain. ACM Press, NY, USA, **95**:9-15.

Micea, M.V., Stratulat, M., Ardelean, D., Aioanei, D., 2001. Implementing professional audio effects with DSPs. *Transactions on Automatic Control and Computer Science, Periodica Politehnica, Timisoara*, **46**(60):55-60.

Mott, R.L., 1990. What is a Sound Effect? Sound Effects: Radio, TV, and Film. Focal Press, Boston, p.53-70.

Newtek, 2005. Retrieved 22 Dec. 2005. Http://www.newtek.com/

O'Brien, J.F., Cook, P.R., Essl, G., 2001. Synthesizing Sounds from Physically Based Motion. Proceedings of SIGGRAPH'01, Computer Graphics Proceedings, Annual Conference Series, p.11-17.

O'Brien, J.F., Shen, C., Gatchalian, C.M., 2002. Synthesizing Sounds from Rigid-Body Simulations. ACM SIGGRAPH Proceedings of Symposium on Computer Animation.

SoundBlaster, 2006. EAX homepage. Retrieved 12 Feb. 2006. Http://www.soundblaster.com/eax/

Takala, T., Hahn, J., 1992. Sound rendering. *ACM Computer Graphics (SIGGRAPH'92)*, **26**(2):211-220. [doi:10.1145/142920.134063]

Unreal Technology, 2005. Retrieved 6 Jan. 2006. Http://www.unrealtechnology.com/html/homefold/home.shtml

van den Doel, K., Kry, P.G., Pai, D.K., 2001. Foley Automatic: Physically-based Sound Effects for Interactive Simulation and Animation. Proceedings of SIGGRAPH'01, Computer Graphics Proceedings, Annual Conference Series, p.537-544.

Zolzer, U., 2002. DAFX: Digital Audio Effects. Wiley, Chichester, England.