3D Sound Design and Technology for the Sensory Environments Evaluations Project: Phase 1

ICT Technical Report
No: ICT TR 01.2001

ICT Technical Report on Sound Design and Technology for SEE: Phase 1
Version 2
By Ramy Sadek, SEE Project Programming Intern
Edited by Dave Miraglia, ICT Sound Design Producer
Project Lead: Jacquelyn Ford Morie

1. Intro

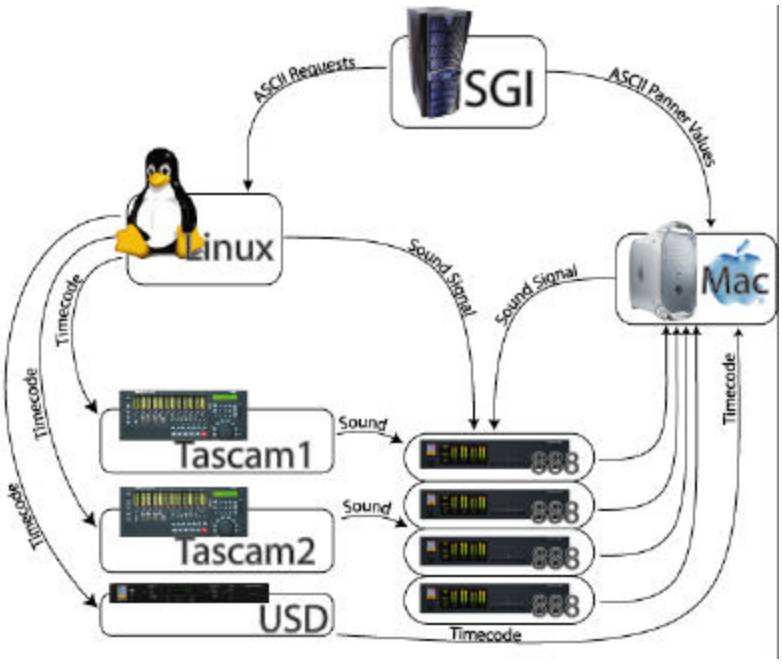
Our current sound solution is multifaceted. The system is comprised of "Pre-mixed stems", "Effects Sounds" triggered off the Linux Box, "Panned Sounds", and the rumble floor. These sounds require a large amount of hardware; in particular we make use of a complete 10.2 sound system (10 full range loudspeakers and two subwoofers), two TASCAMs, a USD, a Linux box with 3 SoundBlaster Sound Cards, a Macintosh G4 with 1 MIX and 1 FARM card connected to 4 888s, and a custom-built rumble floor. The Mac is the keystone of the whole system, since it is the only unit that can send sound to the loudspeakers. For this reason, all sound depends on the Mac. To achieve any sound output, the Mac must be up and running ProTools.

The USD and the 888s are directly connected to the Mac via the ProTools MIX and FARM PCI DSP cards. The 888s stream audio data, while the USD (in this configuration) is the input interface for timecode. By sending timecode to the USD, we effectively send timecode to ProTools.

The TASCAMs are media-playback devices. They are capable of storing and playing a large amount of multi-channel audio data. Our TASCAMs can play 24 channels of sound but we make use of 13, one per channel of our 10.2 plus plus system. The TASCAMs can be operated manually (i.e. by hitting the "play" button) or remotely by sending timecode to them. We use the latter mode of operation.

The 888s are audio interface boxes providing digital and analog audio I/O . They route audio signals out to the speakers and provide the inputs for the Tascams and Linux Box. The MIX and Farm cards are PCI cards in the Mac that house the DSP necessary to process audio data from TDM plug-ins and perform other non-host processes. So, from the Mac, using ProTools software, it is possible to route a set of inputs to a set of outputs through the 888s and play it through the sound system. In fact, this is only possible via the Mac; we have no other way to route the signals.

Figure 1: Control Flow



2. The 4 Sound Types

2.1 Effects Sounds

Understanding how "Effects Sounds" work will simplify comprehension of the other sound types. Effects are those sounds that are triggered immediately upon some condition in the virtual world. The noise of the startled rats is an example of an “Effects Sound.” This type of sound is played directly into the Mac’s audio input (888) by playing the sound through the audio output of one of the Linux box’s sound cards.

I have implemented a client-server API to facilitate playing of “Effects Sounds.” The client (in this case, our VR engine) issues a call to play a sound, this is translated into an ASCII text message and sent over the network via a standard UNIX socket to a server, running on the Linux box. The Linux box then parses this message, and forks off a secondary process to play the sound. (A full list of available API calls follows in the appendices). This secondary process typically involves a call to SOX, which is the open-source sound utility on the Linux box. All “Effects Sound” files are located on the Linux box in a directory accessible by the server process.

The server issues the command to fork off a new SOX process. SOX then plays the analog sound through a single channel on the sound card. Two outputs from the soundcard are hardwired to two discrete audio inputs on the 888’s. The sound effect is appropriately routed within the ProTools software and then played back through the 888s and out to the respective loudspeakers. This is necessary because only the 888s can output to the speakers. For all other sounds, the same routing into and back out of the 888’s is necessary.

2.2 Pre-mixed Sounds

Pre-mixed Sounds are artistically designed multi-channel effects stems. For example, we use this sound type for ambient sound mixes, environmental sounds, truck drive-overs and by's. The term "stem" is used in film mixing to signify a set of multichannel sound files (e.g. dialog stem, music stem, and effects stem) that only when played simultaneously to their respective loudspeaker locations create an immersive effect. For our system, generally a stem is 13 channels, but could be could be comprised of any number of channels. The process for playing them is similar to "Effects sounds" with two major exceptions. Whereas the "Effects" are sound files located on the Linux box, the "Pre-mixed Sounds" are files located on the two TASCAMs. These Pre-mixed sound stems are triggered by sending timecode to the TASCAMs. The call to send timecode works exactly the same way as the Effects calls, except that SOX is called to play a timecode file as opposed to an effect file. The timecode file is effectively an 80 bit analog sound which uses a binaray coded decimal format to maintain frame accurate time coordinates (e.g.. Hour:Min:Sec:Frame) which correspond to the timecode position at which the effects were originally recorded on the device. To trigger the second Tascam, the same play sound call is used, but the timecode is routed to a different soundcard output on the Linux box which is hard wired to the timecode input of the second Tascam.

Since we can at most trigger one "Pre-mixed Sound" at a time on each TASCAM, we utilize both of them to saturate our sound space. For example, while the participant is in the culvert, one TASCAM plays ambient culvert sounds continuously. In order to trigger the sound for a truck driving overhead, we play the appropriate timecode file to the other TASCAM. We can also, via the USD, (Universal Slave Driver) play timecode to ProTools itself, which will then start to play the pre-defined mix it has open. In this case it plays the outdoor soundscape. Consider it to be a virtual third TASCAM.

The current configuration for the SEE demo has:
*Tascam 1 playing the Pre-mixed Soundscape for the Culvert.
*Tascam 2 plays triggered Pre-mixed effects stems for the Culvert and outside (e.g. trucks and the ending dog bark/yell) These particular effects can be triggered at any point in the demo. They differentiate from the soundscapes in that they are event based and not a continuous "loop" or stream of sound that is triggered once and plays until the end of the demo.
*ProTools is used to play back the Pre-mixed continuous background soundscape for the outdoors portion of SEE. ProTools not only plays back the outdoor soundscape, but simultaneously plays the sounds which are dynamically panned.

2.3 Panned Sounds

The most interesting sound types are "Panned" or spatialized sounds. These are sounds that are played in such a way that they seem as though they are sharing the same three-dimensional space as the participants. If there is a sound coming from directly in front of the virtual participant, and he turns his head (virtually, not physically) ninety degrees to the right, the sound pans to the left speaker. This physical manifestation of virtual motion helps the participants to locate themselves in the virtual world.

Fundamental to the workings of the Panned Sounds is the JLA panner developed by Johnny Lee supervised by Chris Kyriakakis. This tool is a real-time audio suite (RTAS) plugin for ProTools. ProTools supports two types of real-time plugins, RTAS and TDM. The JLA panner is an RTAS plugin, which means it runs on the host CPU. As such the panner can only be run on "disk tracks" (i.e. tracks with a soundfile loaded on them) and not on auxiliary input or record enabled tracks. Therefore, it cannot run on sounds that must be "triggered". That is, only sounds with known beginning times or ones which continuously loop can be loaded and processed by the panner. This problem has been a central issue leading to the complexity of the current solution. This and other limitations will be addressed in a later chapter.

The panner construct is a fairly simple one. The plugin has 3 modes: Listener, Receiver, and Sender. The Receivers represent physical loudspeaker locations. Since we have 10 full range loudspeakers, there are 10 Receivers in the ProTools session, one per loudspeaker. The Sender plugins are placed on tracks with sounds that require realtime panning. For example, to create a spatialized cricket sound, one need only create a disk track in ProTools and put a cricket sound on the track, then instantiate a Sender and give the Sender the desired spatial coordinates. The Listener represents the location and orientation of the virtual participant. Thus, according to the relative location of the Listener to the Senders, the Receivers can determine the appropriate volume for the respective speakers, thereby spatializing the sound. This process is entirely dynamic, which means that both the Listener and the Senders can move with respect to one another (note that the sender must have its "relative" mode enabled for this to work, otherwise it ignores the Listener entirely and is scaled only by proximity to the Receivers).

The panner listens for positional data via the network. A client directly establishes a connection with the panner using the MAC's IP address, and updates the panner's parameters on the fly. In the case of our application, the engine forks off another process which monitors the participant's position and orientation. Since the panner calculates only within a virtual cube spanning -100.0 to 100.0 on each axis, this monitor process normalizes the positional values and then re-maps them to this cube in order to produce correct positional correspondence between the visual world and the aural space. Note that the orientation of the virtual aural world and the normal orientation of the engine's virtual world have opposite up-vectors. That is, one is upside-down from the other. To correct for this, we flip the Listener over by giving him a Roll (as in Yaw, Pitch and Roll) value of 180 degrees. All communications are handled via an API implemented by Ramy Sadek and John Deweese, the source code of which will follow in the appendices.

2.4 The Rumble Floor

The last type of sound we support is sound sent to the Rumble Floor. This is a special floor custom-built for ICT. It has 10 transducers in it that can produce sound at subsonic frequencies; below the threshold of audibility for normal human hearing. These transducers are tremendously powerful, and their respective amplifiers are set to allow a maximum volume of only one-half their power. The transducers all share a single mono input, which is sent to a matrix switch, and then out to the mono-input of all ten transducers. In order to switch on or off particular transducers, we communicate with the

matrix switch via a serial (RS232) connection. This allows us to pan sound effects through each of the transducers in the floor. This is much simpler than generating a separate signal for each transducer.

3. Limitations

While this system works very well for a simple, tightly scripted and choreographed scenario, it cannot function in a truly dynamic environment. Discussion of what is needed to create an appropriate general solution is left to the next chapter. This chapter is a discussion in detail of the limitations of the current system.

The most significant shortcoming of the current solution is that it does not and cannot allow dynamically triggered Panned sounds. Because ProTools will only allow the panner to run on a disk track, only sounds that begin at a predetermined time and are loaded prior to the beginning of the virtual exploration can be panned. This is acceptable if there are scene elements that are continuously noisy and repetitive like a cricket, or a helicopter. But for anything with a specific beginning or end, like a truck starting up, doors slamming, voices, etc, the panner cannot be used. This limitation is severe, and must be addressed.

As a work-around, we developed the method with the two TASCAMs which allows us one 13-channel sound stem per TASCAM. That is, we can send a grouped sound to any of 13 particular speakers, but the relationship between sound and speaker is not determined on the fly. Rather its position is hard-encoded within the session the TASCAM is playing, and it ignores the participant's position and orientation entirely. For example, if there is to be a truck to the left of the participant as he faces forward, but he has turned his head, the sound will still come out of the left speaker but will sound to be on his right. In our attempt to create the illusion of a truly realistic acoustic environment, this limitation is extremely problematic. With such restrictions suspension of disbelief cannot be sustained. If the movements of the participant are not carefully scripted, the demonstration will completely violate fundamental laws of psyhoacoustics. In real life, direct sound should emit from the relative location of the object and its orientation to the participant. Reflected sounds must also comply in real time.

The second limitation of this TASCAM method is that we can only play three pre-mixed sets of sound stems, one from each TASCAM and one from ProTools. For example, if we have a set of ambient outdoor sounds, a set of truck sounds, a set of voices, and a jeep, each triggered as a response to different actions by the participant, we could not have all four playing. Instead, the scenario designers must ensure that the participant is never able to trigger a fourth event until the second is finished playing. This severely limits our creative possibilities and hinders our ability to make the scenario truly complex, multi-layered, and dynamic. We could not, for example, have the participant make a mistake leading to his capture immediately after he hears a truck roll by. We must wait until the effect of the truck finishes and then play the "final capture stem". The hardware is the limiting factor in this configuration which has an effect on pacing within the scenario.

It is also important to note that it takes the TASCAM about two seconds to synchronize once the timecode is sent, so it cannot be used for immediate effects like synching a "pop" to a balloon burst, or a dog's bark to its mouth motion, a "thud" with a dropping brick, etc.

We have no way to dynamically trigger immediate sounds without including the Linux box. This increases the complexity of our system dramatically, making it easier to break, and harder to fix.

Another problem is that the transition between ambient SoundScapes is not automatable. In order to change smoothly from one SoundScape to another, an operator must be watching the virtual participant's actions and cross-fade between the two SoundScapes using the faders in ProTools.

Sound in a six-degree-of-freedom virtual world is not remotely immersive unless it can be dynamically triggered and spatialized. While we can spatialize in real time, we cannot trigger which severely cripples creative freedom.

4. Improvements

Clearly the main improvement, and currently the highest priority must be achieving dynamically triggered and spatialized sound. That is, we must be able to, at any time, play a particular sound to any of the 10 full range loudspeakers, 2 subwoofers, and Rumble floor depending on the participant's position and orientation at that time. There is no existing solution to this problem, academic or commercial. We must implement our own solution either via the ProTools Plug-in architectures or by implementing our own application to replace ProTools. This means ICT needs a sound programmer. Additionally, in order to build this solution there must be a development machine separate from, but identical to the Mac in the VR Theater. The theater is a shared resource with is simply in too high a demand to be successfully used for sound development purposes. The sound programmer must be free to work diligently irrespective of demos or the needs of research groups to use the theater for testing, otherwise the task is likely to take years.

There are 3 reasons why solving this problem is extremely difficult. First, our panner is an RTAS panner, which cannot be run on auxiliary or record enabled tracks at this time. This means we cannot feed sound signals in from another source (e.g. the linux box) and pan them dynamically. Since this restriction is dictated by the current version of the ProTools software, there is little we can do about it. The panner could be written as a TDM plugin, but this is a problem of severe complexity. In fact, it is probably impossible to create a panner that will meet the hard-realtime constraint of 1000 DSP cycles.

The second problem to overcome is that we cannot trigger a sound in ProTools except by using timecode which limits us to only playing back ONE Pre-mixed stem at a time. In order to get ProTools to play a sound in a dynamic (as opposed to scripted) fashion is to

write a new application that interfaces with ProTools via DirectConnect. Since DirectConnect is a TDM plugin, no RTAS plugin can follow it at this time.

Another alternative would be to implement an RTAS plugin that will dynamically feed sound samples to the panner. However, there is a hard real-time constraint on RTAS plugins as well. It is not possible to open a file and read from it within the 30ms time allotted. Attempts to do this will invariable lead to a system crash.

Recently DigiDesign introduced a third type of real-time plugin, called HTDM. These plugins allow data from the TDM bus of the MIX cards to be processed on the host CPU. In theory, this could allow the JLA "spatializing" panner to run on dynamically triggered sounds. However, it is unclear how easy it will be to rewrite the panner in such a way that it conforms to the HDTM specifications. In fact, it may not be possible to do so.

The most important problem outside of dynamic panning is the VR Theater's Mac. The VR Theater Mac is completely unreliable. It crashes often even after having a complete disk format and re-install of system software, to the extent that we cannot even count on it to stay operational through the duration of a demo. This makes development and sound design nearly impossible on it. While we have yet to discover the problem (John Deweese, formerly a Developer at Apple was unable to discover it), we are certain it must be fixed or replaced. DigiDesign will soon port ProTools to MacOS X although the shipping date is unclear. This may help minimize the number of crashes. We are still unsure about the cause of the problem and have not run any comprehensive tests on the machine. This unreliability may be an affect of the beta nature of the JLA panner, the Network communication data being sent to the plug-in, or a hardware/PCI card problem.

In the end, the sound solution should be entirely free of the ProTools Software, even if a quick fix involves the ProTools plugin architecture. The ideal situation is a completely general and abstract implementation that allows any type of input to go to any type of plugin, and for any type of plugin to follow any other type of plugin. The commercial ProTools software does not allow any of these options: plugin type order is currently restricted, and allowed plugin type depends on track type. We must be free of such restrictions in order to create successful virtual worlds. It is essential that we develop our own system on top of the DAE (DigiDesign Audio Engine). The DAE is the lower level system that the ProTools runs on top of. We can harness the DAE's ability to access sound files direct from disk and it's capability for multichannel output for our own purposes. We are currently development partners with DigiDesign Corp. with free access to the DAE dev kit.

Ultimately, we would like to have a fully dynamic spatialized sound presentation powered by an Immersive Sound Engine built on top of the DAE. We shall experiment with a custom filter plugin that interprets the dimensions and reflective/absorptive properties of the environment and surfaces in CG environments. This would allow the CG modelers to specify acoustic properties of their objects and spaces using a palette of materials. They could choose from traditionally absorptive materials like cloth and wood, or reflective surfaces like metal and marble. The environment parameters would be relayed to the filter plugins of our Immersive Audio Engine and change as the

participant moved from one environmental space to the next. For example, if the participant was to move from a culvert to an open outdoor environment with no reflective surfaces, the plug-in would change parameters based on data sent from the CG world that would update and modify all effects played in that environment. In it's most simplest form, the footsteps of a soldier moving through a long culvert would be more reverberant than those same footsteps in an open outdoor environment with less reflective surfaces. The custom filter would effectively, render the sound in real-time based on the size and textures of the space.

Modelers should also be able to specify what type of sound is triggered when the participant moves through the world. We may provide a toolset for CG modelers which would allow them to "texture map" sound objects to particular areas of the CG world. If, for instance, the culvert floor in some areas were gravel and in other portions puddles of water, the CG world would need to relay these surface properties to the Immersive Audio Engine which would in turn play the appropriate sound type. The crunch of the soldier's footsteps on defined "gravel regions" should seamlessly change into the sound of a boot splash in water as he moves into the "puddle regions" of the world.