CS 4455: Video Game Design & Implementation

March 31, 2006: Audio

(Insert Disclaimer Here)
Overview

- Today’s Lecture
- What I’m talking about now
- Audio Theory
- Digitizing Sound
- Game Implementation
- High Level APIs
Why is Audio important?
What is audio?

- Inside your ear is an eardrum
- A thin piece of skin
- When it vibrates, your brain interprets this as sound
- Changes in air pressure often cause this vibration
How Audio Works

• An object produces sound when it vibrates
• This moves air particles
• Those particles in turn move other particles
Terms to Note:

- **Wavelength** - distance between repeating points
- **Amplitude** - non-negative height of the wave
Audio Terminology

Terms:

- Period - How long it takes between cycles
- Frequency - How many cycles occur

(These are inverses)
Audio Terminology

The faster they loop, the higher their frequency.

The SI unit for this is **Hertz** (Hz).

1 Hz = once a second,
1 KHz = one thousand times a second
Audio Terminology

- **Intensity** = the “power” of the sound
- A fairly large scale, so usually expressed logarithmically:

\[
I_{\text{dB}} = 10 \log_{10} \left( \frac{I}{I_0} \right) \quad \text{or} \quad P_{\text{dB}} = 10 \log_{10} \left( \frac{P}{P_0} \right),
\]

- With Sound, \( I_0 \approx 10^{-12} \text{ W/m}^2 \)
# Common DB Levels

<table>
<thead>
<tr>
<th>DB Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0dB</td>
<td>Threshold of hearing</td>
</tr>
<tr>
<td>10dB</td>
<td>Human breathing at 3 meters</td>
</tr>
<tr>
<td>30dB</td>
<td>Theatre, no talking</td>
</tr>
<tr>
<td>60dB</td>
<td>Inside of office or restaurant</td>
</tr>
<tr>
<td>70dB</td>
<td>Busy traffic at 5m</td>
</tr>
<tr>
<td>90dB</td>
<td>Loud factory, heavy truck at 1m</td>
</tr>
<tr>
<td>100dB</td>
<td>Jack Hammer at 2m; inside disco</td>
</tr>
<tr>
<td>120dB</td>
<td>Rock Concert</td>
</tr>
<tr>
<td>150dB</td>
<td>Jet engine at 3m</td>
</tr>
<tr>
<td>250dB</td>
<td>Inside tornado; nuclear bomb @ 5m</td>
</tr>
</tbody>
</table>

+10dB means 10 times as powerful
+3dB roughly twice as powerful
<table>
<thead>
<tr>
<th>dB</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>potentially harmful to hearing</td>
</tr>
<tr>
<td>120</td>
<td>unsafe</td>
</tr>
<tr>
<td>150</td>
<td>physical damage to body</td>
</tr>
<tr>
<td>163</td>
<td>windows break</td>
</tr>
<tr>
<td>19x</td>
<td>eardrums rupture</td>
</tr>
<tr>
<td>200</td>
<td>can cause death</td>
</tr>
</tbody>
</table>
Digitizing Sound
At a given interval, “sample” the amplitude of the wave
Sampling: Nyquist Limit

Nyquist Limit - a given sampling rate can only represent frequencies up to one-half that rate
Sampling

- Typical factors on a computer
  - How many times per second?
  - How many levels can we differentiate between?
  - How many channels?
Sampling

- CD Quality audio
- 44kHz (44,100 samples/sec)
- 16-bit (65,536 possible levels)
- 2 channels (left and right)
- 176,400 bytes/sec
- This is approximately 6 seconds per megabyte!
Sampling

- Low Quality Audio
  - 8kHZ (8 samples per second)
  - 8-bit (1 byte)
  - 1 channel
- 8,000 bytes/sec
- About two minutes per megabyte
Sampling Comparison
What can we do?
(Or, a brief history of computer audio)
FM synthesis

- Used in early systems like the GameBoy
- Hardware continually produced one or more sine waves (sometimes other shapes, too)
- Software could modify frequency and amplitude
- Can be done in very little space (the BIGGEST GameBoy games were about 1/2MB)
FM synthesis demo
MIDI synthesis

- Common in PC sound cards and many consoles, such as the Super Nintendo (SNES), Genesis, your cellphone...

- MIDI files contain instructions to turn on or off various instruments

- Instruments are externally defined

- Therefore, small file format

- Sound can differ player-to-player
MIDI synthesis demo
Module Audio

- Like MIDI, but you can (or are required to) supply your own instruments.
- .MOD/.S3M/.XM/.IT file formats.
- Used in the PlayStation, also common on the GameBoy Advance.
- Usually still small, if you can share instruments.
Module Audio Demo
RedBook Audio

- Music is streamed from CD
- Commonly used on the SegaCD
- May need to buffer or avoid when you need to read from the CD!
- Takes a lot of space
RedBook Audio Demo

(Realistic Depiction of Bonus Stage)
Compressed Audio

- General Compression
  - Lossy - small changes okay
  - Loseless - must be 100% preserved
Compressed Audio

Audio Compression

Bit Reduction

DPCM encodes the differences between subsequent samples (the D is for differential or delta)

ADPCM is a more advanced version
Compressed Audio

- Audio Compression
- Psycho-acoustic
- Designed with human hearing in mind
- MP3, AAC, OGG, WMA, .etc
Compressed Audio Demo

Nah.
Gaming Implementation
Sound in most games is divided into two parts:

- Background music
- Sound Effects
Background Music

- Unlike movies and TV, not timed
- Ambient and looping
- Can be streamed
Sound Effects

Characteristics

- Typically very short
- Often tied to an event
- Examples: Gun fire, character is hit, explosion, speech, etc.
- Generally stored in memory
Mixing

- In many game consoles, separate HW for these functions is not unusual
- Hardware MIDI standard in some
- If not, convert to sampled in software
Mixing

- Software mixing is easy
- If sample rates are the same
- Just add!
- Beware of exceeding the max
Mixing Example

def mixSound(dest, source):
    for i in range(1, min(getLength(dest), getLength(source))):
        sourceValue = getSampleValueAt(source, i)
        destValue = getSampleValueAt(dest, i)

        setSampleValueAt(dest, i, sourceValue + destValue)
Buffering

- When reading or converting sound, you need to stay ahead of the audio output device but can’t convert the whole song

- Two techniques for buffers:
  - Circular buffers - read and write in same buffer
  - Buffer chaining - write to buffer, read from the other, swap
High Level APIs

Audio is a lot simpler than graphics (in a game).

A lot of APIs can be condensed to:

- `Play(sample, loops)`
- `Stop(sample)`
- `SetPan`
- `SetVolume`
- `SetSpeed`
Example Sound APIs

- Cross platform 2D APIs
  - Java Sound
  - SDL
  - QuickTime
- Sound APIs with 3D support
  - OpenAL
  - DirectSound (NOT crossplatform)
Example (JavaSound)

```java
Sequence sequence =
    MidiSystem.getSequence(new java.net.URL(url));

// Create a sequencer for the sequence
Sequencer sequencer = MidiSystem.getSequencer();
sequencer.open();
sequencer.setSequence(sequence);
sequencer.start();
```

See packages under javax.sound such as javax.sound.midi and javax.sound.sampled
Positional Audio

Basic Theory

- Sound distance and volume are inversely related
- Sound differences in the ears help determine position
3D Audio

- Problem with traditional two speaker or headphone setup
  - Forward vs. Behind
- Speaker setups available that have 5, 6, or 7 speakers
- Speaker setup tends to differ, so this is difficult!
3D Audio Example

// Load wav data into a buffer.
alGenBuffers(1, &Buffer);
if (alGetError() != AL_NO_ERROR)
    return AL_FALSE;
alutLoadWAVFile("wavdata/Footsteps.wav", &format, &data, &size, &freq, &loop);
alBufferData(Buffer, format, data, size, freq);
alutUnloadWAV(format, data, size, freq);

// Bind buffer with a source.
alGenSources(1, &Source);
if (alGetError() != AL_NO_ERROR)
    return AL_FALSE;
alSourcei (Source, AL_BUFFER, Buffer);
alSourcef (Source, AL_PITCH, 1.0f);
alSourcef (Source, AL_GAIN, 1.0f);
alSourcefv(Source, AL_POSITION, SourcePos);
alSourcefv(Source, AL_VELOCITY, SourceVel);
alSourcei (Source, AL_LOOPING, AL_TRUE);

// Do an error check and return.
if (alGetError() != AL_NO_ERROR)
    return AL_FALSE;
// Initialize OpenAL and clear the error bit.
alutInit(NULL, 0);
alGetError();

// Load the wav data.
if (LoadALData() == AL_FALSE)
    return 0;

SetListenerValues();

// Begin the source playing.
alSourcePlay(Source);

// Loop
ALint time = 0;
ALint elapse = 0;

while (!kbhit())
{
    elapse += clock() - time;
    time += elapse;

    if (elapse > 50)
    {
        elapse = 0;
        SourcePos[0] += SourceVel[0];
        SourcePos[1] += SourceVel[1];
        SourcePos[2] += SourceVel[2];

        alSourcefv(Source, AL_POSITION, SourcePos);
    }
}
Resources

- OpenAL tutorials:
  - http://www.devmaster.net/articles/openal-tutorials/lesson1.php

- Sound Editing Software:
  - http://audacity.sourceforge.net/

- Sound Effects
  - /net/dvfx/hollywood_edge
  - http://www.sounddogs.com/