Multimédia prezentace přednášek Zvuk, DirectSound

Lukáš Polok

Vysoké učení technické v Brně, Fakulta informačních technologií
Božetěchova 2, 612 66 Brno
ipolok@fit.vutbr.cz

28.03.2013

Co je to zvuk

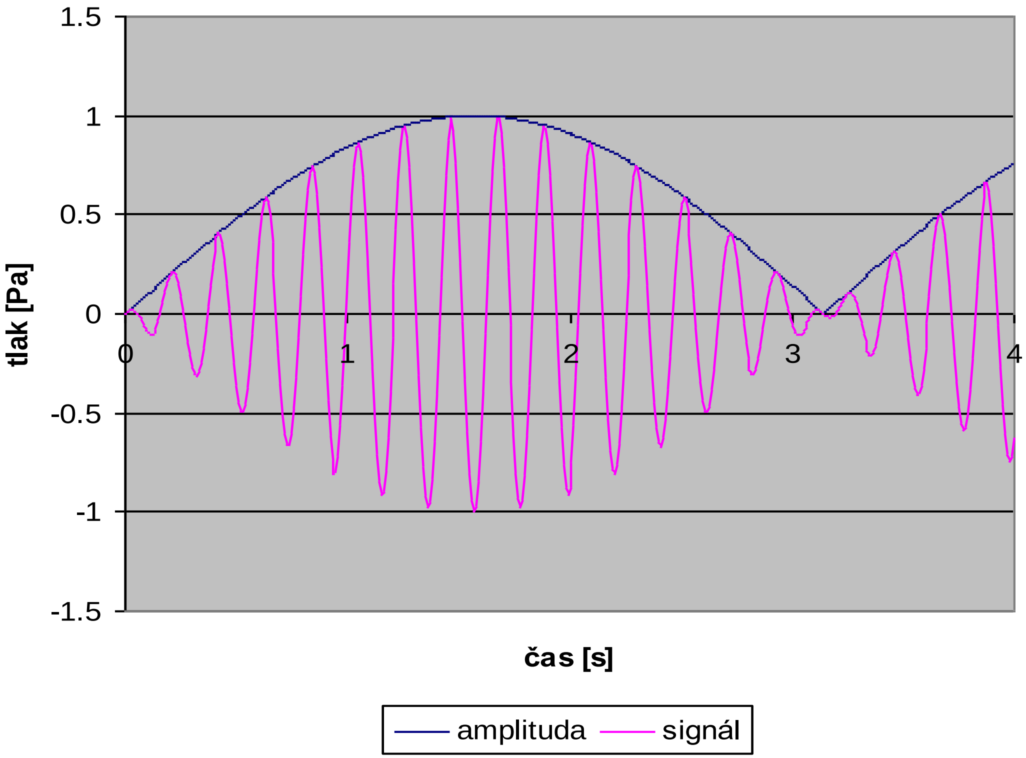
- „Slovo které řeknete, nedostanete zpět ani párem velbloudů“

  \- Ježíš K.

Co je to zvuk (2)

- „Nějaké“ vlnění v čase

- V analogovém provedení je to funkce (změny) tlaku vzduchu v čase
- Má **amplitudu**
- Má **frekvenci**



- Pravidelně **vzorkujeme** naši změnu tlaku v čase
- Dostaneme řadu čísel, které reprezentují zvuk
- Problémy
  - **Jak často** se musí **vzorkovat**
  - Kolik desetinných čísel je třeba si pamatovat (**přesnost**)

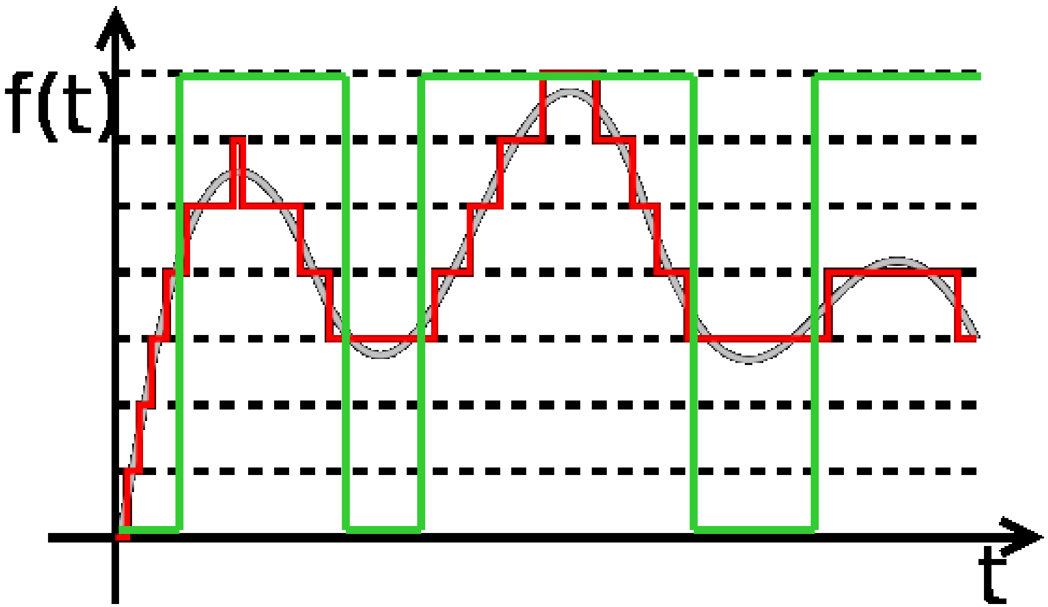
t [sec]	signál
0.00	0.0000
0.01	0.0096
0.02	0.0165
0.03	0.0186
0.04	0.0145
0.05	0.0035
0.06	-0.0136
0.07	-0.0353
0.08	-0.0589
0.09	-0.0813
0.10	-0.0988
0.11	-0.1084

Vzorkovací frekvence

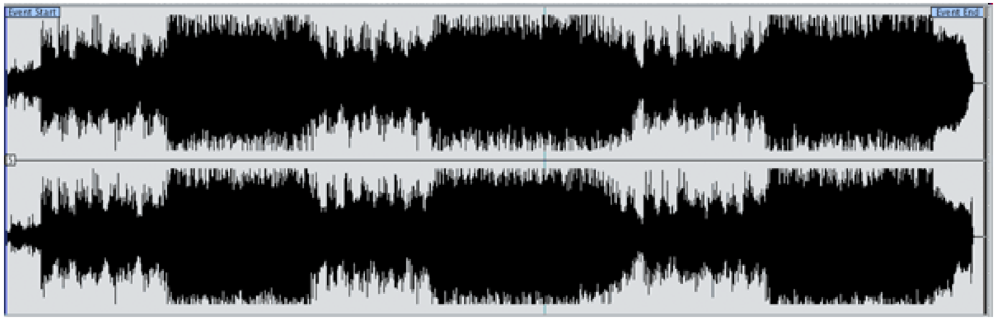
- Shannon Nyquist Kotelnikův teorém
- **Maximální frekvence** která jde přesně zachytit je polovina **vzorkovací frekvence**
- Běžně používané
  - **44100** Hz
  - 48000 Hz
  - 96000 Hz
  - 192000 Hz
- Odvozené od toho, jak (ne)slyšíme

Dynamický rozsah

- Souvisí s přesností záznamu
- Zásadně ovlivňuje naše vnímání
- „Jak velký rozdíl může nahrávka mít mezi tichým místem a hlasitým místem“
- S tím souvisí **počet bitů**
  - **16** bitů / vzorek
  - 24 bitů / vzorek

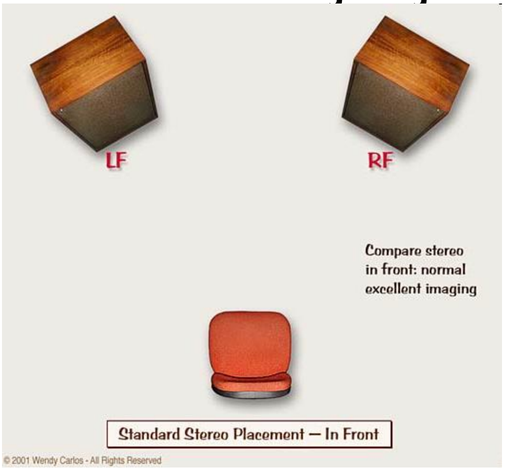


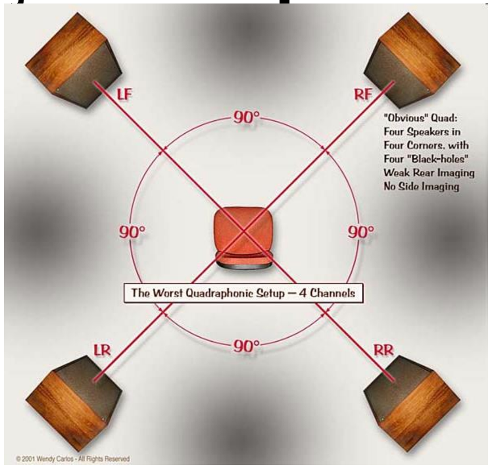
- Pokud je zvuk dlouho hlasitý, ucho se „unaví“
- Pokud se ale intenzita mění, je zvuk pro posluchače zajímavější
- Nirvana - "Heart Shaped Box" (1993)

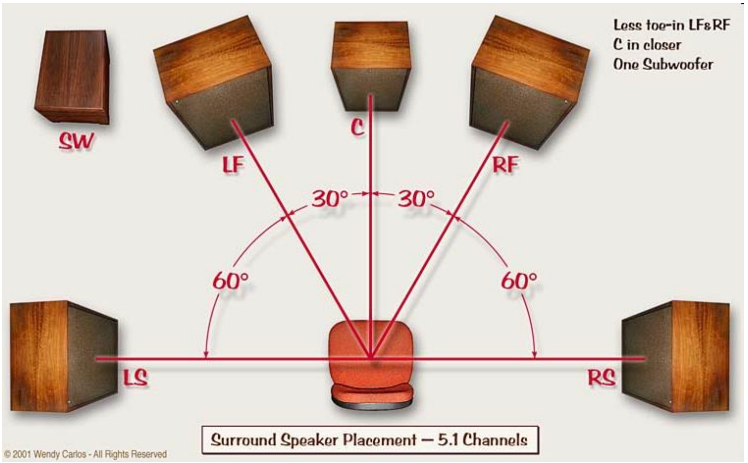


- Stooges - "Search and Destroy" (1997)

- Ovlivňuje jak je zvuk **prostorově** vnímán



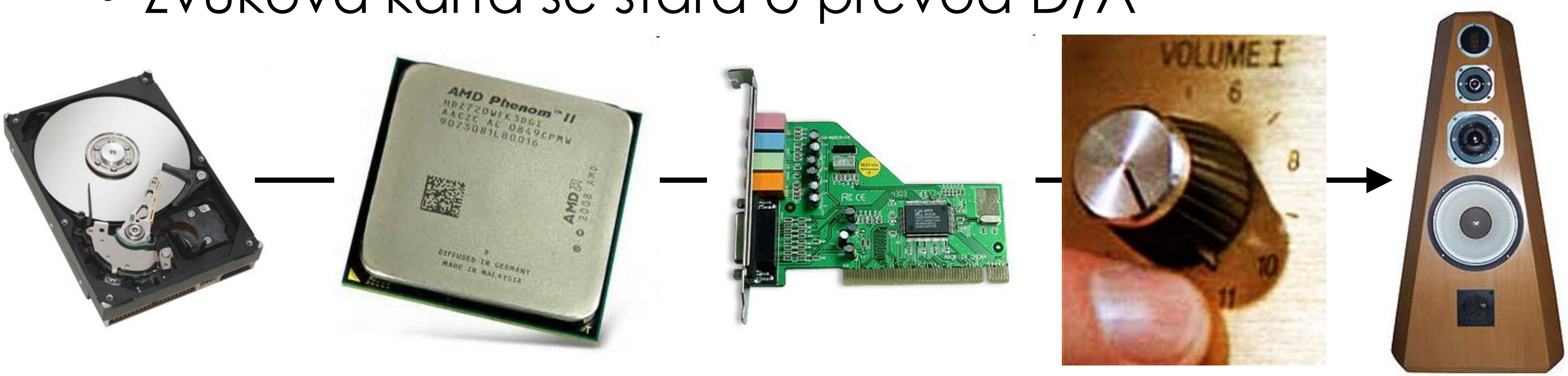




- Mono (1 kanál)
- **Stereo** (2 kanály)
- Quadro (4 kanály)
- 5.1, 7.1, 9.1 (mnoho kanálů, použití ve hrách)
- Každý **kanál** je vlastní stopa, čím víc kanálů, tím víc dat musíme uložit

Zvuk v počítači

- Digitální data
  - **16** bitů x **44100** Hz x **2** kanály = **1.35 MB / sec**
- Pro ukládání je možné data komprimovat
  - 128 kbit MP3 = **16 kB / sec**
- Při přehrávání
  - Je třeba data rozbalit (pokud je použita komprese)
  - Je třeba data nahrát do zvukové karty
  - Zvuková karta se stará o převod D/A



- Problémy: Frank Herbert's Dune

- Problémy: Frank Herbert's Dune

- Problémy: Frank Herbert's Dune

- Problémy: Frank Herbert's Dune

- Problémy: Frank Herbert's Dune

- Problémy: Frank Herbert's Dune

- Problémy: Frank Herbert's Dune

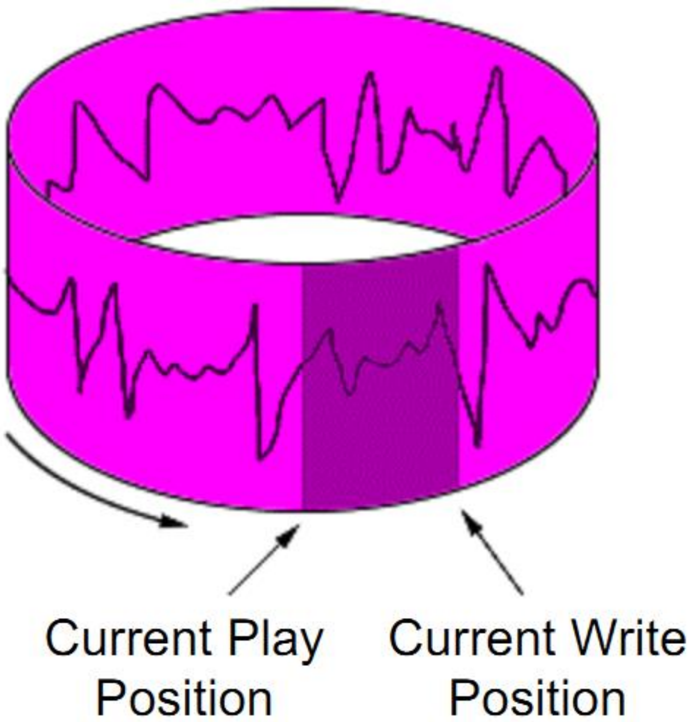
- Problémy: Frank Herbert's Dune

- Základní problém
  - Zvuková karta (hardware) má omezenou paměť
  - Pokud máme např. **5 minut** nahrávku, potřebujeme 5 x 1.35 = **6.75 MB** paměti
  - Většina karet ale má méně než **1 MB buffer**
- Musíme zvuk **streamovat**
  - Tzn. po malých kouskách posílat k přehrávání
  - Nutnost **synchronizace**

- Jednoduché řešení - ping pong buffer

Zvuk v počítači - Streamování (2)

- Jiné řešení - kruhový buffer
- Dvě pozice
  - Přehrávací
  - Zapisovací
- CPU se musí starat, aby ho zvuková karta „nedoběhla“

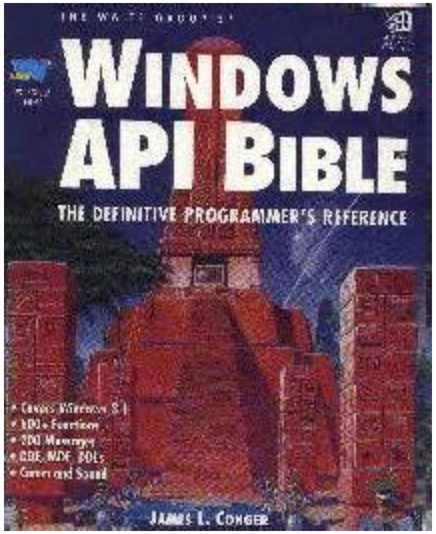


- **Latence** je doba za jak dlouho se zvuk dostane od CPU do reproduktoru
- Většinou chceme malou latenci
  - V počítačových hrách (**desítky mS** jsou ok)
  - V software na skládání hudby (**jednotky mS**)
    - Pro **10 mS** stačí **14 kB** buffer (44100 Hz x 16 bit x 2)
- Čím **větší buffery**, tím **větší latence**
- Ale: čím **menší latence**, tím **častěji** se buffer musí **naplnit**
  - Více práce pro CPU
  - Náročnější na synchronizaci
  - Nebezpečí „vypadávání“ zvuku

- Pro přehrávání zvuků lze použít WinAPI

PlaySound(“tada.wav“, NULL, SND_ASYNC);

- Poměrně málo kontroly nad přehráváním
  - Co přehrát
  - Jednou / donekonečna
  - Čekat / nečekat
- Umí jen .wav formát
- Poměrně málo starostí s přehráváním



- Pro přehrávání zvuků lze použít MCI (Media Control Interface)

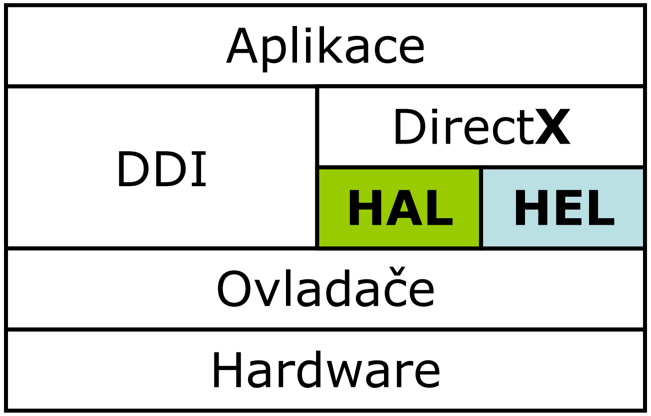
mciSendString(**command_string**, NULL, 0, NULL);

- O něco víc kontroly nad přehráváním
- Umí různé formáty
- Poměrně málo starostí s přehráváním
- Dnes nepoužívané

Zvuk na Windows (3)

- **OpenAL**
  - jako OpenGL, ale pro zvuk
- **ASIO**
  - profi knihovna pro muzikanty
  - vyžaduje speciální HW
  - velmi nízká latence (1 mS není problém)
- **DirectX**
  - Microsoftí knihovna, na Windows
    - Ale také XBOX a další konzole
  - Používaná z jiných knihoven (např. SDL)

- Odstiňuje problémy s ovladači HW
- Obaluje HW s různými funkcemi jednotným rozhraním
- **HAL** – hardware abstraction layer: konzistentní rozhraní pro ovládaná zařízení.
- **HEL** – hardware emulation layer: softwarová implementace funkcí, jež nejsou podporované HW.



Moduly knihoven DirectX 7

• DirectDraw	2D Grafika
• Direct3D	3D Grafika
• DirectSound	Přehrávání zvuků
• DirectMusic	Přehrávání hudby
• DirectInput	Ovládací zařízení
• DirectPlay	Komunikace po síti
• DirectSetup	Nastavení DirectX

Moduly knihoven DirectX 8

• DirectDraw	2D Grafika
• **DirectX Graphics**	2D + 3D Grafika
• DirectSound	Přehrávání zvuků
• DirectMusic	Přehrávání hudby
• DirectInput	Ovládací zařízení
• DirectPlay	Komunikace po síti
• DirectSetup	Nastavení DirectX

Moduly knihoven DirectX 9

• DirectDraw	2D Grafika
• DirectX Graphics	2D + 3D Grafika
• DirectSound	Přehrávání zvuků
• DirectMusic	Přehrávání hudby
• DirectInput	Ovládací zařízení
• DirectPlay	Komunikace po síti
• DirectSetup	Nastavení DirectX

Moduly knihoven DirectX 10

• **Direct2D**	2D Grafika
• **DirectWrite**	Fonty
• Direct3D	3D Grafika
• **DXGI**	Detekce zařízení
• **XACT** (audio creation tool)	Přehrávání zvuků
• DirectMusic	Přehrávání hudby
• **XInput**	Ovládací zařízení
• DirectPlay	Komunikace po síti
• DirectSetup	Nastavení DirectX

Moduly knihoven DirectX 11

• Direct2D	2D Grafika
• DirectWrite	Fonty
• Direct3D	3D Grafika
• DXGI	Detekce zařízení
• **DirectCompute**	GPGPU
• **XACT** (audio creation tool)	Přehrávání zvuků
• DirectMusic	Přehrávání hudby
• XInput	Ovládací zařízení
• DirectPlay	Komunikace po síti
• DirectSetup	Nastavení DirectX

DirectSound

- Základní podpora pro přehrávání zvuků
- Podpora pro rychlé mixování zvukových kanálů
- Přímý přístup na zvukové zařízení
- Podpora pro nahrávání zvuků
- Podpora pro prostorové (3D) efekty zvuku
- Fyzikální efekty např. Dopplerův efekt
- **Latence** mezi spuštěním a přehráním je **~20 mS**

- Základem je objekt `DirectSound`
- Rozhraní objektu se jmenuje `IDirectSound`
- Rozhraní získáme voláním:

```
LPDIRECTSOUND DSound;
DirectSoundCreate(GUID, &DSound, NULL);
```

- Přes toto rozhraní pak voláme funkce popř. získáváme další rozhraní, například:

```
IDirectSoundBuffer
IDirectSoundCapture
IDirectSoundCaptureBuffer
IDirectSound3DBuffer
IDirectSound3DListener
```

DirectSound - Inicilaizace (2)

- **SetCooperativeLevel**(HWND, level)
  - Nutné při inicializaci
  - DSSCL_**NORMAL**
    - možnost spolupráce s dalšími aplikacemi
    - primární formát dat musí odpovídat DirectSound
  - DSSCL_**PRIORITY**
    - lze měnit primární formát dat
    - nový formát se projeví i v jiných aplikacích
  - DSSCL_**EXCLUSIVE**
    - výhradní přístup ke zvukovému zařízení
    - ostatní aplikace se ztlumí, nemůžou přehrávat zvuky
  - DSSCL_**WRITEPRIMARY**
    - přístup k primárnímu bufferu pro mixování
    - je potřeba DirectSound driver, ztráta dalších bufferů

- `IDirectSoundBuffer` je rozhraní pro objekt - paměťový blok pro přehrávání a mixování zvuku
- **Primární zvuková paměť** (buffer) je přehrávána wave zvukovým zařízením
- Do ní se mixují zvuky ze **sekundárních zvukových pamětí**
    - Na zvukové kartě / v RAM počítače
- DirectSound používá **kruhové buffery**
- Pokud se data v kruhovém bufferu nezmění, přehrávají se pořád dokola (např. v **Half Life 2**)
    - www.youtube.com/watch?v=4ddJ1OKV63Q (7:58)

DirectSound - Zvukové paměti (2)

- Jsou podporovány dvě různé verze bufferů
- **Static buffer**
  - Určený pro krátké a často se opakující zvuky.
  - Typické použití je ve hrách - střelba, běh výbuchy.
- **Streaming buffer**
  - Určený pro delší zvuky, které se většinou neopakují (lze po přehrání zahodit).
  - Typické použití je pro přehrávání hudby na pozadí běžící aplikace.

DirectSound - Funkce bufferů

- **CreateSoundBuffer**()
  - Vytvoří buffer
- **Lock**(), **Unlock**()
  - Pro nahrávání dat (Lock vrací pointer)
- **Play**(), **Stop**()
  - Začne / přestane přehrávat co je v bufferu
- **SetCurrentPosition**()
  - Nastaví přehrávání od dané pozice
- **SetNotificationPositions**()
  - Notifikace když přehrávání dosáhne dané pozice
  - Používá se např. jako upomínka „potřebuji data“

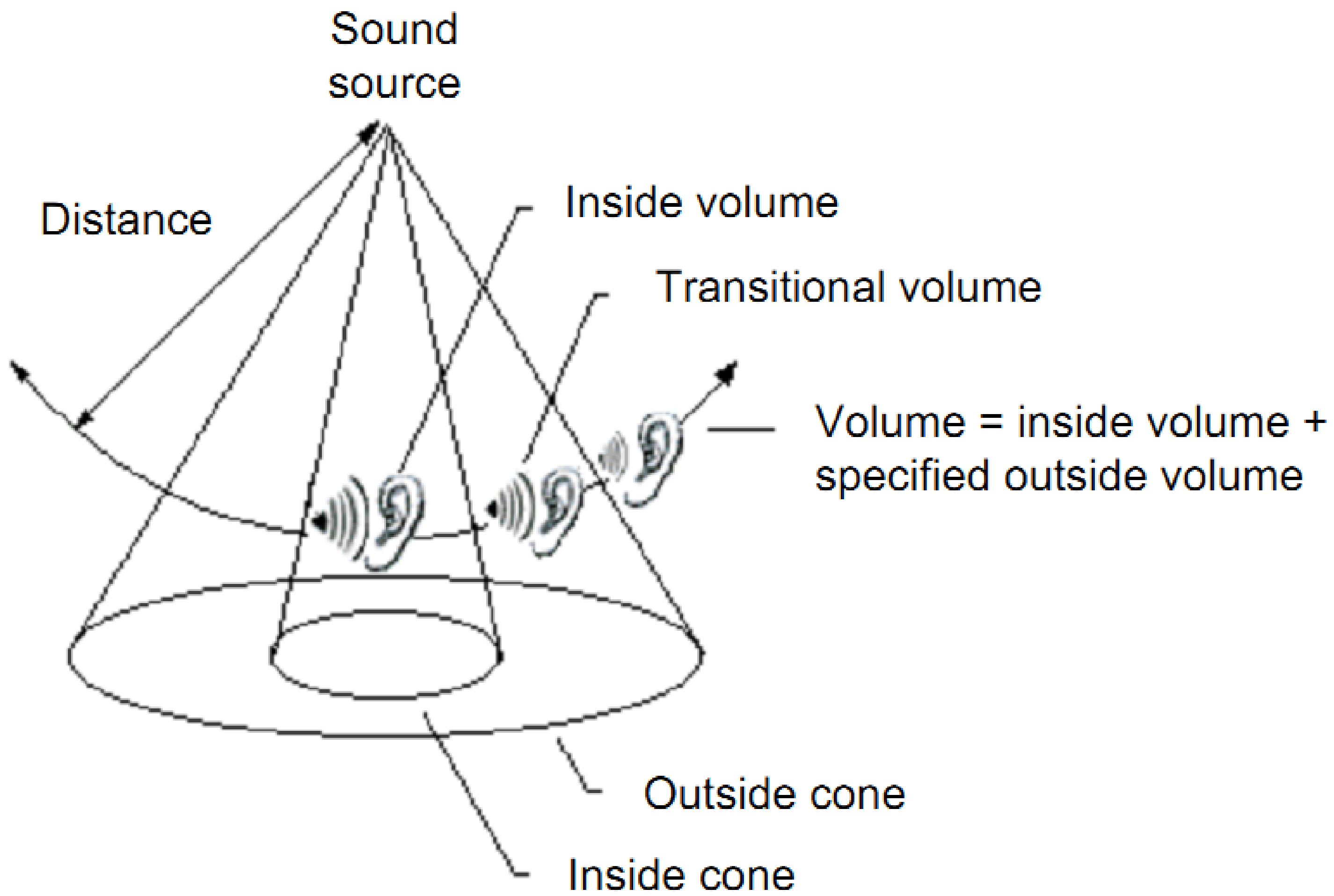
DirectSound - Funkce bufferů (2)

- **GetVolume**(), **SetVolume**()
  - Nastavení hlasitosti
- **GetPan**(), **SetPan**()
  - Nastavení balance levá / pravá strana (stereo)
- **GetFrequency**(), **SetFrequency**(), **GetFormat**(), **SetFormat**()
  - Nastavení formátu a vzorkovací frekvence

DirectSound3D

- Pomocí Direct**Sound**3D lze dosáhnout efektů prostorového zvuku
  - Ve sluchátkách
  - Ve dvou (a více) reproduktorech
- Např. ve hrách - zvuk výbuchu je **mono**
  - **DirectSound3D** nastavuje:
    - **Hlasitost** ~ vzdálenost
    - ***Pan*** ~ pozice (pravé/levé ucho)
    - **Zpoždění doletu** ~ pozice (pravé/levé ucho)
    - **Tlumení** (***muffling***) ~ pozice (vpředu/vzadu)
  - Další efekty jako ozvěna podle přednastavených prostředí (např.: „kamenný koridor“)
    - Také **EAX** (Environmental Audio Extensions)

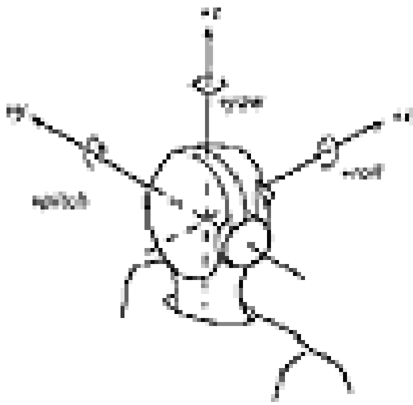
- Zvukový trychtýř: některé zdroje zvuku mohou být směrové



- Poloha a orientace posluchače je určena objektem `IDirectSound3DListener`
- Pokud má posluchač vůči zvukovému zdroji nenulovou relativní rychlost, DirectSound3D automaticky vytvoří Dopplerův jev

yaw

pitch

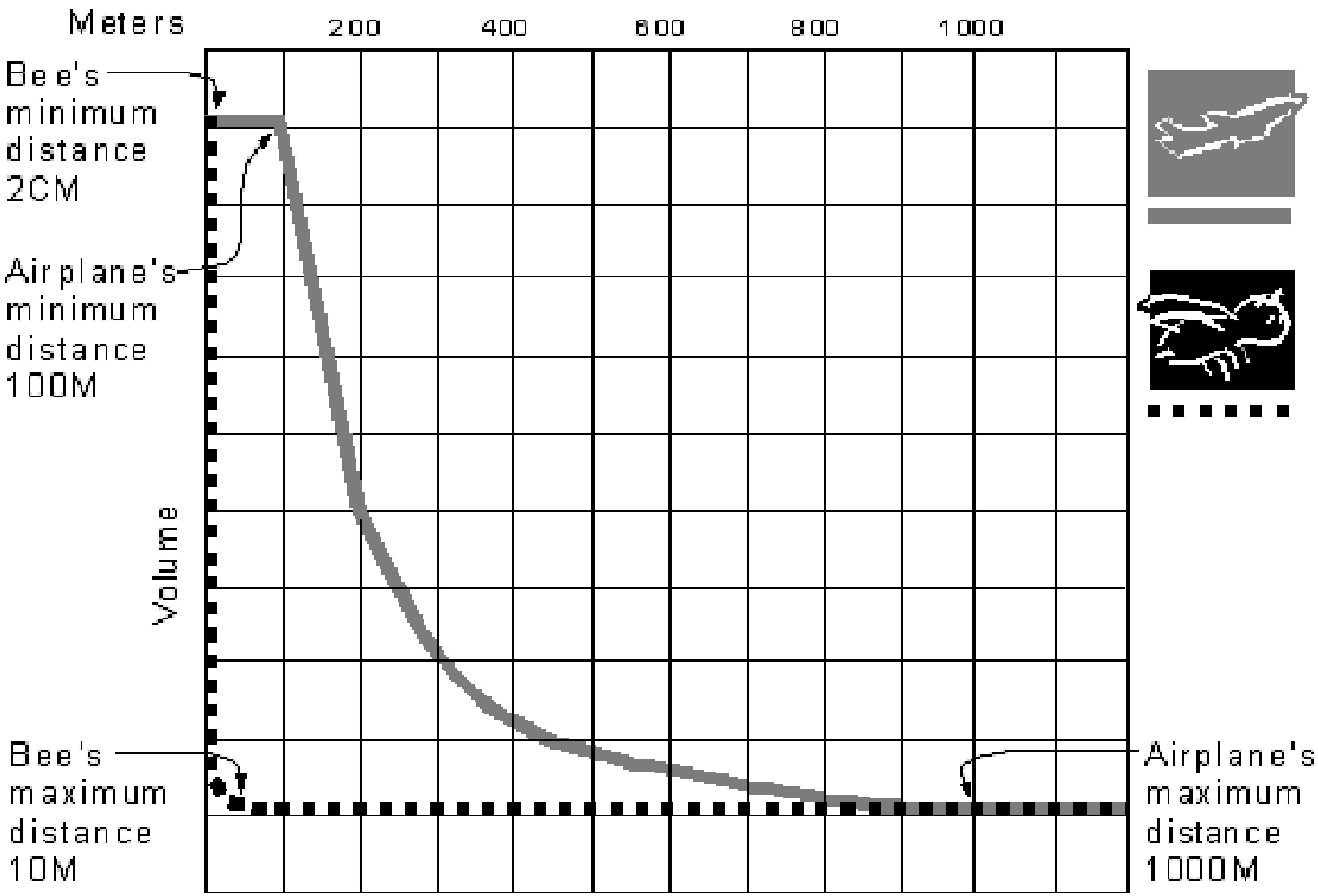


roll

yaw

roll

- Představte si, že máte vosu v uchu: to je zvukově srovnatelné se startujícím letadlem 100 m daleko



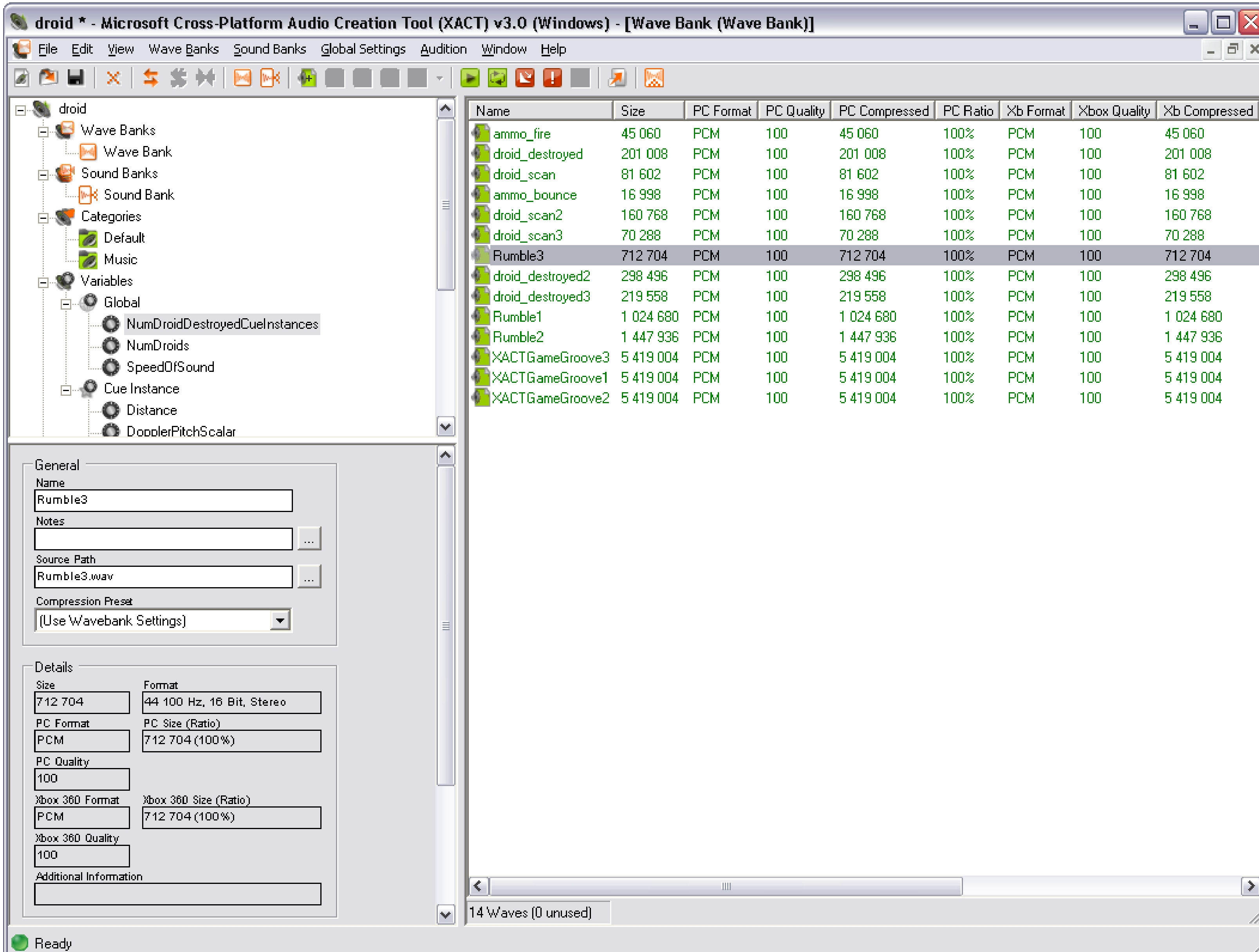
- Podpora pro **MIDI**
- Přehrávání hudby pomocí hardwarového nebo softwarového syntetizéru
- Podpora nástrojů v DLS standardu
  - **D**own**L**oadable **S**ounds
- Obsahuje hudební nástroje Roland Sound Set

- Dnes mrtvé, nahrazené kvalitnější komprimovanou hudbou např. v mp3

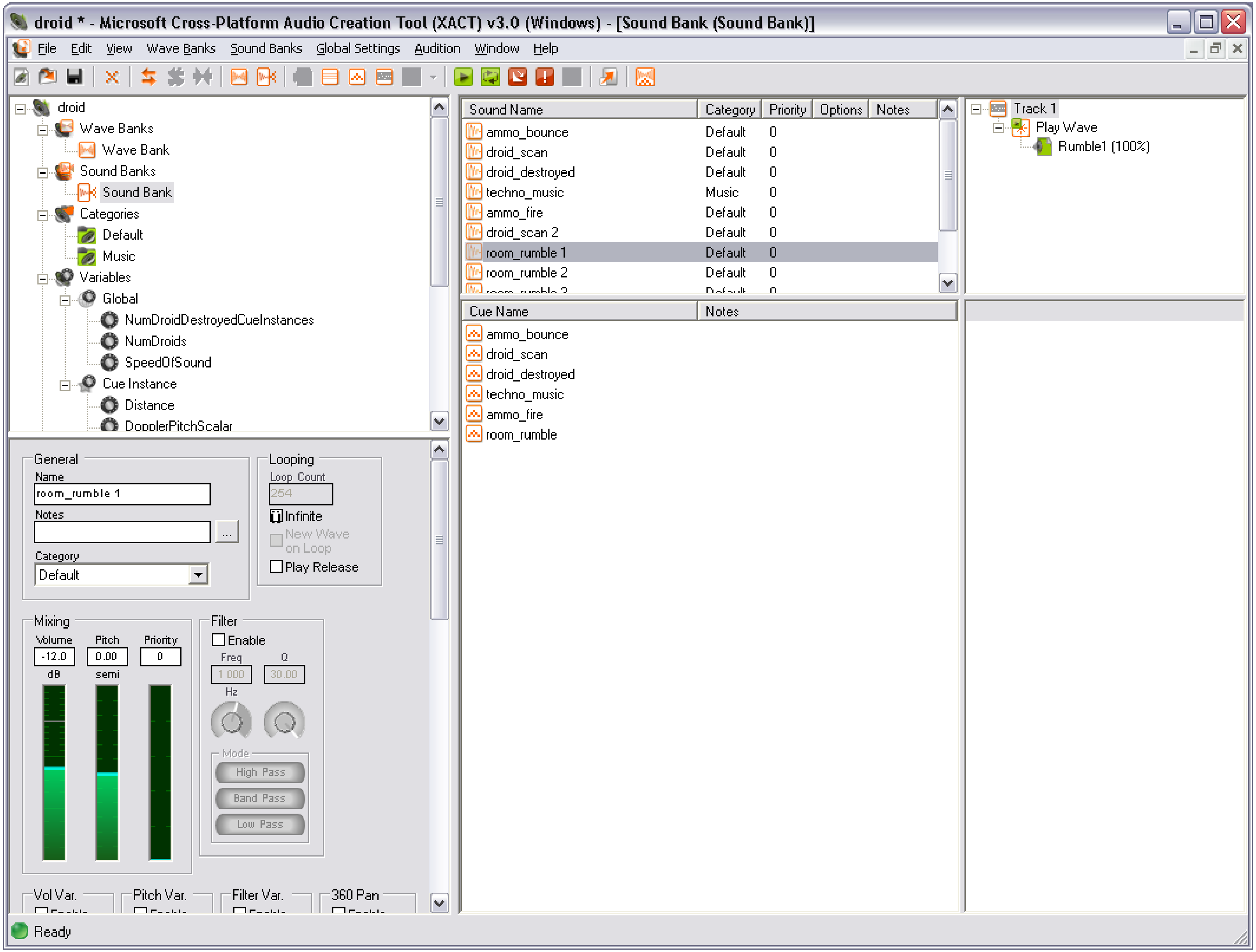
XACT - Cross-platform Audio Creation Tool

- Součást **DirectX** (od verze **10**), **XNA**
- Také port v **Mono** (Android, iOS, Mac, Linux, ...)
- Má dvě části
- Nástroj pro vytváření zvuků (**sound creation**)
  - Podporuje WAV, AIFF, XMA (= WMA)
  - Stereo a 5.1
  - Lze zabalit více zvuků do zvukové banky (XWB)
  - Lze jednoduše „**naklikat“ efekty** (XSB)
- API pro přehrávání zvuků, včetně efektů
  - Není potřeba zdlouhavě nastavovat za běhu
  - Odpadá složité ošetřování zvukových bufferů

XACT - Wave Bank (data zvuků)



XACT - Sound Bank (nastavení a efekty zvuků)



IXACT3Engine *pEngine;

IXACT3WaveBank *pWaveBank;

IXACT3SoundBank *pSoundBank;

XACTINDEX iSound;

XACT3CreateEngine(0, &pEngine);

pEngine->**Initialize**(NULL);

pEngine->**CreateInMemoryWaveBank**(data, size, 0, 0, & pWaveBank);

pEngine->**CreateSoundBank**(data, size, 0, 0, &pSoundBank);

iSound = pSoundBank->**GetCueIndex**("name");

XACT - Přehrávání

- Základní přehrávání zvuku (nebo hudby)

  pSoundBank->**Play**(iSound, 0, 0, NULL);

- 3D zvuk

  XACTGAME_3DCUE cue3D;

  cue3D. bActive = true;

  pSoundBank->**Prepare**(iSound, 0, 0, &cue3D);

  Cue3D.vEmitterPosition = **(x, y, z)**;

  XACT3DCalculate(**x3DInstance**, &**listener**, &**emitter**, &**dspSettings**);

  XACT3DApply(&**dspSettings**, &Cue3D.pCue);

  Cue3D.pCue->**Play**();

Vyzkoušejte si sami

- Stáhněte si **DirectX Software Development Kit**
- Letos na:
  http://www.microsoft.com/en-us/download/details.aspx?id=6812
- Po instalaci máte k disposici složky
  - DXSDK/Samples/C++/XACT/
  - DXSDK/Samples/C++/XAudio2/

Multimédia
prezentace přednášek
Zvuk v Linuxu
Multiplatformní rozhraní

Lukáš Polok

Vysoké učení technické v Brně, Fakulta informačních technologií
Božetěchova 2, 612 66 Brno
ipolok@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

28.03.2013

- OSS - Open Sound System
- Funguje pod Linuxem / Unixem
  - Hlavně na FreeBSD

- Původně free software
- Potom proprietární a pak zase free
  - Některé distribuce OSS opustily

OSS - Programování

- Jednoduché “C“ rozhraní

  `#include <soundcard.h>`

- Podporovaná zařízení

  `/dev/mixer` – nastavení hlasitosti, ne vždy k dispozici

  `/dev/sndstat` – diagnostika, není k prog. Zpracování

  `/dev/dsp`, `/dev/audio` – primární vstup / výstup

  `/dev/sequencer`, `/dev/music` – MIDI aplikace

  `/dev/dmfm` – low-level přístup k FM syntéze

  `/dev/midi`, `/dev/dmmidi` – low-level přístup k MIDI

OSS - Programování

```
#include <ioctl.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/soundcard.h>

int audio_fd;
if((audio_fd = open(“/dev/dspW“, O_WRONLY, 0)) == -1) {
    perror(DEVICE_NAME);
    exit(1);
}
// /dev/dsp/ je 8-bit, /dev/dspW/ je 16-bit, /dev/audio/ je µ-Law

int vol = 80; // vždy používat dočasnou proměnnou!
vol = (vol * 255) / 100; // z 0 – 100 do 0 – 255
vol = vol| (vol << 8); // pro levý a pravý kanál
ioctl(mixer_fd, SOUND_MIXER_WRITE(SOUND_MIXER_VOLUME), &vol);

Close(audio_fd);
// po skončení
```

OSS – Nastavení Přehrávání

```
int format, channels, srate; // vždy používat dočasnou proměnnou!
format = AFMT_S16_LE; // znaménkový, 16-bit, little endian
if(ioctl(audio_fd, SNDCTL_DSP_SETFMT, &format) == -1) {
    perror("SNDCTL_DSP_SETFMT"); exit(-1);
}
if(format != AFMT_S16_LE)
    ; // formát byl odmítnut, ale byl navržen alternativní
channels = 2; // stereo (default je mono)
if(ioctl(audio_fd, SNDCTL_DSP_CHANNELS, &channels) == -1) {
    perror("SNDCTL_DSP_CHANNELS"); exit(-1);
}
if(channels != 2)
    ; // stereo neumí, ale umí tolik kanálů, co je teď v channels
srate = 44100; // standardních 44.1 kHz
if(ioctl(audio_fd, SNDCTL_DSP_SPEED, & srate) == -1) {
    perror("SNDCTL_DSP_SPEED"); exit(-1);
}
if(srate != 44100)
    ; // tohle to neumí, ale přehraje nám to pomaleji nebo rychleji
```

OSS - Přehrávání

```
short left_buffer[1024], right_buffer[1024];
// data „logicky“ rozdělená

unsigned char dev_buffer[4096];
// data pro přehrávání HW

unsigned char *p_dest = dev_buffer;
for(int i = 0; i < 1024; ++ i, p_dest += 4) {
    short L = left_buffer[i], R = right_buffer[i];
    p_dest[0] = L & 0xff; // nižší byte
    p_dest[1] = L >> 8; // vyšší byte
    p_dest[2] = R & 0xff;
    p_dest[3] = R >> 8; // levý a pak pravý vzorek
}
// konvertuje vzorky do podoby jak se dají přehrát

int len;
if((len = write(audio_fd, dev_buffer, sizeof(dev_buffer))) == -1)
    ; // chyba
```

OSS - Synchronizace

- Nahrávání zvuků stejné jako přehrávání, jen místo `write()` je `read()`
- Funkce `write()` a `read()` jsou blokující
  - Nepoužitelné s GUI
  - Lze použít `select()` a `poll()`
- Nebo použít více vláken
  - Jedno vlákno jen ve smyčce čte / zapisuje a čeká
  - Druhé vlákno zpracovává (třeba dekóduje mp3)

- Advanced Linux Sound Architecture
  - Není na FreeBSD
- Nástupce OSS
  - OSS se dneska v Linuxech emuluje ALSA-ou
- Free, GPL licence

- `#include <alsa/asoundlib.h>`

ALSA – Nastavení zařízení

```c
#include <alsa/asoundlib.h>

snd_pcm_t *handle;
if(snd_pcm_open(&handle, "default", SND_PCM_STREAM_PLAYBACK, 0) < 0)
    exit(-1); // chyba
snd_pcm_hw_params_t *params;

snd_pcm_hw_params_alloca(&params); // parametry přehrávání
snd_pcm_hw_params_any(handle, params); // načte výchozí nastavení
snd_pcm_hw_params_set_access(handle, params,
    SND_PCM_ACCESS_RW_INTERLEAVED); // interleaved: L R L R ...
snd_pcm_hw_params_set_format(handle, params, SND_PCM_FORMAT_S16_LE);
snd_pcm_hw_params_set_channels(handle, params, 2); // stereo
int val = 44100, dir;
snd_pcm_hw_params_set_rate_near(handle, params, &val, &dir); // 44.1k
if(snd_pcm_hw_params(handle, params) < 0)
    exit(-1); // chyba

snd_pcm_close(handle);
```

ALSA - Přehrávání

```
snd_pcm_uframes_t frames = 32; int t;
snd_pcm_hw_params_set_period_size_near(handle, params, &frames, &t);
if(snd_pcm_hw_params(handle, params) < 0)
    exit(-1);
// nastaví velikost přehrávacího bufferu (nepovinné)

snd_pcm_hw_params_get_period_size(params, &frames, &t);
// zjistií použitou velikost přehrávacího bufferu

int size = frames * 4; // 2 B / sample, stereo
unsigned char *dev_buffer = malloc(size);
for(int n; !finished_playback();) { // přehrávací smyčka
    decode_audio_data(dev_buffer, size); // naplnit buffer daty
    if((n = snd_pcm_writei(handle, dev_buffer, frames)) == -EPIPE)
        snd_pcm_prepare(handle); // „cvak“ ...
    else if(n < 0)
        exit(-1); // chyba
    else if(n != (int)frames)
        ; // nepodařilo se zapsat celý buffer naráz
}
snd_pcm_drain(handle); // přehrát co je ješte nabufferované
free(device_buffer);
```

Ostatní Linuxové rozhraní

- Pulse audio
  - POSIX-ové distribuce, GNU GPL
  - sources, sinks
  - Podporuje hodně aplikací
  - Umí konvertovat mezi zvukovými formáty

- JACK
  - Profesionální sound server
  - Nízká latence
  - GNU LGPL

OpenAL

- Na Windows, Linux, BSD, Android, Apple/Mac
- Vyvíjené Creative Technology
- Podobné OpenGL

OpenAL - Inicializace

```
ALCdevice *p_device = alcOpenDevice(NULL);
if(!p_device)
    exit(-1);
ALCcontext *p_context = alcCreateContext(p_device, NULL);
if(!p_context)
    exit(-1);
alcMakeContextCurrent(p_context);
if(alGetError() != AL_NO_ERROR)
    exit(-1);
// vyrobí device a kontext, zapne kontext jako aktuální

// hraj zvuky

alcMakeContextCurrent(NULL);
alcDestroyContext(p_context);
alcCloseDevice(p_device);
// odpojí kontext a zruší kontext i device
```

OpenAL – Zdroje zvuku a buffery

```
ALuint buffer, source; // Aluint je trochu jako GLuint
alGenSources(1, &source);
// vyrobit jeden source

alGenBuffers(1, &buffer);
// vyrobit jeden buffer

const int samples = 1024;
const int size = samples * 2 * sizeof(short);
short buff_data[1024 * 2]; // je to stereo -> * 2

decode_audio_data(buff_data[i], size); // naplnit buffer daty

alBufferData(buffer, AL_FORMAT_STEREO16, buff_data, size, 44100);
// poslat data přes OpenAL zvukové kartě

alSourceQueueBuffers(source, 1, &buffer); // zařadit buffer do fronty

alSourcePlay(source); // začít přehrávat
```

```
ALuint buffer[2], source;
alGenSources(1, &source);
alGenBuffers(2, buffer);
int samples = 1024, size = samples * 2 * sizeof(short), state, proc;
short buff_data[1024 * 2];
for(int i = 0, n = 0; !finished_playback(); i = 1 - i) {
    alGetSourcei(source, AL_BUFFERS_PROCESSED, &proc); // přehrané b.
    n -= proc; // aktualizuje počet bufferů
    if(n < 2) { // jsou ve frontě méně než 2 plné buffery?
        decode_audio_data(buff_data[i], size); // naplnit buffer daty
        if(proc) // byl právě vyprázdněn buffer?
            alSourceUnqueueBuffers(source, 1, &i); // odpojit buffer
        alBufferData(buffer[i], AL_FORMAT_STEREO16,
            buff_data, size, 44100); // nahrát data
        alSourceQueueBuffers(source, 1, &buffer[i]); // zařadit do fr.
    } else Sleep(1); // přehrává se, nezbývá než čekat
    alGetSourcei(n_source, AL_SOURCE_STATE, &state);
    if(state != AL_PLAYING)
        alSourcePlay(source); // pokud nám to ještě nehraje
}
```

- Asynchronous Sound I/O
- Vyvíjené Steinberg
- Profesionální low-latency práce se zvukem
  - Přehrávání formátů jako 32-bit / 192 kHz
  - Latence jednotky milisekund
- Linux / Mac / Windows
- Potřebuje speciální HW driver
  - nebo emulace pomocí programu ASIO4ALL
  - v Linuxech lze emulace pomocí branche JACK

ASIO - Inicializace

```
#include <ASIO/asio.h>

extern AsioDrivers *asioDrivers;
bool loadAsioDriver(char *name);
// část driveru se přikompiluje přímo k aplikaci (fujky)

ASIODriverInfo driver;
if(!loadAsioDriver("M-Audio FireWire ASIO") || // nebo "ASIO4ALL"
   ASIOInit(&t_driver) != ASE_OK)
    exit(-1);

// tady si můžeme hrát

ASIOExit();
asioDrivers->removeCurrentDriver();
delete asioDrivers;
asioDrivers = 0; // !!
```

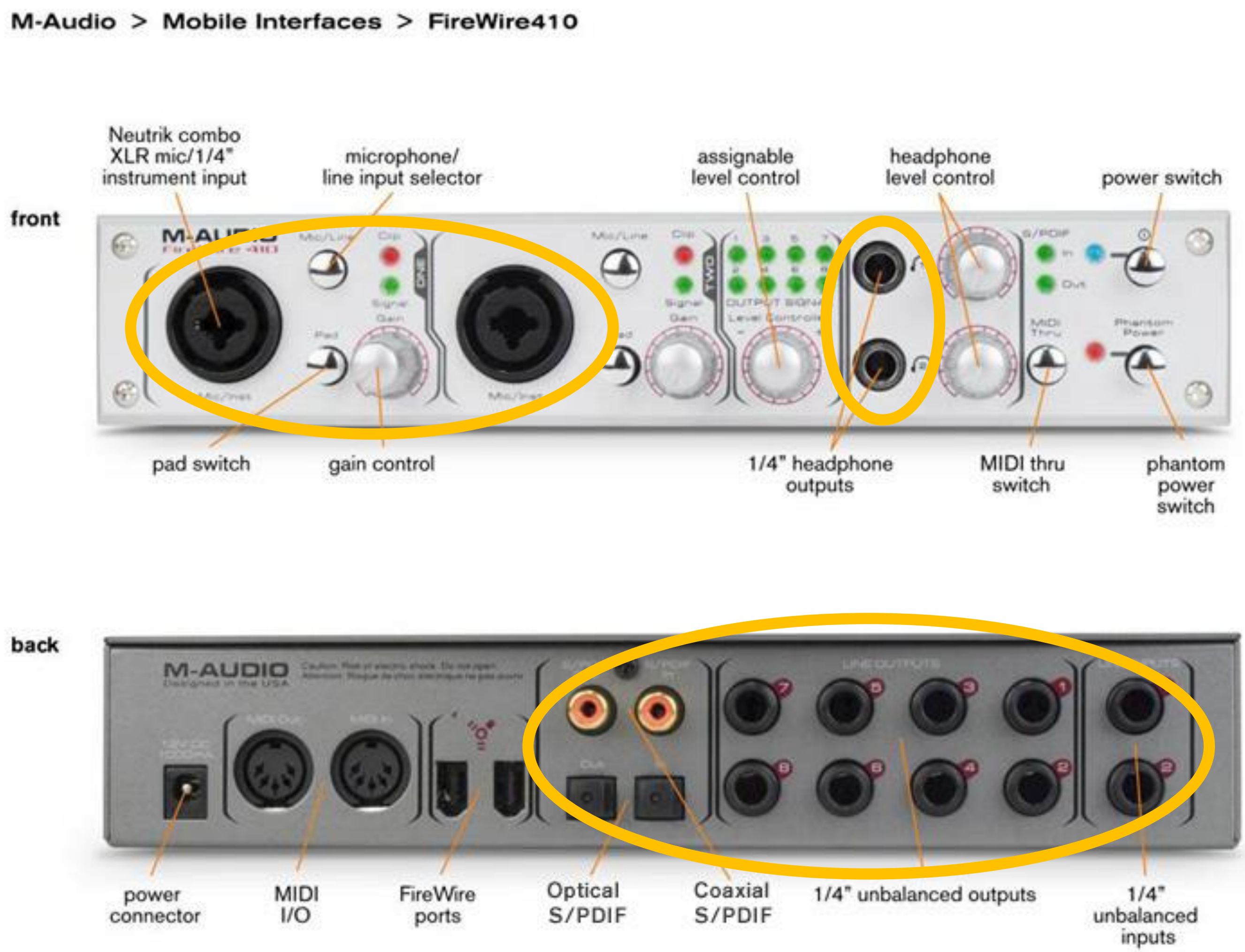
ASIO – Načtení informací o kanálech

```
int in_latency, out_latency;
ASIOGetLatencies(&in_latency, &out_latency);
// vstupní a výstupní latence ve vzorcích

int input_chan_num, output_chan_num;
ASIOGetChannels(&input_chan_num, &output_chan_num);
// počet vstupních a výstupních kanálů na kartě

int min_size, max_size, best_size, granularity;
ASIOGetBufferSize(&min_size, &max_size, best_size, &granularity);
// minimální, maximální, preferovaná velikost bufferů a krok

ASIOChannelInfo chan[32] = {0};
for(int i = 0; i < min(input_chan_num + output_chan_num, 32); ++ i) {
    chan[i].channel = i;
    chan[i].isInput = (i < input_chan_num)? true : false;
    ASIOGetChannelInfo(&chan[i]);
}
// načtení informace o všech kanálech
```


M-Audio > Mobile Interfaces > FireWire410
front
Neutrik combo XLR mic/1/4" instrument input
microphone/ line input selector
assignable level control
headphone level control
power switch
pad switch
gain control
1/4" headphone outputs
MIDI thru switch
phantom power switch
back
power connector
MIDI I/O
FireWire ports
Optical S/PDIF
Coaxial S/PDIF
1/4" unbalanced outputs
1/4" unbalanced inputs

ASIO – Nastavení přehrávání

```
ASIOTime *buffSwitchTimeInfo(ASIOTime *t, long index,
    ASIOBool direct_process);
int messageCallback(long msg, long value, void *message, double *opt)
{    return 0;    }
void buffSwitchCallback(long index, ASIOBool direct_process)
{    buffSwitchTimeInfo(NULL, index, direct_process);    }
void sRateChangeCallback(ASIOSampleRate rate)
{    /* .. */    }

if(ASIOSetSampleRate(44100) != ASE_OK) exit(-1); // 44.1 kHz
ASIOChannelInfo ochan[2]; // stereo
ochan[0] = chan[input_chan_num];
ochan[1] = chan[input_chan_num + 1]; // výstupní kanály za vstupními
ochan[0].type = ochan[1].type = ASIOSTInt32LSB; // 32-bit vzorky

ASIOCallbacks cb;
cb.asioMessage = &messageCallback; // ošetření různých zpráv
cb.bufferSwitch = &buffSwitchCallback; // ošetření výměny bufferů
cb.bufferSwitchTimeInfo = &buffSwitchTimeInfo;
cb.sampleRateDidChange = &sRateChangeCallback; // změna frekvence
ASIOCreateBuffers(ochan, 2, best_size, &cb); // inicializace ASIO
```

ASIO - Přehrávání

```
ASIOStart();

// ...

ASIOStop();

ASIOTime *buffSwitchTimeInfo(ASIOTime *t, long index,
    ASIOBool direct_process)
{
    for(int i = 0; i < 2; ++ i) {
        void *dest = ochan[i].buffers[index];
        switch(ochan[i].type) {
        case ASIOSTInt32LSB: // signed 32-bit na vzorek, lil‘ endian
            decode_data_for_channel((int*)dest, best_size, i);
            break; // data se generují zvlášť pro levý a pravý kanál
        }
    }
    return 0;
}
```

Konec! Otázky?