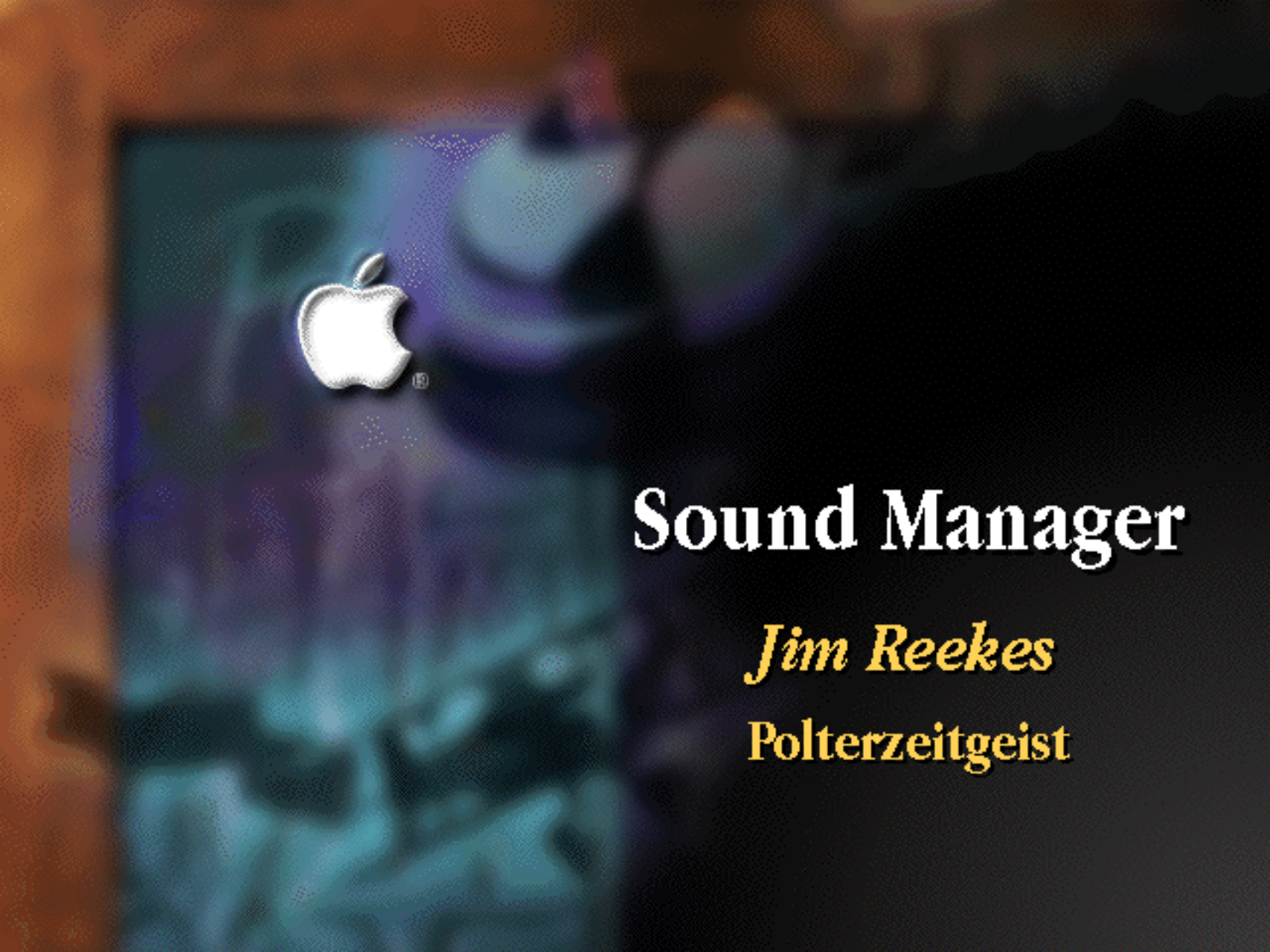

Worldwide
Developers
Conference


Sound Manager
Jim Reekes
Polterzeitgeist

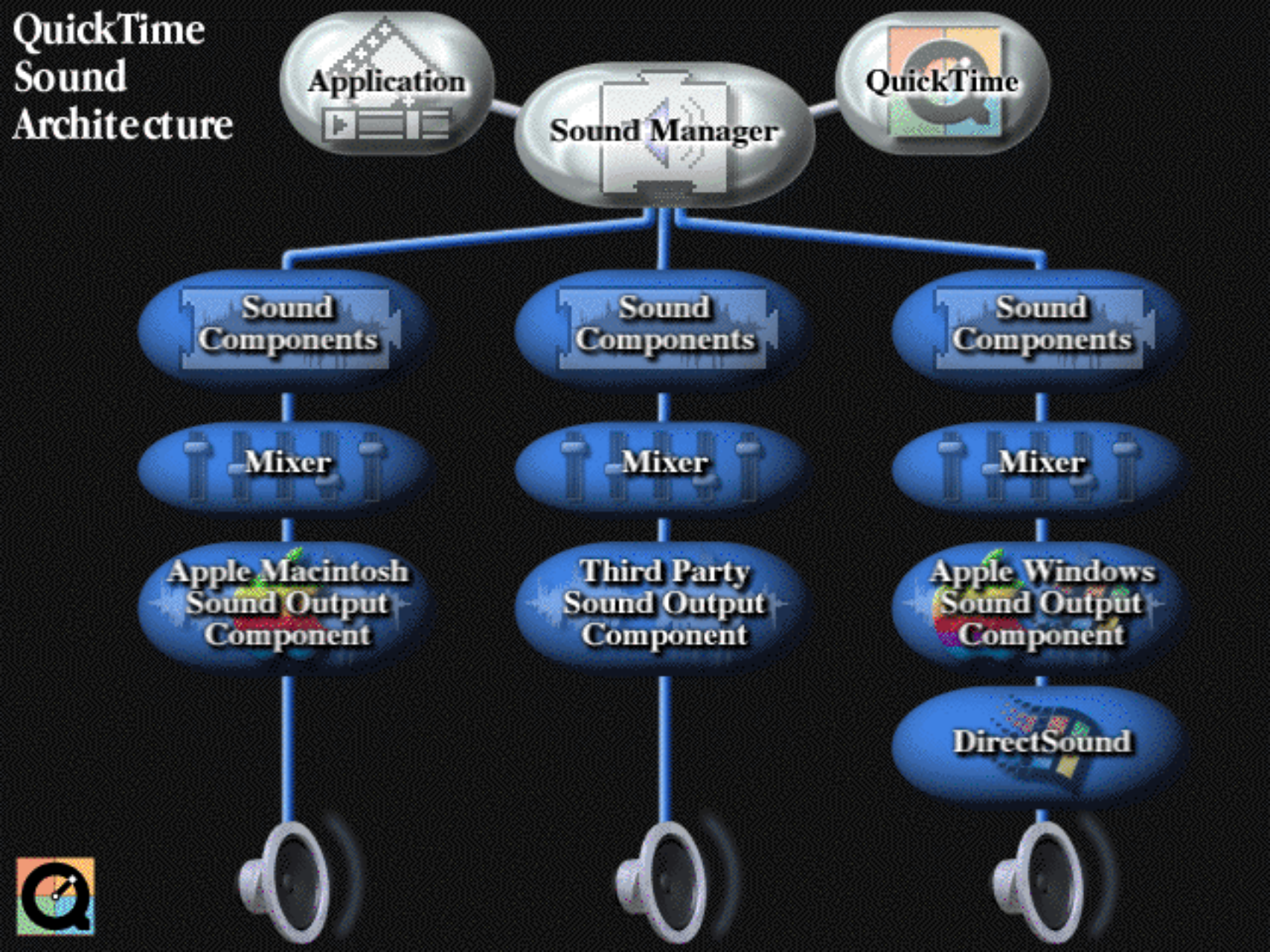
Multi–platform

- Macintosh, Rhapsody, Windows 95 and Windows NT
- Under development for Unix platforms
- Identical APIs, features, and extensibility
  - Supports both playback and capture
- Everything mentioned in this session applies to all platforms

Sound Manager Basics

- System software clients include
  - QuickTime movie sound tracks
  - QuickTime Software Synthesizer
  - Text to Speech Manager
  - SysBeep
- Sound mixing, volume and balance
- Hardware independent
- Data independent
- Integrated sample rate conversion
  - Up to 65kHz sample rates


Gratuitous Movie
Sound Manager
in Action


QuickTime
Sound
Architecture
Application
Sound Manager
QuickTime
Sound Components
Sound Components
Sound Components
Mixer
Mixer
Mixer
Apple Macintosh Sound Output Component
Third Party Sound Output Component
Apple Windows Sound Output Component
DirectSound

Data Format Conversion

- Use QuickTime's services for mixing and conversion within your application
  - Sample rate conversion, compression or decompression, stereo to mono, etc.
- Importing and exporting files
- Services available for both real time and off-line use
  - Off-line uses highest quality

Audio Formats

- Apple provided audio codecs (*new)
  - MACE 3:1
  - IMA
  - BIG/little Endian
  - Microsoft ADPCM*
  - DVI/Intel IMA*
  - MACE 6:1
  - μlaw
  - alaw*
  - DVC*
  - Single and double precision floating point*

Audio Formats

- Apple provided audio codecs (*new)
  - MACE 3:1
  - IMA
  - BIG/little Endian
  - Microsoft ADPCM*
  - DVI/Intel IMA*
  - MACE 6:1
  - μlaw
  - alaw*
  - DVC*
  - Single and double precision floating point*
- Extensible – add your own audio codecs

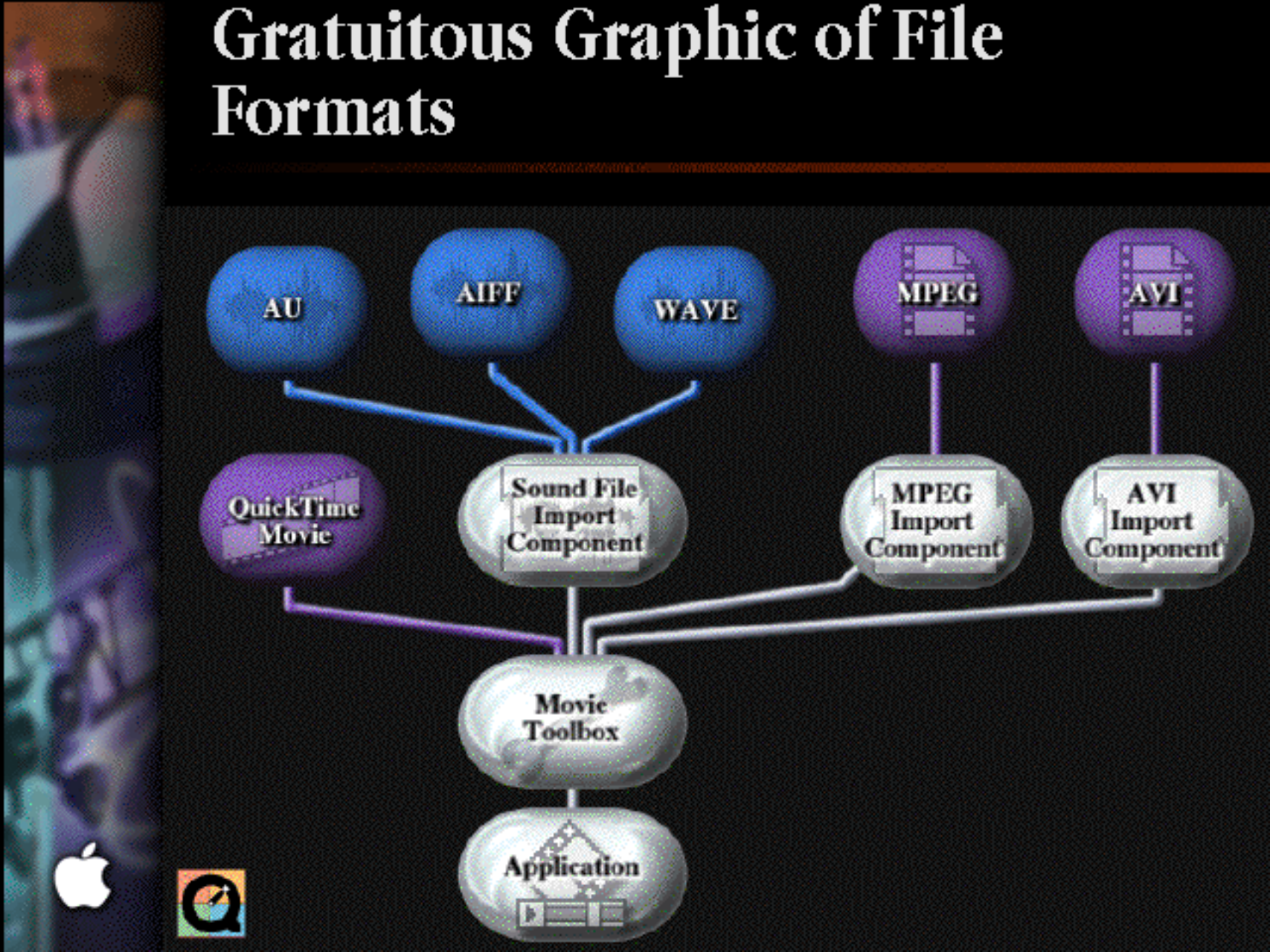
Supported File Formats

- Ability to play audio from a file is provide through QuickTime
- Supported file formats
  - System 7 Sound Files
  - QuickTime Movie
  - Sound Designer II
  - AU
  - AVI
  - OMF
  - AIFF
  - Audio CD tracks
  - WAVE
  - MPEG Layer I/II
  - DVC
- QuickTime rocks

Supported File Formats

- Ability to play audio from a file is provide through QuickTime
- Supported file formats
  - System 7 Sound Files
  - QuickTime Movie
  - Sound Designer II
  - AU
  - AVI
  - OMF
  - AIFF
  - Audio CD tracks
  - WAVE
  - MPEG Layer I/II
  - DVC
- QuickTime rocks
- Extensible – add your own file formats

Gratuitous Graphic of File Formats



Multiple sound output devices

- One default device, selected by user
- Multiple sound output destinations
- Each channel can use a different device
  - SndNewChannel (&sndChannel,
    kUseOptionalOutputDevice,
    (long)soundOutComponent,
    nil)
- Each track in a QuickTime movie can use a different device
  - MediaSetSoundOutputComponent()

Configuration Information

- Determine hardware configuration
  - Optimize performance
  - Understanding latency

```
err = SndGetInfo(chan,
siHardwareFormat, &hwFormat)

hwFormat.format;
hwFormat.numChannels;
hwFormat.sampleSize;
hwFormat.sampleRate;
hwFormat.sampleCount;
```

API changes

- Merged SoundComponents.h and SoundInput.h into Sound.h
- Audio Filters
  - Pre-mixer effects
- Sound Clock
  - Synchronization and scheduling

Audio Filters

- Pre-mixer effects
  - Example: Sound Sprocket 3D Localization
- Ability to add filters into playback chain
  - err = SndSetInfo(chan, siPreMixerSoundComponent, &scLink)
- Create your own
  - EQ, reverb, level metering, etc.

Sound Clock

- Counts samples sent to hardware
- Essential for precise synchronization with other media elements
- Used by QuickTime as default clock for movie playback
  - Available for use by any client
- Synchronizes multiple sound channels
- Schedules into future and past

ScheduledSoundHeader

```
struct ScheduledSoundHeader {
   TimeRecord           startTime;
   union {
      SoundHeader       stdHeader;
      CmpSoundHeader    cmpHeader;
      ExtSoundHeader    extHeader;
   } u;
};
```

Sound Clock Example

```
cmd.cmd = clockComponentCmd;
cmd.param2 = true;
err = SndDoImmediate(chan, &cmd);

cmd.cmd = getClockComponentCmd;
cmd.param2 = (long)&clock;
err = SndDoImmediate(chan, &cmd);

currentTime.base = nil;
err = ClockGetTime(clock, &currentTime);

// set start time and sound header
scheduledSound.startTime = x;
scheduledSound.u.stdHeader = y;
cmd.cmd = scheduledSoundCmd;
cmd.param2 = (long)&scheduledSound;
SndDoImmediate(chan, &cmd);
```

Macintosh Performance Numbers

- Average 23ms/2 latency
  - Macintosh sound hardware interrupt
- CPU Usage on PowerMac 9600/200
  - 16-bit stereo w/ rate conversion - 1.6%
  - 5 channels of 16-bit stereo IMA w/rate conversion - 9.1%

Macintosh Performance Numbers

- Average 23ms/2 latency
  - Macintosh sound hardware interrupt
- CPU Usage on PowerMac 9600/200
  - 16-bit stereo w/ rate conversion - 1.6%
  - 5 channels of 16-bit stereo IMA w/rate conversion - 9.1%
- Sound is cheap

Future Audio Issues

- Lower latency
  - Cut PCI PowerMacs in half
  - Adjustable buffer size
- High quality sample rate conversion
- Variable bit rate compression
- Surround/Dolby Digital Audio
  - Use of Sound Clock

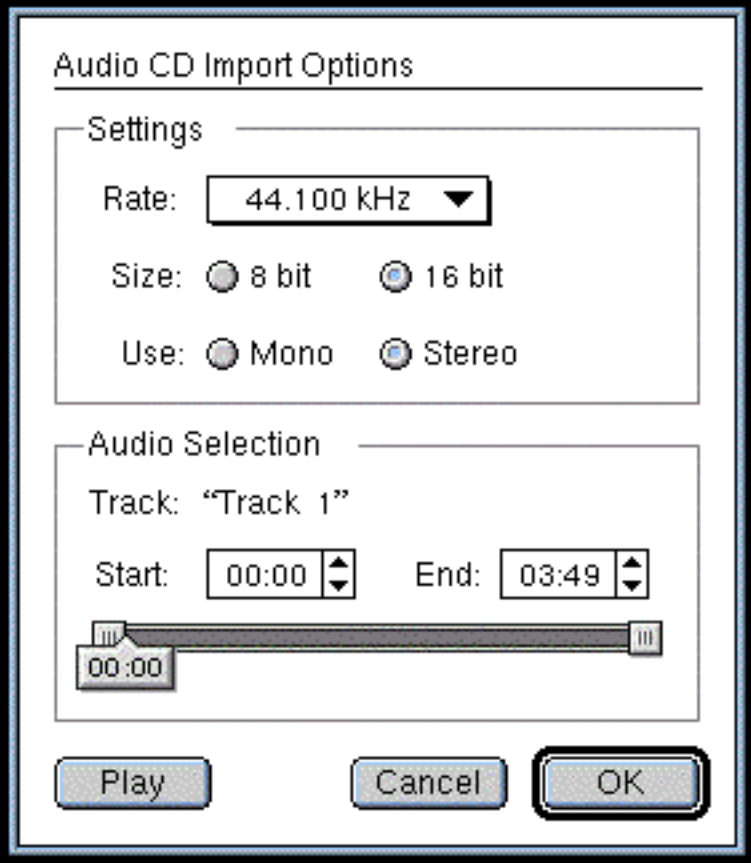
Web Page

- quicktime.apple.com/dev/devsnd.html
  - Inside Macintosh and addendum
  - Latest Interface and Libraries
  - Sample Code
  - develop articles
  - Anything else I find valuable

Demos

- Import and export audio
- Play audio file as movie
- Inserting and adding audio tracks
- Save audio as file
  - change format
  - set sample rate
  - set sample size
  - adjust track volume
  - adjust track balance
  - mix to stereo

Audio CD Import Dialog



Play Audio File

```
// can be AIFF, WAVE, AU, etc.
OpenMovieFile(     &aFile,
                   &movieFileRef,
                   fsRdPerm);

NewMovieFromFile( &theMovie,
                  movieFileRef,
                  nil,
                  nil,
                  newMovieActive,
                  nil);
StartMovie(theMovie);
```

Extract Audio in Memory

```
Handle sndHandle = NewHandle(0);

// set startTime and endTime

PutMovieIntoTypedHandle(
                        theMovie,
                        nil,
                        soundListRsrc,
                        sndHandle,
                        startTime,
                        endTime,
                        0, 0);
```

Inserting an Audio Track

```
// insert (paste) one audio track

// note: if target duration is non-zero,
// then that segment of the target movie
// is deleted before the paste

SetMovieSelection (sourceMovie,
                   sourceStart,
                   sourceDuration);
SetMovieSelection (targetMovie,
                   targetStart,
                   targetDuration);
PasteMovieSelection (targetMovie,
                     sourceMovie);
```

Adding an Audio Track

```
// overlay (add) one audio track

// note: if target duration is 0, then
// source movie's duration is used
// if target duration is not zero, then
// source movie is scaled to fit target

SetMovieSelection (sourceMovie,
                   sourceStart,
                   sourceDuration);
SetMovieSelection (targetMovie,
                   targetStart,
                   targetDuration);
AddMovieSelection (targetMovie,
                   sourceMovie);
```

Setting Volume and Balance

```
// volume range is 0 to 256

SetTrackVolume(aTrack, 256);


// set balance of QuickTime audio track
// balance range is -128 to 128

aMedia = GetTrackMedia(aTrack);
audioHandler = GetMediaHandler(aMedia);
MediaSetSoundBalance(audioHandler, 0);
```

Export Movie to Audio File

```
// do not display dialog
ConvertMovieToFile(theMovie, nil,
                   &outputFile,
                   'WAVE', 'TVOD',
                   -1, nil, 0, 0);

// display dialog to select audio format

ConvertMovieToFile(theMovie, nil,
                   &outputFile,
                   'AIFF', 'TVOD',
                   -1, nil,
                   showUserSettingsDialog,
                   0);
```


QTMA
3.0
David Van Brink
QuickTime Music Architect

QuickTime Music Architecture

The Shortest Summary Ever

- Part of QuickTime
- Plays musical notes
- Uses external General MIDI synthesizers, or
- Uses built-in software synthesizer
- Can be embedded in a QuickTime movie
- Can be imported from Standard MIDI Files

New Features for 3.0

Better, Stronger, Faster

- Abstraction of MIDI Layer
- Support for Mute & Solo
- Improvements to Software Synthesizer
  - Audio Quality
  - More Kinds Of Instruments

Abstraction of MIDI Layer

- New ‘midi’ Component Type
- Support for Existing MIDI Systems
  - MIDI Manager
  - Opcode’s OMS
  - Mark Of The Unicorn’s FreeMIDI
- Allows Support For Other MIDI’s
  - Serial Port PCI Cards
  - Windows & Other OS’s

Abstraction of MIDI Layer

```
pascal ComponentResult
QTMIDIGetMIDIPorts
    (QTMIDIComponent ci,
    QTMIDIPortListHandle *inputPorts,
    QTMIDIPortListHandle *outputPorts)

pascal ComponentResult
QTMIDIUseReceivePort
    (QTMIDIComponent ci,
        long portIndex,
        MusicMIDIReadHookUPP readHook,
        long refCon)
```

Abstraction of MIDI Layer

```
pascal ComponentResult
QTMIDIUseSendPort
   (QTMIDIComponent ci,
      long portIndex,
      long inUse)

pascal ComponentResult
QTMIDISendMIDI
   (QTMIDIComponent ci,
      long portIndex,
      MusicMIDIPacket *mp)
```

About MIDI

Q: What About The MIDI Manager?

- MIDI Manager Provided Three Main Functions
  - MIDI Output To Devices
  - Timing Services
  - Interapplication Routing

About MIDI

A: Just Use QTMA

- QTMA & QuickTime do most of what you want
  - QTMA plays notes in a clean, device independent fashion
    - NANewNoteChannel()
    - NAPlayNote()
  - QuickTime provides rich timing services
  - But: No Interapplication Communication

Support for Mute & Solo

While Playing A Movie

- New Calls In MusicMediaHandler
  - MusicMediaGetPartCount()
  - MusicMediaGetPartInstrument()
  - MusicMediaSetPartInstrument()
  - MusicMediaGetPartMix()
  - MusicMediaSetPartMix()

Support for Mute & Solo

While Playing Sequence Data

- New Calls In The TunePlayer
  - TuneGetPartMix()
  - TuneSetPartMix()

Software Synthesizer Audio

Sounds Better

- Improved Audio On 68k Macs
  - Low pass filter helps alleviate sampling distortion
- Improved Audio on PPC and Up
  - Better feature choices for best sound quality
    - Linear Interpolation
    - More Voices
    - Simple Filter On Each Voice

Instruments

New Knobs

- Two General Purpose Envelope Generators
  - Pitch
  - Filter Cutoff
- All stages of all envelopes may specify linear or logarithmic shape

Instruments

Standard Libraries

- Support for converting new “DLS” format instruments into QuickTime “atomic” instruments
- Support for Roland GS extended instrument library
  - GS MIDI files can use GS instruments via bank select


Demo

Importing A MIDI File

```
StandardGetFile();
```

Importing A MIDI File #2

```
Movie NewRAMMusicMovie(Handle midiFileH)
    {
    Handle ramAlias;
    Movie mo;
    ComponentResult err

    ramAlias = 0;
    mo = NewMovie(newMovieActive);
    if(!mo)
        goto goHome;
    ramAlias = NewHandleClear(4);
    err = SetMovieDefaultDataRef(mo, ramAlias,
'hndl');
    DisposeHandle(ramAlias);
    err = PasteHandleIntoMovie(fileHandle,
'Midi', mo, 0, 0);
    return mo;
    }
```

Part List

```
count = MusicMediaGetPartCount
   (mediaHandler,1);
for (i=1; i <= count; i++)
   {
   AtomicInstrument ai;
   err = MusicMediaGetPartInstrument
      (mediaHandler,1,i,&ai,1);

   AddInstrumentToList(ai,...);
   }
```

Part Solo

```
for(i = 1; i <= partCount)
 {
 mute = shouldWeMute(i);
 err = MusicMediaSetPartMix
  (mediaHandler,1,i,0x00010000,0,
    mute ? kTuneMixMute : 0);
  }
```

Pick Instrument

```
err = NAPickEditInstrument
  (noteAllocator, nil, prompt,
       (long)0, (NoteChannel)0,
       ai,kPickEditAllowPick);

if(err != userCanceledErr)
  err =
MusicMediaSetPartInstrument
    (mediaHandler,1,j,ai);
```

Future Directions

- Computers are *still* getting faster!
  Continue to use lots of cycles
  - More voice expression
  - More realtime audio effects
- Keep pace with music standards
  - Other music file formats
  - Instrument formats

Future Directions

- Cross Platform
  - QTMA -- Like the rest of QuickTime -- is a flexible framework
  - Use native hardware and services on any platform
  - Provide a consistent API for music features on many platforms


Q&A