VK Computer Games

Mathias Lux & Horst Pichler
Universität Klagenfurt

Organisatorisches

- Projektplanung,
  - Legen Sie eine Seite im Wiki an!
  - Deadline ist der 18. Mai ‘08
  - Wer bis dahin nicht als Entwickler genannt ist hat kein Projekt registriert.

Projektbeschreibung

Sample-Project

Ein Re-Make von Spy vs. Spy in einer Netzwerk-Multiplayer-Variante. George Bush jun. (White Spy) versucht Osama Bin Laden (Black Spy) zu überlisten und umgekehrt. Das Spiel läuft in Räumen ab, die von beiden betreten und "verändert" werden können. Die Veränderungen sind Fallen und Selbstschussanlagen, die das Ziel haben den anderen Spy auszuschalten. Betreten beide zur gleichen Zeit denselben Raum können sie sich einen Fauskampf liefern.

Plattform: Java 1.6
Typ: 2D Side Scroller, Action Adventure
Entwickler: Mathias Lux, Max Mustermann & Otto Normalverbraucher

Screenshots: 1, 2, ...

Upload einer Beta-Version, etc.

Link zur Herkunftsseite:
Projekte

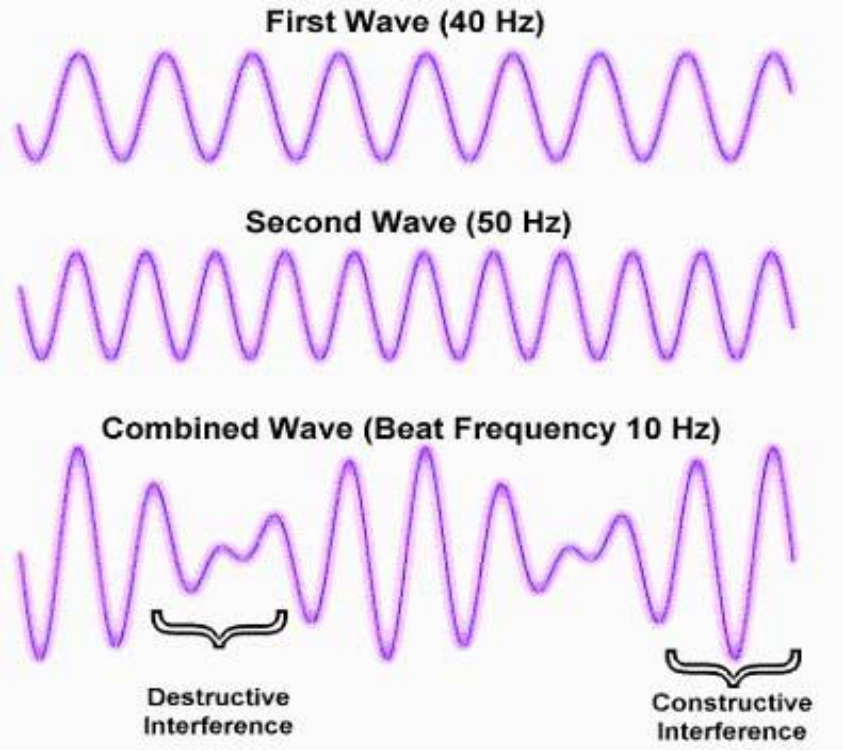
Agenda :: Sound

- Introduction
- Playing sounds
- Java Sound API
- Using external SPIs
- Sound analysis –> DSP

What is sound?

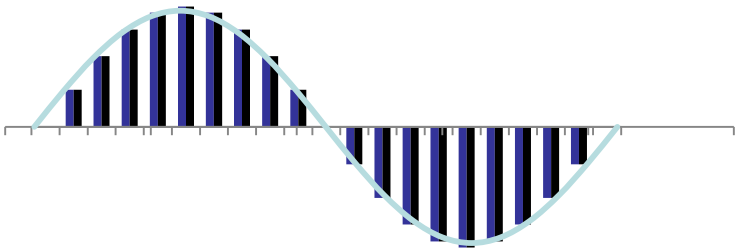
What is sound?

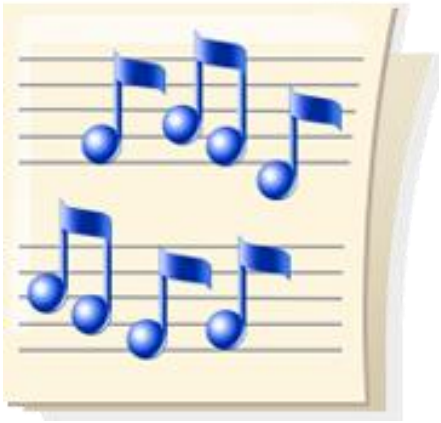
- Multiple sounds at the same time?



What is digital sound?

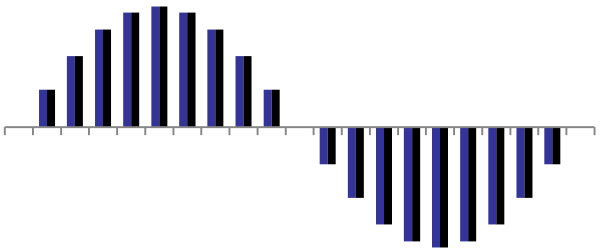
- A digitization of the wave.
  - Either a recipe for reconstruction
  - Or a discrete approximation





Sampled sound

- Wave gets sampled x times a second
  - E.g. 48.000 times -> 48 kHz sampling rate
- Obtained values are stored
  - E.g. 256, 240, 13, -7, -12, -44, ...
  - Quantization to e.g. 2^8 levels -> 8 Bit
- Possibly from different sensors
  - Stereo -> 2 channels

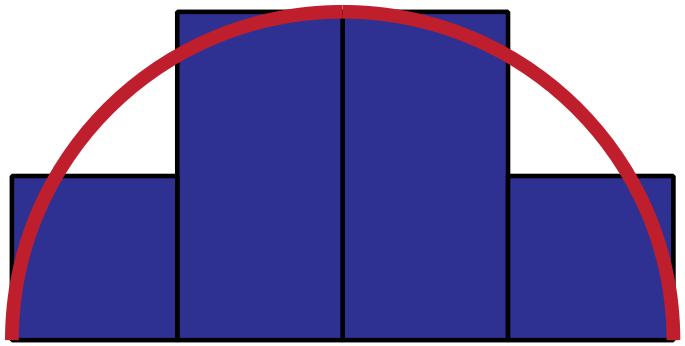


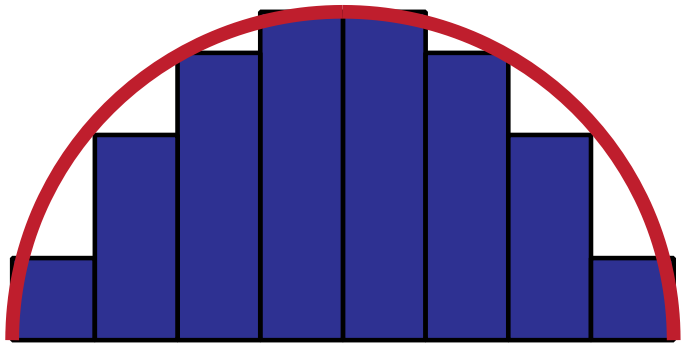
Sampled sound

- Example: 8 kHz, 16 bit Stereo
  - Sound wave is sampled 8.000 times a second
  - Samples are stored in 16 bit numbers
- That's *Pulse Code Modulation* (PCM)
  - Often used in WAV files …
  - Also as input from microphone or line in

Sampling Rates

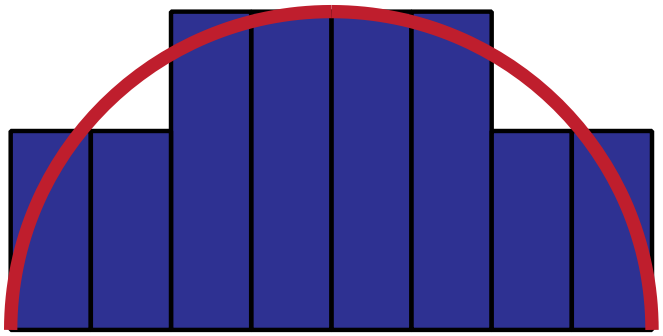
- With sampling rate x we can approximate frequencies up to x/2
- Assume frequency 1
  - sampling rate of 1 -> “0”
  - sampling rate of 2 -> “1,-1”

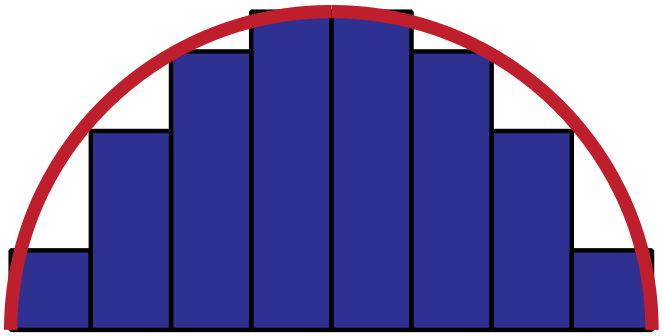




Quantization

- Reduces the possible values of the samples to a certain value
  - 8 Bit -> 256 levels, etc.





What do we want to capture?

- Humans can hear
  - From around 16 – 21 Hz
  - To around 16 kHz – 19kHz
  - 16 bit is enough (CD), 32 bit even better

Sound Formats

- Waveform Audio Format
  - Container for several compression formats
  - Includes PCM, MP3, GSM, μ-Law
- Musical Instrument Digital Interface
  - Control codes for instruments
  - Instruments can be “emulated”
- Compressed Audio Formats
  - MP3, OGG, AAC, …

Agenda :: Sound

- Introduction
- **Playing sounds**
- Java Sound API
- Using external SPIs
- Sound analysis –> DSP

Playing Sound in Java Games

- Using `java.applet.AudioClip`
  - Create an AudioClip per sound file
    - Supports Midi, Wav, ...
    - No MP3!
  - Can play multiple sound files at the same time
  - Very easy to use

Handling clips ...


```
// create a new AudioClip:
clip = Applet.newAudioClip(
  getClass().getResource("sound.mid"));

// play, stop, loop
clip.play();
clip.stop();
clip.loop();
```

Adding Sound to the game: A Walkthrough

- Create a SoundLib.java
  - Preloads music and effects in constructor
  - Provides methods for playing sound
- Add SoundLib class as member to game
  - Create in preloading routine
  - Play sounds on events
    - Shot rocket, Explosion

Demo ...

- Code & Game ...

Agenda :: Sound

- Introduction
- Playing sounds
- **Java Sound API**
- Using external SPIs
- Sound analysis –> DSP

Advanced Sound API: Why should I use this?

- Control
  - Where I/O happens, e.g. multiple devices
  - Volume, e.g. multiple clips + background
- Insert noise
  - Simulation of noisy channels
- Insert reverb
  - Simulation of wide chambers
- Distort and warp

Advanced Sound API: Package `javax.sound`

- Mixers
  - Audio Devices (HW & SW)
- Lines
  - I/O like speakers, mic (cp. Mixer)
- AudioFormat
  - Encoding, channels, etc.
- AudioSystem
  - Entry point ...

Advanced Sound API: Output ...

- Using Clips
  - Decoded portions of sound stored in memory

```
shot = AudioSystem.getClip();
shot.open(AudioSystem.getAudioInputStream(
   getClass().getResource("sfx/phase_array.wav")));
```

Advanced Sound API: Playing Clips


```
if (shot.isActive()) {
   shot.stop();
   xplode.stop();
}
if (!xplode.isActive()) {
   xplode.setFramePosition(1);
   xplode.start();
}
```

Check if running ...

Set to start

Agenda :: Sound

- Introduction
- Playing sounds
- Java Sound API
- **Using external SPIs**
- Sound analysis –> DSP

Compressed Sound

- Typically OGG Vorbis coded sound is used
  - Compression format and container are “free”
  - Superior to Midi
    - Sounds the same everywhere
- Use Vorbis SPI to play OGG files
  - Adds OGG decoder to Java Sound API

Advanced Sound API: Using MP3 / Ogg

- Sound API allows for external Service Providers
  - Provide decoding for various formats
  - MP3 SPI:
    - http://www.javazoom.net/mp3spi/mp3spi.html
  - Vorbis SPI:
    - http://www.javazoom.net/vorbisspi/vorbisspi.html

Loading an MP3 / OGG

- Put SPI in your classpath
- Everything else remains the same
  - Except ...

```
bgMusic = AudioSystem.getClip();
AudioInputStream in = AudioSystem.getAudioInputStream(
    getClass().getResource("sfx/rool-oh_why.mp3"));
AudioFormat inFormat = in.getFormat();
AudioFormat decodedFormat = new AudioFormat(
        AudioFormat.Encoding.PCM_SIGNED,
        inFormat.getSampleRate(), 16, inFormat.getChannels(),
        inFormat.getChannels() * 2, inFormat.getSampleRate(),
        false);
bgMusic.open(AudioSystem.getAudioInputStream(decodedFormat, in));
```

Loading an MP3 / OGG

- Note that this decodes the MP3
  - The raw sound data is stored in memory
  - Not feasible with this example
    - With a 4 minute song -> 117 MB mem usage
- Better approach
  - Decode on the fly
  - Read frames from input line
  - Write frames to output line

Decode on-the-fly: Create AudioInputStream


```
AudioInputStream in =
   AudioSystem.getAudioInputStream(getClass().getResource(
   "sfx/rool-oh_why.mp3"));
AudioFormat inFormat = in.getFormat();
AudioFormat decodedFormat = new
   AudioFormat(AudioFormat.Encoding.PCM_SIGNED,
        inFormat.getSampleRate(), 16, inFormat.getChannels(),
        inFormat.getChannels() * 2, inFormat.getSampleRate(),
        false
);
bgMusic = new SoundThread(decodedFormat,
        AudioSystem.getAudioInputStream(decodedFormat, in));
```

Decode on-the-fly: Write to “sound card”


```
byte[] data = new byte[4096];
SourceDataLine line = getLine(targetFormat);
if (line != null) {
   line.start(); // Start
   int nBytesRead = 0, nBytesWritten = 0;
   while (nBytesRead != -1) {
        nBytesRead = din.read(data, 0, data.length);
        if (nBytesRead != -1)
              nBytesWritten = line.write(data, 0, nBytesRead);
   }
   // Stop
   line.drain(); line.stop();
   line.close(); din.close();
}
```

Other approaches: JLayer


```
URL song = getClass().getResourceAsStream(
  "/resources/sound/song.mp3");
Player p = new Player(song);
p.play();
```

- Differences to the previous approach
  - Cannot control caching / prefetching
  - Cannot modify it
  - Need not synchronization, etc.

Advanced Sound API

- More information on Java Sound:
  - See Java Tutorial
  - http://java.sun.com/docs/books/tutorial/sound/TOC.html
- Tools to edit sound
  - Try Audacity ... audacity.sourceforge.net/
- “Free” sound
  - See e.g. *electrobel.be* or CC sites.

Agenda :: Sound

- Introduction
- Playing sounds
- Java Sound API
- Using external SPIs
- **Sound analysis –> DSP**

Sound from the Microphone: Format


```
public static AudioFormat getAudioFormat() {
   float sampleRate = 8000.0F; // use e.g. 8000, 11025, 16000, 22050, 44100, 48000
   int sampleSizeInBits = 8; // use e.g. 8 or 16 ... 8 bits -> we use a single byte
   int channels = 1; // make it mono. stereo would be 2
   boolean signed = true; // for 8 bits this is -128 to 127 (or something like that :)
   boolean bigEndian = false;
   return new AudioFormat(sampleRate,
                sampleSizeInBits,
                channels,
                signed,
                bigEndian);
}
```

Microphone: Open Line


```
TargetDataLine line;
// create an AudioFormat object:
AudioFormat format = getAudioFormat();
// create a DataLine
DataLine.Info info = new
        DataLine.Info(TargetDataLine.class, format);
if (!AudioSystem.isLineSupported(info)) {
   System.err.println("Line not supported");
}
```

Microphone: Analyse Data


```
// open the line and start capturing ...
line = (TargetDataLine) AudioSystem.getLine(info);
line.open(format);
AudioInputStream is = new AudioInputStream(line);
byte[] buffer = new byte[1024];
line.start();
while (is.read(buffer) > 0) {
   int max = -128;
   for (int i = 0; i < buffer.length; i++) {
        max = Math.max(Math.abs(buffer[i]), max);
   }
   if (max > 5) do.something();
}
```

Analyse Data: What for?

- Using input sound as control element
  - Take for instance moon lander
  - Sound energy fires up engine …
- Employ DFT / FFT
  - Which frequency band has maximum energy?
  - Use this as input to the game
    - E.g. high sounds left, low sounds right …

Thanks ...

That’s it for sound ...