Computer Games 2011
Sound & Physics

Dr. Mathias Lux
Klagenfurt University

Agenda

- **Motivation & Introduction**
- Digital sound & your computer
- Recording & Production
- FMOD
  - Code examples

Motivation

- Sound is critical to success
  - No game without sound …
  - Sound design is a profession
- Diverse hardware
  - Headphone
  - Speakers (2, 2.1, 5.1, 7.1)
  - Hifi || !Hifi

Examples

- Dead Nation?
- Little Big Planet?
- Musikspiele?
  - GH, RB, SS …

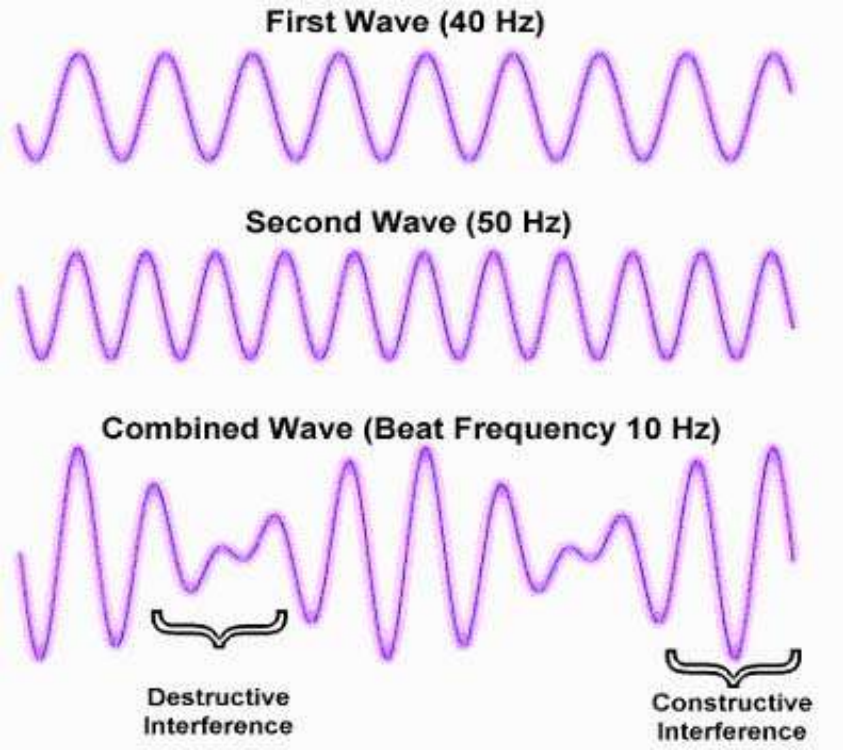
Agenda

- Motivation & Introduction
- **Digital sound & your computer**
- Recording & Production
- FMOD
  - Code examples

What is sound?

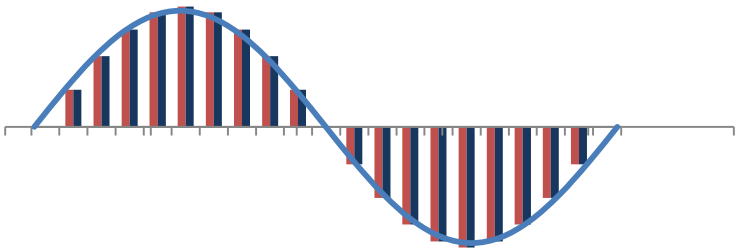
What is sound?

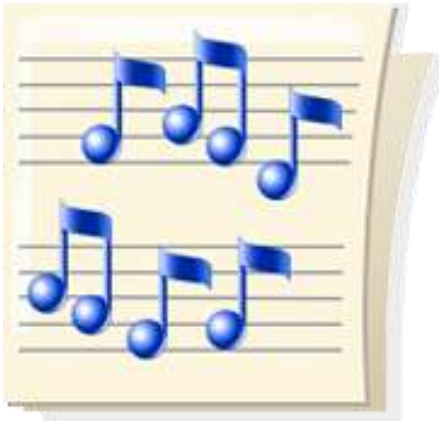
- Multiple sounds at the same time?



What is digital sound?

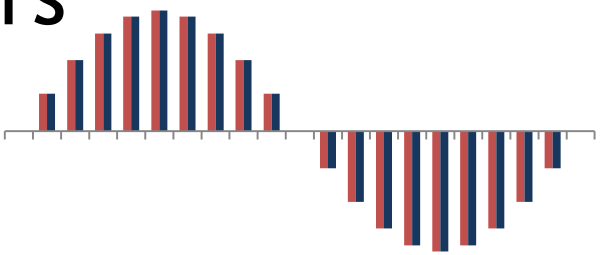
- A digitization of the wave.
  - Either a recipe for reconstruction
  - Or a discrete approximation





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

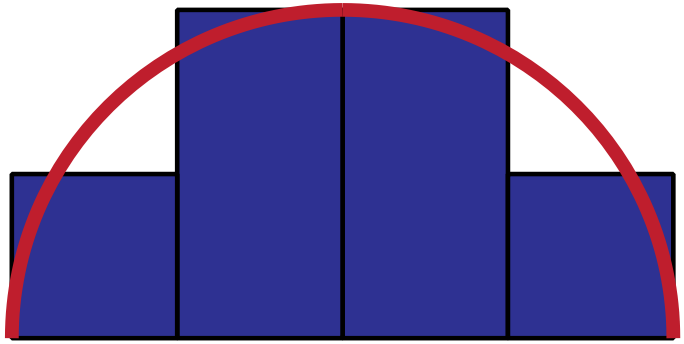


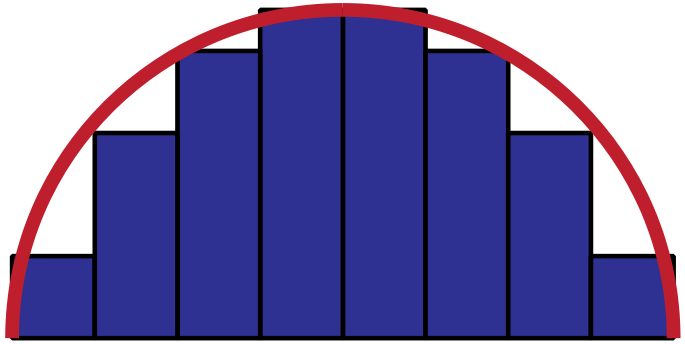
Sampled sound

- 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

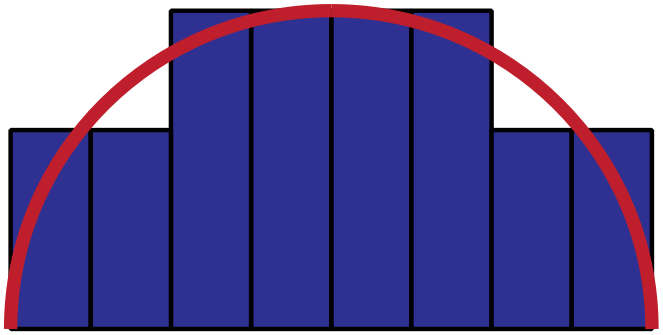
- With sampling rate x we can approximate frequencies up to x/2
- Assume frequency 1
  - sampling rate of 1 -> “0”
  - sampling rate of 2 -> “1,-1”

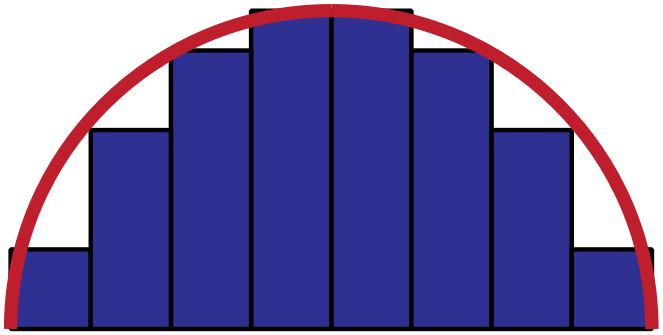




Quantization

- 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

Types of sounds

- Samples
- MIDI
- Loops

Types of sounds: Samples

- Rendered sound
- Typically sampled and edited from “real sounds”
- Can be altered with effects
  - Possibilities limited, e.g. no exchange or silencing of single musical instrument, voice, etc.

Types of sounds: Midi

- Musical Instrument Digital Interface
- Communication and synchronization of
  - musical instrument and
  - computer equipment
- Example: master keyboard
  - keyboard is played,
  - just commands -> computer, no “sound”
  - compute”renders” sound based on commands

Types of sounds: Loops

- Loops are „variable length songs“
  - intro ||: loop :|| outro
- Plays along with a level
- Ends with the end of the level
  - Not before or after …
- Highlights special events

- See example …

Sound Formats & Compression

- Waveform Audio Format
  - Container for several compression formats
  - Includes PCM, MP3, GSM, μ-Law
- Musical Instrument Digital Interface
  - File format for MIDI
- Compressed Audio Formats
  - MP3, OGG, AAC, …

Multiple sounds at once: Mixer

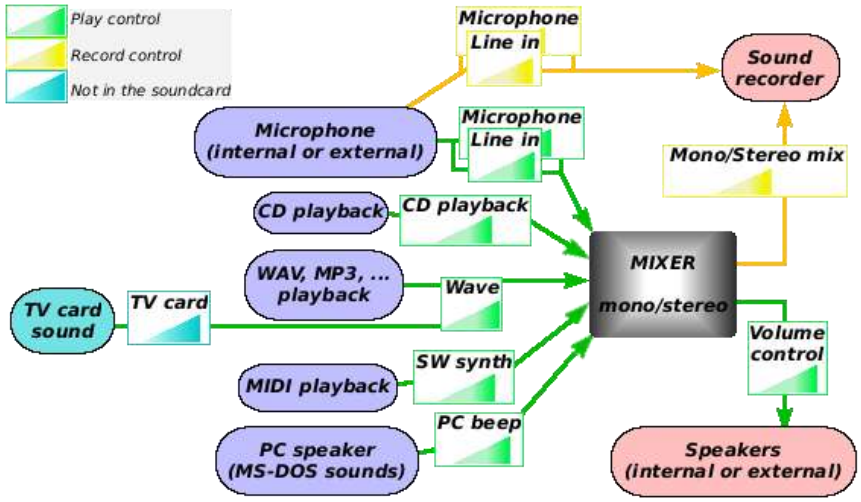
- Soundcard mixer receives inputs from
  - external connectors and the
  - sound card's digital-to-analog converters
- It mutes, amplifies and adds signals
- It can be controlled by software

src. Wikipedia

Soundcard Mixer: Typical In- & Outputs

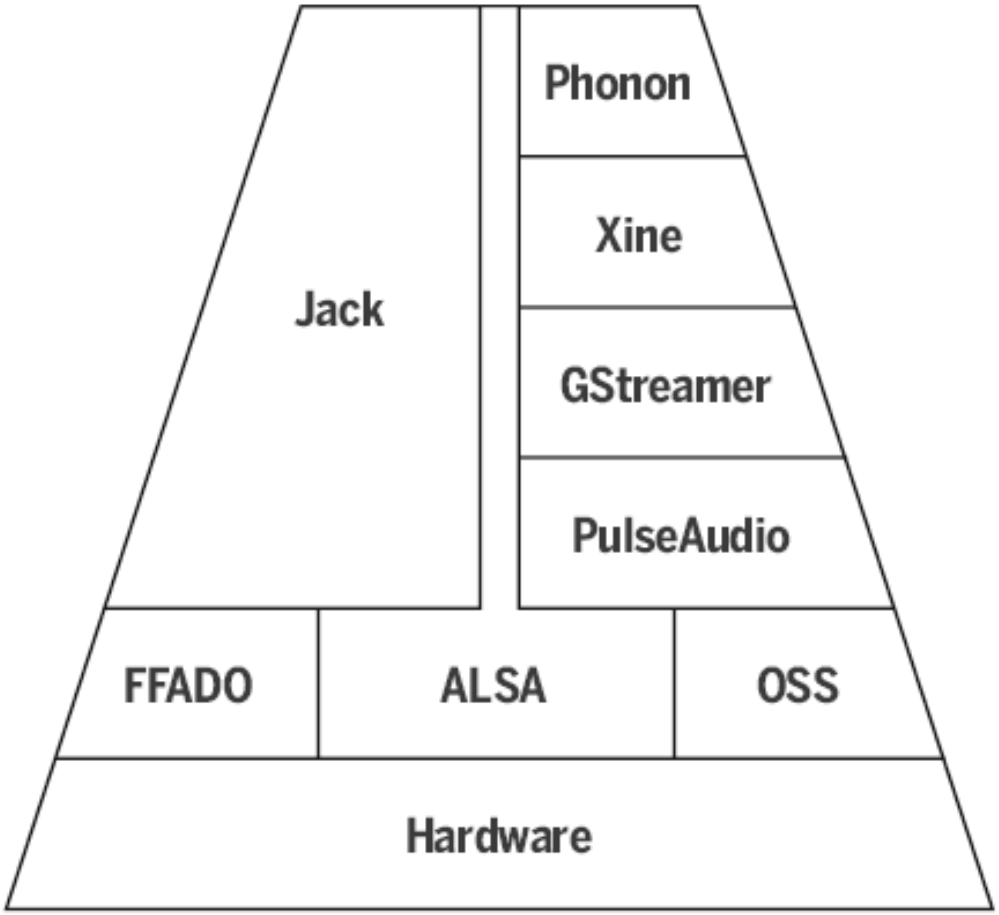
- Wave / PCM
- MIDI
- CD Playback
- Microphone
- Line/Aux In
- PC Speaker
- SPDIF

- Line / Aux out
- SPDIF



src. Wikipedia

Advanced Linux Sound Architecture (ALSA)



src. tuxradar.com

Advanced Linux Sound Architecture (ALSA)

- Provides API between
  - software & hardware
- Has compatibility layer
- Communicates with sound driver for hardware
- Utilizes hardware if possible
- Uses software if no hardware support
- User configurable

src. tuxradar.com

Agenda

- Motivation & Introduction
- Digital sound & your computer
- **Recording & Production**
- FMOD
  - Code examples

Voice Recording: Equipment

- Hardware
  - Microphone
  - Cable
  - Soundcard
  - Studio
- Software
  - For recording and editing

Voice Recording: Microphone

- Different types of microphones
  - Dynamic, condenser, piezoelectric, etc.
- XLR versus TRS (tip, ring sleeve)
- Minimize the way of the analogue signal

Voice Recording: Microphone

- Pop filter
- Microphone spider
- Stand

Voice Recording: Cable & Soundcard

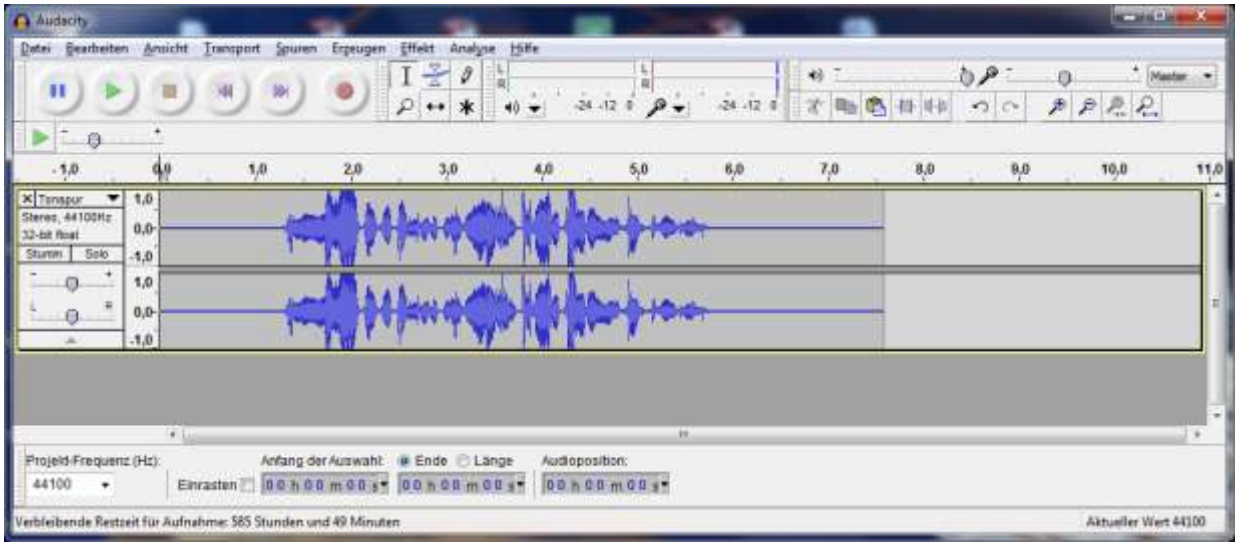
- Prefer XLR to TSR
- The nearer the soundcard the better
- Soundcards properties
  - Firewire vs. USB
  - 16 vs. 24 bit sampling rate
  - 48 vs. 96 vs. 192 kHz

Voice Recording: Studio

- Balance recording level
- Minimize background noise
  - cars, talking, barks, walking, etc.
  - computers, mobile phones, tv, etc.
- Select a suitable speaker
  - slightly annoying voice for books
  - calming, nice voice for explanations
  - high voice for phone-like experience
- Normalize the speakers condition
  - having a cold, time of day, etc.

Voice Recording: Software

- Many different programs available
- Open source: Audacity
  - Multiplatform, multiformat (lame & ffmpeg)



Audacity Demo

- Record talk
- Cut and trim
- Change pitch
- Add reverb

Sound Production: Other software

- Sequencer
  - Fruity Loops
  - MU.LABS Free
- Recording
  - Audacity
  - Cubase
- …

Agenda

- Motivation & Introduction
- Digital sound & your computer
- Recording & Production
- **FMOD**
  - Code examples

FMOD

- FMOD Ex
  - Low level API (mixer, interfaces, dsp, 3D, …)
- FMOD Event System
  - Built on FMOD Ex
  - Access to events designed in Designer
- FMOD Designer
  - Tool for Event creation
  - Looping, effects, etc.

FMOD Licenses

- Non Commercial License
  - free license
- Commercial License
  - 1,500 – 6,000 $
- Casual License
  - 500 – 3,000 $
- Shareware License
  - 100 $ per platform & product

Agenda

- Motivation & Introduction
- Digital sound & your computer
- Recording & Production
- FMOD
  - **Code examples**

Hands-On

- Integrating FMOD Ex
- Playing a sound
- Looping a sound
- Fade outs
- DSP
- 3D

FMOD & XNA

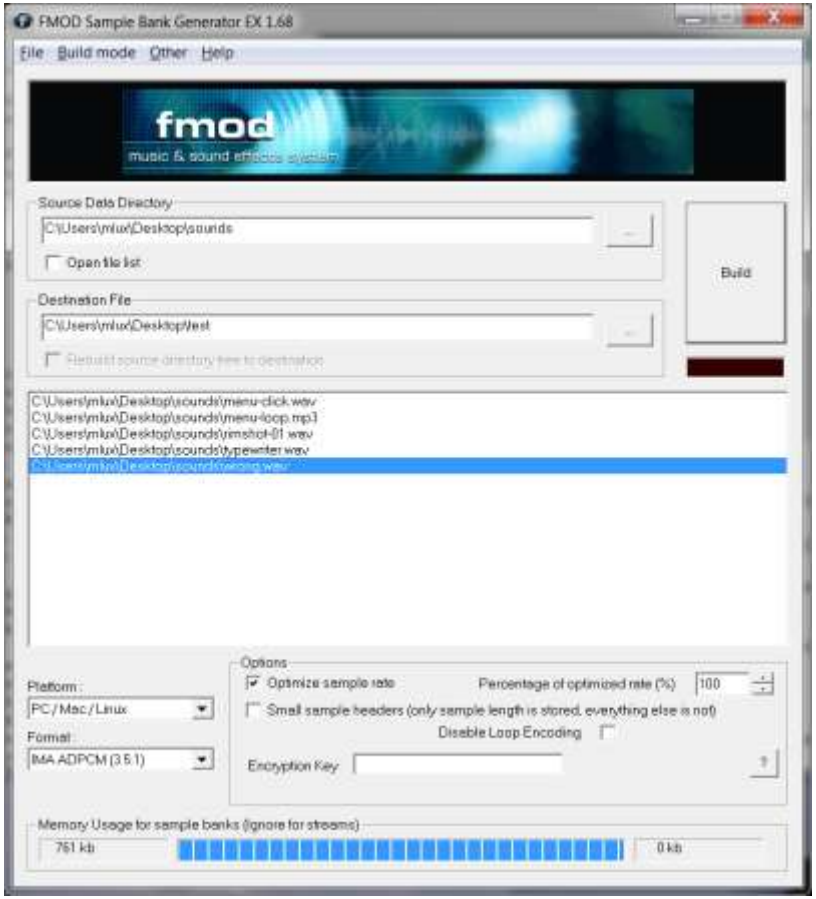
- Install FMOD SDK
- Include .cs files
  - `fmod.cs`
  - `fmod_dsp.cs`
  - `fmod_error.cs`
- Copy `fmod.dll` to search path

Play

```
// create structure for error & success reports
FMOD.RESULT result;
// create FMOD system
result = FMOD.Factory.System_Create(ref system);
ERRCHECK(result);
// init system
result = system.init(32, FMOD.INITFLAGS.NORMAL, (IntPtr)null);
ERRCHECK(result);
// create background sound
result = system.createSound("../../../menu-loop.mp3", FMOD.MODE.SOFTWARE, ref menuLoop);
```

```
result = system.createSound("../../../../../examples/media/wave.mp3",
    (FMOD.MODE._2D | FMOD.MODE.HARDWARE | FMOD.MODE.CREATESTREAM), ref sound);
ERRCHECK(result);
```

Memory usage …



Loop

```
[...]
// set to loop:
result = menuLoop.setMode(FMOD.MODE.LOOP_NORMAL);
ERRCHECK(result);
result = menuLoop.setLoopCount(-1);
ERRCHECK(result);
// play
result = system.playSound(FMOD.CHANNELINDEX.FREE, menuLoop, false, ref channelbgmusic);
ERRCHECK(result);
```

Sound Groups & Volume

```
[...]
// set to loop:
result = menuLoop.setMode(FMOD.MODE.LOOP_NORMAL);
ERRCHECK(result);
result = menuLoop.setLoopCount(-1);
ERRCHECK(result);
// add to sound group:
result = system.createSoundGroup("bgmusic", ref bgmusic);
ERRCHECK(result);
bgmusic.setVolume((float)0.2);
menuLoop.setSoundGroup(bgmusic);
// play
result = system.playSound(FMOD.CHANNELINDEX.FREE, menuLoop, false, ref channelbgmusic);
ERRCHECK(result);
```

Fade out

```
public override void Update(GameTime gameTime) {
    if (fade) {
        if (volume > 0) {
            volume -= (float)0.001;
            result = game.bgmusic.setVolume((float)volume);
            game.ERRCHECK(result);
        } else {
            fade = false;
            result = game.bgmusic.stop();
            result = game.bgmusic.setVolume((float)0.2);
            game.ERRCHECK(result);
        }
    }
    result= game.system.update();
[...]
}
```

DSPs

```
FMOD.Sound demoSound = null;
FMOD.RESULT result;
result = game.system.createSound("../../../woohoo.wav", FMOD.MODE.SOFTWARE, ref demoSound);
demoSound.setMode(FMOD.MODE.LOOP_OFF);
FMOD.DSP dsp = null;
FMOD.DSPConnection conn = null;
count++;
game.system.createDSPByType(types[count%types.Length], ref dsp);

FMOD.Channel demoChannel = null;

game.system.playSound(FMOD.CHANNELINDEX.FREE, demoSound, false, ref demoChannel);

demoChannel.addDSP(dsp, ref conn);
```

DSPs (ctd.)

```
FMOD.DSP_TYPE[] types = {
                                FMOD.DSP_TYPE.ECHO,
                                FMOD.DSP_TYPE.CHORUS,
                                FMOD.DSP_TYPE.DELAY,
                                FMOD.DSP_TYPE.COMPRESSOR,
                                FMOD.DSP_TYPE.HIGHPASS,
                                FMOD.DSP_TYPE.LOWPASS,
                                FMOD.DSP_TYPE.REVERB,
                            };
```

3D Sound

- Initialization

```
float DISTANCEFACTOR = 1.0f; // meter
FMOD.VECTOR pos1 = new FMOD.VECTOR();
FMOD.VECTOR vel1 = new FMOD.VECTOR();


public Soundmanager() {
    pos1.x = 0.0f * DISTANCEFACTOR; pos1.y = -0.0f; pos1.z = 0.0f;
    vel1.x = 0.0f; vel1.y = 0.0f; vel1.z = 0.0f;
```

3D Sound

- Initialization

```
result = system.createSound("../../../rimshot.wav", (FMOD.MODE.SOFTWARE | FMOD.MODE._3D), ref testA)
ERRCHECK(result);
result = system.createSound("../../../wrong.wav", (FMOD.MODE.SOFTWARE | FMOD.MODE._3D), ref testB);
ERRCHECK(result);

// 3D
system.setSpeakerMode(FMOD.SPEAKERMODE.STEREO);
ERRCHECK(result);

result = system.set3DSettings(1.0f, DISTANCEFACTOR, 1.0f);
ERRCHECK(result);

// unverändert bis 1m, ab 50m nix mehr zu hören
result = testA.set3DMinMaxDistance(1.0f * DISTANCEFACTOR, 50.0f * DISTANCEFACTOR);
ERRCHECK(result);
```

3D Sound

Init:

```
result = system.createSound("../../../rimshot.wav", (FMOD.MODE.SOFTWARE | FMOD.MODE._3D), ref testA)
ERRCHECK(result);
result = system.createSound("../../../wrong.wav", (FMOD.MODE.SOFTWARE | FMOD.MODE._3D), ref testB);
ERRCHECK(result);

// 3D
system.setSpeakerMode(FMOD.SPEAKERMODE.STEREO);
ERRCHECK(result);

result = system.set3DSettings(1.0f, DISTANCEFACTOR, 1.0f);
ERRCHECK(result);

// unverändert bis 1m, ab 50m nix mehr zu hören
result = testA.set3DMinMaxDistance(1.0f * DISTANCEFACTOR, 50.0f * DISTANCEFACTOR);
ERRCHECK(result);
```

3D Sound

- Play

```
public void playSoundA()
{
    result = system.playSound(FMOD.CHANNELINDEX.FREE, testA, true, ref channel);
    ERRCHECK(result);
    result = channel.set3DAttributes(ref pos1, ref vel1);
    ERRCHECK(result);
    result = channel.setPaused(false);
    ERRCHECK(result);
}
```

SoundDemoGame

- Walkthrough …

Additional Information

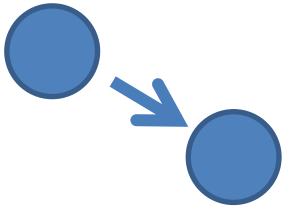
- FMOD SDK
  - API documentation
  - C# samples
- Online
  - FMOD wiki
  - Mailing list

Collision Detection

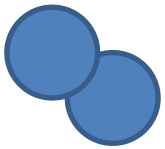
- Collision occurs before the overlap
- So an overlap indicates a collision

- Example Pool Billard

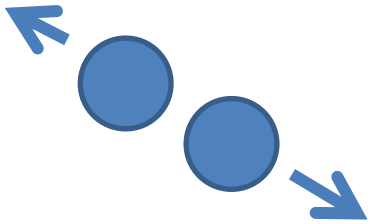
Before



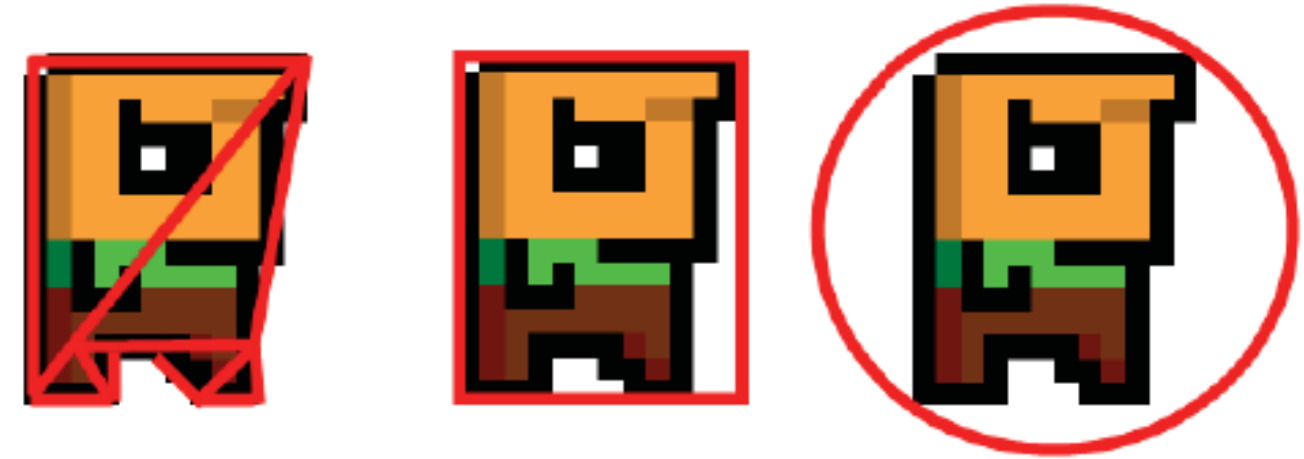
Computed



Actual



Bounding Boxes



src. M. Zechner, Beginning Android Games, Apress, 2011

Triangle Mesh

- Convex hull, convex polygon
- Tight approximation of the object
- Requires storage
- Hard to create
- Expensive to test

src. M. Zechner, Beginning Android Games, Apress, 2011

Axis Aligned Bounding Box

- Smallest box containing the object
- Edges are aligned to x-axis and y-axis
- Fast to test
- Less precise than triangle mesh
- Stored as corner + width + height

src. M. Zechner, Beginning Android Games, Apress, 2011

Bounding Circle

- Smalles circle that contains the object
- Very fast test
- Least precise method
- Stored as
  center + radius

src. M. Zechner, Beginning Android Games, Apress, 2011

Bounding shape updates

- Every object in the game has
  - position, scale, oritention &
  - bounding shape
- Position, scale and orientation are influenced by the game play
  - Bounding shape has to be update accordingly

src. M. Zechner, Beginning Android Games, Apress, 2011

Bounding shape updates

- Changes in position
  - Move vertices, box corner or circle center
- Changes in scale
  - Easy if center of change is center of object
  - Scale vectors in relation to center,
  - or change radius of a bounding circle

src. M. Zechner, Beginning Android Games, Apress, 2011

Bounding shape updates

- Change in orientation
  - if center of rotation is center of object …
  - for triangle meshes triangles are rotated
  - bounding circles remain untouched
  - for bounding boxes corner points have to be rotated & a new box has to be found

src. M. Zechner, Beginning Android Games, Apress, 2011

Phases in Collision Detection

- Broad phase
  - Identify potentially colliding objects
  - With less precise bounding shapes
- Medium phase
  - Determine actually colliding objects
- Narrow phase
  - Find when (contact time) and where (contact set)

src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Collision Culling: Circles

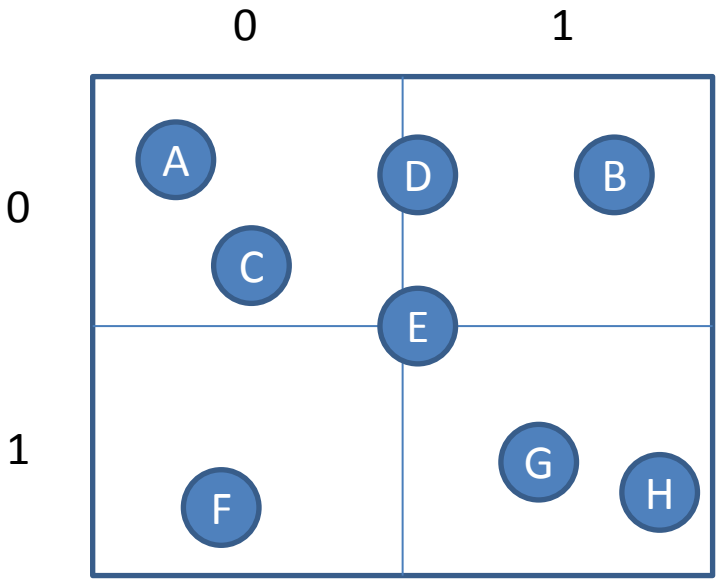
- Bounding Circles
  - with center C_i and radius r_i
  - potential collision when $| C_i - C_j | < r_i + r_i$
  - number of collision tests c with c_p as time for one test

$$c = \frac{n(n-1)}{2} c_p$$

src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Collision Culling: Circles

- Bounding Circles
  - speed up with grid
  - eg. B_{00} = {A, C, D, E}
- Supposing we have b circles in each bin:

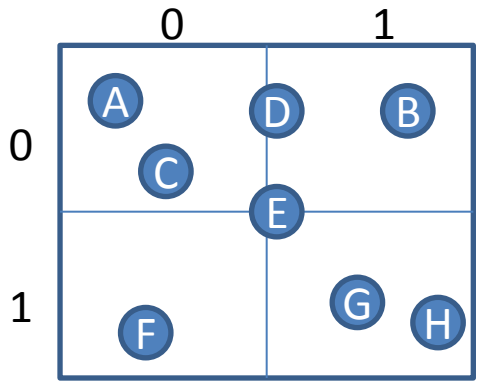


$$\tilde{c} = \frac{n/b(n/b-1)}{2} c_p$$

src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Collision Culling: Circles

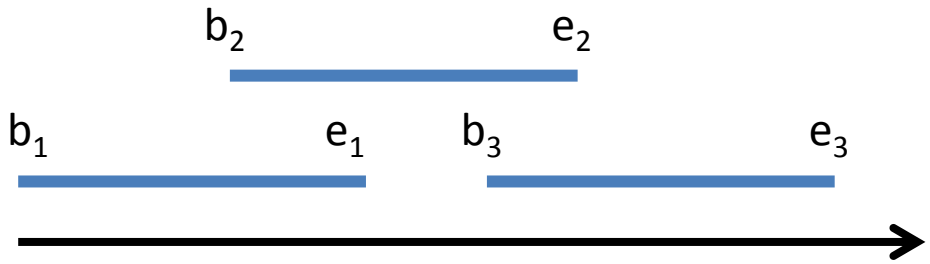
- Still missing the cost for determining the bin …
- Naive approach tests against axis-aligned borders
- Trade-off between
  - grid size (det. bin)
  - culling cost c
- Solution: add time coherence



src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Collision Culling: Axis-Aligned Bounding Boxes

- Intersection of AABBs based on intervals
- Sorting andand update in real time
- Intervals reflect edges parallell to an axis
  - $I_i = [b_i, e_i]$

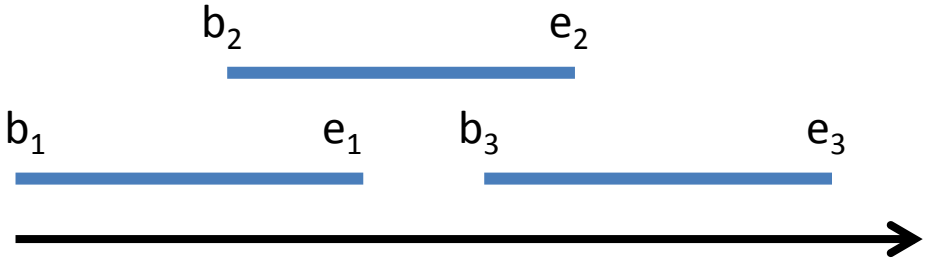


src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Collision Culling: Axis-Aligned Bounding Boxes

Start from left hand side

1. b1 encountered, A={} -> no intersection, update A={I1}
2. b2 encountered, intersection I1+I2, update A={I1, I2}
3. e1 encountered, update A={I2}
4. b3 encountered, intersection I2+I3, update A={I2, I3}
5. … and so on



src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

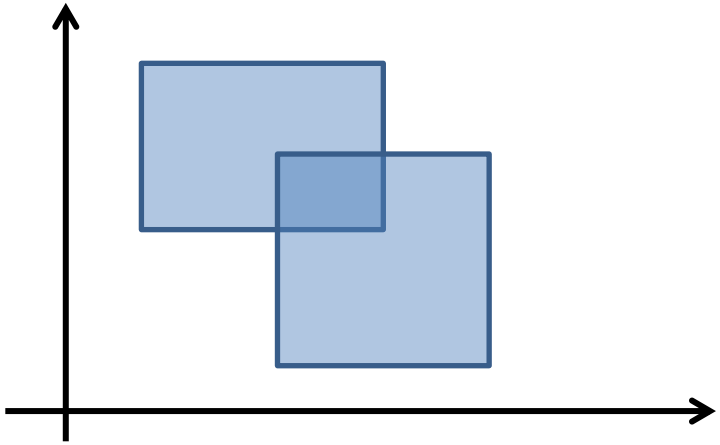
Collision Culling: Axis-Aligned Bounding Boxes

- Apply “dirty” tag for sorting
  - Faster if fewer changes
- Performance depends on the number of intersections m
  - sweep is O(n), sort is O(n log n), intersection report is O(m)
  - worst case: all intervals overlap, m=n^2

src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Rectangle Intersection

- Rectangles intersect when
  - they intersect on x- axis and y-axis
- So just test the candidates on the other axis



src. David H. Eberly, Game Physics 2nd Edt., Morgan Kaufmann, 2010

Physics

- *force* = *mass* x *acceleration*
  - *acceleration* & *force* are vectors
- *acceleration* = *force* / *mass*
  - the more mass the more force we need for acceleration
- Ignoring mass & force for simple games
  - acceleration is (0, -9.81), ~100 m/s max.

Euler Integration

- Numerical Euler integration
  - not stable, error prone

```
Vector2 position = new Vector2();
Vector2 velocity = new Vector2();
Vector2 acceleration = new Vector2(0, -10);
while(simulationRuns) {
   float deltaTime = getDeltaTime();
   velocity.add(acceleration.x * deltaTime, acceleration.y * deltaTime);
   position.add(velocity.x * deltaTime, velocity.y * deltaTime);
}
```

Velocity Verlet

- Simplified version of Verlet Integration

$$\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{\vec{a}(t) + \vec{a}(t+\Delta t)}{2}\Delta t$$

- Algorithm

Calculate: $\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\vec{a}(t)\,\Delta t^2$

Derive $\vec{a}(t+\Delta t)$

Calculate: $\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{1}{2}\,(\vec{a}(t) + \vec{a}(t+\Delta t))\,\Delta t$

src. http://en.wikipedia.org/wiki/Verlet_integration#Velocity_Verlet

Newtons Laws

1. In the absence of forces objects remain at rest
2. *Force* = *mass* x *acceleration*
3. If a force is exerted on a body there is an equal but opposite direction on some body that interacts with it

Forces

- Gravitational forces
  - 9.81 m/s on Earth
- Spring forces
  - force depends on distance to a point
- Friction & viscosity
  - sliding on surfaces and through gases or fluids
- Torque

Tools & Frameworks

- RAD Game Tools
- Autodesk Scaleform
- Havok
- Bullet
- Box2D

RAD Game Tools

- Privately held company owned by Jeff Roberts & Mitch Soule
- Based in Kirkland, Washington
- Develops video game software technologies
- Hires one specific person to write, document and support each single product
- refuses to patent any of their technologies

src. Wikipedia (en)

RAD Game Tools

- 1988 Windows dev & consulting
- 1993 Smacker, 8-bit video codec (time of CDs)
- 1994 Fastest growing company in Utah, 63rd in U.S.
- 1995 Aquisition of Miles Sound System
- 1999 Release of Bink
- 2002 Pixomatic (software 3D fallback system)

src. Wikipedia (en)

Bink

- Video game codec for games
  - licensed for > 5,800 games
- Range of bitrates & resolutions
  - 75 - 1200 kBps (720p)
- Full control through Bink SDK
  - no callbacks etc.
- Self-contained, no 3rd party SW

src. http://www.radgametools.com/

Bink

- Up to 10 times faster
  - than modern codecs
- Ressource efficient
  - up to 16 MB less memory
- Multiple platforms
  - same API on 14 different platforms
- Supported by many game engines
  - eg. already integrated in Unreal Engine

src. http://www.radgametools.com/

Bink

- Audio codec with VBR
  - up 15:1 compression, no add. licenses
- Integrated mixer
  - 5.1 & 7.1 support
- Optimized for many platforms
  - Takes advantage of SPUs, SSE2, assembly optimizations …
- Extremely robust
  - and well received by customers

src. http://www.radgametools.com/

Miles Sound System

- Licensed for > 5,000 games
- Integrates
  - high-level sound authoring with 2D and 3D digital audio
  - featuring streaming, environmental reverb, DSP filtering, multichannel mixing
  - highly-optimized audio decoders (MP3, Ogg and Bink Audio)
- Designer for Windows
- Deployment to 14 platforms supported

src. http://www.radgametools.com/

Miles Sound System

- MP3 decoder and license
- Bink audio decoder
  - 30% smaller memory footprints
- Ogg Vorb audio decoder
  - 40% faster than libvorbis
- Robust & simple and clean API

src. http://www.radgametools.com/

Iggy Game UI

- User Interface library for
  - Windows, Xbox 360, Sony PS3, Nintendo 3DS, and Nintendo Wii
- Utilizes Flash content (up to v9)
  - includes ActionScript 3 interpreter
- Interactive performance tools
  - for memory and performance debugging

src. http://www.radgametools.com/

Iggy Game UI

- A triangulator to convert vector shapes to triangles
- A bitmap font rasterizer
- Conversion of wide lines to triangles
- A system to generate edge anti-aliasing the polygons
- Linear and circular gradient support
- Shader based GPU-accelerated implementation of filter effects defined by Flash

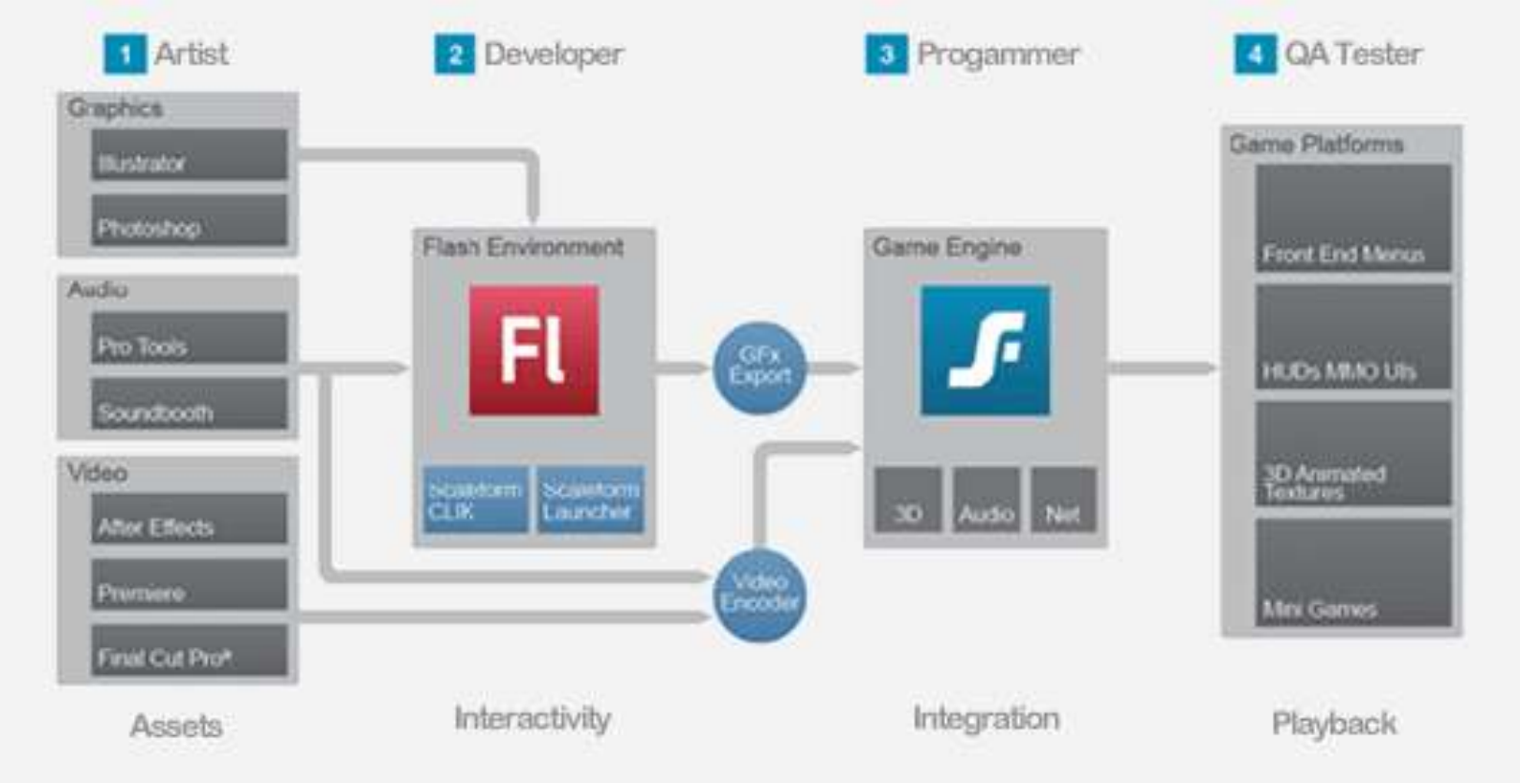
src. http://www.radgametools.com/

Iggy Game UI

- A full audio mixer
- MP3 decoder from Miles (including patent rights)
- Custom memory management
  - Multiple instances
- Debugging tools:
  - Iggy Explorer (to view the hierarchies of the artists),
  - Telemetry bindings
  - and full memory allocation tracking.

src. http://www.radgametools.com/

Autodesk Scaleform



Src. http://gameware.autodesk.com/scaleform/features/overview

Scaleform

1. Create UI assets, including bitmaps, vectors, audio, and video using the Adobe Creative Suite
2. Import assets into Flash Studio and add interactivity and animation
3. Export Flash content to game & connect UI
4. Test the Flash content navigation, localization and functionality running in game

Autodesk GAMEWARE

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT | WIEN GRAZ

Src. http://gameware.autodesk.com/scaleform/features/overview

Havok Physics

- Irish company, since 1998
  - owned by Intel
- Real-time physics simulation
  - collision and dynamics of rigid bodies
  - ragdolls & character animation
  - movement of vehicles
- *http://youtu.be/Vgxtt9dHw_8?hd=1*

Havok Cloth

- Simulation of cloth, hair, foliage and other deformable objects
- Animate believable behaviors of character garments and environmental cloth

- http://youtu.be/qTOzOgAdYWk?hd=1

Havok Destruction

- Tool set & runtime SDK
- Fracture meshes and use in-game
- Destruction of complex structures
- Simulate destruction in runtime

- http://youtu.be/clcg5eotZlY?hd=1

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

havok®

Havok Animation

- Integration with Havok Physics
- Repurpose existing content from one character to another with runtime retargeting and mirroring
- Integration of characters into dynamic environments with procedural look at, foot placement and reaching
- http://youtu.be/1y3G0ZxA_eE?hd=1

Havok AI

- Pathfinding & path following in highly dynamic and destructible environments
- Flying, swimming and wall climbing
- Individual character motions even in congested crowd scenes
- http://youtu.be/rlbjGiP104M?hd=1

Autodesk

- Maya & 3ds Max
- Educational programs
  - Autodesk University
  - Student licenses (http://students.autodesk.com)

Autodesk
Education Community

Other tools

- Bullet
- Box2D

Vielen Dank …

… für die Aufmerksamkeit