Cross-Plattform Spieleentwicklung mit der SDL

Anhand der Spiele-Neuentwicklung "*BlinkenSisters - Hunt for the Lost Pixels*", einem Jump'n'Run, werde ich zeigen, wie ein solches Spiel entwickelt werden kann.

Was ist SDL?

SDL steht für “**S**imple **D**irectmedia **L**ayer”

- Cross-Plattform multimedia library für low-level Zugriff auf Grafik, Audio, Tastatur, Maus, Joystick.
- Grafik über 2D Framebuffer oder 3D-Hardware über OpenGL
- Erweiterbar über div. Zusatzlibraries, z.B. SDL_net für Netzwerkzugriff

SDL: Unterstützte Betriebssysteme

- Offiziell: Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, und QNX
- Inoffiziell: AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2

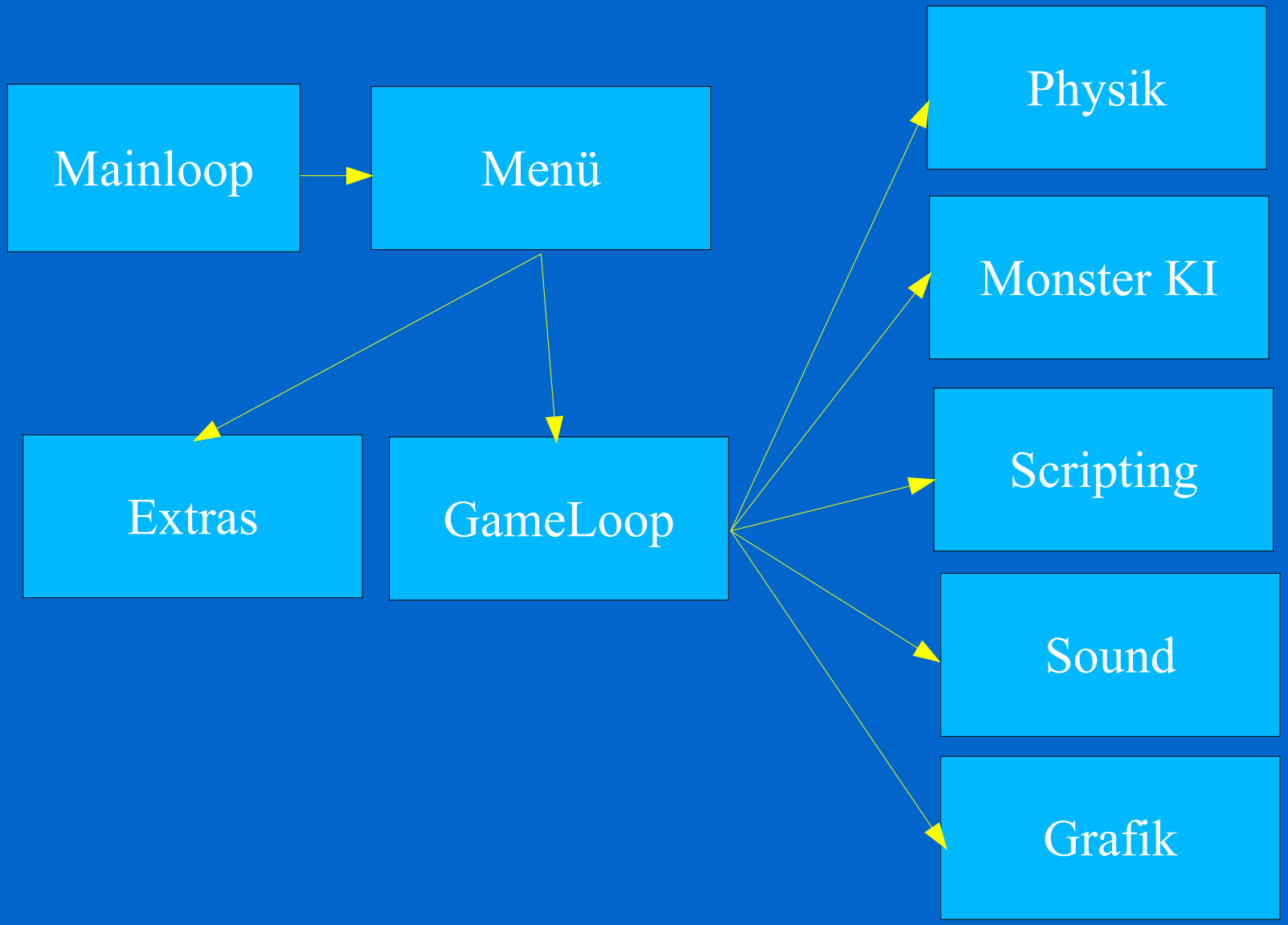
Was ist BlinkenSisters?

- Jump'n'Run im Stil alter C64 und Amiga-Games, jedoch mit aktueller Technologie implementiert
- Lern- und Demonstrationsprojekt für die Entwicklung einfacher Spiele
- Erweiterbar durch einfache Dateiformate und Scripting-Support
- Interessant, lustig und mit coolen Features :-)

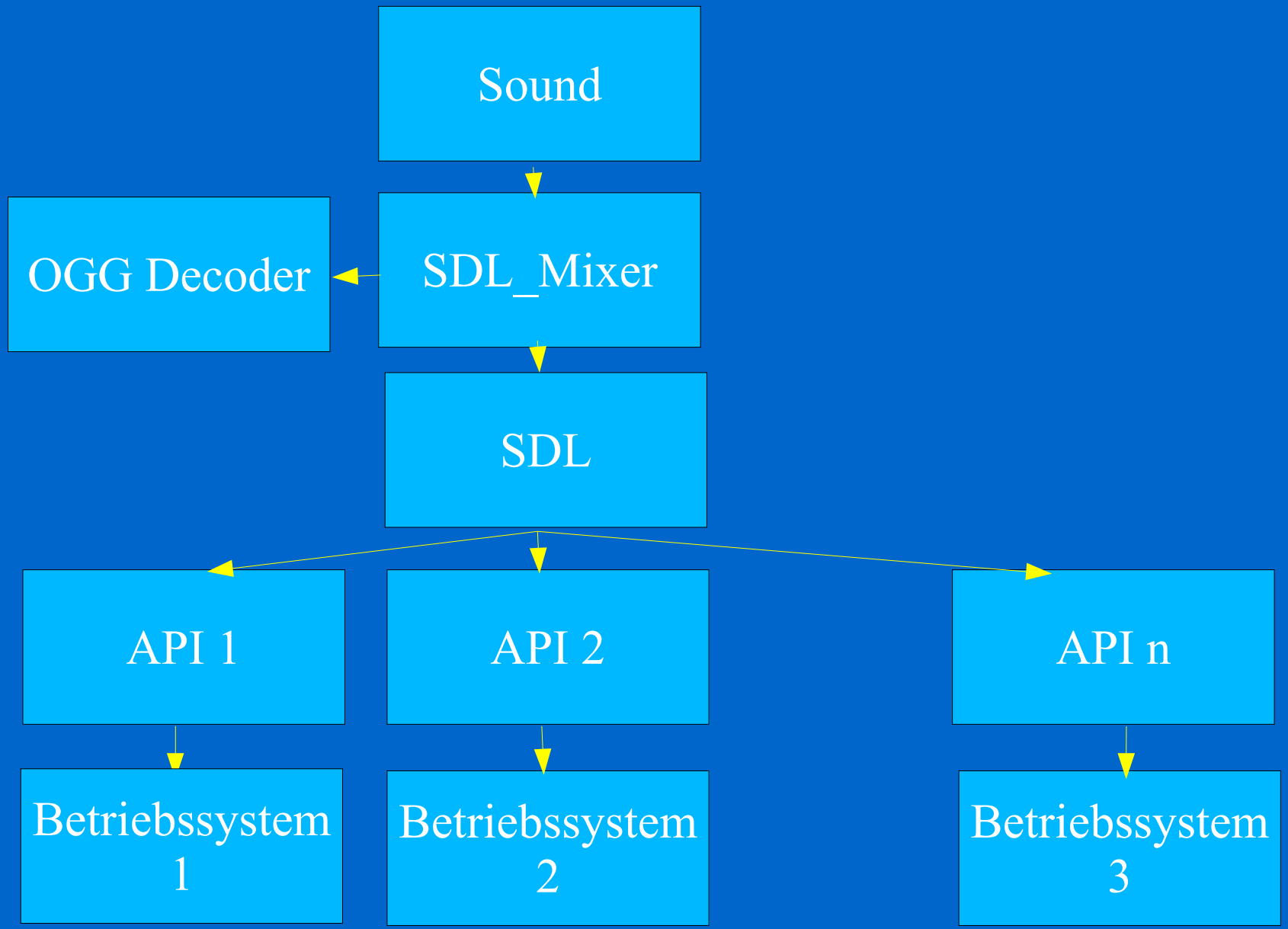
BS: Technische Daten

- Spielprinzip: 2D Jump'n'Run
- Grafik: 640x480 Pixel/24 Bit
- Sound: OGG/Vorbis
- Input: Keyboard, Joystick, Gamepad
- Levels: Tile-basiert
- Videos: BMF
- Add-Ons: BMF über HTTP
- Sprache: C/C++
- Scripting: LUA

BS: Aufbau der Engine



BS und SDL: Beispiel Sound



Achtung bei Cross-Plattform Entwicklung

- Datentypen in C/C++ passen ihre Größe dem jeweiligen Betriebssystem und der Hardware an
- Verschiedene Hardware-Plattformen ordnen Bytes eines Multibyte-Wertes verschieden an (Big/Little-Endian Problem)
- Build-Systeme sind nicht auf allen Plattformen gleich

Problem: Datentypen

- Der einzige (normalerweise) verlässliche Datentyp ist CHAR (1 Byte)
- LONG entspricht (meistens) der Größe von VOID
- VOID entspricht meistens der BUS-Breite oder dem größten addresierbaren Speicher (2, 4, 8 oder 16 Byte)
- Für INT gilt: CHAR < INT <= LONG

- Um verlässlich Dateien zu lesen und über Netzwerke zu kommunizieren, müssen vom Entwickler eigene Datentypen definiert werden

Lösung: Datentypen

- Für praktisch alle Anwendungszwecke werden von der SDL Datentypen mit definierter Größe angeboten

- Beispiel Ganzzahl mit 4 Byte (32 Bit):

Uint32: Ganzzahl mit 32 Bit ohne Vorzeichen
Sint32: Ganzzahl mit 32 Bit mit Vorzeichen

Problem: Byteorder

- Die Reihenfolge der Bytes eines Multibyte-Wertes hängt von der Hardware ab

- Beispiel an der Zahl 256:

Big-Endian: 0x0100 (z.B. MacOSX/PowerPC)
Little-Endian: 0x0001 (z.B. Windows/i386)

- Wird also z.B. Ein Wert binär aus einer Datei gelesen, so müssen die Bytes unter Umständen entsprechend gedreht werden.

Lösung: Byteorder (1)

- Für viele dieser Probleme bietet die SDL bereits Lösungen.

- Beispiel: SDL_net

SDLNet_Read32() liest einen Uint32 aus einem TCP-Stream und konvertiert automatisch von "Network Byte Order" auf die lokale Byteorder

Lösung: Byteorder (2)

- Manche Konvertierungen weiterhin selbst implementiert werden.
- Abhilfe: Die SDL setzt das define SDL_BYTEORDER

- Anwendungsbeispiel:

```
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
    do_some_conversion();
#endif
```

Problem: Build-System

- Auf verschiedenen Betriebssystemen werden verschiedene Build- und Paketverwaltungssysteme eingesetzt
- Alle SDL Libraries müssen/sollten mit dem gleichen Compiler übersetzt werden wie das Zielprogramm

Lösung: Build-System

- Die SDL unterstützt viele Build-Systeme und Compiler; eine Einbindung in weitere Build-Systeme ist relativ leicht zu machen.
- Für die Verwaltung der Build-Systeme des Zielprogramms ist natürlich weiterhin der Entwickler des Programms zuständig

Demonstration von BS: LostPixels

- MacOSX mit Xcode
- Windows mit VisualC 6
- Linux mit GNU make

Fragen und Antworten

- Ihr stellt Fragen
- Ich habe hoffentlich Antworten

Links

- http://www.blinkensisters.org
- http://www.libsdl.org
- http://www.lua.org
- http://www.kekkai.org/roger/sdl/

ENDE

DANKE FÜRS ZUHÖREN!