**Soundausgabe:**

**Soundeffekt Media Dateien:**

Übersicht Websites:

<http://unity3diy.blogspot.de/2014/08/top-16-best-free-music-and-sound-effect.html>

<http://soundbible.com/tags-game.html>

Soundeffekt Generator:

Sfxr Synthesizer

Ideen:

**Heartbeat:**

Falls Leben nahe Ende, 2 Stufen

Slow\_HeartBeat-Mike\_Koenig-1853475164

Hearbeat\_2-Mike\_Koenig-143666461

**Regen:**

Wetter wird grafisch schlechter -> Regen setzt ein

Rain\_Background-Mike\_Koenig-1681389445.wav

**Gewinnton:**

Short\_triumphal\_fanfare-John\_Stracke-815794903

**Schlag:**

Strong\_Punch-Mike\_Koenig-574430706

**Powerup aufgenommen:**

Power\_Up\_Ray-Mike\_Koenig-800933783

**Sprung:**

Mario\_Jumping-Mike\_Koenig-989896458

Jump-SoundBible.com-1007297584

**Biertrinken:**

**Zeltsound:**

**Natursound:**

Nature Ambiance-SoundBible.com-1444637890

**Krankenwagensound:**

City Ambiance-SoundBible.com-1513196434

**Objektorientierte Analyse:**

**Fragen:**

Preload von Wave Dateien nötig (200 MB Ram)? -> keine Delays

Wie sehen Game Engine Ausgaben aus, kann Abstand der states zum Spieler mit abgespeichert werden -> Lautstärke variabel?

**Idee:**

Game speichert in game\_states alle 30 FPS die aktuellen Spielzustände in game\_states. (Beispiel player\_jump, item\_beerflying, player\_life\_low, player\_lifeverylow, player\_drunken, game\_won, game\_lost, item\_powerup, background\_outside, background\_tent, background\_mainmenu)

Audio vergleicht diese Spielzustände mit einem Dictionary welches alle Spielzustände aufzählt zu denen Audiofiles vorhanden sind und spielt diese Files solange ab bis der state aus game\_states gelöscht wird.

High Level Approach (1D Audio):

Klasse: **Audio**

Audio stellt public eine Funktion **play (states)** zur Verfügung, welche beginnt den Sound aller Keys im Dictionary states bei vorhandenen Sounddateien (getaudio) mit berechneter Lautstärke gleichzeitig abzuspielen (startoutput). Läuft einer der states bereits (check), so wird für diesen nichts unternommen. Zeigt ein Vergleich mit vorherigen audiostates, dass states nicht mehr vorhanden sind wird die wiedergabe der states gestoppt (stopoutput). (eine Art Main funktion)

Audio besitzt die private Funktion **stopoutput(states)** welche den Sound aller elemente des states dictionary stoppt

Audio besitzt die private Funktion **getaudio(states)** welche einen Vergleich der states mit den Keys aller vorhandenen Wave Dateien (audiofiles) durchführt und bei Vorhandensein von wavedateien die samples zu den zugehörigen keys mit Lautstärkeinformation in audiostates speichert.

Audio besitzt die private Funktion **check(states)** welche überprüft ob sound der states gerade abgespielt wird (keine Abstand Spieler information, Lautstärke eines Sounds immer gleich dem Aufruf)

Audio besitzt die private Funktion **startoutput(states)** welche die in audiostates gespeicherten samles in jeweils richtigen lautstärken in Endlosschleife auf der Hardware abspielt.

Low Level Approach (2D Audio):

Lautstärken variieren während Objekt sich im Raum bewegt ( beispielweise Bier fliegt auf Player zu, Japaner kommt näher und wird lauter). Effekte werden hinzugefügt (Beispiel besoffen -> Modulation des Ausgabessignals)

Idee: Alle audio\_states werden in jedem Frame mit aktueller Lautstärkinformation zusammengemischt (**mix**) ein gemeinsames Ausgabessignal (audio\_block) generiert. Auf das gemeinsame Signal können Effekte (**effect**) gelegt werden, welche von game\_states abhängen, danach Ausgabe von audio\_block (startoutput). Nach jedem Frame wird auf geänderte Lautstärkeinformation und geänderte audio\_states geprüft und ein neuer audio\_block zusammengefügt und ausgegeben.

von GUI gesendete Werte aktueller Ereignisse:

**game\_states**: Qhash Dicitionary

Keyname state: [Abstand state zum Spieler (float), usw. einfach ergänzbar]

Alle aktuell ablaufenden Zustände -> wenn Spiel oder ein Zustand beendet wird, Dicitonary Wert(e) löschen -> Audio abspielen stoppt

**- audio\_states**:

Keynamen aktuell gespielter audiofiles: [Abstand state zum Spieler (float), Samples Audio file]

**- audio\_files:**

Keyname vorhandener audiofiles: [Speicherort]

Von Game Engine aufruf:

Die Funktion play(actions) wird mit jedem Frame von der Game Engine mit dem QhasDictionary gamestates neu aufgerufen.

play (actions) prüft im Anschluss durch Vergleich mit der Liste „playactions“ und audiofiles ob der sound aller actionen aus actionen bereits abgespielt wird bzw. ein soundfile dafür vorlieg

liegt ein soundfile vor und es wird nicht abgespielt:

wird die funktion getsound(action) aufgerufen, welche das wave file von action einliest und in .

wird die funktion audiooutput() aufgerufen, welche das Audio Signal zu den anderen bereits ablaufenden hardware soundausgaben hinzufügt

**Control Audio**

- Dictionary audio\_states: Dicitionary mit Zeiger auf Audioobjekte

+Reset\_audio\_states()

+set\_audio\_states(game\_states)

+get\_audio\_

**Audio**

- double samplerate

- double samplenumber

- string filepath

- double volume

+Get\_filepath()

+Get\_samplerate()

+Get\_samplenumber()