

Lite-C FMOD wrapper DLL v 0.1

für die A7 game engine (Lite-C)

Copyright © 2006 – 2007 - Christian Behrenberg - All Rights Reserved

REFERENZ

Inhaltsverzeichnis

Einführung	3
Was ist FMOD?	3
Warum eine externe Audioengine?	3
Der Unterschied zwischen der FMOD.dll und der Lite-C wrapper-plugin DLL	3
Befehlsatzänderungen	4
Konstanten	4
Initialisierung & Beenden von FMOD	4
Unterstützte APIs und FMOD Features	5
Vorraussetzungen	5
Installation der Beispiele	6
Integration von FMOD in Ihr Projekt	6
Erklärungen und die dazugehörigen Beispiele	7
Streams	7
Samples	8
CD	8
Channels	9
System	9
FSOUND API Implementationsübersicht	10
Pre Initialization / Initialization / Enumeration Functions	10
Global Run-Time Update Functions	10
Global Run-Time Information Functions	10
Sample Functions	11
Channel Functions	11
3D Sound Functions	12
Stream Functions	12
CD Functions	13
DSP Functions	13
FX Functions	14
Recording Functions	14
FAQ	14
Fehlerbeseitigung	15
Lizenzbestimmungen	15
Development Information	17

Stand der Dokumentation: 2007-05-08

Einführung

Willkommen zu der Referenz der FMOD Wrapper DLL für die A7 game engine. Im Folgenden finden Sie eine Einführung in FMOD, den Unterschieden zum Audiosystem der A7 engine, dem workflow des Wrappers und der Befehlssatzübersicht.

Was ist „FMOD“?

FMOD ist eine plattformübergreifende Audioengine, die es Ihnen einfach ermöglicht, aktuelle Audio-Technologien in ihrer Software zu nutzen. Keine andere Audioengine bietet eine solch reichhaltige und aktuelle Plattformunterstützung. FMOD gibt es nicht nur für das Windows 32 bit System, sondern auch für 11 weitere Systeme, darunter fallen z.B. Windows 64bit (AMD64), Linux, Macintosh und alle auf dem Markt erhältlichen Spielekonsolen. FMOD wird und wurde von über hundert kommerziellen Spielen benutzt, zum Beispiel in „World of Warcraft“ oder „Flat Out“.

Warum eine externe Audioengine?

Die Audiofähigkeiten der A7 engine sind recht beschränkt. Das streaming der Dateien von der Festplatte ist sehr langsam: der Start verzögert sich merklich und friert auch kurz das Spiel ein – für professionelle Spieleproduktionen undenkbar. Abgesehen von pause und stop hat man auch keine Möglichkeiten, den stream einzeln anzusprechen, zu steuern und zu regeln. Es können keine Spezialeffekte auf streams oder samples gelegt werden. Man kann diese auch nicht analysieren: die aktuelle (Laufzeit-) Lautstärke, eine Frequenzanalyse, oder beat detection sind daher nicht abfragbar. File-tags kann man zum Beispiel auch nicht auslesen. Diese Features und noch viel mehr sind jetzt durch FMOD möglich!

Der Unterschied zwischen der FMOD.dll und der Lite-C wrapper-plugin DLL

Die FMOD Audioengine verwaltet Objektstrukturen, die nicht kompatibel zum prozeduralen ANSI-C Standard sind. Um mit den Funktionen arbeiten zu können, müssten Sie daher selbst diese Objekte verwalten. Da dies so nicht in Lite-C möglich ist, verwaltet die Wrapper DLL alle erzeugten Objekte (Streams, Samples, Channels, DSP units, etc.) selbst und vergibt ihnen eindeutige Identifikationsnummern, die Sie über den Wrapper mitgeteilt bekommen (ähnlich einem handle der media_* Anweisung der A7 engine). Über diese Ids können Sie dann ganz bequem auf diese Objekte zugreifen. Es ist ratsam, bei der Arbeit mit dem FMOD Wrapper nicht nur diese Referenz geöffnet zu haben, sondern auch die Referenz der FMOD engine selbst. Zunächst einmal lauten die Befehle etwas anders als die originalen FMOD Befehle, die Parameterlisten sind eventuell etwas modifiziert, es gibt zusätzliche Befehle, Einschränkungen, (noch) nicht unterstützte Befehle, Hinweise, etc. Deshalb ist es ratsam, aufmerksam beide Referenzen simultan zu lesen.

Befehlssatzänderungen

Sie können die FMOD Befehle nicht direkt ansprechen, dies übernimmt der Wrapper. Sie können dies veranlassen, indem Sie entsprechende Hüll-, bzw. Wrapper-Funktionen aufrufen. Diese nehmen die Parameter entgegen, verarbeiten diese eventuell dementsprechend und geben diese an die FMOD DLL weiter. Die Lite-C Wrapper-Funktionen beginnen alle mit einem vorangestellten „LC_“ vor dem Funktionsnamen eines FMOD Befehls. Beispiel: Wenn Sie den FMOD Befehl `FSOUND_Stream_Open()` zum öffnen eines Streams benutzen wollen, würden Sie `LC_FSOUND_Stream_Open()` in Lite-C benutzen.

Die wichtigste Neuerung ist die Verwaltung der FMOD Objekte. Streams, Samples und (eigene) DSP Units werden von der Wrapper DLL selbst verwaltet und gemanaged. Sie erhalten immer anstatt dieser Objekte integer Zahlen zurück, die eine eindeutige ID besitzen. Wenn Sie als einen FMOD Befehl benutzen wollen und ein Objekt übergeben müssen, so denken Sie daran, dem Wrapper anstatt dessen die (Integer-) ID mitzuteilen. Der Wrapper sucht sich dann das dementsprechende Objekt dann selbst heraus. Lesen Sie dazu die dementsprechenden Signaturen der Funktionen in der Header-Datei des Plugins durch!

Konstanten

Im Gegensatz zum Befehlssatz wurden die Konstanten, die in FMOD Verwendung finden, nicht abgeändert. Sie können Sie ganz normal benutzen. Beispiel: Sie wollen die Tracks einer CD grundsätzlich in Schleife spielen lassen. Sie würden folgenden Aufruf tätigen: `LC_FSOUND_CD_SetPlayMode(0, FSOUND_CD_PLAYLOOPED)`; In diesem Fall stellt die Konstante `FSOUND_CD_PLAYLOOPED` diejenige FMOD Konstante dar, die FMOD signalisiert, die Tracks des Standard CD-Laufwerks immer in Schleife zu spielen.

Initialisierung & Beenden von FMOD

Im Gegensatz zur FMOD engine, die Sie vielleicht in ein anderes Projekt, z.B. in C++, einbinden würden, müssen Sie nun keine Gedanken mehr an die Initialisierung verschwenden. Dies übernimmt die FMOD Wrapper DLL automatisch, wenn Sie den ersten Befehl ausführen. Allerdings ist es auch kein Problem, selber die FMOD engine über den Wrapper zu initialisieren. Der Wrapper erkennt dies und führt dann den gewünschten Befehl direkt aus. Beim Herunterfahren der game engine muss FMOD wieder beendet werden. Dies müssen Sie manuell einrichten, mehr dazu erfahren Sie im Kapitel „Integration von FMOD in Ihr Projekt“.

Unterstützte APIs und FMOD Features

Zur Zeit unterstützt der FMOD Wrapper nur eine Untermenge des kompletten Befehlssatzes von FMOD. Bisher werden nur die FSOUND API Befehle implementiert, der FMUSIC API Befehlssatz bisher noch nicht (nur bei Bedarf). Es werden folgenden Kernfeatures bedient:

- Initialisierung, Herunterfahren und Einstellungen
- Samples
- Streams (sowohl von der Festplatte, als auch aus dem Internet)
- Channels
- CD
- Frequenz Analyse

Diese Features sind (fast) vollständig implementiert, allerdings einige nicht. Bitte vergleichen Sie jedes Feature, welches Sie benutzen wollen, mit der Befehlssatzdokumentation des Wrappers und dem entsprechenden Eintrag in der FMOD Dokumentation!

Die folgenden Bereiche sind derzeit in Entwicklung (oder geplant) und noch nicht freigegeben. Bitte haben Sie dafür Verständnis. Dies betrifft die folgenden Bereiche ohne gesonderte Reihenfolge der Prioritäten:

- 3D Sound, multiple listener Objekte, Surround Sound
- runtime channel effects
- Samples, die aus einer WRS herausgelesen werden kann
- Aufnahme von Sounds (über das Mikrofon zum Beispiel)
- Syncpoints und Callbacks in Musikdateien; Systeminterne Callbacks
- FMUSIC API

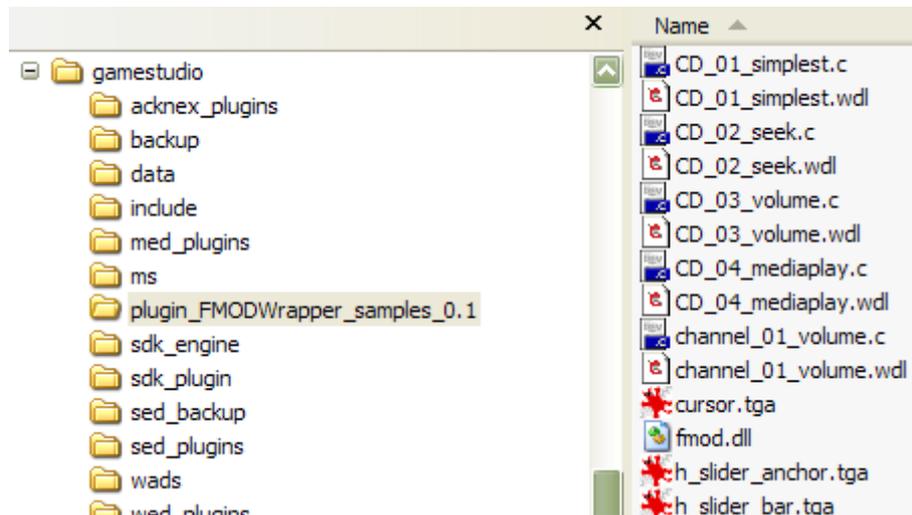
Voraussetzungen

Der FMOD Wrapper unterstützt nur Lite-C. Wenn Sie mit dem Wrapper arbeiten wollen, so müssen Sie folgende Kriterien erfüllen:

- Sie besitzen die A6 engine und sind Betatester: dann können Sie mit der Lite-C beta den Wrapper mit A6 benutzen (jedoch nur mit der beta-Version!) oder
- Sie besitzen die A7 engine und zeitgleich eine valide Lite-C Kopie oder
- Sie besitzen keine Kopie von Gamestudio. Laden Sie sich daher die kostenlose Variante von Lite-C auf der Conitec Download-Seite herunter: <http://www.conitec.de>
- Sie besitzen eine Kopie der FMOD 3.75 DLL. Die FMOD DLL wird NICHT mitgeliefert! Sie müssen die DLL selbst herunterladen, wenn Sie dies nicht bereits getan haben! Diese ist kostenlos auf der FMOD Internetseite herunterzuladen: <http://www.fmod.de>

Installation der Beispiele

Theoretisch können Sie das Archiv mit den Beispieldateien überall extrahieren. Wenn Sie jedoch die mitgelieferten RUN_*.bat Dateien benutzen wollen, um die Beispiele direkt auszuführen, so extrahieren Sie das Archiv einfach in ihr Gamestudio, bzw. Lite-C Verzeichnis. Zum Beispiel:



Wenn Sie dann die *.bat Dateien starten, wird das entsprechende Beispiel gestartet! Stellen Sie sicher, dass Sie den Wrapper UND die FMOD.dll in das Beispielverzeichnis kopieren (der Wrapper und die FMOD.dll sind im Archiv der Beispiele NICHT enthalten!).

Integration von FMOD in Ihr Projekt

Die Installation von FMOD und des Wrappers ist ein Kinderspiel. Im Folgenden eine Schritt für Schritt Anleitung mit Erläuterungen, wie Sie die FMOD Audioengine in ihr Projekt integrieren:

1. Kopieren Sie die Dateien fmod.dll, LC_FMOD_WRAP.dll und LC_FMOD_WRAP.h in ihr Projekt.
2. Inkludieren Sie die Headerdatei ein: `#include "LC_FMOD_WRAP.h"`; Diese ist unabhängig und VOR allen Aufrufen einzubinden! Platzieren Sie den include einfach direkt nach den acknex.h und default.c includes.

```

1  //- includes -----
2
3      #include <acknex.h>
4      #include <default.c>
5
6      #include "LC_FMOD_WRAP.h"; //FMOD plugin header file
7

```

3. Wenn die engine geschlossen wird, muss die FMOD engine wieder heruntergefahren werden – ansonsten erhalten Sie einen Speicherfehler! Dies können Sie nun auf verschiedene Arten lösen, je nachdem, wie Sie ihr Projekt verfassen, suchen Sie sich die

passendste Methode aus:

- Fahren Sie die Engine automatisch herunter (indem Sie den folgenden Befehl einfach hinschreiben, z.B. vor `sys_exit()`):

```
#ifdef FMOD_DLL
    LC_FSOUND_Close();
#endif
```

- Sie können die Engine automatisch über den `on_exit` event herunterfahren lassen:
 - Wenn Sie noch keinen `on_exit` event benutzen, dann schreiben Sie eine void Funktion, in der Sie FMOD herunterfahren und setzen die Funktion als `on_exit` event:

```
void sysExit_event ()
{
    #ifdef FMOD_DLL
        LC_FSOUND_Close();
    #endif
}

void main ()
{
    //...
    on_exit = sysExit_event;
}
```

- Wenn Sie bereits einen `on_exit` Event besitzen, kopieren Sie sich einfach den obigen Code hinein!
- Alternativ können Sie auch direkt die Anweisung dem `on_exit` event zuweisen: `on_exit = LC_FSOUND_Close();`

Um die Initialisierung müssen Sie sich nicht kümmern: Wenn Sie das erste Mal einen relevanten FMOD Befehl ausführen, wird die engine *automatisch* hochgefahren – und nicht sofort beim Systemstart. **Es ist nur wichtig, dass Sie die engine auch wieder herunterfahren!**

Damit wäre FMOD und der Wrapper erfolgreich in ihr Projekt eingebunden. Wenn Sie sich zum ersten Mal mit FMOD befassen, so empfehle ich Ihnen, sich zunächst durch die Tutorials durchzuarbeiten.

Erklärungen und die dazugehörigen Beispiele

Im Folgenden werden alle wichtigen Funktionsgruppen von FMOD thematisiert und auf die Beispiele verwiesen.

Streams

Bei allen Musikstücken die groß sind und/oder selten gespielt werden, ist es sinnvoll die Musikdatei als sogenannten Stream abzuspielen. Ein Stream ist ein Strom der zwischen der Datei und dem Programm geöffnet wird. Dabei wird dann, während die Musikdatei abgespielt wird, immer ein kleiner Teil an Daten in den Speicher geladen und die alten Teile wieder entfernt. Somit ist der Speicherverbrauch viel geringer, als wenn man die ganze Datei als Sample in den Speicher lädt – dies macht sich besonders bei langen Musikstücken bezahlt, oder Sprecherkommentaren (voice overs). In FMOD kann man Streams zu Dateien auf der Festplatte, Datenträgern (CD, DVD) und Dateien im Internet öffnen.

stream_01_simplest.c	Hier wird demonstriert, wie man auf einfachste Weise einen Stream initialisiert und abspielt.
stream_02_seek.c	Mit FMOD kann man auch innerhalb eines Streams hin und her springen (engl. „to seek“). Dieses Beispiel zeigt, wie es geht.
stream_03_stereo.c	Während ein Stream läuft, ist es auf dem Stereokanal zur Laufzeit unterschiedlich laut und leise. In diesem Beispiel wird gezeigt, wie man diese sog. „levels“ abfängt und auswertet.
stream_04_mediaplay.c	Hier wird ein einfacher Mediaplayer realisiert. Über eine Play, Pause und Stop Schaltfläche kann man den Stream mit simpelsten Mitteln steuern.
stream_05_information.c	Man kann statische als auch Laufzeit-gebundene Informationen eines Streams abfragen. Dieses Beispiel zeigt es anhand der wichtigsten Werte.
stream_06_tags.c	Es ist Standard bei vielen Mediaplayern eine Information bezüglich des Interpreten und des Song-Titels anzuzeigen, indem die Musikdatei ausgewertet wird. Dies kann man auch mit FMOD und dieses Beispiel zeigt es!
stream_07_spectrum.c	Eines der wirklich aufregendsten Features eines modernen Mediaplayer ist es, ein Spektrum des gerade laufenden Liedes zwecks der Visualisierung der Musik anzuzeigen. FMOD kann dies auch und mithilfe einer einfachen Abfrage kann man selbst solche Spektrogramme zeichnen oder auswerten (zum Beispiel zur Erkennung eines beats!)

Samples

Samples sind Musikdateien, die komplett in den Speicher geladen (extrahiert) werden. Aufgrund Speicherplatzbelegung ist es nur dann sinnvoll, Musik als Sample in den Speicher zu laden, wenn die Datei kurz ist und häufig abgespielt werden soll (typische Effekte wie Schüsse, Explosionen, etc.).

sample_01_simplest.c	Dieses Beispiel lädt ein Sample und über einen Knopfdruck spielen Sie es ab.
sample_02_loop.c	Bei Programmstart wird ein Sample geladen und in Schleife abgespielt.

CD

Mit FMOD kann man auch Musikstücke von einer Audio CD (bzw. dem Audio Sektor einer hybriden CD) abspielen. Die FMOD engine besitzt eine eigene Routine, um Audio CDs abzuspielen. Im Gegensatz zu konventionellen Mediaplayern (wie z.B. dem Windows Media Player), wird die CD nicht jederzeit nach dessen Status abgefragt. Bei Spielen würde dies zu stotternden Framerates und Störungen im Spielablauf führen. Während ein Lied abgespielt wird, fragt FMOD zum Beispiel das Laufwerk niemals ab, was zu einer 0%-tigen CPU Auslastung führt und damit gegenüber konventionellen Lösungen überlegen ist.

CD_01_simplest.c	In dieser Demo wird gezeigt, wie man auf einfachste Weise den ersten Track einer eingelegten CD im primären Disc-Laufwerk abspielen kann.
CD_02_seek.c	Wie bei den Streams, kann man auch in einem Track einer CD hin und her springen. Dies wird in dieser Demo realisiert.
CD_03_volume.c	Dieses kleine Beispiel zeigt, wie man die Lautstärke der CD ändern kann.
CD_04_mediaplay.c	Hier wird wie bei dem entsprechenden Stream-Beispiel ein kleiner Mediaplayer gezeigt. Der kleine Unterschied bei der CD-Version ist das man zum nächsten oder zum vorherigen Lied wechseln kann.

Channels

Anstatt nun Streams, Samples usw. einzeln zu betrachten, kann man diese nun in Channels, also sozusagen „Spuren“ aufteilen oder man könnte auch sagen: gruppieren. Man kann dann einen channel anders mischen als andere Channels (Lautstärke ändern, Effekte hinzufügen, etc.). Beispiel: Wenn der Spieler verwundet ist, senkt man die Lautstärke der Umgebungsgeräusche stark herab und erhöht die eigenen Geräusche sehr stark und fügt einen leichten Echo-Effekt hinzu, um ein eindringlicheres Spielgefühl zu simulieren (z.B. in Call of Duty). Dies würde man nun erreichen indem man den einen channel lauter macht und einen zweiten leiser. Man hat keine Auskunft darüber, welche Streams und/oder Samples dort drin arrangiert sind, aber der Effekt ist der gleiche als wenn man es wüsste und alle Streams und Samples einzeln anfasst. Man muss dann nur beim Erzeugen der Audio-Objekte darauf achten, dass sie im dementsprechenden Channel initialisiert werden.

channel_01_volume.c	Diese Demonstration zeigt, wie man ganz einfach einen Stereokanal nach links oder rechts „verschieben“ kann (engl.: „panning“) und die Lautstärke, bzw. die Frequenz ändert.
---------------------	--

System

system_01_information.c	In diesem Beispiel werden einige wichtige und auch interessante Daten über die FMOD engine angezeigt.
-------------------------	---

FSOUND API Implementationsübersicht

Im Folgenden wird der Befehlssatz der Wrapper-DLL aufgeführt. Die Kategorien sind entsprechend der FMOD Referenz sortiert, damit man einen einfachen Abgleich durchführen kann. Die Befehle sind farblich codiert, um hervorzuheben, ob sie implementiert sind oder nicht, bzw. welche Besonderheit sie haben. **Die Übersicht geht nicht auf die Signaturen (Rückgabewert, Parameterliste) der Befehle ein!** (Bitte lesen Sie sich diesbezüglich die Signaturen der Prototypen in der Headerdatei des Plugins durch und vergleichen Sie diese mit der jeweiligen Signatur des entsprechenden FMOD Befehls)

Legende:	Function	unterstützte Funktion
	Function	(noch) nicht unterstützte Funktion
	Function	substituierte Funktion
	<i>Function</i>	Ersatzfunktion
	Function	Zusätzliche Funktion, die es NICHT in FMOD gibt, aber im Wrapper
	Function	Aus technischen Gründen wird diese Funktion niemals implementiert

Pre Initialization / Initialization / Enumeration Functions

LC_FSOUND_Close
~~LC_FSOUND_File_SetCallbacks~~
 LC_FSOUND_Init
 LC_FSOUND_SetBufferSize
 LC_FSOUND_SetDriver
 LC_FSOUND_SetHWND
 LC_FSOUND_SetMaxHardwareChannels
~~LC_FSOUND_SetMemorySystem~~
 LC_FSOUND_SetMinHardwareChannels
 LC_FSOUND_SetMixer
 LC_FSOUND_SetOutput

Global Run-Time Update Functions

LC_FSOUND_SetPanSeparation
 LC_FSOUND_SetSFXMasterVolume
 LC_FSOUND_SetSpeakerMode
 LC_FSOUND_Update

Global Run-Time Information Functions

LC_FSOUND_GetCPUUsage
 LC_FSOUND_GetChannelsPlaying
 LC_FSOUND_GetDriver
LC_FSOUND_GetDriverCaps
 anstatt dessen diese Funktionen verwenden:
 LC_FSOUND_GetDriverCaps_HARDWARE

LC_FSOUND_GetDriverCaps_EAX2
LC_FSOUND_GetDriverCaps_EAX3
LC_FSOUND_GetDriverName
LC_FSOUND_GetError
LC_FSOUND_GetMaxSamples
LC_FSOUND_GetMaxChannels
LC_FSOUND_GetMemoryStats
LC_FSOUND_GetNumDrivers
LC_FSOUND_GetNumHWChannels
alternativ kann man auch diese Funktionen verwenden:
LC_FSOUND_GetNumHWChannels_num2d
LC_FSOUND_GetNumHWChannels_num3d
LC_FSOUND_GetNumHWChannels_total
LC_FSOUND_GetOutput
LC_FSOUND_GetOutputHandle
LC_FSOUND_GetOutputRate
LC_FSOUND_GetSFXMasterVolume
LC_FSOUND_GetVersion

Sample Functions

LC_FSOUND_Sample_Alloc
LC_FSOUND_Sample_Free
~~LC_FSOUND_Sample_Get~~
LC_FSOUND_Sample_GetDefaults
LC_FSOUND_Sample_GetDefaultsEx
LC_FSOUND_Sample_GetLength
LC_FSOUND_Sample_GetLoopPoints
LC_FSOUND_Sample_GetMinMaxDistance
LC_FSOUND_Sample_GetMode
LC_FSOUND_Sample_GetName
LC_FSOUND_Sample_Load
(Hinweis: int index wird in der Signatur unterstützt, zur Zeit aber mit FSOUND_FREE überschrieben.)
~~LC_FSOUND_Sample_Lock~~
LC_FSOUND_Sample_SetDefaults
LC_FSOUND_Sample_SetDefaultsEx
LC_FSOUND_Sample_SetMaxPlaybacks
LC_FSOUND_Sample_SetMinMaxDistance
LC_FSOUND_Sample_SetMode
~~LC_FSOUND_Sample_SetLoopPoints~~
~~LC_FSOUND_Sample_Unlock~~
~~LC_FSOUND_Sample_Upload~~

Channel Functions

LC_FSOUND_PlaySound
~~LC_FSOUND_PlaySoundEx~~
LC_FSOUND_StopSound
LC_FSOUND_SetFrequency
LC_FSOUND_SetLoopMode
LC_FSOUND_SetMute
LC_FSOUND_SetPan
LC_FSOUND_SetPaused
LC_FSOUND_SetPriority

LC_FSOUND_SetReserved
LC_FSOUND_SetSurround
LC_FSOUND_SetPriority
LC_FSOUND_SetVolume
LC_FSOUND_SetVolumeAbsolute
[LC_FSOUND_SetVolumeAbsolute_global](#)
LC_FSOUND_GetVolume
LC_FSOUND_GetAmplitude
~~LC_FSOUND_3D_SetAttributes~~
~~LC_FSOUND_3D_SetMinMaxDistance~~
LC_FSOUND_SetCurrentPosition
LC_FSOUND_GetCurrentPosition
LC_FSOUND_GetCurrentSample
LC_FSOUND_GetCurrentLevels
LC_FSOUND_GetFrequency
LC_FSOUND_GetLoopMode
LC_FSOUND_GetMixer
LC_FSOUND_GetMute
LC_FSOUND_GetNumSubChannels
LC_FSOUND_GetPan
LC_FSOUND_GetPaused
LC_FSOUND_GetPriority
LC_FSOUND_GetReserved
LC_FSOUND_GetSubChannel
LC_FSOUND_GetSurround
LC_FSOUND_IsPlaying
~~LC_FSOUND_3D_GetAttributes~~
~~LC_FSOUND_3D_GetMinMaxDistance~~

3D Sound Functions

~~LC_FSOUND_3D_Listener_GetAttributes~~
~~LC_FSOUND_3D_Listener_SetAttributes~~
~~LC_FSOUND_3D_Listener_SetCurrent~~
~~LC_FSOUND_3D_SetDistanceFactor~~
~~LC_FSOUND_3D_SetDopplerFactor~~
~~LC_FSOUND_3D_SetRolloffFactor~~

Stream Functions

~~LC_FSOUND_Stream_AddSyncPoint~~
LC_FSOUND_Stream_Close
~~LC_FSOUND_Stream_Create~~
~~LC_FSOUND_Stream_CreateDSP~~
~~LC_FSOUND_Stream_DeleteSyncPoint~~
LC_FSOUND_Stream_FindTagField
LC_FSOUND_Stream_GetLength
LC_FSOUND_Stream_GetLengthMs
[LC_FSOUND_Stream_GetLengthSecs](#)
LC_FSOUND_Stream_GetMode
LC_FSOUND_Stream_GetNumSubStreams
LC_FSOUND_Stream_GetNumSyncPoints
LC_FSOUND_Stream_GetNumTagFields
LC_FSOUND_Stream_GetOpenState

~~LC_FSOUND_Stream_GetPosition~~
~~LC_FSOUND_Stream_GetSample~~
~~LC_FSOUND_Stream_GetSyncPoint~~
~~LC_FSOUND_Stream_GetSyncPointInfo~~
~~LC_FSOUND_Stream_GetTagField~~
~~LC_FSOUND_Stream_GetTime~~
~~LC_FSOUND_Stream_GetTime_secs~~
~~LC_FSOUND_Stream_Net_GetBufferProperties~~
~~LC_FSOUND_Stream_Net_GetLastServerStatus~~
~~LC_FSOUND_Stream_Net_GetStatus~~
~~LC_FSOUND_Stream_Net_SetBufferProperties~~
~~LC_FSOUND_Stream_Net_SetMetadataCallback~~
~~LC_FSOUND_Stream_Net_SetProxy~~
~~LC_FSOUND_Stream_Open~~
~~LC_FSOUND_Stream_Play~~
~~LC_FSOUND_Stream_PlayEx~~
~~LC_FSOUND_Stream_SetBufferSize~~
~~LC_FSOUND_Stream_SetEndCallback~~
~~LC_FSOUND_Stream_SetLoopCount~~
~~LC_FSOUND_Stream_SetLoopPoints~~
~~LC_FSOUND_Stream_SetMode~~
~~LC_FSOUND_Stream_SetPosition~~
~~LC_FSOUND_Stream_SetSubStream~~
~~LC_FSOUND_Stream_SetSubStreamSentence~~
~~LC_FSOUND_Stream_SetSyncCallback~~
~~LC_FSOUND_Stream_SetTime~~
~~LC_FSOUND_Stream_SetTime_secs~~
~~LC_FSOUND_Stream_Stop~~

CD Functions

~~LC_FSOUND_CD_GetNumTracks~~
~~LC_FSOUND_CD_GetPaused~~
~~LC_FSOUND_CD_GetTrack~~
~~LC_FSOUND_CD_GetTrackLength~~
~~LC_FSOUND_CD_GetTrackLength_secs~~
~~LC_FSOUND_CD_GetTrackTime~~
~~LC_FSOUND_CD_GetTrackTime_secs~~
~~LC_FSOUND_CD_GetVolume~~
~~LC_FSOUND_CD_OpenTray~~
~~LC_FSOUND_CD_Play~~
~~LC_FSOUND_CD_SetPaused~~
~~LC_FSOUND_CD_SetPlayMode~~
~~LC_FSOUND_CD_SetTrackTime~~
~~LC_FSOUND_CD_SetVolume~~
~~LC_FSOUND_CD_Stop~~

DSP Functions

~~LC_FSOUND_DSP_ClearMixBuffer~~
~~LC_FSOUND_DSP_Create~~
~~LC_FSOUND_DSP_Free~~
~~LC_FSOUND_DSP_SetActive~~
~~LC_FSOUND_DSP_GetActive~~

LC_FSOUND_DSP_GetBufferLength
LC_FSOUND_DSP_GetBufferLengthTotal
LC_FSOUND_DSP_SetPriority
LC_FSOUND_DSP_GetPriority
~~LC_FSOUND_DSP_GetClearUnit~~
~~LC_FSOUND_DSP_GetClipAndCopyUnit~~
~~LC_FSOUND_DSP_GetMusicUnit~~
~~LC_FSOUND_DSP_GetSFXUnit~~
~~LC_FSOUND_DSP_GetFFTUnit~~
Ersatzfunktionen:
LC_FSOUND_DSP_OpenFFTUnit
LC_FSOUND_DSP_CloseFFTUnit
LC_FSOUND_DSP_GetSpectrum
~~LC_FSOUND_DSP_MixBuffers~~

FX Functions

~~LC_FSOUND_FX_Disable~~
~~LC_FSOUND_FX_Enable~~
~~LC_FSOUND_FX_SetChorus~~
~~LC_FSOUND_FX_SetCompressor~~
~~LC_FSOUND_FX_SetDistortion~~
~~LC_FSOUND_FX_SetEcho~~
~~LC_FSOUND_FX_SetRanger~~
~~LC_FSOUND_FX_SetGargle~~
~~LC_FSOUND_FX_Set3DL2Reverb~~
~~LC_FSOUND_FX_SetParamEQ~~
~~LC_FSOUND_FX_SetWavesReverb~~

Recording Functions

~~LC_FSOUND_Reverb_SetProperties~~
~~LC_FSOUND_Reverb_GetProperties~~
~~LC_FSOUND_Reverb_SetChannelProperties~~
~~LC_FSOUND_Reverb_GetChannelProperties~~

FAQ

- **Q:** Der Wrapper funktioniert nur im Zusammenhang mit der A7 engine. Gibt es auch eine Version für die A6 game engine in Verbindung mit der C-Script scripting language? - **A:** Der Wrapper funktioniert nur mit Lite-C. Eine Portierung auf C-Script ist zur Zeit nicht geplant, weil die Entwicklung der Lite-C Version noch nicht einmal fertiggestellt ist. Sollten Sie dringends eine Portierung eines bestimmten Bereichs (z.B. das streaming) haben, so setzen Sie sich mit mir in Verbindung.
- **Q:** Kann ich die Lite-C Variante des Wrappers auch in meinem A6 Projekt verwenden? - **A:** Jein. Wenn Sie Betatester sind, haben Sie auch Zugriff auf eine Lite-C beta, sodass Sie den Wrapper auch mit einer A6 – betaversion benutzen könnten. Sie könnten auch die A6

engine in ein C++ Programm binden, allerdings macht es dann keinen Sinn, den Wrapper zu benutzen, da Sie dann auch vollen Zugriff auf die „echte“ Version von FMOD hätten.

Fehlerbeseitigung

- **Q:** Wenn ich meinen ersten FMOD Befehl ausführe, dann stürzt immer die engine ab. Kann es sein, dass die Wrapper DLL einen Fehler hat? - **A:** Dies kann natürlich sein, aber wenn immer beim ersten Befehl die engine abstürzt, dann liegt das wohl daran, dass entweder die FMOD.dll und/oder die Wrapper – DLL nicht gefunden worden ist. Stellen Sie sicher, dass ihr plugin-dir richtig eingestellt ist. Schauen Sie sich die zusätzlichen .wdl Dateien der Beispielpprogramme an - dort wird dies genau gezeigt.
- **Q:** Wenn ich das Spiel beende, dann erhalte ich immer eine Fehlermeldung, dass die Acknex auf einen ungültigen Speicherbereich zugreift. Wieso passiert das? - **A:** Aller Voraussicht nach haben Sie vergessen, die FMOD engine herunterzufahren. Fügen Sie die entsprechende Anweisung in Ihren Programmcode ein (siehe: Integration in Ihr Projekt).
- **Q:** Ich fahre bereits FMOD herunter, wenn ich das Spiel beende, trotzdem erhalte ich diese Fehlermeldung. - **A:** Achten Sie darauf, dass Sie nach dem Herunterfahren keine Zugriffe mehr durchführen! Wenn Sie immer noch Probleme haben, wenden Sie sich bitte an mich.

Lizenzbestimmungen

It is at the express consent of the parties that the present agreement be written in English. C'est à la demande expresse des parties que cette convention soit rédigée en anglais.

READ THE TERMS OF THIS AGREEMENT ("AGREEMENT") CAREFULLY BEFORE USING THE SOFTWARE PACKAGE.

The subject of this license agreement is the computer program "Lite-C FMOD wrapper DLL", the program description and the instruction manual, as well as other associated written and electronic materials, in the following termed as software. Christian Behrenberg wants to point out the fact that it is not possible at the state of the art to provide computersoftware in such a way, that it works in all applications and combinations error free. The subject of this contract is therefore only a software, which is in the sense of the programs description and the instruction manual principlly useful. **BY USING THIS SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT.**

Christian Behrenberg is both author and owner of the software, as well of in this software used algorithms and procedures. Christian Behrenberg keeps the legal claim and tenure at the software. You accept that this granted license is not a sale of the software and that this license does'nt entitle you to assert a claim on patents, duplication, industry secrets, registered trademarks or on other rights.

You are allowed to copy, distribute and use the software freely (freeware principle), as far as the following regulations are considered: The software has to be distributed in the EXE- or ZIP archives which are being provided by Christian Behrenberg and you may not alter or add a file(s) in the distributed contents. The licensee may use the software in commercial products. If the licensee uses the software and thereby publishes a product with it, you have to inform Christian Behrenberg and leave him a free copy of that product. Furthermore the licensee has to add Christian Behrenberg as developer to his documentation of the product specified as "additional audio engineering", "plugin programming", or as "freelance audio engineer", "freelance audio programmer" or "freelance game developer". Optionally the licensee is permitted to print a reference to the internet-website of Christian Behrenberg in the documentation of the product (<http://www.christian-behrenberg.de>). The licensee is bound to the license agreements of all external software, which he uses automatically by the use of this software. Christian Behrenberg is not responsible for license injuries of third parties by the inappropriate use of this software. The licensee accepts all license agreements of external software with this license agreement. This concerns the "FMOD audio engine" in this case, copyright © 2001-2002 Firelight Technologies, Pty, Ltd. All rights reserved.

YOU MAY NOT, AND YOU MAY NOT PERMIT OTHERS TO REVERSE ENGINEER, DECOMPILE, DECODE, DECRYPT, DISASSEMBLE, OR IN ANY WAY DERIVE SOURCE CODE FROM THE SOFTWARE; YOU MAY NOT, AND YOU MAY NOT PERMIT OTHERS TO SELL, RENT, LEASE OR OTHERWISE EXPLOIT THE SOFTWARE.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL CHRISTIAN BEHRENBURG BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL , INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF CHRISTIAN BEHRENBURG HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THE SOFTWARE AND ACCOMPANYING WRITTEN MATERIALS ARE PROVIDED ON AN "AS IS" BASIS, WITHOUT ANY WARRANTIES OF ANY KIND, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY CHRISTIAN BEHRENBURG SHALL CREATE A WARRANTY, OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY, AND YOU MAY NOT RELY ON ANY SUCH INFORMATION OR ADVICE. CHRISTIAN BEHRENBURG DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF USE, OF THE SOFTWARE OR WRITTEN MATERIALS IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE, AND THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. IF THE SOFTWARE OR WRITTEN MATERIALS ARE DEFECTIVE, YOU, AND NOT CHRISTIAN BEHRENBURG ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION OTHER THAN EXPRESSLY DESCRIBED ABOVE.

Any action related to this Agreement will be governed by German law. No choice of law rules of any jurisdiction will apply. You acknowledge that you have read this Agreement, understand it, and agree to be bound by its terms and conditions.

Development Information

Die FMOD Wrapper DLL wird von Christian Behrenberg entwickelt. Nähere Informationen finden Sie auf der Website <http://www.christian-behrenberg.de> für Informationen, News, Korrekturen, Ankündigungen und neue releases.

Copyright © 2006 – 2007 - Christian Behrenberg - All Rights Reserved