Chuck's FMOD EX Guide

Introduction

I'm using "FMOD Ex", C++, and Visual Studio 2005.

Download

Go to http://www.fmod.org/index.php/download and download the Windows 32 bit version. Install.

Visual Studio Configuration

Click Menu, Tools, Options, Projects and Solutions, VC++ Directories.

Select "Include Files" from "Show directories for". Add a reference to the api location. On my computer it is C:\Program Files\FMOD SoundSystem\FMOD Programmers API Win32\api\inc

Select "Library Files" from "Show directories for". Add a reference to the api location. On my computerit is C:\Program Files\FMOD SoundSystem\FMOD Programmers API Win32\api\lib

I copied the FMODEX.DLL file from the API folder to the project folder with the VS2005 solution.

Select Project, Properties, Configuration Properties, Linker, Command Line: Add "fmodex_vc.lib".

More Resources

http://www.fmod.org/files/FMODEducationalResources.zip

Open VS and Create Empty Console Application

```
#include <iostream>
#include <cstdlib>
#include <fmod.h>
#include <fmod.hpp>
#include <windows.h>

using namespace std;

int main(){

// create system
FMOD::System* system;
FMOD_RESULT  result = FMOD::System_Create(&system);

// load sound
FMOD::Sound* sound;
system->setOutput(FMOD_OUTPUTTYPE_DSOUND );
system->init(32, FMOD_INIT_NORMAL, 0);
```

```
result = system->createSound("sounds/forest.wav", FMOD_LOOP_NORMAL,NULL,
&sound);

// play sound
FMOD::Channel* channel = 0;
result = system->playSound(FMOD_CHANNEL_FREE, sound, false, &channel);
system->update();
if(result == FMOD_OK)
::MessageBox(0, "werwer", "wer", 0);

// release resources
result = sound->release();
result = system->release();

  return 0;
}
```

Simple Keyboard Control

Number keys 1, 2, 3, and 4 start, stop, pause, and unpause the sound file.

```
#include <iostream>
#include <cstdlib>
#include <fmod.h>
#include <fmod.hpp>
#include <windows.h>

using namespace std;

bool keyDown(int key)
{
  if(GetAsyncKeyState(key) == 0) //key is not pressed, return false
    return false;
  else                           //key IS pressed, return true
    return true;
}

int main(){
      bool quit = false;

      //declare variable for FMOD system object
      FMOD::System* system;

      //allocate memory for the FMOD system object
      FMOD_RESULT  result = FMOD::System_Create(&system);

      //initialize the FMOD system object
      system->init(32, FMOD_INIT_NORMAL, NULL);

      //declare variable for the sound object
      FMOD::Sound* sound;

      //created sound object and specify the sound
```

```
      result = system->createSound("sounds/ambient1.mp3",
FMOD_LOOP_NORMAL,NULL, &sound);

      // play sound - 1st parameter can be combined flags (| separator)
      FMOD::Channel* channel = 0;

    bool pauseSound = false;

      while(!quit){
            //start sound
            if(keyDown('1') == true){
                  channel->isPlaying(&pauseSound);
                  if(pauseSound == false)
                    result = system->playSound(FMOD_CHANNEL_FREE, sound,
false, &channel);
            }

            //stop sound
            if(keyDown('2') == true){
                  channel->stop();
            }

            //pause sound
            if(keyDown('3') == true){
                  channel->setPaused(true);
            }

            //unpause sound
            if(keyDown('4') == true){
                  channel->setPaused(false);
            }

            if(keyDown(VK_ESCAPE))
              quit = true;

      system->update();
      }
      // release resources
      result = sound->release();
      result = system->close();
      result = system->release();

  return 0;
}
```

Keyboard Control (Two Sounds)

```
#include <iostream>
#include <cstdlib>
#include <fmod.h>
#include <fmod.hpp>
#include <windows.h>

using namespace std;

bool keyDown(int key)
```

```
{
  if(GetAsyncKeyState(key) == 0) //key is not pressed, return false
    return false;
  else                           //key IS pressed, return true
    return true;
}

int main(){
      bool quit = false;

      //declare variable for FMOD system object
      FMOD::System* system;

      //allocate memory for the FMOD system object
      FMOD_RESULT  result = FMOD::System_Create(&system);

      //initialize the FMOD system object
      system->init(32, FMOD_INIT_NORMAL, NULL);

      //declare variable for the sound object
      FMOD::Sound* sound[2];

      //created sound object and specify the sound
      result = system->createSound("sounds/ambient1.mp3",
FMOD_LOOP_NORMAL,NULL, &sound[0]);
      result = system->createSound("sounds/foreman.mp3",
FMOD_LOOP_NORMAL,NULL, &sound[1]);

      // play sound - 1st parameter can be combined flags (| separator)
      FMOD::Channel* channel[2];
      channel[0] = 0;
      channel[1] = 0;

    bool pauseSound[2];
      pauseSound[0] = false;
      pauseSound[1] = false;

      while(!quit){
            //start sound
            if(keyDown('1') == true){
                  channel[0]->isPlaying(&pauseSound[0]);
                  if(pauseSound[0] == false)
                    result = system->playSound(FMOD_CHANNEL_FREE, sound[0],
false, &channel[0]);
            }

            //stop sound
            if(keyDown('2') == true){
                  channel[0]->stop();
            }

            //pause sound
            if(keyDown('3') == true){
                  channel[0]->setPaused(true);
            }

            //unpause sound
```

```
            if(keyDown('4') == true){
                  channel[0]->setPaused(false);
            }


            if(keyDown('5') == true){
                  channel[1]->isPlaying(&pauseSound[1]);
                  if(pauseSound[1] == false)
                    result = system->playSound(FMOD_CHANNEL_FREE, sound[1],
false, &channel[1]);
            }

            //stop sound
            if(keyDown('6') == true){
                  channel[1]->stop();
            }

            //pause sound
            if(keyDown('7') == true){
                  channel[1]->setPaused(true);
            }

            //unpause sound
            if(keyDown('8') == true){
                  channel[1]->setPaused(false);
            }

            if(keyDown(VK_ESCAPE))
              quit = true;

      system->update();
      }
      // release resources
      result = sound[0]->release();
      result = sound[1]->release();
      result = system->close();
      result = system->release();

  return 0;
}
```

Audio Singleton Class (Two Sounds Hardcoded)

CAudioManager.h File

```
//CAudioManager.h
#ifndef CAUDIOMANAGER_H
#define CAUDIOMANAGER_H

#include <iostream>
#include <cstdlib>
#include <windows.h>
#include <string>
#include <cstring>
#include <vector>
#include <fmod.h>
```

```
#include <fmod.hpp>
#include "fmod_errors.h"

//file loading
struct SOUND_FILE_DATA{
  int soundID;
  std::string filename;
  std::string description;
};

class CAudioManager{
public:
  static CAudioManager *Instance();
  void PlaySound(int num);
  void StopSound(int num);
  void PauseSound(int num);
  void UnpauseSound(int num);
  void Update();

protected:
  CAudioManager();
  ~CAudioManager();

private:
  static CAudioManager *pInstance;

FMOD::System* system;
      //declare variable for the sound object
      FMOD::Sound* sound[2];

      // play sound - 1st parameter can be combined flags (| separator)
      FMOD::Channel* channel[2];

    bool pauseSound[2];
};

#endif
```

CAudioManager.cpp File

```
#include "..\include\CAudioManager.h"

CAudioManager *CAudioManager::pInstance = 0;

CAudioManager *CAudioManager::Instance(){
  if(CAudioManager::pInstance == 0)
    CAudioManager::pInstance = new CAudioManager;

  //else
  return CAudioManager::pInstance;
}

//initialize sound system
CAudioManager::CAudioManager(){

      //allocate memory for the FMOD system object
      FMOD_RESULT  result = FMOD::System_Create(&system);
```

```
	if(result == FMOD_OK){
		::MessageBox(0, "FMOD System created OK!", "Success!", 0);
	}

	//initialize the FMOD system object
	system->init(32, FMOD_INIT_NORMAL, NULL);
	if(result == FMOD_OK){
		::MessageBox(0, "FMOD System initialized!", "Success!", 0);
	}

	//created sound object and specify the sound
	result = system->createSound("sounds/ambient1.mp3",
FMOD_LOOP_NORMAL,NULL, &sound[0]);
	result = system->createSound("sounds/foreman.mp3",
FMOD_LOOP_NORMAL,NULL, &sound[1]);

	// play sound - 1st parameter can be combined flags (| separator)
	channel[0] = 0;
	channel[1] = 0;

    //bool pauseSound[2];
	pauseSound[0] = false;
	pauseSound[1] = false;
}

//close sound system if it initialized correctly
CAudioManager::~CAudioManager(){
	FMOD_RESULT  result;
	// release resources
	result = sound[0]->release();
	result = sound[1]->release();
	result = system->close();
	result = system->release();
}

void CAudioManager::PlaySound(int num){
	FMOD_RESULT  result;
	channel[num]->isPlaying(&pauseSound[num]);
	if(pauseSound[num] == false)
	  result = system->playSound(FMOD_CHANNEL_FREE, sound[num], false,
&channel[num]);
}

void CAudioManager::StopSound(int num){
	channel[num]->stop();
}

void CAudioManager::PauseSound(int num){
	channel[num]->setPaused(true);
}

void CAudioManager::UnpauseSound(int num){
	channel[num]->setPaused(false);
}

void CAudioManager::Update(){
	system->update();
```

```
}
```

Main.cpp Test File

```
#include <iostream>
#include <cstdlib>
#include <fmod.h>
#include <fmod.hpp>
#include <windows.h>
#include "../include/CAudioManager.h"

using namespace std;

bool keyDown(int key)
{
  if(GetAsyncKeyState(key) == 0) //key is not pressed, return false
    return false;
  else                           //key IS pressed, return true
    return true;
}

int main(){
      bool quit = false;

      CAudioManager *pAudio = CAudioManager::Instance();

      while(!quit){
            if(keyDown('1') == true){
                  pAudio->PlaySound(0);
            }
            if(keyDown('2') == true){
                  pAudio->StopSound(0);
            }
            if(keyDown(VK_ESCAPE))
              quit = true;

            pAudio->Update();

      }

      return 0;
}
```