사운드 재생하기

사운드는 게임에서 매우 중요한 요소 중하나입니다. 게이머는 사운드를 통해서 게임 환경에 대한 정보를 알아낼 수도 있고, 게임에 더 몰입하게 할 수도 있습니다. FPS게임을 예로 들어보죠, 바닥에서 팅! 팅! 하는 소리를 통해 주변에 수류탄이 떨어졌다는 것을 알 수 있습니다. 그리고 총을 조준하고 적을 향해 총알을 발사할 때, 타타타탕하는 총소리가 나지 않는다면, 정말 현실감 떨어지겠죠.

우리가 다룰 사운드의 분야는 극히 일부분에 지나지 않습니다. 일단 필요로 하는 단순한 기능은 그저 음악 파일을 재생하는 것일 뿐입니다. 그리고 정말 거대한 분야인 컴퓨터 사운드에 대해 많은 지식을 갖게 되었으면 좋겠습니다. 그리고 오디오(Audio)라는 단어는 녹음된 사운드를 의미합니다. 오디오와 사운드라는 단어는 자주 혼용되서 쓰이곤 합니다, 그러나 컴퓨터 사운드 분야에서는 사실상 같은 의미라고 보셔도 무방합니다. 컴퓨터가 재생하는 사운드는 전부 오디오니까요.

기본적인 C/C++ 표준 라이브러리에는 사운드 관련 기능이 전혀 존재하지 않습니다. 사운드 기능을 사용하기 위해서는 각 OS마다 고유의 라이브러리를 사용해야 했습니다. 예를 들어 Windows 계열에서는 DirectSound 와 XAudio2를 사용합니다. 이후 이런 모든 운영체제에서 사용가능한 범용 사운드 라이브러리가 개발되고 제공되었습니다. 사운드 API의 핵심 기능 중 하나는 다양한 사운드 형식의 파일을 읽어 들이는 것입니다. 그러나 여기에는 한 가지 문제점이 있습니다. Mp3(MPEG Layer-3)와 같은 형식은 어마어마한 사용료를 내야 합니다. 물론 PCM-Wave, Flac, Ogg, WMV와 같은 공개 형식들도 존재하나, 이러한 음원 형식들에 걸려있는 특허와 사용허가권(라이선스)를 잘 확인해야 합니다.

이러한 이유에서인지 상당수의 사운드 라이브러리들은 유료로 사용해야 하거나, 다양한 형식의 사운드 파일을 읽는 기능이 없습니다. 우리는 이번 장에서 게임 엔진인 Irrlicht(일리힛)엔진의 IrrKlang 라이브러리를 사용해서 사운드를 재생해 볼 것입니다. 이번 장에는 주파수나, 비트레이트 같은 컴퓨터 사운드의 이론과 관련된 내용들은 다루고 있지 않습니다. 그러한 고급 내용들은 11장 사운드 엔진 만들기에서 알아볼 것입니다.

IrrKlang 다운로드하기

IrrKlang은 http://www.ambiera.com/irrklang/ 에서 다운로드 가능합니다. IrrKlang은 비상업적 용도에 한해 무료로 이용 가능합니다. 만약 IrrKlang을 사용해서 게임을 만들어서 판매하려 한다면, IrrKlang Pro의 사용권을 구매해서 사용해야 합니다. IrrKlang과 IrrKlang Pro는 상업적 사용의 허가와 dll파일이 필요없는 점을 제외하면 모든 기능이 동일합니다(2014년 9월 기준).

홈페이지의 좌측 Downloads 카테고리에 들어가서, 여러분의 컴퓨터의 운영체제에 알맞은 버전의 IrrKlang 압축 파일을 다운로드하시기 바랍니다.

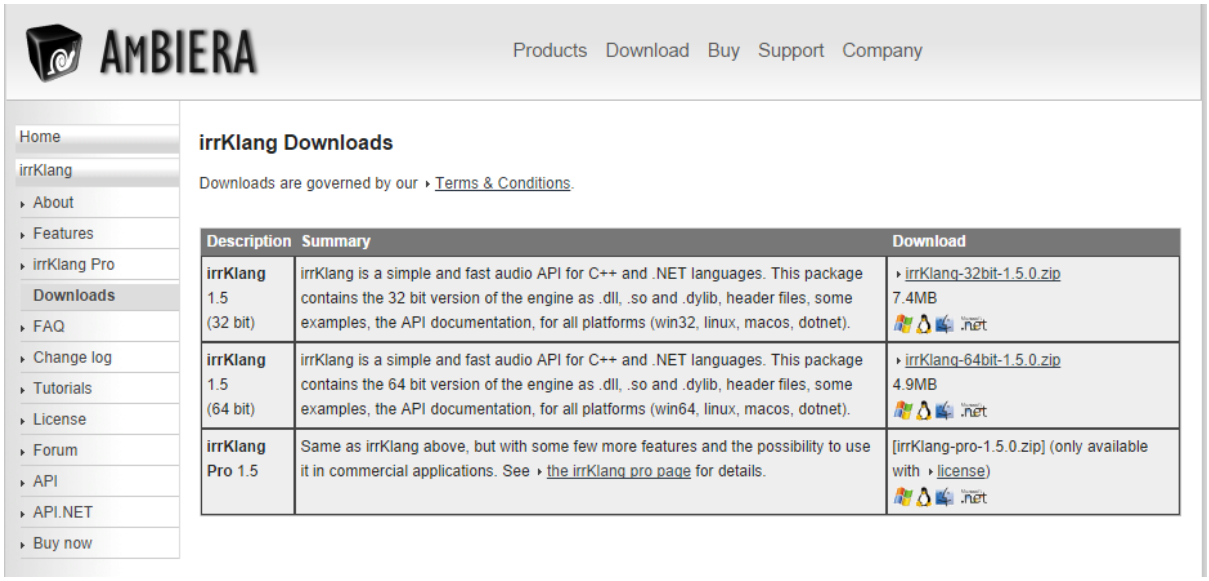

그림 1. IrrKlang 다운로드 페이지

다운로드 받은 압축 파일 안에는 프로그래밍에 필요한 라이브러리, 해더 파일 등과 참고용 문서, 사용허가권 파일 등이 들어있습니다. Lib 폴더 안의 라이브러리 파일들과 bin 폴더 안의 dll파일들, 그리고 include 폴더 안의 헤더 파일들이 필요합니다. 여러분이 IrrKlang을 사용할 프로젝트에 이 파일들을 추가해 주세요. 방법을 모르시는 분께서는 부록의 라이브러리 사용하기를 참고해 주세요. 특히나 사용한 컴파일러에 따라서 폴더 내부에서 나뉘어진 경우도 있습니다. 그럴 때는 여러분에게 맞는 컴파일러를 사용하셔야 합니다. 대부분의 경우는 Visual Studio를 사용하실 테니 32비트 운영체제의 경우 win32-VisualStudio 폴더 안의 파일들을 사용하시면 됩니다.

사운드 재생하기

이제 IrrKlang을 사용할 준비가 되었으니 예제를 통해서 IrrKlang을 사용해 봅시다. 간단하게 Ogg 형식의 오디오 파일을 재생해 보는 예제입니다.

```
// irrKlang을 사용하기 위해서 irrKlang.h을 포함합니다.
#include <irrKlang.h>
#include <iostream>

using namespace std;


int main()
{
	// 사운드 엔진을 생성합니다.
	irrklang::ISoundEngine *engine =
		irrklang::createIrrKlangDevice();
```

```
	cout << "Play from stream. " << endl;

	// play2D 메서드로 ogg 파일을 재생합니다.
	// 두 번째 인자가 true면 재생이 끝나도 계속해서 반복 재생하게 됩니다.
	engine->play2D("../media/classroom_bgm.ogg", true);
	system("pause");

	// 사운드 엔진 제거
	engine->drop();

	return 0;
}
```

예제 1. Play2D Streaming.cpp

IrrKlang의 모든 기능들은 네임스페이스 irrklang 안에 선언 및 정의되어 있습니다. IrrKlang에서 사운드를 관리하는 클래스는 ISoundEngine 클래스 입니다. 이 클래스의 객체는 createIrrKlangDevice() 함수로 얻을 수 있습니다. 수 많은 인자들이 기본 인자로 되어 있기 때문에 인자하나 없이 사용할 수 있습니다. 인자를 채워야 할 정도로 세세하게 사용할 필요는 없습니다.

사운드 파일의 재생은 너무나도 간편합니다. 바로 엔진 객체의 play2D 메서드만을 사용하면 되죠, play2D 함수의 프로토타입은 아래와 같습니다.

```
virtual ISound* irrklang::ISoundEngine::play2D (
	const char * soundFileName,
	bool   playLooped = false,
	bool   startPaused = false,
	bool   track = false,
	E_STREAM_MODE       streamMode = ESM_AUTO_DETECT,
	bool   enableSoundEffects = false
	)
```

1. soundFileName
   재생할 사운드 파일의 이름 혹은 경로입니다.

2. playLooped
   사운드를 반복해서 재생할 지 여부입니다.

3. startPaused
   false면 곧바로 사운드를 재생합니다. 사운드를 곧바로 재생하지 않기를 원할 때 유용합니다.

4. track
   track은 사운드를 제어하고 싶은 경우에 true를 주어 ISound 객체가 반환되게 합니다. ISound 객체는 재생을 정지하거나, 속도/음량 제어 등의 기능을 사용할 수 있습니다.

5. streamMode
   사운드 파일의 데이터를 읽어오는 방식을 지정합니다. 기본값은 irrklang::ESM_AUTO_DETECT로 IrrKlang 라이브러리가 알맞은 방식을 찾아서 사용하게 합니다.

6. enableSoundEffects
   사운드 효과를 활성화할지 여부를 결정합니다.

반환값은 ISound 클래스의 객체입니다. 이 객체는 재생할 사운드에 대한 제어 기능을 가지고 있습니다. 따라서 제어 기능이 필요할 때, 즉 play2D() 메서드의 인자인 startPaused, track, enableSoundEffects 중 하나라도 true일 경우에만 반환됩니다. 혹은 사운드 파일을 재생할 수 없을 경우에도 null이 반환됩니다.

사운드 제어하기

ISound 클래스는 재생하는 사운드를 제어하는 기능들을 메서드로 가지고 있습니다. 몇몇 유용한 메서드들만을 살펴보죠.

프로토타입	**설명**
void setIsPaused(bool paused)	Paused 가 true면 재생을 정지하고, false 면 정지중인 사운드를 재생합니다.
bool getIsPaused()	사운드가 정지중인지를 반환합니다.
void setIsLooped(bool looped)	Looped가 true면 사운드가 반복적으로 재생하게 하고, false면 그렇게 하지 않습니다.
bool isLooped()	사운드가 반복재생중인지를 반환합니다.
void stop()	사운드 재생을 중단하고 사운드 자원을 제거합니다.
setPlaybackSpeed(float speed)	재생 속도를 지정합니다. 예를 들어 1.5를 인자로 주면 재생 속도가 1.5배가 됩니다.
float getPlaybackSpeed()	재생 속도를 반환합니다.
void setPan(float pan)	팬 값을 지정합니다. 인자 pan의 범위는 -1부터 1까지 입니다. 팬 값에 대한 설명은 아래에 나와 있습니다.
float getPan()	팬 값을 반환합니다.
void setPlayPosition(unsigned int pos)	현재 재생중인 위치를 지정합니다, 인자 pos의 단위는 밀리초입니다. 예를들어 14500을 주면 사운드의 14.5초 부분을 재생하게 됩니다.
unsigned int getPlayPosition()	현재 재생중인 위치를 반환합니다. 단위는 역시나 밀리초입니다.

Pan은 여러 개의 스피커에서 사운드가 나오는 비율을 지정하는 값입니다. IrrKlang에서 Pan의 범위는 -1 ~ 1 인데요. 2개의 스피커(이를 스테레오라고 합니다)를 기준으로 -1로 갈수록 한쪽 스피커로 나오는 음량이 더 많아지고 반면 반대쪽 스피커로 나오는 음량이 더 적어집니다. 만약 1로 간다면 그 반대입니다. Pan값이 0이라면 두 스피커에서 고르게 소리가 나오게 됩니다.

Sound Control 예제프로젝트는 콘솔 창에서 명령어를 입력 받아 위의 메서드들을 사용해 재생중인 사운드를 처리하는 소스 코드입니다. 주석을 참고해서 코드를 참고해 보시기 바랍니다.

원본 관리하기

우리가 엔진의 `play2D()` 메서드를 사용하면 그 즉시 해당 경로의 사운드 파일을 읽어 들이면서 재생을 시작합니다. 이런 방식에는 문제가 있습니다. 사운드 파일을 찾고 읽어 들이는 과정은 많은 시간을 요구하는 작업입니다. 때문에 게임과 같이 빠른 반응을 필요로 하는 프로그램에서는 미리 사운드 파일을 읽어 두거나, 파일의 크기가 클 경우 미리 파일을 열어놔야 합니다. 이런 기능 역시 IrrKlang은 ISoundSource를 통해서 지원합니다. Sound Source, 즉 음원 객체는 미리 사운드 데이터를 읽어 들인 후 재생을 원할 때 제공하며 빠르고 효율적으로 사운드 재생을 할 수 있습니다. 음원은 ISoundSourceEngine 클래스의 addSoundSourceFromFile() 혹은 addSoundSourceFromMemory() 메서드로 추가할 수 있습니다.

함수	설명
ISoundSource* addSoundSourceFromFile (const char* filename, E_STREAM_MODE mode = ESM_AUTO_DETECT, bool preload = false);	filename 으로부터 음원을 읽어와서 IrrKlang 내부 저장소에 추가한 뒤 반환합니다. 파일명을 이름으로 저장합니다. Preload 인자가 true이면 play2D()와 같은 메서드를 통해 재생을 시작하지 않아도 곧바로 로딩합니다. mode 는 스트림 모드를 지정합니다. 스트림 모드에 대해선 이후에 설명하도록 하겠습니다.
ISoundSource* addSoundSourceFromMemory (void *memory, int sizeInBytes, const int *soundName, bool copyMemory=true);	memory 라는 배열의 시작주소로부터 데이터를 sizeInBytes 만큼 읽어들여 음원을 추가합니다. 음원을 추가할 때 IrrKlang 내부에서 soundName 으로 저장됩니다. 여기서 memory는 사운드 파일 하나를 통째로 읽어서 저장한 메모리 공간을 의미합니다. copyMemory 인자가 true일 경우 새 메모리 공간을 할당하며 메모리를 복사하지만, false일 경우 인자로 온 주소를 그대로 사용합니다.

	이 함수는 네트워크 통신과 같이 시스템 드라이브와 같은 파일이 아닌 곳에서 얻어온 사운드 파일 데이터로부터 음원을 생성하기 위한 메서드입니다.

addSoundSourceFromFile("abc.wav") 와 같이 음원을 추가하게 된다면, 이후 추가적인 로딩 없이play2D("abc.wav") 할 때 자동적으로 미리 로딩된 abc.wav 파일을 재생하게 됩니다. 만약 이게 마음에 들지 않는다면 반환된 ISoundSource 객체를 이용하면 됩니다. ISoundEngine::addSoundSourceAlias() 함수를 사용하면 특정 음원에 원하는 이름을 지정할 수 있습니다.

스트림 모드란 이렇게 데이터를 읽어들이는 방식을 의미합니다. 예컨데 3~5분가량 되는 가요 음악들은 용량을 아끼기 위해서 기본적으로 압축되어 있습니다. 그럼에도 불구하고 약 5MB가량의 저장공간을 사용합니다. 이것들의 압축을 해제하면 20MB 나 되는 많은 메모리를 차지하게 됩니다. 이러한 점은 분명히 성능에 문제가 될 것입니다. 따라서 이렇게 긴 음악 파일은 일부분씩 읽어 들이면서 사운드를 재생하는데, 이 방식을 스트리밍(Streaming)이라고 합니다. 하지만 이전에 말했듯 파일을 읽는 작업은 성능이 많이 요구되는 힘든 작업입니다. 따라서 효과음 같은 길이가 짧은 음원은 전부 읽어들여서 사용합니다. E_STREAM_MODE 열거형은 이러한 스트리밍 방식을 지정하는 값들을 가지고 있습니다. 하나하나 살펴보죠.

E_STREAM_MODE 열거형	
ESM_AUTO_DETECT	대부분의 함수에서 기본값입니다. 차지하는 메모리 량을 계산해서 아래 두 값 중 적절한 값을 선택합니다.
ESM_STREAMING	음원을 로딩할 때 스트리밍 기법을 사용하도록 합니다.
ESM_NO_STREAMING	음원을 로딩할 때 스트리밍하지 않고 전부 읽어들이도록 합니다. 이 값을 선택하였다 해도 IrrKlang이 거부하여 스트리밍 모드로 바뀔 수도 있습니다.

아래 예제에선 ISoundSource 클래스를 활용하여 미리 읽어들인 음원을 재생해 봅니다. 등록된 음원을 ISoundEngine::removeSoundSource() 메서드로 할 수 있습니다.

```
#include <irrKlang.h>
#include <iostream>

using namespace std;


int main()
```

```
{
	// 사운드 엔진을 생성합니다.
	irrklang::ISoundEngine *engine =
		irrklang::createIrrKlangDevice();

	cout << "Play from source. " << endl;

	irrklang::ISoundSource *source =
		engine->addSoundSourceFromFile("../media/classroom_bgm.ogg");

	engine->addSoundSourceAlias(source, "ClassRoom");

	// 아래 3개는 모두 똑같이 동작합니다.
	//engine->play2D(source, true);
	//engine->play2D("../media/classroom_bgm.ogg");
	//engine->play2D("ClassRoom");

	int abc;
	cout << "아무 키나 누르고 엔터!" << endl;
	cin >> abc;


	// 만약 음원을 없애고 싶다면?
	engine->removeSoundSource("ClassRoom");
	engine->removeSoundSource("../media/classroom_bgm.ogg");
	engine->removeSoundSource(source);

	// 음원을 전부 지워버려!
	engine->removeAllSoundSources();

	// 사운드 엔진 제거
	engine->drop();

	return 0;
}
```

예제 2. Play2D Source.cpp

IrrKlang 의 추가적인 유용한 기능들

대부분의 게임들은 배경음, 효과음 2가지의 볼륨을 나누어서 설정할 수 있도록 해줍니다. 또는 모든 사운드의 볼륨을 조절하거나, 재생하지 않게 할 수도 있습니다. ISoundSource클래스와 ISoundEngine클래스는 이러한 보조 기능을 지원합니다. ISoundEngine::setSoundVolume() 메서드는 모든 재생중인 사운드의 볼륨을 조절합니다. 하지만 이 설정된 값으로 바뀌는 건 아닙니다. 0.5

볼륨으로 음원이 재생 중일 때 전체 음원의 볼륨을 0.7로 바꿔버린다면 0.5로 재생되던 음원의 볼륨은 0.5 × 0.7 = 0.35 가 됩니다. ISoundSource::setDefaultVolume() 메서드는 이 음원을 재생할 때의 기본 볼륨을 지정합니다. 기본 값은 1입니다.

이 이외에도 IrrKlang은 3D Sound를 지원합니다. 3D Sound란 청취자와 발생지의 위치를 지정해서 거리만큼 음악의 볼륨을 감쇄시키거나, 한쪽 스피커에서 더 큰 소리가 나게 하는 것을 말합니다. 3D 사운드와 관련된 내용들은 여기서 담지는 않겠습니다. IrrKlang 공식 홈페이지에는 관련된 튜토리얼들과 API문서들이 존재합니다. 또한 웹 상의 다른 튜토리얼 문서들을 활용하시면 자료들을 쉽게 얻으실 수 있습니다/