사운드

2014-07-21

1. 디지털 사운드 개론

우리는 귀로 소리를 들을 수 있다. 귀는 사람의 입력 장치로써, 다양한 주변 환경들에 대한 정보를 얻을 수 있게 한다. 소리가 없는 게임은 정말로 어색할 것이다. 소리는 게이머를 게임에 더 집중하게 만드는 핵심 요소 중 하나이다. 우리가 일반적으로 듣는 소리를 사운드라고 한다. 발걸음 소리나 바람 소리 같은 것에서부터 음악 역시 소리이다. 그러나 그 중에서 녹음된 소리를 오디오(Audio)라고 부른다. 컴퓨터가 재생하는 소리는 사실상 모두 녹음된 오디오라고 할 수 있다. 그래도 이 튜토리얼에서는 통틀어서 사운드라고 칭하겠다. 컴퓨터로 듣는 사운드는 크게 2D 사운드와 3D 사운드로 나눌 수 있다. 2D 사운드는 일반적인 사운드를 의미하지만 3D 사운드는 소리의 발생 위치와 듣는 위치를 지정해서 그 거리에 따라 다양한 효과를 준다. 예를 들어 지하철 역에서 기차가 올 때, 기차가 도착하기 직전의 소리는 매우 크지만 기차가 멀리서 오고 있을 때의 소리는 작게 들린다. 여기서 3D 사운드에 대해서도 다루지는 않는다. 그러나 그다지 어려울 건 없다. 오늘날에는 흔하게 사용되는 기술이기 때문에 검색해보면 금방 답을 찾을 수 있을 것이다.

소리는 진동이다. 물체의 진동이 대기를 따라서 전달되어 귀로 들어간 다음 고막을 떨게 한다. 고막이 떨며 다시 그 진동이 귀 안으로 들어가서 청각 세포를 자극시켜 전류가 뇌로 흘러가서 소리가 들리게 된다. 즉 소리는 물체를 떨게 함으로써 발생된다. 현악기나 성대, 스피커 모두 물체를 떨게 해서 소리를 낸다. 매우 큰 스피커가 소리를 낼 때 만져보면 부르르 떨리는 것을 느낄 수 있다. 우리의 뇌는 이 진동을 형태에 따라서 다르게 인식한다. 1초에 떨리는 횟수를 주파수(Frequency)라고 하고 단위를 Hz로 표기한다. 또한 진동하는 물체는 계속 떨면서 반복운동을 하는데 한번 반복운동을 하는 데 걸린 시간을 주기(Cycle)이라 하고 1주기동안 움직인 거리를 진폭(Amplitude)라고 한다. 주파수가 클수록(많이 떨수록) 더 높은 음정의 소리가 나고, 작을수록(적게 떨수록) 더 낮은 음정의 소리가 난다. 진폭이 클수록 큰 소리가 나고, 진폭이 작을수록 작은 소리가 난다.

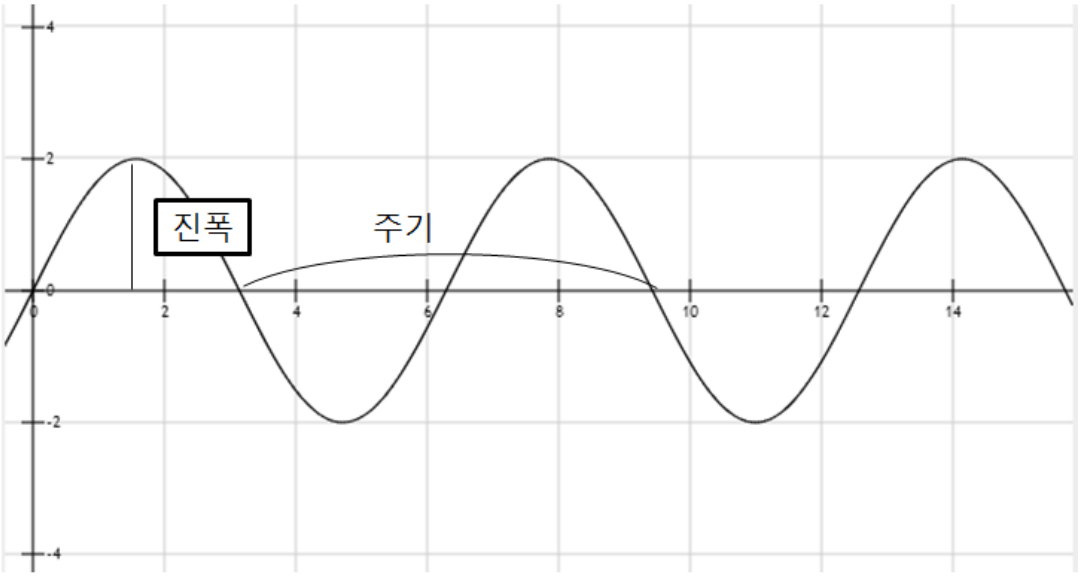

그림 1. 사운드의 구성

디지털 사운드는 정해진 주파수에 진폭들의 흐름으로 나타낸다. 하지만 컴퓨터는 데이터를 어느 정도까지만 세밀하게 저장할 수밖에 없다. 즉 아날로그(Analog) 데이터를 디지털(Digital) 데이터로 변환해야(Convert) 하는데, 이를 수행하는 기계를 ADC(Analog-To-Digital)라고 한다. 따라서 나타낼 수 있는 가장 가까운 값으로 실제 음성 데이터들을 양자화(Quantization)해야 한다.

아래 그림 2에서 점선은 실제 사운드이고, 막대는 양자화된 디지털 사운드이다. 만약 주파수가 더 크다면 위 그림에서 막대들의 간격은 더 좁아지고 더 세밀하게 데이터들을 표현할 수 있을 것이다. 하지만 그만큼 차지하는 기억 장치의 양은 늘어날 것이다. 또한 단순히 주파수만을 늘리고 기존의 사운드 데이터는 그대로 유지한다면 한 진폭의 시간은 줄어들기 때문에 재생 속도는 늘어나게 되고 주파수가 빨라져서 음정도 올라가게 된다. 동영상을 빠르게 재생할 때 목소리가 웃기게 되고, 느리게 재생할 때 저음의 이상한 소리로 바뀌는 것도 이러한 이유 때문이다. 그리고 위 그림에서 막대 하나의 높이를 나타내는 것이 진폭이다. 진폭을 표본(Sample)이라고 한다. 그렇기 때문에 주파수를 표본율(Sample Rate)라고도 한다. 한 표본을 나타내는 데 사용한 메모리의 크기를 깊이(Depth)라고 한다. 이처럼 아날로그 신호를 주파수와 깊이로 나타내는 방식을 PCM(펄스 부호 변조, Pulse-Code Modulation)이라고 한다. 그런데 깊이가 같은 표본이라도 형식(Format)이 다를 수 있다. 정수형이나 부동소수점 형식으로 나타낼 수도 있기 때문이다. 오늘날의 사운드는 대부분이 16비트 깊이를 가지고 고품질 오디오는 32비트를 사용한다. 사운드 데이터는 매우 많은 메모리 용량을 차지한다. 예를 들어 44KHz(=44000Hz) 주파수의 16 비트 4채널의 1분짜리 오디오는 $44000(\text{Frequency}) \times 4(\text{Channels}) \times 2(\text{Depth}) \times 60(\text{Time}) = 21120000\text{bytes}$ 인데 이는 약 20MB이다.

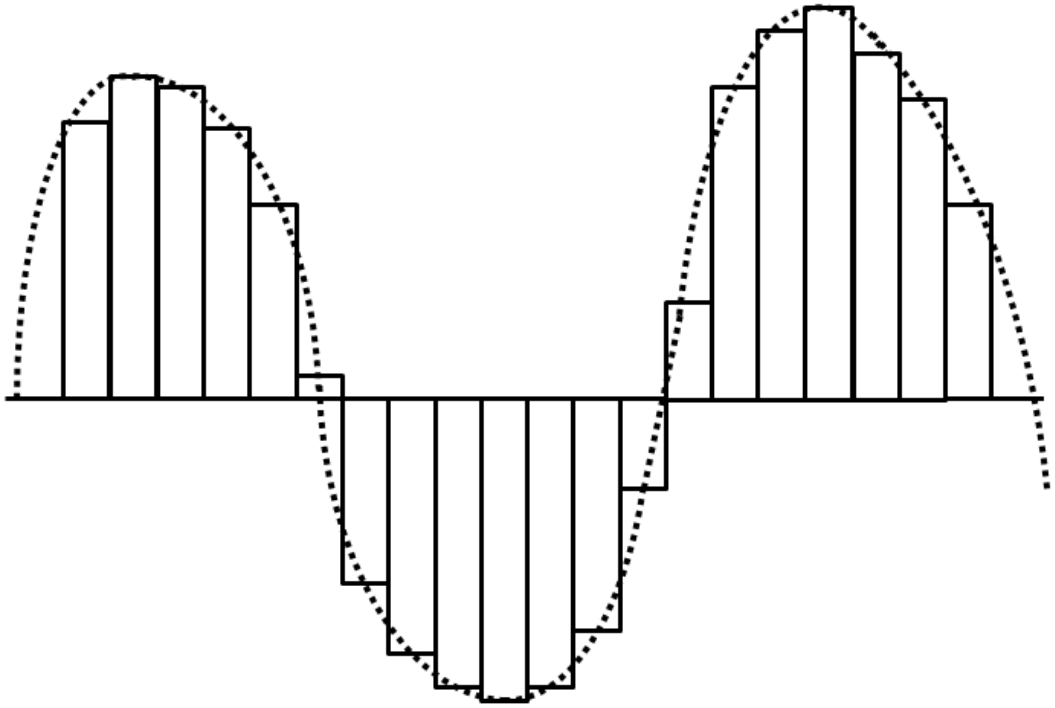

그림 2. 실선은 아날로그 음원, 막대는 양자화된 디지털 음원을 의미한다).

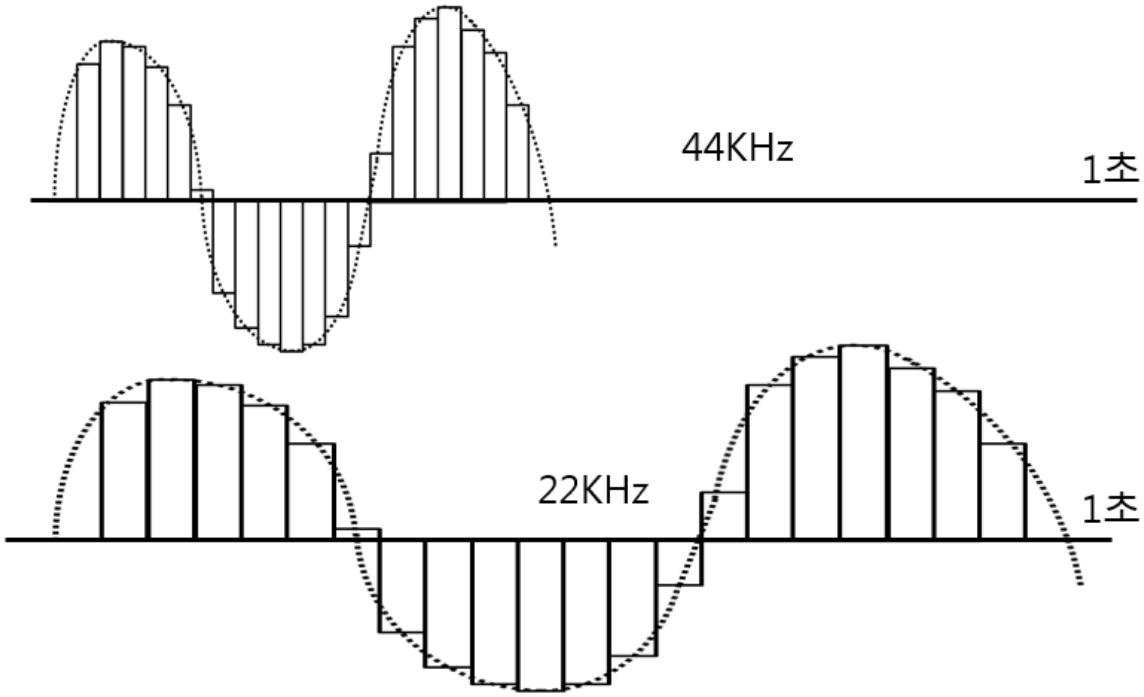

그림 3. 단순히 음원의 주파수만을 2배 늘렸을 때. 1초동안 재생되어야 할 음원이 0.5초만에 재생되게 되며, 주파수가 빨라진 만큼 음정도 높아진다.

이러한 정보들은 메모리에 저장되었다가 사운드 카드를 통해서 스피커로 전송되고 소리가 나오게 된다. 사운드 카드는 사운드 연산을 대신 처리해주는 장치인데, 대부분은 CPU에 내장되어 있고 그래픽카드처럼 범용적이지 않기 때문에 대부분 전문가들만이 사용하는 고급 하드웨어이다. 오늘날의 Sound API들은 이 사운드 카드를 통해서 사운드를 처리(Processing)할 수 있게 해주는 기능들을 제공한다. 뿐만 아니라 오늘날 발전한 GPU의 성능을 이용해서 GPU로 사운드 처리를 하기도 한다. Windows의 사운드 API로는 과거 DirectSound가 존재해 왔었다. DirectSound는 충분하고 다양한 기능을 가지고 있었지만 시대가 더더욱 발전하면서 새로운 기능들이 필요해지기 시작했다. 그러나 DirectSound는 버전 8을 끝으로 한참 동안이나 새로운 API가 등장하지 않았다. 그러던 중 Microsoft에서 XAudio라는 새로운 사운드 API를 발표했다. 2014년 현재는 XAudio2를 이용하여 다양한 사운드 처리를 할 수 있다. 기본적으로 사운드 재생을 포함해서 메모리 관리를 도와주거나 재생되는 사운드에 효과를 줄 수도 있다. 또한 여러 개의 사운드를 합쳐서 믹싱(Mixing)을 해서 더 부드럽고 효과적으로 여러 개의 사운드를 재생할 수도 있다.

만약 XAudio2처럼 다양한 고급 기능들을 제공하는 저수준(Low-Level) API가 사용하기 벅차다면, 단순히 2D, 3D 사운드를 재생하는 간단한 기능들만을 원한다면 FMod나 일리힛 같은 엔진에 포함된 사운드 엔진을 사용하길 권장한다. XAudio2는 그 자체로서 사운드 엔진이라기 보다는, 훌륭한 사운드 엔진을 개발하기 위한 도구의 성격이 강하다. 그만큼 배울 것도 많다는 소리이다. 필자도 XAudio2의 수많은 기능들을 사용해 보진 않았으나 함께 공부한다는 생각으로 글을 적는다.

2. XAudio2 개요

아래 내용은 MSDN의 XAudio2 API Guide 에서 개요부분(XAudio2 Introduction)을 번역한 내용입니다.[1]

"XAudio2는 저수준(Low-level) 오디오 API입니다. 이전의 DirectSound나 XAudio와 유사하게 게임을 위한 부호 처리와 혼합 기능을 제공합니다. XAudio2는 오랫동안 기다려 온 DirectSound의 대체 API입니다. 몇 가지 눈에 띄는 문제들과 기능 요청들을 해결해 줍니다.

XAudio는 게임을 위한 고성능 오디오 엔진을 개발하기 위한 목적으로 설계되었습니다. 음향 효과와 배경음악을 최신 게임에 넣고 싶은 게임 개발자들을 위해서, XAudio2는 낮은 지연 시간(Low-latency)을 가진 오디오 그래프와 혼합 엔진을 제공하고 동적 버퍼와 정확한 샘플 재생 동기화, 그리고 암시적 원본 재생율 변환과 같은 기능들을 지원합니다. WASAPI와 비교하면, XAudio2는 복잡한 오디오 문제를 해결하기 위한 최소한의 코드만을 요구합니다. Media Foundation

[1] "XAudio2 Introduction", http://msdn.microsoft.com/en-us/library/windows/desktop/ee415813(v=vs.85).aspx

Engine과 비교해 보아도, XAudio2는 게임에서 사용되기 위한 저수준이고, 낮은 지연 시간을 가진 C++ API 입니다. 단순히 음악을 재생하고 싶은 응용 프로그램이라면, Media Foundation 엔진이 응용프로그램에 있어서 아마도 더 좋은 선택이 될 것입니다."

위 설명글의 "부호 처리" 같은 내용을 전부 이해하지 못한다고 해도 별 상관 없다. XAudio2 의 변화는 차차 코드를 보면서 설명할 것이다. XAudio2와 관련된 헤더 파일과 라이브러리들은 Windows 8용 Windows SDK 안에 포함되어 있다. 그렇지만 Windows 7에서도 사용 가능하고, XAudio2를 사용하는 어플리케이션은 DirectX Runtime이 설치된 환경이라면 얼마든지 실행할 수 있다.[2] XAudio2.h 를 #include 하고 XAudio2.lib을 링크하고, XAudio2Utils.h 와 XAudio2Utils.cpp를 작성해 보자. 이 두 파일에는 XAudio2 와 관련된 보조 기능들을 작성할 것이다. 코드에 작성된 주석들을 꼭 읽어보길 바란다.

XAudio2Utils.h

```
// Include guard
#ifndef _XAudioUtils_h_
#define _XAudioUtils_h_

#include <Windows.h>
#include <xaudio2.h>
#pragma comment(lib, "xaudio2.lib")


// XAudio의 기본적인 객체들을 생성해서
// 반환해 준다.
HRESULT InitXAudio2(
              IXAudio2 **audio,
              IXAudio2MasteringVoice **masterVoice );


// COM 인터페이스를 해제해주는 함수
template <typename T>
inline void SafeRelease(T t) {
       if(t)
              t->Release();
}

// XAudio2 보이스를 제거하는 함수
inline void SafeDestroy(IXAudio2Voice* voice)
```

[2] XAudio2 Versions. MSDN. http://msdn.microsoft.com/ko-kr/library/windows/desktop/ee415802(v=vs.85).aspx

```
{
	if(voice)
		voice->DestroyVoice();
}

// C++ new 로 할당된 메모리를 제거하는 함수.
template <typename T>
inline void SafeDelete(T t) {
	if(t)
		delete t;
}

#endif
```

XAudio2Utils.cpp

```
#include "XAudio2Utils.h"

HRESULT InitXAudio2(
		IXAudio2 **audio,
		IXAudio2MasteringVoice **masterVoice )
{
	HRESULT hr = S_OK;

	// CoInitialize() 함수로 COM 초기화를 해 주어야 한다.
	hr = CoInitialize(nullptr);


	if(SUCCEEDED(hr))
	{
		// XAudio2Create 함수로 XAudio2 객체를 생성한다.
		// 2, 3번째 인자는 기본 값을 사용하며
		// 2번째 인자는 XAudio 객체의 특징을 명시하는데,
		// 아직까진 기본적으로 0을 주어야 하고,
		// 세번째 인자는 사용할 CPU를 지정하는데,
		// 기본 CPU(1번)을 사용한다.
		hr = XAudio2Create(audio, 0, XAUDIO2_DEFAULT_PROCESSOR);
	}


	if(SUCCEEDED(hr))
	{
		// 이제 마스터링 보이스를 생성한다.
		// 마스터링 보이스는 재생되는 모든 보이스들을
		// 제어하는 역할을 한다.
		hr = (*audio)->CreateMasteringVoice (masterVoice);
```

```
        }

        return hr;
}
```

SafeRelease()와 SafeDelete()는 포인터가 널인지 확인하고 Release()/delete 해주는 함수이다. 이 두 함수는 컴퓨터 그래픽스 예제에서도 등장한다. 그런데 SafeDestroy() 라는 함수가 추가되었다. 이 함수는 XAudio2의 IXAudio2Voice 객체가 널인지 확인하고 Destroy()함수를 호출해서 제거한다. XAudio2의 IXAudio2Voice 및 이를 상속하는 클래스의 객체들은 IUnknown을 상속받는 COM인터페이스의 형식을 따르지 않는다.

IXAudio2 인터페이스는 XAudio2와 관련된 모든 객체들을 생성하는 기능들을 갖고 있다. 이는 XAudio2Create 함수를 통해서 생성할 수 있다. XAudio2에는 음성(Voice)라는 개념이 존재한다. 음성은 오디오 데이터를 가공해서 사운드 출력 장치에 전달하는 역할을 수행한다. XAudio2에는 3가지 음성이 있다. 마스터 음성(Mastering Voice), 서브믹스 음성(Submix Voice), 소스 음성(Source Voice) 각각 역할과 기능들이 다르지만 공통적으로 할 수 있는 기능들도 있다.

종류	**특징 및 기능**
공통	음량(Volume)을 제어할 수 있다. 채널별 음량 제어가 가능하다. 다양한 효과(Effect)를 적용할 수 있다.
소스 음성	오디오 데이터를 갖고 있다가 XAudio2 처리 파이프라인에 전달하는 역할을 수행한다.
서브믹스 음성	서브믹스 음성은 주로 효과 처리를 하거나 성능을 향상시키기 위해서 사용된다. 소스 음성과 같이 오디오 데이터를 갖고 있지는 않는다. 주파수변환(SRC), 효과 사슬(Effect Chain), 혹은 너무 많은 음성이 동시에 재생되는 걸 방지할 수 있다.
마스터 음성	마스터 음성은 하나의 사운드 출력 장치를 나타낸다. 사운드 출력 장치로 나가는 모든 오디오 데이터는 마스터 음성을 거친다. 즉 마스터 음성을 제어하면 모든 음성을 제어할 수 있다.

각각 IXAudio2의 CreateSourceVoice(), CreateSubmixVoice(), CreateMasteringVoice() 메서드를 통해서 생성할 수 있다. 마스터 음성을 생성할 때, CreateMasteringVoice() 메서드의 두 번째 인자와 세 번째 인자를 사용하여 채널 수와 주

파수를 원하는 대로 바꿀 수 있다. 하지만 출력 장치가 주파수 지정을 지원하지 않는 경우는 자동적으로 출력 장치의 기본 값으로 변환된다. 소스 음성을 생성할 때 소스 음성이 전달될 서브믹스 음성을 지정할 수 있다. 만약 서브믹스 음성이 지정되지 않았다면 오디오 데이터는 곧바로 마스터 음성으로 전달되지만, 서브믹스 음성이 지정될 경우 서브믹스 음성으로 전달된다. 서브믹스 음성은 전달받은 음성을 변환한 뒤 다시 또 다른 음성으로 전달한다. 전달하는 음성은 여러 개가 될 수도 있다. 이렇게 소스 음성들이 서브믹스 음성과 이어지며 결국에는 마스터 음성으로 가게 되는데 이것을 오디오 그래프라고 한다.

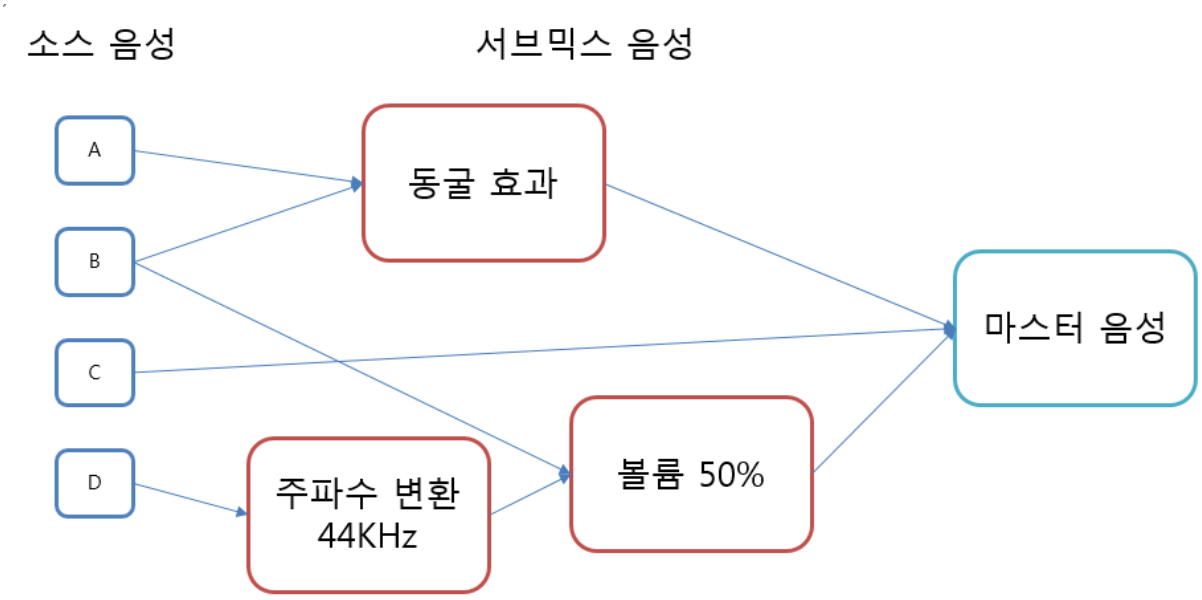

그림 4. 오디오 그래프의 예.

위의 그림 4는 오디오 그래프의 한 예이다. A, B, C, D라는 소스 음성이 있고 3가지 서브믹스 음성과 한 개의 마스터 음성이 있다. A는 동굴 효과가 적용되어 재생될 것이며, C는 아무런 효과 없이 곧바로 재생되고, D는 오디오 데이터의 주파수 44KHz로 변환된 뒤 볼륨이 50% 줄어들어서 재생되게 된다. B는 2가지 서브믹스 음성을 가졌는데, 이런 경우는 두 가지 변화가 동시에 일어나게 된다. 볼륨이 50%가 되면서 동굴 효과가 적용된다.

참고자료

1. Indiana University, "What is amplitude?"
   http://www.indiana.edu/~emusic/acoustics/amplitude.htm

2. GNV, "진동이란?", http://www.globalv.co.kr/techdata_01

3. www.win32developer.com, "XAudio2 Tutorial 1",
   http://www.win32developer.com/tutorial/xaudio/xaudio_tutorial_1.shtm

4.