Sound Programming

제5장 Sound Effect를 위한 Sound Control Basics

음향 효과를 위한 Sound Control 기본

- 고급 음향 효과 프로그래밍을 위해서 알아야 할 기본적인 사운드 컨트롤
  - 여러 사운드를 동시에(concurrently) 재생하기
  - 반복(looping) 재생하기
  - 볼륨 컨트롤
  - 여러 사운드를 연속(streaming) 재생하기

Playing Sounds Concurrently

- 이전의 예에서는 한 **Wav** 사운드를 반복 재생하려면,
  - "재생 시작과 종료"를 계속 반복할 수 밖에 없었음.
  - 같은 사운드건 서로 다른 여러 개의 사운드건, 이들을 동시 또는 반복 재생하려면 일단 ***Source를 여러 개 마련해야*** 함.
- **OpenAL**에서 여러 음원 동시 재생 순서

동시 재생할 **Wav** 파일 준비

Wav 파일 개수 만큼의 **Buffer** 생성(alGenBuffers())

각 **Buffer**에 음원 데이터를 적재(ALFWLoadWaveToBuffer())

Wav 파일 개수 만큼의 **Source** 생성(alGenSources())

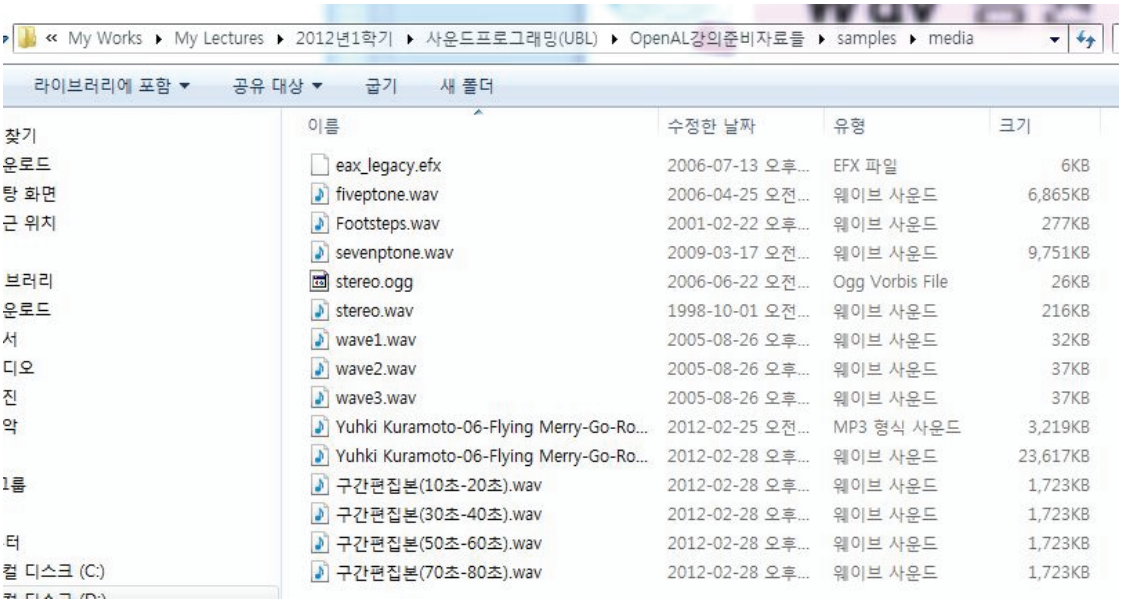
생성한 **Buffer**와 생성한 **Source**를 1:1로 하나씩 연결한다(alSourcei()).

alSourcePlay() 대신 alSourcePlayv()를 이용하여 동시 재생 시작!

재생 중인 **Source** 모두에 대해 상태를 얻어(alGetSourcei()) 재생 종료 여부를 검사!

Wav 음원 준비

- **…\samples\media** 폴더 밑에 재생할 **wav** 파일들을 미리 저장해둔다.



PlayStaticControl 예제 - main() 함수 흐름

- **OpenAL 초기화 및 Buffer 생성**

```
int main()
{
	ALuint uiMyBuffer[4];
	ALuint uiMySource[4];
	ALint iMyState[4];
	WAVEID MyWaveID[4];

	// Initialize Framework
	ALFWInit();
	ALFWprintf("Concurrently Playing Test Application\n");

	if (!ALFWInitOpenAL()) {
		ALFWprintf("Failed to initialize OpenAL\n");
		ALFWShutdown();
		return 0;
	}

	alGenBuffers(4, uiMyBuffer);  // 재생할 Wav 파일이 4개 이므로 버퍼를 4개 생성한다.
	……
```


PlayStaticControl 예제 - main() 함수 흐름 (계속)

- **Buffer에 음원데이터 적재 → Source 생성 → Buffer와 Source 연결**

```
……
// framework 라이브러리 함수를 이용해 각 버퍼에 음원 데이터를 적재한다.
if (!ALFWLoadWaveToBuffer(ALFWaddMediaPath("음원1.wav"), uiMyBuffer[0], &MyWaveID[0])) {
	ALFWprintf("Failed to load %s\n", " 음원1.wav");
}
if (!ALFWLoadWaveToBuffer(ALFWaddMediaPath("음원2.wav"), uiMyBuffer[1], &MyWaveID[1])) {
	ALFWprintf("Failed to load %s\n", "음원2.wav");
}
if (!ALFWLoadWaveToBuffer(ALFWaddMediaPath("음원3.wav"), uiMyBuffer[2], &MyWaveID[2])) {
	ALFWprintf("Failed to load %s\n", "음원3.wav");
}
if (!ALFWLoadWaveToBuffer(ALFWaddMediaPath("음원4.wav"), uiMyBuffer[3], &MyWaveID[3])) {
	ALFWprintf("Failed to load %s\n", "음원4.wav");
}
// Source도 4개 생성한다.
alGenSources(4, uiMySource);
// Buffer와 Source를 연결한다.
alSourcei(uiMySource[0], AL_BUFFER, uiMyBuffer[0]);
alSourcei(uiMySource[1], AL_BUFFER, uiMyBuffer[1]);
alSourcei(uiMySource[2], AL_BUFFER, uiMyBuffer[2]);
alSourcei(uiMySource[3], AL_BUFFER, uiMyBuffer[3]);
……
```

PlayStaticControl 예제 - main() 함수 흐름 (계속)

- **alSourcePlay()**가 아닌 **alSourcePlayv()**로 동시 재생 → 동시재생 중인 모든 **source**에 대해 종료 검사

```
……
// 동시 재생(즉, 섞임 재생) 버전
alSourcePlayv(4, uiMySource);

// 동시 재생 중인 모든 source들에 대해 상태를 검사하여 재생 종료 여부를 검사한다.
do {
        Sleep(100);
        ALFWprintf(".");
        // Get Source State
        alGetSourcei( uiMySource[0], AL_SOURCE_STATE, &iMyState[0]);
        alGetSourcei( uiMySource[1], AL_SOURCE_STATE, &iMyState[1]);
        alGetSourcei( uiMySource[2], AL_SOURCE_STATE, &iMyState[2]);
        alGetSourcei( uiMySource[3], AL_SOURCE_STATE, &iMyState[3]);
} while (iMyState[3] == AL_PLAYING || iMyState[2] == AL_PLAYING ||
         iMyState[1] == AL_PLAYING || iMyState[0] == AL_PLAYING);
……
```


반복 재생(Looping)

- 같은 사운드를 중복 재생하려면?
  - 재생 전 Source의 상태를 AL_LOOPING 상태로 설정

```
// Source의 상태를 looping 상태로 세팅
// AL_TRUE: 무한 루프 재생 ON, AL_FALSE: 무한 루프 재생  OFF
alSourcei(uiSource, AL_LOOPING, AL_TRUE);

// Play Source
alSourcePlay( uiSource );
ALFWprintf("Playing Source ");

do {
   Sleep(100);
   if (_kbhit()) break;          // 무한 재생을 중간에 종료할 수 있도록
   ALFWprintf(".");
   // Get Source State
   alGetSourcei( uiSource, AL_SOURCE_STATE, &iState);
} while (iState == AL_PLAYING);
```

  - 단, 무한 재생이므로 적당한 시점에 무한 재생을 OFF 시켜주거나, 재생을 중지 시켜야 함.

볼륨 컨트롤

- **OpenAL** 재생에서 볼륨 제어는 **Source**에서 가능
  - 용어: volume 대신 *GAIN*을 사용(입력 대비 출력의 증폭비율)
  - Source 생성 후 디폴트 gain 값: 1.0.
    - 1.0: 현 스피커 상태에서 낼 수 있는 최대값의 의미
    - 이 값을 줄였다 키웠다 하며 볼륨을 조절할 수 있다.

```
// 현재 gain 값(볼륨 값)을 알아냄.
ALfloat volume;
alGetSourcef(uiSource, AL_GAIN, &volume);
ALFWprintf("value of gain: %f\n", volume);      // 1.0이 현재 스피커 볼륨이 낼 수 있는 최대 값임.

// Play Source
……

do {
    Sleep(100);
    volume *= (ALfloat)0.9;      // gain 값을 줄여가며 볼륨이 줄어드는지 확인함.
    alSourcef(uiSource, AL_GAIN, volume);
    if (_kbhit()) break;         // 무한 재생을 중간에 종료할 수 있도록
    ALFWprintf(".");
    // Get Source State
    alGetSourcei( uiSource, AL_SOURCE_STATE, &iState);
} while (iState == AL_PLAYING);
```


Playing Stream

- 여러 음원을 순서대로 배열한 후 순서대로 자동 재생하는 기법
- 코딩 순서

Stream 형태로 재생할 Wav 파일들 준비

Wav 파일 개수 만큼의 Buffer 생성(alGenBuffers())

각 Buffer에 음원 데이터를 적재(ALFWLoadWaveToBuffer())

Source는 한 개만 생성(alGenSources())

n 개의 Buffer들을 Source에 Queuing한다
(alSourceQueueBuffers(uiSource, n, uiBuffers)).

alSourcePlay() 를 이용하여 Stream 재생 시작!

재생 중인 Source 에 대해 상태를 얻어 재생을 완료한 버퍼의 개수를 알아내는 방식으로 전체 재생 종료 여부를 알아낸다
(alGetSourcei(uiSource, AL_BUFFERS_PROCESSED, &iBuffersProcessed))

Code for Stream Playing

```
#define NUMBUFFERS                  (4)         // 스트림 재생에서 사용할 Wav 파일 개수
#define SERVICE_UPDATE_PERIOD       (500)       // 스트림 재생 종료 검사 간격(밀리초 단위)
……
ALuint uiStreamBuffer[NUMBUFFERS];
ALuint uiStreamSource;
ALint iStreamState;
WAVEID StreamWaveID[NUMBUFFERS];
ALint iBuffersProcessed;            // 이미 처리된(재생된) 버퍼의 개수
ALint iQueuedBuffers;               // 아직 재생 완료되지 않고 stream 큐에 남아 있는 버퍼의 개수
const char* stream_wav_files[NUMBUFFERS] = {
      "구간편집본(70초-80초).wav",
      "구간편집본(50초-60초).wav",
      "구간편집본(30초-40초).wav",
      "구간편집본(10초-20초).wav"
};

// 원하는 개수만큼 버퍼를 생성한다.
alGenBuffers(NUMBUFFERS, uiStreamBuffer);

// stream playback을 위한 Source 한 개를 생성한다.
alGenSources( 1, &uiStreamSource );

// 루프를 돌며 버퍼에 Wav 파일을 로드한다.
for (int i = 0; i < NUMBUFFERS; i++) {
      if (!ALFWLoadWaveToBuffer(ALFWaddMediaPath(stream_wav_files[i]), uiStreamBuffer[i],
                                &StreamWaveID[i])) {
            ALFWprintf("Failed to load %s\n", stream_wav_files[i]);
      }
}
```


Code for Stream Playing(계속)

```
// 각 버퍼에 Wav 파일을 로드했으면 이 버퍼를 Source에 Queuing 한다.
alSourceQueueBuffers(uiStreamSource, NUMBUFFERS, uiStreamBuffer);

// Start playing source
alSourcePlay(uiStreamSource);
ALFWprintf("Start stream playing\n");

// stream 재생이 끝났는지 검사한다.
while (!ALFWKeyPress()) {
      Sleep( SERVICE_UPDATE_PERIOD );

      // 스트림 재생이 끝난 버퍼의 개수를 알아낸다.
      iBuffersProcessed = 0;
      alGetSourcei(uiStreamSource, AL_BUFFERS_PROCESSED, &iBuffersProcessed);
      alGetSourcei(uiStreamSource, AL_BUFFERS_QUEUED, &iQueuedBuffers);
      ALFWprintf("Buffers Processed %d, Remaining Buffers %d\r", iBuffersProcessed,
                                                                 iQueuedBuffers-iBuffersProcessed);
      alGetSourcei(uiStreamSource, AL_SOURCE_STATE, &iStreamState);
      if (iStreamState != AL_PLAYING)          // 재생 중이 아니면 나간다.
            break;
} // end of while (!ALFWKeyPress())

// 재생이 끝났으면 Source를 재생 정지시키고, Queue를 clear 한다.
alSourceStop(uiStreamSource);
alSourcei(uiStreamSource, AL_BUFFER, 0);

// 생성했던 Source와 버퍼들을 제거한다.
alDeleteSources(1, &uiStreamSource);
alDeleteBuffers(4, uiStreamBuffer);
```

그 밖의 재생 관련 기능들

`void alSourcePause(ALuint source)`

- 재생 중인 Source의 재생을 일시 멈춘다.
- Source 상태는 AL_PLAYING → AL_PAUSED 로 바뀜.
- alSourcePlay()를 다시 호출해주면 멈춘 위치부터 재생 재개.

`void alSourceRewind(ALuint source)`

- Source 재생을 멈추고, Source의 상태를 AL_INITIAL로 되돌린다.
- alSourcePlay()를 다시 호출하면 처음부터 재생

Report

- **5장에서 배운 재생 제어 기능들은 중간 프로젝트를 위해 매우 중요한 내용입니다. 다음과 같은 기능들을 코딩하고 Full Document가 달린 소스 코드와 보고서를 제출하시오.**
  - ① 배경음악으로 사용될 음원 30초짜리를 준비하고, 볼륨을 상당히 낮춰 무한 반복 재생시킨 후,
  - ② 본인이 원하는 음악 5개를 준비하고, 이를 Stream Play 시킨 후, 재생이 종료될 때까지 대기한다. 재생이 종료되면,
  - ③ 배경음악 재생을 중지시키고, ②에서 준비했던 5개의 음원을 동시 재생 시킨다. 모든 재생이 종료될 때까지 대기한다.

Q & A