Sound Programming

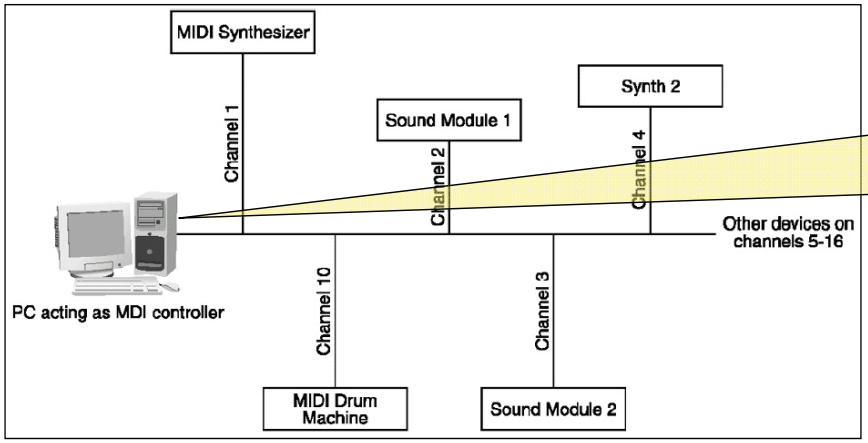
제6장 MIDI Playback in OpenAL

Overview

- 지금까지는 OpenAL에서
  - Wave 사운드 파일을 Load 하고, 재생하고, 재생 효과를 어떻게 하는지에 대해 다루었음.
- 이번 장에서는 **MIDI Playback**에 대해 알아본다.
  - MIDI는 배경음악으로 많이 이용됨
- **MIDI**
  - 대표적인 Synthesized Static Music의 일종
    - Static Music 이란 Wav 같은 음원을 담고 있지 않고 "Note on 같은 명령을 담고 있다"는 의미
  - 불행히도 OpenAL에서는 MIDI playback을 지원하지 않음.
  - 이번 장에서는 Win32에서 지원하는 MCI 함수들을 이용하여 MIDI 재생 라이브러리를 직접 만들고, 재생시켜본다.

How MIDI Works

- **Musical Instrument Digital Interface**
  - 전자 음악 악기들과 컴퓨터 사이의 통신을 위한 일종의 de-facto standard로 자리 잡음.



컴퓨터나 **Synthesizer**가 미디 케이블 상에 떠다니는 **"Note On"** 메시지를 잡아 저장하고 재생하는 방식으로 **MIDI** 파일을 관리할 수 있음!

  - 위 장비들 중 하나에서 MIDI cable에 떠다니는 "Note On" 신호를 잡아 재생하거나, 저장할 수 있으면 OK!

"Note On" message

- **MIDI 파일 재생 예**
- **Note On 메시지**
  - MIDI 장치에서 사운드를 발생시켰을 때 MIDI 케이블에 전송되는 데이터
  - "옥타브" + "볼륨" + "길이" 등의 정보로 구성
  - 메시지 해석은 각 MIDI 장치마다 다를 것임!
- **MIDI message의 유형**
  - 가장 대표적인 MIDI 메시지가 "Note On" 메시지,
  - 음색을 바꾸라고 장비에게 지시하는 메시지도 있고,
  - 악기가 지원하는 떨림/굴절 같은 특수 음향 효과를 내라는 특수 메시지도 있음.

Sequencer Software

- **Sonar, Sibelius, …**
  - MIDI Port로 들어오는 MIDI 메시지를 순서대로 모으고, 합성해주는 기능
  - MIDI에 16개의 채널만 있다고 해서 동시에 재생 가능한 Note가 16개 인 것은 아님!
    - 한 악기에서 동시에 여러 Note를 재생할 수 있기 때문
- **Win32 MCI로 MIDI 사운드를 재생하려면,**
  - 출력을 H/W MIDI port가 아닌 윈도우 내에 있는 software synthesizer로 함으로써, 거기서 합성된 후 스피커로 나가도록 하면 됨.
- 이번 장에서는 ***Win32 MCI를 이용한 MIDI 파일 재생 클래스 라이브러리***를 이용해 **MIDI** 파일을 재생할 것임

CMidi 클래스 구조

```
class CMidi {
public:
    CMidi();                    // 생성자
    virtual ~CMidi();           // 소멸자

    int GetDevices(void);       // 시스템에 연결되어 있는 MIDI 출력 장치의 개수를 알아본다.
    BOOL Open(const char* Filename);  // MIDI 파일을 open한다.
    void Close(void);                 // MIDI 파일을 close 한다.
    BOOL Play(void);                  // MIDI 파일 재생 시작
    void Stop(void);                  // 재생 중인 MIDI 파일 재생 중지
    void Pause(void)                  // 재생 중인 MIDI 파일 재생 일시 중지
    // 입력된 MIDI 파일의 총 재생 길이를 분과 초로 나누어 넘겨준다.
    void GetMidiLength(int *pnMinutes, int *pnSeconds);
    int GetMinutes(void);             // 현재 재생의 분 위치를 알아낸다.
    int GetSeconds(void);             // 현재 재생의 초 위치를 알아낸다.
    BOOL IsPlaying(BOOL *pbPaused);   // 현재 open된 MIDI 파일이 재생 중인지 알아낸다.
    // 주어진 분과 초 위치로 재생 위치를 이동하여 재생한다.
    BOOL SeekTo(int nMinute, int nSecond);
    void Forward(int nSeconds);       // 주어진 초 만큼 건너뛴다.
    void Backward(int nSeconds);      // 주어진 초 만큼 뒤로 건너 뛴다.
    int GetTempo(void);               // 현재 재생 템포를 알아낸다.
    void SetTempo(int nTempo);        // 새로운 재생 템포를 설정한다.

private:
    BOOL m_bOpened;             // 현재 open되어 있는지의 여부
    BOOL m_bPaused;             // 현재 재생이 일시 중지되어 있는지의 여부
    BOOL m_bPlaying;            // 현재 재생 중인지의 여부
    WORD m_wDeviceID;           // open된 MIDI 파일 또는 장치의 번호
};
```

CMidi 클래스를 이용한 .mid 파일 재생

```
#include "Framework.h"
#include "Midi.h"  // CMidi 클래스를 사용하려면 반드시 포함해야 함

int main()
{
    ……

    // MIDI 재생 샘플 코드 시작
    CMidi m_Midi;  // mid 파일 하나당 CMidi 클래스 인스턴스 하나
    const char* midi_file_name = "aria.mid";
    int nMinutes, nSeconds;
    BOOL bPaused;
    DWORD dwCurrentSecs,dwTotalSecs;

    if (!m_Midi.Open(ALFWaddMediaPath(midi_file_name))) {
          ALFWprintf("%s 파일 오픈 실패!\n", ALFWaddMediaPath(midi_file_name));
          return -1;
    }

    // Display the total play length
    m_Midi.GetMidiLength (&nMinutes, &nSeconds);
    if (nMinutes == -1) {
          nMinutes = 0;
          nSeconds = 0;
    }
    dwTotalSecs=(nMinutes*60)+nSeconds;
    ALFWprintf("총 재생 시간: %02d분 %02d초\n", nMinutes, nSeconds);
```


CMidi 클래스를 이용한 mid 파일 재생 (계속)

```
    // 재생 시작
    m_Midi.Play();
    Sleep(1000);    // MIDI 파일을 울릴 시간을 준다.

    // 종료 검사
    do {
         nMinutes = m_Midi.GetMinutes();
         nSeconds = m_Midi.GetSeconds();
         dwCurrentSecs = (nMinutes*60)+nSeconds;
         ALFWprintf("%02d:%02d\r", nMinutes, nSeconds);
         Sleep(250);
    } while (m_Midi.IsPlaying(&bPaused));
    ALFWprintf("\n");
}
```

기타 CMidi 클래스 멤버 함수들

```
m_Midi.Stop();              // 재생 정지
m_Midi.Pause();             // 재생 일시 멈춤
m_Midi.Forward(5);          // 앞 방향으로 5초 전진 재생
m_Midi.Backward(5);         // 뒷 방향으로 5초 후퇴 재생

// 재생 중 임의의 위치로 이동하여 재생하는 기능
int nSliderPos, nSliderSecs;
int nMinutes, nSeconds, nTotalSecs;

m_Midi.GetMidiLength(&nMinutes,&nSeconds);
nTotalSecs = (nMinutes*60)+nSeconds;
nSliderSecs = (nTotalSecs/100)*비율;     // 원하는 초 설정

nMinutes = nSliderSecs/60;
nSeconds = nSliderSecs%60;

m_Midi.SeekTo(nMinutes,nSeconds);
```


Tempo 변경하기

- **Tempo** 조절 : **MIDI** 파일에서만 가능한 사운드 효과
  - 긴박한 장면에서는 템포를 빨리 하여 긴장감을 높일 수 있다면…
  - MP3나 Wav 파일 재생 시에는 할 수 없는 기능임.
- **CMidi** 클래스의 **GetTempo()/SetTempo()** 메소드 이용

```
// 종료 검사 시 템포 조절하는 예제
do {
      nMinutes = m_Midi.GetMinutes();
      nSeconds = m_Midi.GetSeconds();
      tempo = m_Midi.GetTempo();
      dwCurrentSecs = (nMinutes*60)+nSeconds;
      ALFWprintf("%02d:%02d\r", nMinutes, nSeconds);
      Sleep(250);
      if (ALFWKeyPress()) {
            key = _getch();
            if (key == 'u')
                  m_Midi.SetTempo(tempo + 50);
            else if (key == 'd')
                  m_Midi.SetTempo(tempo - 50);
      }
} while (m_Midi.IsPlaying(&bPaused));
ALFWprintf("\n");
```

Homework

- **CMidi** 클래스를 이용해 **.mid** 파일을 재생하면서 다음의 **Cmidi** 클래스 멤버 함수들을 사용하는 프로그램을 작성하라.
  - 종료 검사를 해야 함.
  - 'f'가 입력되면 5초 Fast Forward
  - 'b'가 입력되면 5초 backward
  - 'u'가 입력되면 템포 증가. 증가 정도는 각자 판단
  - 'd'가 입력되면 템포 감소. 감소 정도는 각자 판단
- 제출물
  - Full Comment가 달린 소스코드가 포함된 프로젝트 폴더
  - 프로그램 제작 설명서
    - 여기엔 소스 코드 포함하면 안 됨.

Q & A