# Direct Sound

305890
2008년 봄학기
5/14/2007
박경신
kpark@dankook.ac.kr

## DirectSound

- DirectSound
  - DirectX에서는 사운드를 사용하기 위해서 다이렉트 사운드 (DirectSound) API를 제공한다.
  - DirectSound를 사용하기 위해 d3d9.lib, d3dx9.lib, winmm.lib, dsound.lib, dxerr9.lib, dxguid.lib 을 프로젝트에 링크

## DirectSound API

- DirectSound API
  - 게임에서 사운드 효과를 부여함
  - 윈도우 상에서 작동되는 사운드 카드를 지원함
  - PCM형태의 디지털 사운드 출력 (WAV 파일)
  - 2차원 사운드와 3차원 사운드로 분류
  - 3차원 사운드 emulating
  - 2-스피커, 4-스피커 등의 옵션 지정가능
  - 파일 입출력을 지원하지 않음

## Direct Sound

- DirectSound가 지원하는 일
  - WAV 형식의 파일 재생
  - 여러 개의 사운드를 동시에 재생
  - 하드웨어 버퍼에 고음질의 사운드 저장
  - 3차원 환경에서 사운드 재생
  - 에코, 코러스 등의 효과
  - 마이크나 다른 입력 장치로부터 사운드 녹음
- DirectSound 사운드 버퍼
  - 사운드 데이타를 사운드 버퍼(sound buffer)를 사용하여 관리한다.
  - 다양한 사운드를 가진 여러 개의 버퍼를 가질 수 있다.
  - 버퍼가 제어되고 재생될 수 있으며, 또는 새로운 버퍼 자료를 생성하기 위해 믹싱(mixing)도 가능하다.
  - 사운드 버퍼는 사운드 데이터를 갖는 영역이다.
  - 예를 들어, WAV 파일을 사운드 버퍼에 읽어 들여서, 파일에 있는 사운드 자료를 버퍼에 넣어 재생 또는 제어를 할 수 있다.

## Direct Sound

- 사운드 버퍼의 종류
  - 주버퍼 (primary buffer)
    - 모든 재생되고자 하는 사운드가 주버퍼에서 믹싱(mix)되어 사운드 카드에서 출력된다.
  - 보조 버퍼 (secondary buffer)
    - 보조버퍼는 우리의 응용프로그램에서 필요한 모든 사운드 데이터를 가지는 버퍼이다.
    - 다이렉트 사운드는 하나 이상의 보조버퍼를 사용하여 여러 개의 사운드를 동시에 재생할 수 있다.
  - 정적 버퍼 (static buffer)
    - 사운드 데이터가 일정 크기를 가지고 있을 때 정적 버퍼를 사용한다.
    - 정적 버퍼는 특정 사운드 전체를 메모리로 읽어 들일 수 있다.
  - 스트리밍 버퍼 (streaming buffer)
    - 사운드 데이터가 너무 커서 메모리에 한꺼번에 들어가지 않을 때 스트리밍 버퍼를 사용한다.
    - 사운드가 재생되면서 스트리밍 버퍼에 새로운 사운드 데이터를 읽어 들인다.

## Direct Sound Interface

- IDirectSound8: 사운드 카드의 기능을 결정하고 재생을 위한 버퍼 생성에 사용
- IDirectSoundBuffer8: 재생 가능한 사운드 데이터를 담기 위한 버퍼
- IDirectSound3DBuffer8: 3차원 사운드를 담기 위한 버퍼
- IDirectSound3DListerner8: 3차원 listener를 표현에 사용
- IDirectSoundCapture8: Capture buffer 생성에 사용
- IDirectSoundCaptureBuffer8: 마이크와 같은 장치에서 녹음한 데이타를 저장하기 위한 버퍼
- IDirectSoundNotify8: 사운드 재생 종료를 알림
- IKsPropertySet8: 사운드 카드 제조업체에서 새로운 기능을 넣기 위해 사용하는 인터페이스

## DirectSound 설정

- DirectSound 설정
  1. DirectSound의 객체를 생성한다. DirectSoundCreate8() 함수 사용.
  2. 협력 레벨을 설정한다. IDirectSound8::SetCooperativeLevel() 함수 사용.
  3. 보조 사운드 버퍼를 생성한다. IDirectSound8::CreateSoundBuffer() 함수 사용.
  4. 생성된 보조 사운드 버퍼에 사운드 정보를 복사한다.
  5. 마지막으로, 프로그램 종료 시에는 반드시 사용중인 객체들을 해제시킨다.

## Using DirectSound

- 다이렉트 사운드 (DirectSound)
  - 다이렉트 사운드를 사용하려면 반드시 초기화가 필요하다.
  - 첫 번째로, IDirectSound8 인터페이스로 표현되는 다이렉트 사운드 장치를 사용해야 한다.
- 다이렉트 사운드 장치 (DirectSound Device)
  - 컴퓨터 안에서 사운드 하드웨어 특정 부분의 인터페이스를 말한다.
  - 다이렉트 사운드가 작동되도록 하려면 사운드 카드를 선택하고 다이렉트 사운드 장치를 생성해야 한다.
  - 다이렉트 사운드 장치 생성은 DirectSoundCreate8를 사용한다.

## Using DirectSound

```
HRESULT DirectSoundCreate8(LPCGUID lpcGuidDevice,
        LPDIRECTSOUND8 *ppDS8, // LPDIRECTSOUND8의 포인터 변수
        LPUNKNOWN pUnkOuter); // 항상 NULL이다
```

- lpcGuidDevice
  - GUID는 사용하고자 하는 사운드 카드 장치를 말한다
  - DSDEVID_DefaultPlayback 나 NULL을 사용한다.
  - NULL은 디폴트 사운드 장치 (default sound device)를 사용한다는 의미
- ppDS8
  - 생성된 다이렉트 사운드 장치를 지칭하는 변수의 포인터
- pUnkOuter
  - 제어하고자 하는 객체의 IUnknown interface이다. 항상 NULL이다.

## Using DirectSound

```
// variable that will hold the return code
HRESULT hr;

// variable that will hold the created DirectSound device
LPDIRECTSOUND8 m_pDS = NULL;

// Attempt to create the DirectSound device
hr = DirectSoundCreate8(NULL, &m_pDS, NULL);

// Check the return value to confirm that a valid device was created
if (FAILED(hr)) return false;
```

## Setting the Cooperative Level

- 하드웨어 장치에 접근할 수 있게 협력 레벨을 지정해야 한다.
- 다이렉트 사운드가 제공하는 4가지의 협력 레벨
  - DSSCL_NORMAL
    - 일반적인 협력 레벨. 이 레벨은 다른 이벤트 발생을 가능하게 해서 다른 응용프로그램과 같이 쓰기에 가장 좋다.
    - 응용프로그램이 활성화 중일 때만 사운드를 출력한다.
    - 주버퍼를 제어하기 위한 코드를 작성할 필요가 없으며, 대부분이 이 설정을 사용한다.
    - 하드웨어 장치가 다른 응용프로그램과 공유해서 쓰기 때문에 주버퍼의 포멧을 바꿀 수 없다.
  - DSSCL_PRIORITY
    - 우선순위 협력레벨. 사운드 카드에 접근할 수 있는 권한을 가지게 되며, 주 사운드 버퍼의 일반적인 제어권이 프로그래머에게 있게 된다.
    - 이 레벨은 응용프로그램에서 사운드 압축과 같은 주버퍼의 데이터 형식 변경 작업을 필요로 할 경우 사용된다.
    - 대부분의 게임에서는 이 레벨을 사용하고 있다.

## Setting the Cooperative Level

- 다이렉트 사운드가 제공하는 4가지의 협력 레벨
  - DSSCL_EXCLUSIVE
    - 독점적 협력레벨. 우선순위 협력레벨과 비슷하지만 응용프로그램이 활성화되어 있을 때만 사운드를 출력할 수 있다.
  - DSSCL_WRITEPRIMARY
    - 가장 높은 협력레벨로서, 이 레벨은 응용프로그램에게 주버퍼의 모든 제어 권한을 준다.
    - 즉, 재생, 정지 등의 동작들을 프로그래머가 직접 제어해야 한다.
    - 자신만의 사운드 믹서, 사운드 엔진을 만들 때에만 이 레벨을 사용한다.

## Setting the Cooperative Level

- 협력 레벨을 설정하기 위해서 SetCooperativeLevel 함수를 사용한다.

HRESULT SetCooperativeLevel(HWND hWnd, DWORD dwLevel);

- hWnd
  - 협력레벨 바꾸기를 요청하려는 응용프로그램의 윈도우 핸들
- dwLevel
  - 협력레벨

## Setting the Cooperative Level

```
HRESULT hr;

// Create the DirectSound device
LPDIRECTSOUND8      g_pDS = NULL;
hr = DirectSoundCreate8(NULL, &g_pDS, NULL);

// Set the DirectSound cooperative level
hr = g_pDS->SetCooperativeLevel(hWnd, DSSCL_PRIORITY);

if (FAILED(hr)) return false;
```

## Sound Files

- 사운드 데이터가 다이렉트 사운드의 보조 버퍼로 읽혀서 사용된다.
- 사운드 데이터는 정적 버퍼 또는 스트리밍 버퍼로 읽혀 들인다.
- 정적 버퍼 (Static buffer)
  - 사운드 전체를 읽어 들일 수 있는 고정된 길이의 버퍼이다.
- 스트리밍 버퍼 (Streaming buffer)
  - 버퍼가 수용할 수 있는 양보다 더 큰 대용량 사운드를 사용할 때
  - 작은 버퍼가 사용된다.
  - 사운드 데이터의 일부가 계속 읽어 들이면서 재생된다.

## The Secondary Buffer

- 사운드 데이터가 재생되는 단계
  - 다이렉트 사운드는 오디오 데이터를 저장하는데 버퍼를 사용한다.
  - 재생하려는 오디오 데이터를 저장하기 위해 보조 버퍼를 생성해야 한다.
  - 버퍼가 생성된 후에 사운드 전체를 버퍼에 올려놓는다. (또는 스트리밍 사운드의 경우 사운드의 일부를 올려놓는다.)
  - 그리고 사운드를 재생한다.
- 다이렉트 사운드는 여러 개의 보조 버퍼를 지원하며 동시에 재생 가능하다. 이 때 여러 개의 재생되는 사운드는 주버퍼에서 믹싱된다.
- 보조버퍼를 생성하기 전에 사운드의 포멧을 알아야 한다.
- 다이렉트 사운드는 같은 포멧으로 버퍼를 사용해야 한다.
- 예를 들어, 16-bit 2-채널 WAV파일 경우 보조버퍼도 이 파일 포멧으로 생성되어야 한다.

## WAVEFORMATEX structure

- 다이렉트 사운에서 버퍼의 포멧 지정: WAVEFORMATEX
  - 만약 wav 파일 포멧이 알려진 것이라면, 일반적인 WAVEFORMATEX 구조체를 생성한다.
  - 만약 파일 포멧이 모르는 것이라면, 오디오 파일을 열은 후에 이 구조체의 값을 채워야 한다.

```
typedef struct {
    WORD wFormatTag;
    WORD nChannels;
    DWORD nSamplesPerSec;
    DWORD nAvgBytesPerSec;
    WORD nBlockAlign;
    WORD wBitsPerSample;
    WORD cbSize;
} WAVEFORMATEX;
```

## WAVEFORMATEX structure

- wFormatTag
  - 오디오의 종류
  - 1-채널 또는 2-채널 PCM 자료의 경우 이 값은 WAVE_FORMAT_PCM이다.
- nChannels
  - 사운드의 채널 수. 1일 때는 MONO이고 2일 때는 STEREO이다.
- nSamplesPerSec
  - 샘플링 주기, 즉 주파수 (Mhz)
- nAvgBytesPerSec
  - 평균 데이터-전송 비율 (in bytes per second)
- nBlockAlign
  - 블럭당 전체 데이터를 설정 (in bytes)
  - nChannels * wBitsPerSample / 8
- wBitsPerSample
  - 샘플당 비트 수 (8 또는 16)
- cbSize
  - 추가적으로 더 필요한 바이트 수. 항상 0 이다.

## The Secondary Buffer

- 보조 사운드 버퍼 구조체: DSBUFFERDESC

```
typedef struct {
    DWORD dwSize;
    DWORD dwFlags;
    DWORD dwBufferBytes;
    DWORD dwReserved;
    DLPWAVEFORMATEX lpwfxFormat;
    GUID guid3DAlgorithm;
} DSBUFFERDESC, *LPDSBUFFERDESC;
```

## DSBUFFERDESC structure

- dwSize
  - DSBUFFERDESC 구조체의 크기 (in bytes)
- dwFlags
  - 보조 사운드 버퍼의 플래그 정보를 지정
- dwBufferBytes
  - 새로운 사운드 버퍼의 크기 (in bytes)
  - 이 버퍼가 가지고 있는 사운드 자료의 바이트 수
- dwReserved
  - 사용하지 않는다. 항상 0이다.
- lpwfxFormat
  - WAVEFORMATEX 구조체 정보를 받는다.
- guid3DAlgorithm
  - 사용하는 두 개의 스피커 가상 알고리즘 GUID 식별자

## DSBUFFERDESC structure

- dwFlags에 자주 사용되는 플래그
  - DSBCAPS_CTRLALL: 버퍼는 모든 제어 기능을 가진다.
  - DSBCAPS_CTRLDEFAULT: 버퍼는 기본 제어 옵션을 가진다. 이 값은 DSBCAPS_CTRLVOLUME, DSBCAPS_CTRLFREQUENCY를 지정하는 것과 동일하지만, DirectX6.0이후부터 없어졌다.
  - DSBCAPS_CTRLFREQUENCY: 버퍼가 주파수 제어 기능을 가진다.
  - DSBCAPS_CTRLPAN: 버퍼가 팬 (pan) 기능을 가진다.
  - DSBCAPS_CTRLVOLUME: 버퍼가 볼륨제어 기능을 가진다.
  - DSBCAPS_STATIC: 버퍼가 정적 사운드 데이터에 사용될 것임을 알린다. 대부분 하드웨어 (사운드카드) 메모리에 생성한다.
  - DSBCAPS_LOCHARDWARE: 메모리가 사용가능 하다면 하드웨어 메모리에 사운드 버퍼를 생성하며 하드웨어 믹싱을 사용한다.
  - DSBCAPS_LOCSOFTWARE: 시스템 메모리(RAM)에 사운드 버퍼를 생성하며 소프트웨어 믹싱을 사용한다.
  - DSBCAPS_PRIMARYBUFFER: 주 사운드 버퍼로 생성한다. 이 플래그를 주지 않으면 기본값으로 보조 사운드 버퍼로 생성된다.

## Creating a Secondary Buffer

- DSBUFFERDESC 구조체를 지정한 후, 보조 사운드 버퍼를 생성한다.

```
HRESULT CreateSoundBuffer(LPCDSBUFFERDESC pcDSBufferDesc,
                          LPDIRECTSOUNDBUFFER *ppDSBuffer,
                          LPUNKNOWN pUnkOuter);
```

  - pcDSBufferDesc
    - 이미 지정된 DSBUFFERDESC 구조체를 가리킨다.
  - ppDSBuffer
    - 새롭게 생성된 버퍼를 가질 변수를 가리킨다.
  - pUnkOuter
    - 제어하고자 하는 객체의 IUnknown interface를 가리킨다.
    - 항상 NULL이다.

## Creating a Secondary Buffer

```
// Define a WAVEFORMATEX structure
WAVEFORMATEX wfx;

// Clear the structure to all zeros
ZeroMemory(&wfx, sizeof(WAVEFORMATEX));

// Set the format to WAVE_FORMAT_PCM
wfx.wFormatTag = (WORD) WAVE_FORMAT_PCM;
wfx.nChannels = 2; // set channels by 2
wfx.nSamplesPerSec = 22050;
wfx.wBitsPerSample = 16;
wfx.nBlockAlign = (WORD) (wfx.wBitsPerSample / 8 * wfx.nChannels);
wfx.nAvgByPerSec = (DWORD) (wfx.nSamplesPerSec * wfxnBlockAlign);
```

## Creating a Secondary Buffer

```
DSBUFFERDESC dsbd;
ZeroMemory(&dsbd, sizeof(DSBUFFERDESC));
dsbd.dwSize = sizeof(DSBUFFERDESC);
dsbd.dwFlags = 0;
dsbd.dwBufferBytes = 64000;
dsbd.guid3DAlgorithm = GUID_NULL;
dsbd.lpwfxFormat = &wfx;

LPDIRECTSOUNDBUFFER DSBuffer = NULL;
hr = g_pDS->CreateSoundBuffer(&dsbd, &DSBuffer, NULL);
if (FAILED(hr)) return NULL;
```

## Locking the Sound Buffer

- 사운드 버퍼의 잠금
  - 사운드 버퍼 잠금은 버퍼에서 사운드 데이터를 제어하고 바꿀 수 있게 한다.
  - 잠금 후에는 버퍼에 사운드 데이터를 불러올 수 있다.
  - 사용 후에 잠금 상태를 풀어줘야 한다.

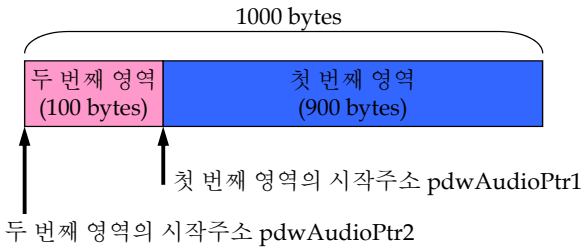
```
HRESULT Lock(
    DWORD dwOffset,
    DWORD dwBytes,
    LPVOID *ppvAudioPtr1,
    LPDWORD pdwAudioBytes1,
    LPVOID *ppvAudioPtr2,
    DPDWORD pdwAudioBytes2,
    DWORD dwFlags);
```

## Locking the Sound Buffer

- dwOffset
  - 어디서부터 잠글 것인가를 설정
- dwBytes
  - 얼마만큼 잠글 것인가를 설정 (in bytes)
- ppAudioPtr1
  - 잠글 버퍼에서 첫 번째 영역의 주소
- pdwAduioBytes1
  - 첫 번째 영역의 길이 (in bytes)
- pdwAudioPtr2
  - 잠글 버퍼에서 두 번째 영역의 주소
  - 만약 사운드 데이터로 버퍼 전체를 채운다면, NULL을 사용한다.
- pdwAudioBytes2
  - 두 번째 영역의 길이 (in bytes)
  - pdwAudioPtr2 가 NULL이면, 이 값도 NULL을 사용한다.

## Locking the Sound Buffer

- dwFlags
  - 어떻게 잠글 것인가를 설정하는 플래그
  - DSBLOCK_FROMWRITECURSOR: 버퍼의 현재 기록커서 (write cursor)로 부터 잠겨지게 된다.
  - DSBLOCK_ENTIREBUFFER: 전체 버퍼가 잠겨지게 된다. 이 플래그가 설정되면, dwBytes 값은 무시한다.




첫 번째 버퍼에 있는 사운드를 재생하는 동안에 두 번째 버퍼에는 나머지 사운드 부분을 저장한다. 그리고, 첫 번째 버퍼에 있는 사운드 재생이 끝나면 곧바로 두 번째 버퍼에 저장된 사운드를 재생하게 된다. 이렇게 해서, 사운드의 처리 속도를 빠르게 하고 사운드 재생이 끊기지 않고 반복될 수 있게 한다.

## Unlocking the Sound Buffer

- 사운드 버퍼의 잠금 상태를 풀어야 한다.
  - 사운드 데이터를 버퍼에 복사하고 난 후, Unlock을 한다.

```
HRESULT Unlock(LPVOID pvAudioPtr1, DWORD dwAudioBytes1,
               LPVOID pvAudioPtr2, DWORD dwAudioBytes2);
```

- pvAudioPtr1
  - 잠금에서 사용됐던 첫 번째 영역의 주소
- dwAudioBytes1
  - pvAudioPtr1에 사용됐던 첫 번째 영역의 길이 (in bytes)
- pvAudioPtr2
  - 잠금에서 사용됐던 두 번째 영역의 주소
- dwAudioBytes2
  - pvAudioPtr2에 사용됐던 두 번째 영역의 길이 (in bytes)

## Reading the Sound Data into the Buffer

- Loading Sound Data
  - 다이렉트 사운드에는 사운드 파일을 불러오는 기능이 없다.
  - 따라서, DirectX DSK에 포함되어있는 dsutil.cpp 사용하여 사운드 파일을 불러오는 것을 할 것이다.
- Loading Process
  1. CWaveFile 객체를 생성한다.
  2. WAV 파일을 Open한다.
  3. WAV 데이터를 담고 있을 보조 버퍼를 생성한다.
  4. 버퍼를 잠근다.
  5. 사운드 데이터를 읽고 복사한다.
  6. 버퍼 잠금 상태를 푼다.

## Reading the Sound Data into the Buffer

1. CWaveFile 객체를 생성한다.

```
CWaveFile wavFile = new CWaveFile();
```

2. WAV 파일을 Open한다.
   - 아래는 test.wav 파일을 열고 읽는 예제이다.
   - 만약 파일에 데이터가 없다면 (즉, size=0) 이면 중단한다.

```
// open "test.wav"
wavFile->Open("test.wav", NULL, WAVEFILE_READ);
// Check to make sure that the size of data within the wave file is valid
if (wavFile->GetSize() == 0) return false;
```

3. WAV 데이터를 담고 있을 보조 버퍼를 생성한다.

## Reading the Sound Data into the Buffer

4. 버퍼를 잠근다.

```
HRESULT hr;
VOID *pDSLockedBuffer = NULL; // pointer to locked buffer memory
DWOR dwDSLockedBufferSize = 0; // size of the locked buffer
// Start the beginning of the buffer
hr = DSBuffer->Lock(0,
                    // This assumes a buffer of 64000 bytes
                    64000,
                    // The variable holds a pointer to the start of the buffer
                    &pDSLockedBuffer,
                    // holds the size of the locked buffer
                    &dwDSLockedBufferSize,
                    NULL, // No secondary is needed
                    NULL, // No secondary is needed
                    DSBLOCK_ENTIREBUFFER); // Lock the entire buffer
if (FAILED(hr)) return NULL;
```

## Reading the Sound Data into the Buffer

5. 사운드 데이터를 읽고 복사한다.
   - WAV 파일을 열고 데이터를 읽기 전에 resetFile을 한다.
   - 그리고 나서, 데이터를 read한다.

```
HRESULT hr; // variable to hold the return code
// the amount of data read from the wav file
DWORD dwWaveDataRead = 0;
 // reset the WAV file to the beginning
wavFile->ResetFile();
// read the WAV file
hr = wavFile->Read((BYTE *) pDSLockedBuffer,
                              dwDSLockedBufferSize, &dwWaveDataRead);
if (FAILED(hr)) return NULL;
```

## Reading the Sound Data into the Buffer

6. 버퍼 잠금 상태를 푼다.

```
DSBuffer->Unlock(pDSLockedBuffer, dwDSLockedBufferSize,
NULL, NULL);
```

## Playing Sound in a Buffer

- 다이렉트 사운드 버퍼에 데이터가 올려지면 Play함수를 사용하여 사운드를 재생한다.

```
HRESULT Play(DWORD dwReserved1, DWORD dwPriority,
                DWORD dwFlags);
```

  - dwReserved1
    - 항상 0이다.
  - dwPriority
    - 사운드의 재성 우선순위
    - 0부터 0xFFFFFFFF의 값
    - 버퍼 생성시 DSBCAPS_LOCDEFER 플래그가 지정되지 않았다면, 0으로 한다.
  - dwFlags
    - 사운드가 어떻게 재생될 지를 지정하는 플래그, e.g. DSBPLAY_LOOPING

```
DSBuffer->Play(0, 0, DSBPLAY_LOOPING);
```

## Stopping a Sound

- 사운드 재생의 정지를 위해서는 Stop함수를 사용한다.

```
HRESULT Stop();

HRESULT hr;

hr = DSBuffer->Stop();
if (FAILED(hr)) return false;
```

## Controling the Volume

- 사운드의 볼륨 조절은 SetVolume 함수를 사용한다.
  - 볼륨은 DSBVOLUME_MIN (음소거)에서부터 DSBVOLUME_MAX (사운드의 원래 크기)까지 중에 지정할 수 있다.

```
HRESULT SetVolume(LONG lVolume);
```

  - lVolume
    - 0 (DSBVOLUME_MAX)에서 부터 -10000 (DSBVOLUME_MIN)까지 중의 값으로 지정할 수 있다.

- 현재 재생중인 사운드의 볼륨을 얻고자 할 때는 GetVolume 함수를 사용한다.

```
HRESULT GetVolume(LPLONG plVolume);
```

## Panning the Sound

- 사운드가 왼쪽 오른쪽 스피커 사이를 팬 (Panning)하기 위해서는 SetPan함수를 사용한다.
  - 한쪽 스피커에서 사운드의 크기를 줄이면서 다른 쪽 스피커의 사운드의 크기를 늘리면서 패닝 효과를 얻는다.
  - 그래서 사운드가 마치 움직여 다니는 것과 같은 효과를 얻는다.

HRESULT SetPan(LONG lPan);

- lPan
  - DSBPAN_LEFT에서부터 DSBPAN_RIGHT까지의 값을 갖는다.
  - DSBPAN_LEFT (-10000) 은 오른쪽 스피커의 사운드가 완전히 안들릴 때까지 왼쪽 스피커에서 사운드 크기를 증가시킨다.
  - DSBPAN_RIGHT (10000) 은 반대편에서 위와 같이 한다.
  - DSBPAN_CENTER를 0으로 지정하면 양쪽 스피커의 사운드 크기를 원음 크기로 되돌려 놓는다.

## Panning the Sound

- 현재 팬 사운드 값을 받고자 하면, GetPan함수를 사용한다.

HRESULT GetPan(LPLONG plPan);

- NOTE:
  - SetPan 또는 GetPan 함수를 사용하기 전에 이 콘트롤을 사용한다고 버퍼를 지정해야 한다.
  - 즉, 보조버퍼 생성 시 DSBUFFERDESC구조체에 DSBCAPS_CTRLPAN 플래그를 설정해야 한다.