11장 DLL, 사운드, 좌표 변환

2019년도 1학기 윈도우 프로그래밍

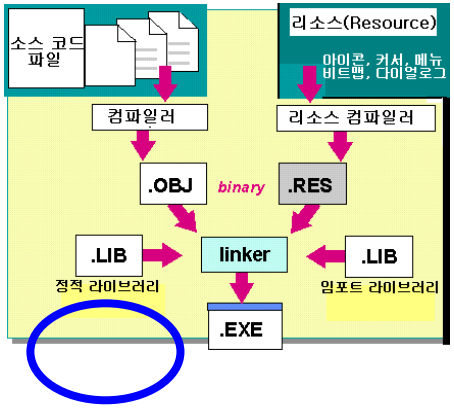
학습 내용

- 학습목표
  - DLL 만들기
  - 사운드 사용하기
  - 좌표 변화 관련 유용한 함수에 관하여 배운다.

- 내용
  - 동적 라이브러리 만들기
  - 사운드 라이브러리 사용하기
  - 몇 가지 유용한 함수
  - 좌표변환 함수

정적 링크 라이브러리(Static Link Library)

- 표준 C 라이브러리 함수: printf()같은 함수로 프로그래머를 위해서 제공하는 함수
  - 전부 컴파일되어 .lib파일 형태로 존재
  - 이 라이브러리 파일: 소스 프로그램 컴파일 후 링커에 의해 함께 링크되어 실행파일에 포함
- 이러한 라이브러리 연결 방법→ 정적 링크(static link)
  - 정적 링크 라이브러리: 컴파일시 링크가 되며 실행파일의 일부로 포함
  - 표준 함수뿐만 아니라 자신이 많이 사용하는 모듈을 .lib파일로 만든 후 현재 프로젝트에 포함시키고 컴파일/링킹을 하면 언제든지 관련 모듈은 따로 프로그래밍할 필요가 없다
- 정적 라이브러리를 사용하는 프로그램은 이미 실행 파일에 관련 모듈이 포함되어 있으므로 실행할 때는 해당 .lib파일 없이 실행할 수 있는 장점을 갖는다.
  - 그러나 .lib파일이 실행 파일의 일부가 되므로 실행파일의 크기가 대체로 커진다.



1. 동적 링크 라이브러리(DLL: Dynamic Link Library)

- DLL은 컴파일시가 아닌 실행시에 관련 모듈을 링크
  - 링크시 실행파일에 포함되지 않는다.
  - 링크 시에는 라이브러리 이름과 라이브러리에서 사용한 함수의 이름만 실행파일에 기록된다.
  - 실제로 실행파일이 실행되면 그때 라이브러리가 메모리에 따로 로드되고 라이브러리의 함수를 호출하게 되면 그 함수로 실행이 넘어가게 된다.

- DLL을 사용했을 때,
  - 일반적으로 DLL을 사용하는 프로그램은 실행파일 크기가 작다.
  - DLL을 사용하는 프로그램은 DLL만 교체하면 프로그램의 버전업을 쉽게 할 수 있다.

DLL의 특징

- 정적 라이브러리는 소스만 제공할 수 있지만 동적 라이브러리는 리소스도 제공할 수 있는 장점이 있다.
  - 자주 사용하는 비트맵 리소스가 있다면 이를 DLL파일에 저장하여 필요할 때 연결해 쓸 수 있다.
  - 대표적으로 글꼴 파일(.fon, .ttf)이 리소스만 갖는 DLL이다.

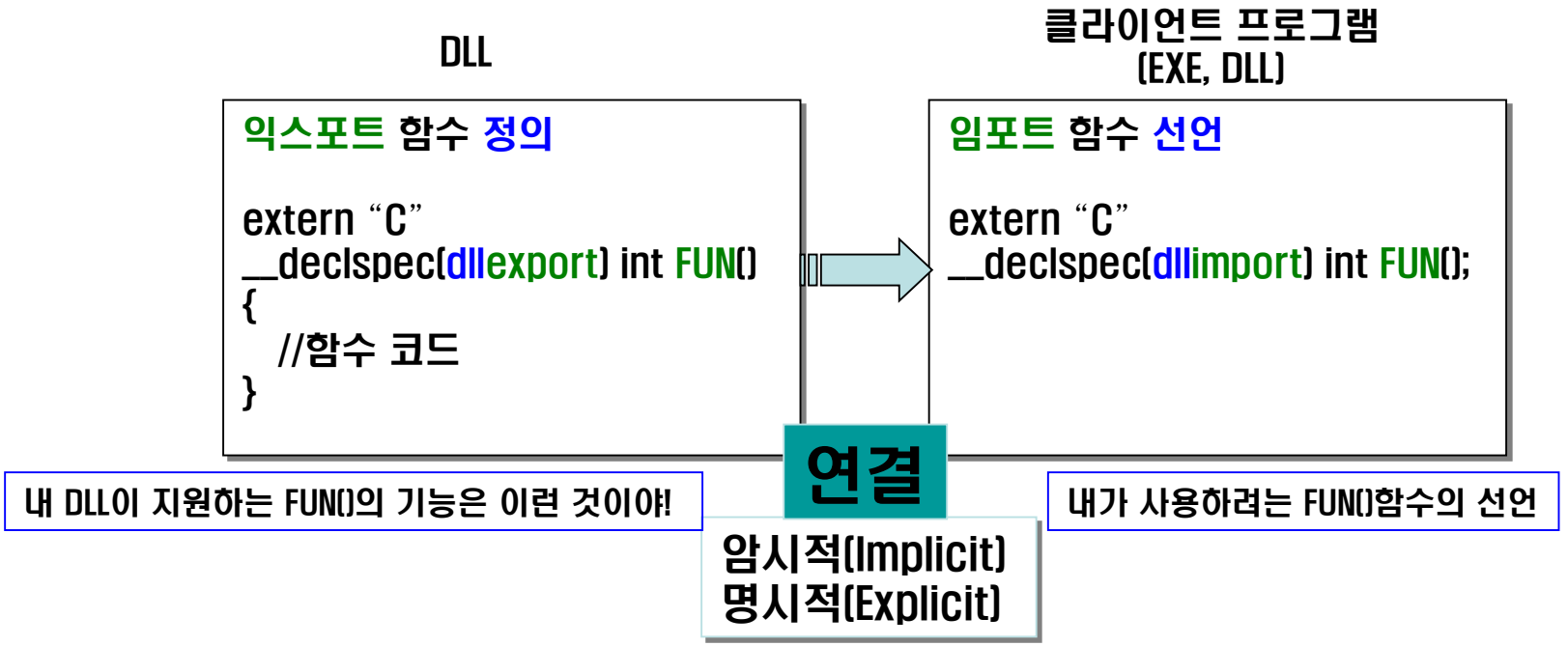
- DLL은 확장자가 반드시 .dll일 필요는 없지만 확장자가 .dll인 DLL만이 윈도우즈에 의해 자동적으로 로드된다.
  - DLL이 다른 확장자를 가지면, LoadLibrary()나 LoadLibraryEx()함수를 사용하여 강제로 로드한다.

- DLL을 사용할 경우 실행파일의 크기는 작아지지만 DLL파일을 별도로 배포해야 하며, 프로그래밍하기 복잡하고 버전관리 등에 신경을 써야 한다.
  - 디바이스 드라이버(keyboard.drv mouse.drv 등)나 각종 커스텀 컨트롤들은 모두 DLL로 만들어진다.
  - 공통 컨트롤은 comctl32.dll로 제공되며, 공통 대화상자는 comdlg32.dll로 제공된다.

DLL을 만들고 사용할 때의 규칙

- **익스포트(export)**
  - 함수를 제공하는 DLL에서는 자신이 제공하는 함수들의 정보를 공개해야 한다.
- **임포트(import)**
  - 만들어진 DLL를 사용하는 클라이언트(EXE나 DLL)에서는 어떤 DLL에 있는 어떤 함수를 사용한다는 선언을 해야 한다.

- 클라이언트에서 DLL의 함수를 호출하는 방법에는 암시적(implicit)방법과 명시적(explicit)방법이 있는데 다음 그림은 암시적 방법을 나타낸다.



DLL 만들고 사용할 때의 규칙

- DLL에서 함수를 익스포트하거나 클라이언트에서 임포트할 때는
  - **__declspec과 extern"C"** 를 같이 써야 한다.

  - **DLL에서 익스포트되는 함수 선언**
    extern "C" __declspec (dllexport)

  - **클라이언트에서는 임포트하는 함수의 원형을 사용**
    extern "C" __declspec (dllimport)

DLL 만들기 (dll1.dll)

- Visual C++에서 Win32 API로 DLL을 만들 때는 **응용 프로그램에서 DLL을 선택**
  - Project name: dllTest
  - 디폴트로 설정되어 있는 "An empty DLL project“
  - File/New/C++ Source File로 옆에 있는 사칙연산을 하는 4개의 함수를 갖는 cpp파일을 추가

- 디버그 모드로 컴파일하면 Debug폴더에 dllTest.dll과 dllTest.lib파일이 생긴다.

- dllTest.lib파일을 임포트 라이브러리(import library)라고 한다.
  - 이 파일은 DLL에 익스포트되는 함수 정보와 그 함수 코드가 어떤 DLL에 정의되어 있는지에 대한 정보를 가진 파일로 앞에서 설명한 일반 정적 lib파일과는 다르다.

- Visual C++은 DLL을 컴파일할 때 .dll과 같은 이름의 임포트 라이브러리(.lib)를 자동으로 만들어 준다

DLL 만들기

// 예) dll1.cpp 파일: 4개의 함수를 가지고 있는 dll 파일을 만든다.

```
extern "C"
__declspec (dllexport) int Add (int x, int y)    // 함수 이름을 변경하지 않게 C문법으로 컴파일해야 한다.
                                                 // 함수를 제공하는 DLL 에서는
{                                                // 자신이 제공하는 함수들의 정보를 공개
   return x+y;
}

extern "C"
__declspec (dllexport) int Sub (int x, int y)
{
   return x-y;
}

extern "C"
__declspec (dllexport) int Mul (int x, int y)
{
   return x*y;
}

extern "C"
__declspec (dllexport) double Div (int x, int y)
{
   return (double)x/y;
}
```

DLL 사용하기

- 만든 DLL을 사용하기 위해서는 사용하는 클라이언트(EXE나 DLL)프로그램이 만들어진 DLL을 연결해야 한다.

- **암시적(implicit) 방법**
  - .dll파일과 함께 제공되는 임포트 라이브러리(.lib)로 연결하는 방법이다.
  - 프로그램이 실행되면 DLL도 함께 로드되며 프로그램이 종료되면 DLL도 함께 해제된다.

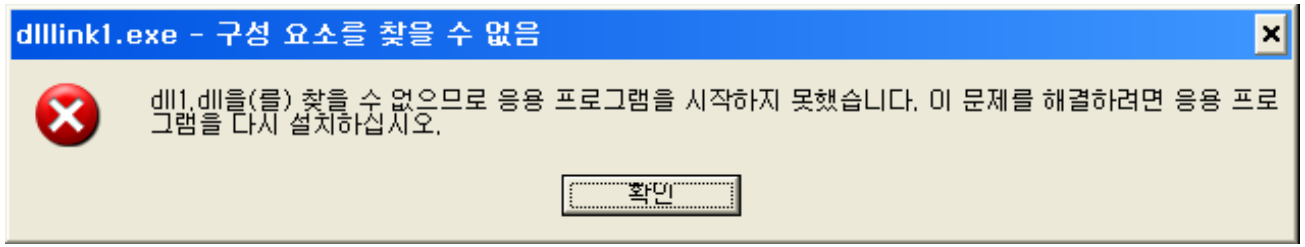
- **명시적(explicit) 방법**
  - LoadLibrary(), GetProcAddress(), FreeLibrary()같은 Win32 API함수를 사용하는 방법
  - 암시적 방법에 비해 장점이 있으나 함수 포인터 같은 어려운 개념을 알고 있어야 한다.

DLL 연결하기

	암시적(implicit)방법	명시적(explicit)방법
방 법	임포트 라이브러리(.lib)사용	API함수 사용
DLL이 메모리에 로드되는 시점	프로그램이 실행되면 DLL도 함께 로드됨	LoadLibrary()함수 호출시
DLL이 메모리에 해제되는 시점	프로그램이 종료되면 DLL도 함께 해제	FreeLibrary()함수 호출시
익스포트 함수 사용 시점	프로그램이 실행되는 동안 어디서나	GetProcAddress()함수 호출 후
장 점	함수 호출이 간편	메모리 소비 적고, DLL로딩 실패시에도 대응코드 가능
단 점	메모리 낭비와, DLL로딩 실패시 프로그램 실행 불가	함수 호출이 복잡
사용 예	1) .lib와 .dll을 현재 프로젝트 폴더에 복사 2) Project/Setting/Link탭을 열어 임포트 라이브러리 포함시킴 3) 임포트 함수 선언 4) 함수 사용	1) LoadLibrary() 2) GetProcAddress() 3) 함수 사용 4) FreeLibrary()

암시적 (implicit) 방법

- 앞에서 만들어 놓았던 dllTest.lib와 dllTest.dll을 **현재 프로젝트 폴더에 복사한다.**
- 이 프로그램을 실행할 때 반드시 다음의 디렉토리 중 하나에는 dllTest.dll파일이 있어야 하며 윈도우즈는 다음과 같은 순서로 DLL을 찾는다.
  - 실행 파일이 로드된 디렉토리
  - 현재 디렉토리
  - 윈도우즈 시스템 디렉토리(C:₩Windows₩System32)
  - 윈도우즈 디렉토리(C:₩Windows)
  - PATH환경변수로 지정된 디렉토리
- 만약 위의 순서대로 찾아서 해당 DLL이 없으면 이 프로그램을 실행할 때 다음과 같은 에러 메시지가 출력된다.



- 하나의 "Win32 Application"프로젝트를 생성하고 다음 페이지 소스를 cpp파일로 프로젝트에 포함시킨다.
- "프로젝트/속성" 메뉴 -> 링커/입력 -> 추가종속성에 임포트 라이브러리(*.lib)를 포함시킨다.

```
#include <windows.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
HINSTANCE g_hInst;

//임포트함수 선언
extern "C" __declspec(dllimport) int Add(int, int);
extern "C" __declspec(dllimport) int Sub(int, int);

int APIENTRY WinMain( HINSTANCE hInstance,HINSTANCE
    hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)
{
  static char szClassName[]  = "ddd";
  static char szTitle[] = "DLL";

  MSG          msg;
  HWND         hWnd;
  WNDCLASSEX    wc;
  g_hInst=hInstance;

  wc.cbSize=sizeof(WNDCLASSEX);
  wc.style          = CS_HREDRAW | CS_VREDRAW;
  wc.lpfnWndProc    = WndProc;
  wc.cbClsExtra     = 0;
  wc.cbWndExtra     = 0;
  wc.hInstance      = hInstance;
  wc.hIcon          = LoadIcon(NULL,IDI_APPLICATION);
  wc.hCursor        = LoadCursor(NULL,IDC_ARROW);
  wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
  wc.lpszMenuName  = NULL;
  wc.lpszClassName = szClassName;
  wc.hIconSm=LoadIcon(NULL,IDI_APPLICATION);
  RegisterClassEx(&wc);

  hWnd = CreateWindow(
              szClassName, szTitle, WS_OVERLAPPEDWINDOW,
             CW_USEDEFAULT,  CW_USEDEFAULT,
             CW_USEDEFAULT,  CW_USEDEFAULT,
             NULL, NULL, hInstance, NULL );
 ShowWindow(hWnd, nCmdShow);
 UpdateWindow(hWnd);

  while( GetMessage( &msg, NULL, 0, 0) )
  {    TranslateMessage( &msg );
       DispatchMessage( &msg );
  }
  return msg.wParam;
}
```

```
extern "C"
__declspec(dllexport) int Add(int x, int y)
{
   return x+y;
}
extern "C"
__declspec(dllexport) int Sub(int x, int y)
{
   return x-y;
}
```

dllTest.dll의 내용

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg,
                                    WPARAM wParam, LPARAM lParam)
{
   char add[50];
   char sub[50];

   switch ( uMsg ){
   case WM_LBUTTONDOWN:
     wsprintf(add, "Add()함수 호출 결과 = %d",Add(5,6));
    MessageBox(hWnd,add,"더한 결과",MB_OK);
   break;

   case WM_RBUTTONDOWN:
    wsprintf(sub, "Sub()함수 호출 결과= %d",Sub(5,6));
    MessageBox(hWnd,sub,"뺀 결과",MB_OK);
   break;

   case WM_DESTROY:
       PostQuitMessage(0);
       break;

   default :
       return DefWindowProc( hWnd, uMsg, wParam, lParam );

   }
   return 0;
}
```

명시적(explicit) 방법

- LoadLibrary(), GetProcAddress(), FreeLibrary()같은 Win32 API함수를 사용한다.
  - 사용할 DLL을 메모리에 로드하기 위해 LoadLibrary()함수를 사용하고, FreeLibrary()함수로 사용한 DLL을 메모리에서 해제한다.
  - 익스포트 함수를 사용하기 위해서는 GetProcAddress()함수를 호출한다.

- 이 방법은 사용할 시점에만 DLL을 메모리에 로드하므로 메모리 소비가 적다.
  - DLL로딩 실패 시에도 대응코드가 가능하다.
  - 함수 호출 방법이 복잡하다는 단점을 갖는다.

LoadLibrary(), FreeLibrary(), GetProcAddress()

- 사용할 DLL을 메모리에 로드하는 LoadLibrary()함수

```
HINSTANCE LoadLibrary(
    LPCTSTR lpLibFileName          // address of filename of executable module
);
```

- 익스포트 함수를 사용하기 위한 GetProcAddress()함수

```
FARPROC GetProcAddress(
    HMODULE hModule,               // handle to DLL module
    LPCSTR lpProcName              // name of function
);
```

- 사용한 DLL을 메모리에서 해제하는 FreeLibrary()함수

```
BOOL FreeLibrary(
    HMODULE hLibModule             // handle to loaded library module
);
```

```
#include <windows.h>
#include <stdio.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
HINSTANCE g_hInst;

int APIENTRY WinMain( HINSTANCE hInstance,HINSTANCE
    hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)
{
  static char szClassName[]  = "ddd";
  static char szTitle[] = "DLL";

  MSG          msg;
  HWND         hWnd;
  WNDCLASSEX    wc;
  g_hInst=hInstance;

  wc.cbSize=sizeof(WNDCLASSEX);
  wc.style          = CS_HREDRAW | CS_VREDRAW;
  wc.lpfnWndProc    = WndProc;
  wc.cbClsExtra     = 0;
  wc.cbWndExtra     = 0;
  wc.hInstance      = hInstance;
  wc.hIcon          = LoadIcon(NULL,IDI_APPLICATION);
  wc.hCursor        = LoadCursor(NULL,IDC_ARROW);
  wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
  wc.lpszMenuName  = NULL;
  wc.lpszClassName = szClassName;
  wc.hIconSm=LoadIcon(NULL,IDI_APPLICATION);
  RegisterClassEx(&wc);

  hWnd = CreateWindow(
              szClassName, szTitle, WS_OVERLAPPEDWINDOW,
             CW_USEDEFAULT,  CW_USEDEFAULT,
             CW_USEDEFAULT,  CW_USEDEFAULT,
             NULL, NULL, hInstance, NULL );
  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);

  while( GetMessage( &msg, NULL, 0, 0) )
  {    TranslateMessage( &msg );
       DispatchMessage( &msg );
   }
  return msg.wParam;
}
```

```
extern "C"
__declspec(dllexport) int Add(int x, int y)
{
   return x+y;
}
extern "C"
__declspec(dllexport) double Div(int x, int y)
{
   return (double)x/y;
}
```

dllTest.dll의 내용

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg,
                         WPARAM wParam, LPARAM lParam)
{
   char add[50];
   char div[100];
   HINSTANCE hDllInst;
   int(*pAdd)(int, int);
   double(*pDiv)(int, int);

   switch ( uMsg ){
   case WM_LBUTTONDOWN:
     hDllInst=LoadLibrary("dll1.dll");
     pAdd=(int (*)(int, int))GetProcAddress(hDllInst,"Add");
     wsprintf(add, "Add()함수 호출 결과 = %d",pAdd(5,6));
     MessageBox(hWnd,add,"더한 결과",MB_OK);
     FreeLibrary(hDllInst);
   break;

   case WM_RBUTTONDOWN:
    hDllInst=LoadLibrary("dll1.dll");
    pDiv=(double(*)(int, int))GetProcAddress(hDllInst,"Div");
    wsprintf(div, "Div()함수 호출 결과= %lf",pDiv(5,6));
    MessageBox(hWnd,div,"나눈 결과",MB_OK);
    FreeLibrary(hDllInst);
   break;

   case WM_DESTROY:
       PostQuitMessage(0);
       break;

   default :
       return DefWindowProc( hWnd, uMsg, wParam, lParam );

   }
   return 0;
}
```

명시적 방법으로 dll을 연결하여 사용하는 프로그램

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    char add[50];
    char div[100];
    HINSTANCE hDllInst;
    int(*pAdd)(int, int);
    double(*pDiv)(int, int);

    switch ( uMsg ){
    case WM_LBUTTONDOWN:
        hDllInst = LoadLibrary ("dll1.dll");
        pAdd = (int (*)(int, int))GetProcAddress (hDllInst,"Add");
        wsprintf (add, "Add()함수 호출 결과 = %d",pAdd (5,6)); //pAdd함수 사용
        MessageBox (hWnd,add,"더한 결과",MB_OK);
        FreeLibrary (hDllInst);
    break;

    case WM_RBUTTONDOWN:
        hDllInst = LoadLibrary ("dll1.dll");
        pDiv = (double(*)(int, int))GetProcAddress (hDllInst,"Div");
        wsprintf (div, "Div()함수 호출 결과= %lf",pDiv(5,6));
        MessageBox (hWnd,div,"나눈 결과",MB_OK);
        FreeLibrary (hDllInst);
    break;
          :
```

사용할 두 개의 함수 포인터를 선언

dll1.dll파일을 메모리에 로드하며 리턴값을 hDllInst변수에 저장

hDllInst에 있는 Add함수의 주소를 얻어 pAdd에 저장

리턴값을 함수 포인터형으로 캐스트연산

DLL 만들어 보기

- 실습에서 많이 사용했던 함수들을 모아 DLL로 만들어 보기
  - 예) 좌우상하 이동
  - 예) 점프하기
  - 예) 보드 그리기

2. 사운드 이용하기: PlaySound 함수

- BOOL **PlaySound** (LPCSTR pszSound, HMODULE hmode, DWORD fdwSound);
  - 32비트 사운드 재생 함수
  - Wav 파일 재생
  - **링커에 winmm.lib 링크 추가**
  - **#include <mmsystem.h> 추가**

  - 인자값
    - lpszSound: 재생할 파일 명
    - hmode: 리소스의 wave 파일을 연주할 경우 리소스를 가진 실행 파일의 핸들 지정, 그 외에는 NULL로 지정
    - fdwSound: 사운드의 연주 방식과 연주할 사운드의 종류 정의
      - SND_ASYNC: 비동기화된 연주, 연주를 시작한 직후 다른 작업을 바로 시작할 수 있다. 연주를 중지하려면 pszSound를 null값으로 하여 함수를 호출
      - SND_LOOP: 지정한 함수를 반복적으로 계속 연주 (SND_ASYNC와 함께 사용)

  - 사용예) PlaySound ("Sample.wav", NULL, SND_ASYNC|SND_LOOP);
  - 사운드를 멈출 때: PlaySound (NULL, NULL, NULL); 호출

사운드 이용하기: FMOD 엔진

- FMOD 사운드 엔진:
  - FireLight Technologies에서 만든 음향 미들웨어
  - **www.fmod.com**에서 해당 엔진을 다운로드 받아 설치한다.
    - Download → FMOD API and Low Level Programmer API에서 해당 버전을 다운로드 및 설치한다.
  - 프로젝트에 설정하기
    - 프로젝트 속성에서
      - VC++ 디렉토리 → 라이브러리 디렉터리 → C:₩Program Files (x86)₩FMOD SoundSystem₩FMOD Studio API Windows₩api₩lowlevel₩lib
      - VC++ 디렉토리 → 포함 디렉터리 → C:₩Program Files (x86)₩FMOD SoundSystem₩FMOD Studio API Windows₩api₩lowlevel₩inc
    - 링커 → 입력 → 추가종속성 → fmod_vc.lib 추가
  - **프로젝트 폴더에 fmod.dll 저장하기**
  - 프로그램에 **#include <fmod.h>** 추가하기

FMOD 사용하기

- 시스템 생성 및 초기화
  - 시스템 오브젝트를 생성하고 초기화 한다.
    - FMOD_SYSTEM* soundSystem; // 선언
    - FMOD_System_Create (&soundSystem); // 생성
      - FMOD_RESULT **FMOD_System_Create** (FMOD_SYSTEM **system);
        - » system: fmod 시스템
    - FMOD_System_Init (soundSystem, 10, FMOD_INIT_NORMAL, NULL); // 초기화
      - FMOD_RESULT **FMOD_System_Init** (FMOD_SYSTEM *system, int MaxChannels, FMOD_INITFLAGS flags, void *ExtraDriverdata);
        - » MaxChannels: FMOD에서 사용될 최대 채널 수
        - » flags: 시작할 때 FMOD 상태 (FMOD_INIT_NORMAL)
        - » ExtraDriverdata: 출력 플러그인에 보내질 드라이버 (NULL로 설정하면 무시한다.
  - FMOD_RESULT: FMOD_OK / FMOD_ERR_(error meaning)

FMOD 사용하기

- 사운드 로드
  - FMOD_SOUND *soundFile; // 사운드 선언
  - FMOD_CHANNEL *channel; // 채널 선언

  - FMOD_System_CreateSound (soundSystem, "bgm.mp3", FMOD_LOOP_NORMAL, 0, &soundFile);
    - FMOD_RESULT **FMOD_System_createSound** (FMOD_SYSTEM *system, const char *name_or_data, FMOD_MODE mode, FMOD_CREATESOUNDEXINFO *exinfo, FMOD_SOUND **sound);
      - » name_or_data: 파일 이름
      - » mode: 사운드 모드 (FMOD_LOOP_NORMAL / FMOD_DEFAULT / FMOD_LOOP_OFF…);
      - » exinfo: 사운드에 대한 추가적인 확장 정보를 위한 FMOD_CREATESOUNDEXINFO 포인터 (NULL로 설정하면 무시한다)
      - » sound: 새로 만든 sound 객체

FMOD 사용하기

- 사운드 재생
  - FMOD_System_PlaySound (system, SoundFile, NULL, 0, &Channel);
    - FMOD_RESULT **FMOD_System_PlaySound** (FMOD_SYSTEM *system, FMOD_SOUND *SoundFile, FMOD_CHANNELGROUP *ChannelGroup, FMOD_BOOL paused, FMOD_CHANNEL **Channel);
      - SoundFile: 재생할 사운드 파일
      - ChannelGroup: 채널 그룹
      - paused: 채널이 멈추었을 때 시작할지 아닐지 (true/false)
      - Channel: 새롭게 재생될 채널 (NULL이면 무시한다)
- 사운드 볼륨 조절
  - FMOD_Channel_SetVolume (Channel, 0.5);
    - FMOD_RESULT **FMOD_Channel_SetVolume** (FMOD_CHANNEL *Channel, float volume);
      - volume: 0.0 ~ 1.0 사이의 볼륨
- 사운드 정지
  - FMOD_Channel_Stop (channel)
- 시스템 닫기
  - FMOD_System_Release (System);

3. PeekMessage () 함수

- 지금까지 우리가 사용했던 메시지루프

```
while( GetMessage ( &msg, NULL, 0, 0) ){
        TranslateMessage ( &msg );
        DispatchMessage ( &msg );
}
```

- GetMessage()함수는 메시지 큐에 대기중인 메시지가 없을 경우 메시지가 전달될 때까지 리턴하지 않고 무한히 대기한다.

- 특별한 일을 하지 않고 대기하는 시간에 다른 일을 하려면 이 함수 대신 PeekMessage()함수를 사용하는 것이 좋다.

PeekMessage () 함수

- 두 함수의 원형
  - BOOL GetMessage (LPMSG lpMsg, HWND hWnd,
    UINT wMsgFilterMin, UINT wMsgFilterMax);

  - BOOL PeekMessage (LPMSG lpMsg, HWND hWnd,
    UINT wMsgFilterMin, UINT wMsgFilterMax, UINT wRemoveMsg);
    - wRemoveMsg: 메시지 처리 방법 지정 플래그
      - PM_NOREMOVE: 메시지를 읽은 후 큐에서 메시지를 제거하지 않는다.
      - PM_REMOVE: 메시지를 읽은 후 큐에서 메시지를 제거한다.
      - PM_NOYIELD: 다른 스레드로 제어를 양보하지 않는다.
    - 리턴값: 메시지 큐에 메시지가 있으면 0이 아닌 값을 리턴, 메시지가 없으면 0을 리턴

PeekMessage () 함수

- PeekMessage() 함수는
  - GetMessage()함수처럼 메시지 큐에서 메시지를 읽는다.
  - 이 함수는 GetMessage()함수와 달리 읽은 메시지를 무조건 제거하지 않으며 큐가 비어 있을 경우 대기하지 않고 곧바로 리턴한다.
  - 메시지를 읽지 않고 단순히 메시지가 있는지 확인만 할 수 있으며 이런 특성은 백그라운드(background) 작업에 적절하다.

- PeekMessage()함수로 메시지 루프를 구현했을 경우 **WM_QUIT메시지에 대한 예외적인 처리를 반드시 해주어야 한다.**
  - GetMessage()함수는 WM_QUIT메시지를 받으면 FALSE를 리턴하여 무한 메시지 루프를 빠져나올 수 있도록 하지만 PeekMessage()함수는 메시지 존재 여부만 알려주므로 무한 메시지 루프를 빠져 나올 수 없다.

PeekMessage () 함수

- PeekMessage()함수를 사용한 메시지 루프는 다음과 같이 구현한다.

```
While (1){
    if ( PeekMessage (&msg, NULL, 0, 0, PM_REMOVE ) ){
        if(msg.message == WM_QUIT )
                break;
        TranslateMessage (&msg);
        DispatchMessage (&msg);
       }
}
```

- 시간이 비교적 오래 걸리는 함수나 코드 부분을 실행할 때 함수나 코드내에 PeekMessage()함수로 메시지 존재 여부를 판단하는 코드를 추가하여 사용자 입력 같은 이벤트에 즉각적으로 반응하여 처리할 수 있도록 해야 한다.

PeekMessage () 함수

- 사용 예) 카운트를 세는 작업을 백 그라운드로 하고 있다.

```
for (;;)
{
	if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {
		if (msg.message == WM_QUIT)
			break;
	TranslateMessage(&msg);
	DispatchMessage(&msg);
	}
	else {
		count++;
		wsprintf(str, _T("현재 카운터는 %d입니다"), count);
		TextOut(hdc, 10, 10, str, lstrlen(str));
	}
}
```

4. GetAsyncKeyState () 함수

- SHORT **GetAsyncKeyState** (int vKey);
  - 현재 키 상태를 알아오는 함수로 키가 눌려졌을 때(down)나 떨어졌을 때(up) 호출됨
  - WM_KEYDOWN 메시지는 키가 눌러질 때 보내지는 메시지로, 키를 계속 누르고 있다는 것을 알려주지는 않는다.
    - 한 개의 키를 누른 상태에서 다른 키를 누르면, 두 번째 누른 키만 전달된다.
  - 키의 현재 상태, 즉 키가 눌러졌는지 아닌지를 조사해야 한다.
  - GetAsyncKeyState는 메시지 처리 시점의 키 상태를 조사
  - 메시지 큐로 가지 않고 시스템에서 바로 상태 파악

  - 키가 눌려진 시점에서 0x8000값을 리턴. 함수가 호출되었을 때 이전에 키가 눌려진 적이 있을 때는 0x0001이 리턴된다. 정확한 연산을 위하여 0x8000값과 & 연산을 사용한다.

GetAsyncKeyState () 함수

- SHORT GetAsyncKeyState (int vKey);
  - vKey: 가상키 코드 값, 확인하고자 하는 키를 입력
  - 리턴 값:
    - 0x0000: 이전에 누른 적이 없고 호출 시점에도 눌려있지 않은 상태
    - 0x0001(최하위 비트가 세팅): 이전에 누른 적이 있고 호출 시점에는 눌려있지 않은 상태
    - 0x8000(최상위 비트가 세팅): 이전에 누른 적이 없고 호출 시점에는 눌려있는 상태
    - 0x8001: 이전에 누른 적이 있고 호출 시점에도 눌려있는 상태

- 키의 눌림 체크
  - GetAsyncKeyState(KEYCODE) & 0x8000 == 0 → 눌리지 않음
    GetAsyncKeyState(KEYCODE) & 0x8000 == 0x8000 → 눌림
    GetAsyncKeyState(KEYCODE) & 0x0001 == 1 → 눌려져 있었음.

  - SHORT status = GetAsyncState (KEYCODE);
    if (status & 0x8000 == 0) // 떼짐
    if (status & 0x8000 != 0) // 눌림
    if (status & 1 == 1) // 눌러져 있었음

GetAsyncKeyState () 함수

- 사용 예: 스페이스키의 현재 상태

```
case WM_KEYDOWN:
        if ( GetAsyncKeyState (VK_SPACE) & 0x8000)
                // 스페이스키가 눌려짐
        else
                // 스페이스키가 안 눌려짐
```

- 사용 예: 컨트롤 키와 left 키를 함께 누른 경우

```
case WM_KEYDOWN:
        if ( (wParam == VK_LEFT) && (GetAsyncKeyState (VK_CONTROL) & 0x8000) )
                                                        // control와 Left 키
                Combo ();
        else                    // 안 눌려짐
                SingleBullet ();
```

GetAsyncKeyState () 함수

- **SHORT GetKeyState (int vKey):**
  - 해당 키가 눌려졌는지, 현재 토글 상태는 무엇인지 알아낼 때 사용하는 함수
  - 해당 키가 눌렸으면 음수값을 반환
    - 최상위 비트가 1로 설정되어 반환됨
  - 누르고 있는 경우 무한으로 입력 받는 함수

  - 사용 예)

```
case WM_KEYDOWN:
        if ( (wParam == VK_LEFT) && (GetKeyState(VK_CONTROL) < 0) )
            //컨트롤 키와 왼쪽화살표 키가 눌림
```

- GetAsyncKeyState 와 GetKeyState 차이점
  - 메시지가 처리될 때(현재 키 상태)의 상황을 조사하는가: GetAsyncKeyState
    - 호출된 시점에서 키 상태를 조사, 메시지 큐를 거치지 않고 바로 리턴해주므로 키 입력을 바로 처리해줄 수 있다.
  - 메시지가 발생되었을 때의 키상태를 조사하는가: GetKeyState
    - 호출된 시점에서 메시지 큐를 거치며, 메시지 발생 후의 상태를 리턴하게 된다.

동시 키 입력을 통한 이펙트 음악 재생하기 샘플 코드

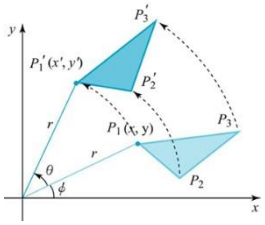
```
FMOD_SYSTEM *System;
FMOD_SOUND *bgmSound[SOUND_COUNT];
FMOD_SOUND *effectSound[EFFECT_COUNT];
FMOD_CHANNEL *Channel[CHANNEL_COUNT];

void Sound_Setup()
{
	char string[100];

	FMOD_System_Create (&System);
	FMOD_System_Init (System, 10, FMOD_INIT_NORMAL, NULL);

	for (int i = 0; i < SOUND_COUNT; i++)
	{
		wsprintf (string, "sound%d.mp3", i);
		FMOD_System_CreateSound (System, string, FMOD_LOOP_NORMAL, 0, &bgmSound[i]);
	}

	FMOD_System_CreateSound (System, "effect0.mp3", FMOD_DEFAULT, 0, &effectSound[0]);
	FMOD_System_CreateSound (System, "effect3.wav", FMOD_DEFAULT, 0, &effectSound[1]);
}


LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage,    WPARAM wParam, LPARAM lParam)
{
	HDC hdc;
	PAINTSTRUCT ps;
	static int x[2], y[2], sound, xDir;
	int i;

	switch (iMessage) {
		case WM_CREATE:
			x[0] = 0; y[0] = 100;
			x[1] = x[0] + nWidth;y[1] = y[0] + nHeight;
			xDir = 3;
			Sound_Setup();
			SetTimer (hWnd, 1, 100, NULL);              // object moving
			break;
```

동시 키 입력을 통한 이펙트 음악 재생하기 샘플 코드

```
        case WM_PAINT:
                hdc = BeginPaint(hWnd, &ps);
                Rectangle(hdc, x[0], y[0], x[1], y[1]);
                EndPaint(hWnd, &ps);
                  break;

        case WM_TIMER:
                for (i = 0; i < 2; i++)
                  x[i] += xDir;
                if (x[0] > 750) xDir *= -1;
                else if (x[0] < 0) xDir *= -1;
                InvalidateRect(hWnd, NULL, true);
                  break;

        case WM_KEYDOWN:
                FMOD_Channel_Stop (Channel[1]);
                if ((wParam == VK_LEFT) && (GetAsyncKeyState(VK_CONTROL)))                  // control와 Left 키
                  FMOD_System_PlaySound (System, effectSound[0], NULL, 0, &Channel[1]);     // 이펙트 0번을 1번 채널에서 생성
                else if ( (wParam == VK_RIGHT) && (GetKeyState(VK_CONTROL) < 0) )           // control와 Right 키
                  FMOD_System_PlaySound (System, effectSound[1], NULL, 0, &Channel[1]);     // 이펙트 1번을 1번 채널에서 생성
                  break;

        case WM_CHAR:
                switch (wParam) {
                        case 'p':
                                soundNum %= (soundNum++);
                                FMOD_Channel_Stop (Channel[0]);// 채널0번에서 현재 생성되고 있는 사운드 종료
                                FMOD_System_PlaySound (System, bgmSound[soundNum], NULL, 0, &Channel[0]); // 배경 사운드를 0번 채널에서 재생
                                break;
                        }
                InvalidateRect (hWnd, NULL, FALSE);
                  break;

        case WM_DESTROY:
                for (i = 0; i < EFFECT_COUNT; i++)          FMOD_Sound_Release (effectSound[i]);
                for (i = 0; i < SOUND_COUNT; i++)           FMOD_Sound_Release (bgmSound[i]);
                FMOD_System_Release (System);
                KillTimer(hWnd, 1);
                PostQuitMessage(0);

        return 0;
        }
        return (DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

5. 좌표계/ 이미지 변환 함수

- 좌표계를 변환하여 이미지를 변환할 수 있다.
  - 이동:
    - $x' = x + tx$
    - $y' = y + ty$
  - 회전:
    - $x' = r\cos(\Phi+\theta) = r\cos\Phi\cos\theta - r\sin\Phi\sin\theta = x\cos\theta - y\sin\theta$
    - $y' = r\sin(\Phi+\theta) = r\cos\Phi\sin\theta + r\sin\Phi\cos\theta = x\sin\theta + y\cos\theta$
  - 신축:
    - $x' = x * sx$
    - $y' = y * sy$

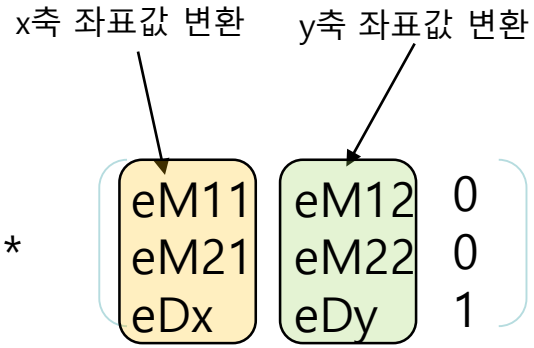


    - 위의 세 변환은 기준점 (원점)에 대하여 변환한다.

좌표계/ 이미지 변환 함수

- int SetGraphicsMode (HDC hdc, int iMode)
  - 그래픽스 모드를 변경한다.
  - iMode: GM_COMPATIBLE / **GM_ADVANCED (←이 모드로 설정)**

- XFORM 구조체
  - typedef struct _XFORM {
    FLOAT eM11;
    FLOAT eM12;
    FLOAT eM21;
    FLOAT eM22;
    FLOAT eDx;
    FLOAT eDy;
    } XFORM



$$[x' \; y' \; 1] = [x \; y \; 1] * \begin{bmatrix} eM11 & eM12 & 0 \\ eM21 & eM22 & 0 \\ eDx & eDy & 1 \end{bmatrix}$$

$$x' = x*eM11 + y*eM21 + eDx$$
$$y' = x*eM12 + y*eM22 + eDy$$

- Bool SetWorldTransform (HDC hdc, CONST XFORM *lpXform);
  - 좌표계 변환 함수

좌표계/ 이미지 변환 함수

- 사용 예)

```
XFORM xFormRotate;
int degree;


case WM_PAINT:
        hdc=BeginPaint (hWnd, &ps);
        degree = 30;

        SetGraphicsMode (hdc ,GM_ADVANCED);                 // 모드를 설정한다.
        xFormRotate.eM11 = (float)cos(degree*3.14/180);     // 적용할 변환 값을 설정한다.
        xFormRotate.eM12 = (float)sin(degree*3.14/180);
        xFormRotate.eM21 = (float)-sin(degree*3.14/180);
        xFormRotate.eM22 = (float)cos(degree*3.14/180);
        xFormRotate.eDx = 00;
        xFormRotate.eDy = 00;
        SetWorldTransform (hdc, &xFormRotate);              // 그릴 DC에 변환을 적용한다.

        StretchBlt (hdc, 10, 10, 100, 100, MemDC, 0, 0, bmp.bmWidth, bmp.bmHeight, SRCCOPY);
        EndPaint (hWnd, &ps);
        break;
```