<제2강 console 게임 제작>

4. 아스키 아트

※ 참고 사이트
- 원하는 폰트로 직접 입력하여 사용하는 사이트: https://patorjk.com/software/taag
- 만들어진것들 사용하는 사이트: https://ascii.co.uk/art
- ASCII ART Archive: https://www.asciiart.eu/
- 직접 그려서 사용하는 사이트: https://www.i2symbol.com/ascii-art-generator-for-facebook-comments
- 아스키아트 생성기(첨부파일o): https://wepplication.github.io/tools/asciiArtGen/
- TEXT-IMAGE(첨부파일o): https://www.text-image.com/convert/ascii.html

※ 인코딩이란?
: 컴퓨터에서 인코딩은 사람의 언어(문자 집합)에서 컴퓨터 언어(0,1)로 변환하는 과정을 통틀어 의미. 코드화, 암호화를 의미.
컴퓨터는 on/off(1, 0)의 2진 신호로 데이터를 처리하게 되어있고, 따라서 숫자는 진법 변환으로 인간과 컴퓨터가 같이 이해를 할 수 있다. 그러나 문자는 그렇지 못하기 때문에 특정 문자와 숫자(코드)를 연결하는 코드가 필요해지는데 이것을 문자 인코딩이라고 한다.

3) 멀티바이트 문자집합과 유니코드 문자집합의 차이점?
- 멀티바이트 : 한 문자에 할당하는 공간이 일정하지 않다 (영어 : 1바이트, 다국어 2바이트)
char str = "문자열"; // 멀티바이트
ex. cout, cin : 멀티바이트

- 유니코드 : 항상 2바이트 할당. 외국어 윈도우에서도 한글이 깨지지 않고, 다국어버전을 만들기 쉽다
wchar_t str = L"문자열"; // 유니코드
ex. wcout, wcin:유니코드

구분	유니코드(_UNICODE)	멀티바이트(_MBCS)
기본 자료형	wchar_t	char
단일 문자의 크기	2Byte (16bit)	1Byte ~ 2Byte (영문, 숫자를 포함한 ASCII는 1바이트로 표현되고 나머지 한글, 한자, 일본 가나 등은 2바이트로 표현)
포함하는 문자셋	와이드 문자 및 utf-16으로 인코딩된 문자열	유니코드를 제외한 문자셋 (ANSI, UTF-8 등)

5. 아스키 아트 출력을 위한 함수들

1) _setmode(): mode에서 제공하는 파일 변환 모드인 fd로 설정됩니다.
_O_TEXT를 mode로 전달하면 텍스트(즉, 변환된) 모드가 설정됩니다.
#include<io.h> 필요: input, output 헤더
- 성공하면 이전 변환 모드를 반환합니다

```
_Check_return_
_ACRTIMP int __cdecl _setmode(
    _In_ int _FileHandle,
    _In_ int _Mode
    );
```

_FileHandle: 파일 설명자, _Mode: 새 변환 모드

ex. _setmode(_fileno(stdout), _O_U16TEXT);

=> 알파벳 O임!!

2) _fileno: 스트림에 연결된 파일 설명자를 가져옵니다.

```
_Check_return_
_ACRTIMP int __cdecl _fileno(
    _In_ FILE* _Stream
    );
```

#include<fcntl.h> 필요: file control options 헤더 => file형태

6. Sound 생성

: 컴퓨터 게임에서 출력되는 화면도 중요하지만 사운드의 역할도 굉장히 중요하다. 우리가 자주 듣는 파일은 mp3 파일이지만, 콘솔용 게임이 주로 나왔던 시기에는 mp3 보다는 스피커에서 간단한 톤을 생성해서 그 톤으로 음악을 만들었다. 여기서 그러한 사운드를 만들기 위한 간단한 방법에 대해서 이야기해 보고 샘플로 '반짝반짝 작은 별'을 스피커음으로 들어 보자.

※ Beep함수: 마이크로소프트제작

- 함수의 원형:

```
BOOL
WINAPI
Beep(
    _In_ DWORD dwFreq,
    _In_ DWORD dwDuration
    );
```

- dwFreq: 헤르츠에서 소리의 주파수, 이 매개변수는 37에서 32,767(0x25 ~ 0x7FFF) 범위에 있어야 합니다.
- dwDuration: 소리의 지속 시간(단위: ms)

- 음에 대한 기준 주파수는 440Hz이다. 이것은 1834년 독일 자연 연구회의에서 결정된 표준 음고로 계명은 A4(라)에 해당한다. 피아노 건반은 한 옥타브당 12개이고 한 옥타브 올라가는 데 2배 주파수의 톤이 나온다.

계명	이름	주파수(Hz)	계산	비고
A4	라	440	440	기준 주파수
A4#	라#	466.16	440*2^(1/12)	
B4	시	493.88	440*2^(2/12)	
C5	도	523.25	440*2^(3/12)	
C5#	도#	554.37	440*2^(4/12)	
D5	레	587.33	440*2^(5/12)	
D5#	레#	622.25	440*2^(6/12)	
E5	미	659.26	440*2^(7/12)	
F5	파	698.46	440*2^(8/12)	
F5#	파#	739.99	440*2^(9/12)	
G5	솔	783.99	440*2^(10/12)	
G5#	솔#	830.61	440*2^(11/12)	
A5	라	880	440*2^(12/12)	
A5#	라#	932.33	880*2^(1/12)	다음 옥타브
B5	시	987.77	880*2^(2/12)	
C6	도	1046.50	880*2^(3/12)	

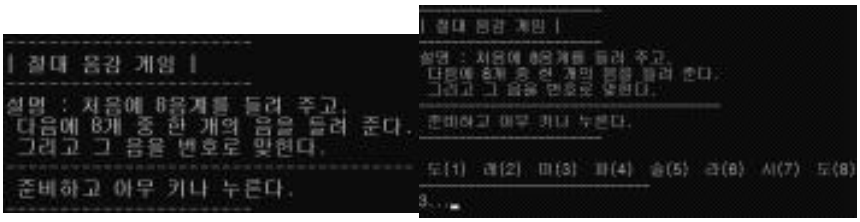
<① 절대 음감 게임>

- 게임 설명:

1. 8음계를 Beep함수를 이용하여 먼저 차례대로 들려준다.
2. 난수를 발생해서 선택된 한 음을 들려준다.
3. 플레이어는 그 음이 무엇인지 선택하고, 맞거나 틀린 정보를 보여준다.

- 실행 예시

(실행초기화면) (키 하나를 입력한 화면) (숫자를 하나 입력한 화면)



7. Sound 삽입 - PlaySound(): .wav 형식의 파일만 지원함.

```
BOOL
WINAPI
PlaySoundW(
    _In_opt_ LPCWSTR pszSound,
    _In_opt_ HMODULE hmod,
    _In_ DWORD fdwSound
    );
```

※ 사운드(.wav) 참고 사이트
- effect: https://sfxr.me/
- bgm: https://freesound.org/

- 필요 헤더: #include <Windows.h> #include <mmsystem.h>
- 필요 라이브러리: winmm.lib

Q) lib란?

=> 정적 라이브러리로, 컴파일 후 obj와 라이브러리가 연결됨.

※ dll: 동적 라이브러리

=> 정적 라이브러리의 단점으로는 라이브러리를 포함하면서 프로그램의 크기가 커지거나, 동일한 라이브러리를 포함한 프로그램들이 동시에 실행되면 불필요한 코드들이 생긴다.

=> 동적 라이브러리는 필요할 때만 메모리로 부르지만, 프로그램 배포시에 exe파일과 dll파일을 같이 배포해야 한다.

- 라이브러리 추가하는법.

① 프로젝트 - 속성 - 링커 - 입력 - 추가종속성 - ∨ 표시 - 편집 - winmm.lib 추가

② #pragma comment(lib, "winmm.lib")

- 단점

① 이 함수는 .wav 형식의 파일만 지원함. 프리웨어 음악파일 변환사이트를 이용해서 .wav로 변환시켜야 함. 단, 변환 과정이 제대로 이루어지지 않으면 인식을 하지 못함.
(http://koyotstar.free.fr/indexEn.html)

② 이 함수는 두 개 이상의 파일을 한번에 재생이 불가능합니다. 즉! 배경음 + 효과음을 할 수 없습니다.

- 함수 인자 설명

```
BOOL
WINAPI
PlaySoundW(
    _In_opt_ LPCWSTR pszSound,
    _In_opt_ HMODULE hmod,
    _In_ DWORD fdwSound
    );
```

① 첫 번째 인자(pszSound): 연주할 사운드의 이름을 지정함. 세번째 인자인 fdwSound의 플래그에 따라 이 값을 해석하는 방법이 달라짐.
=> 일단은 "연주할 사운드 파일의 이름이다". 라고 생각하면 됨. 이 값이 NULL이면 연주중인 사운드 파일의 연주를 멈춤.
② 두 번째 인자(hmod): 리소스의 Wave 파일을 연주할 경우 리소스를 가진 실행 파일의 핸들을 지정하며 그 외의 경우는 NULL로 지정함.
③ 세 번째 인자(dwSound): 사운드의 연주 방식과 연주할 사운드의 종류를 정의하는 플래그임.
★ SND_ASYNC: 비동기화 연주. 연주시작과 동시에 리턴하므로 다른작업을 바로 할수 있습니다. 비동기화 연주를 중지하려면 pszSound를 NULL값으로 하여 PlaySound 함수를 한 번더 호출해야 함.
=> 일반적으로 효과음이나 배경음악은 백그라운드에서 재생해야 하므로 주로 사용됨.
- SND_SYNC: 동기화연주. 사운드의 연주가 완전히 끝난 후 리턴합니다.
★ SND_LOOP: 해당 사운드를 계속 반복합니다. (반드시, SND_ASYNC와 같이 사용)
- SND_FILENAME: pszSound에 파일을 넣을경우 반드시 넣어야 하는 옵션
- SND_RESOURCE: pszSound에 리소스를 넣을경우 반드시 넣어야 하는 옵션
- SND_NOSTOP: 새로운 사운드 연주명령에 의해 지금 연주되고 있는 사운드가 중지되지 않도록 합니다. 기본적으로는 PlaySound가 호출될 때 미리 연주되고 있던 사운드는 중단됩니다.
- 그 외: SND_ALIAS, SND

- 순서
① 리소스를 다운받아 둔다.
②-1(리소스): vs에서 리소스 파일(마우스 오른쪽 버튼) - 추가 - 리소스 클릭 - 가져오기 - 파일 확장자를 모든 파일로 변경 - 다운 받은 리소스 파일 추가 - 저장 - "resource.h" 더블클릭 덮어쓰기 - IDR_WAVE 확인
②-2(파일): vs에서 작업경로에 다운받은 파일 이동

③ 함수
- 시스템 메시지 사운드 출력

```
PlaySound(L"SystemDefault", NULL, SND_ASYNC);
```

- 리소스를 통해서 사운드 출력

```
PlaySound(MAKEINTRESOURCE(IDR_WAVE1), NULL, SND_RESOURCE | SND_ASYNC | SND_LOOP);
```

=> SND_RESOURCE가 반드시 포함되어야 함. 리소스가 포함되어 있어 실행 파일이 무거워지지만, 빠름.
- 파일경로 통해 사운드 출력

```
PlaySound(TEXT("explosion.wav"), NULL, SND_FILENAME);
```

=> SND_FILENAME가 반드시 포함되어야 함. 사운드 파일을 읽어야해서 느리고, 실행 파일과 같이 배포되어야 함.
※ TEXT()매크로: 유니코드를 사용할 경우에 TEXT("문자열")이 L"문자열"로 변환되고, 그렇지 않으면 "문자열"로 변환됨. => 유니코드 사용 유무에 상관없이 문자열 처리 가능.