WinAPI
FileIn_Out

SoulSeek

파일핸들러 생성

HANDLE CreateFile(LPCTSTR **lpFileName**, DWORD **dwCreatedAccess**, DWORD **dwShareMode**, LPSECURITY_ATTRIBUTES **lpSecurityAttributes**, DWORD **dwCreationDisposition**, DWORD d**wFlagsAndAttributes**, HANDLE **hTemplateFile**)

```
HANDLE hFile = CreateFile(OFN.lpstrFile,GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                          FILE_ATTRIBUTE_NORMAL, 0);
```

BOOL ReadFile(HANDLE **hFile**, LPVOID **lpBuffer**, DWORD **nNumberOfBytesToRead**, LPDWORD **lpNumberOfBytesRead**, LPOWERLAPPED **lpOverlapped**);

```
ReadFile(hFile, &g_map[i][j], sizeof(int), &writeB, NULL);
```

BOOL WriteFile(HANDLE **hFile**, LPCVOID **lpBuffer**, DWORD **nNumberOfBytesToWrite**, LPDWORD **lpNumberOfBytesWritten**, LPOVERLAPPED **lpOverlapped**);

```
WriteFile(hFile, &g_map[i][j], sizeof(int), &writeB, NULL);
```

BOOL CloseHandle(HANDLE **hObject**);

```
CloseHandle(hFile);
```

WinAPI DailogBoxFileOpen

SoulSeek

BOOL GetOpenFileName(LPOPENFILENAME lpofn)

BOOL GetSaveFileName(LPOPENFILENAME lpofn)

```
if (GetSaveFileName(&saveFile) != 0)
```

```
if (GetOpenFileName(&openFile) != 0)
```

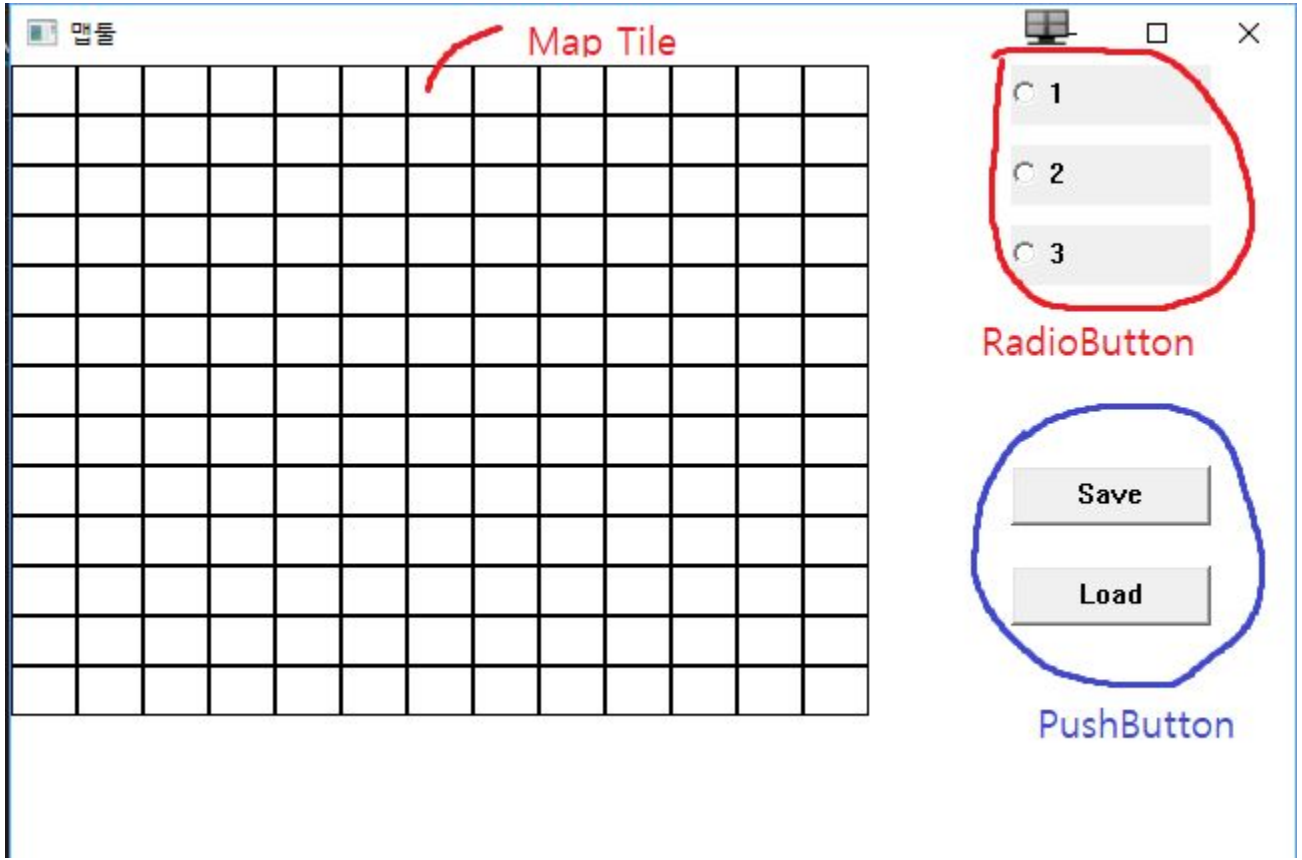
OPENFILENAME 구조체

```
OPENFILENAME saveFile;
char lpstrSaveFile[256] = "";
memset(&saveFile, 0, sizeof(OPENFILENAME));
saveFile.lStructSize = sizeof(OPENFILENAME);
saveFile.hwndOwner = hWnd;
saveFile.lpstrFilter = "Every File(*.*)\0*.*\0TExt File\0*.txt;*.ini\0";
saveFile.lpstrFile = lpstrSaveFile;
saveFile.nMaxFile = 256;
saveFile.lpstrInitialDir = NULL;
```

WinAPI MapTool

SoulSeek

*Tile Map을 사용하는 게임의 맵툴을 제작해보자.



*타일과 선택 오브젝트에 대한 변수 선언 및 초기화

```
//한블록당 width, height를 디파인.
#define BL_WIDTH 33
#define BL_HEIGHT 25
```

```
//맵 블록의 이차원 배열
int g_map[13][13];
//현재 선택한 오브젝트 속성
int cur_select = 0;
```

*윈도우 크기 설정.

```
//정확한 윈도우크기를 설정하자.(윈도우 구성 설정 값을 가져온다.)
int         nWidth, nHeight;//윈도우 크기
//화면 크기 + 양쪽 옆의 경계테두리 프레임 크기
nWidth = 650 + GetSystemMetrics(SM_CXFRAME) * 2;
//화면 크기 + 아래, 위의 경계 테두리 크기 + 캡션의 높이(존재한다면) + 한줄 메뉴의 높이(존재한다면)
nHeight = 384 + GetSystemMetrics(SM_CYFRAME) * 2 +
    GetSystemMetrics(SM_CYCAPTION) + GetSystemMetrics(SM_CYMENU);
```

WM_CREATE에서

- 오브젝트 선택에 쓰일 버튼 생성.

```
//오브젝트 배치 라디오 버튼
CreateWindow("button", "1", WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON | WS_GROUP
    , 500, 0 , 100, 30 , hWnd, (HMENU)0, g_hInst, NULL);
CreateWindow("button", "2", WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON
    , 500, 40, 100, 30, hWnd, (HMENU)1, g_hInst, NULL);
CreateWindow("button", "3", WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON
    , 500, 80, 100, 30, hWnd, (HMENU)2, g_hInst, NULL);
```

- 파일 저장, 열기에 사용될 버튼 생성

```
//Save, Load 버튼
CreateWindow("button", "Save", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON
    , 500, 200, 100, 30, hWnd, (HMENU)100, g_hInst, NULL);
CreateWindow("button", "Load", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON
    , 500, 250, 100, 30, hWnd, (HMENU)101, g_hInst, NULL);
```

```
//맵 블록의 크기만큼 메모리 초기화 한다.
memset(g_map, 0, sizeof(int) * 13 * 13);
```

WM_LBUTTONDOWN

- 라디오 버튼으로 선택된 오브젝트 정보를 타일에 배치

```
case WM_LBUTTONDOWN:
    {
        POINT pt;
        pt.x = LOWORD(lParam);
        pt.y = HIWORD(lParam);
```

```
//커서가 클릭됐을때 어느 타일에 속해 있는지 파악하고 해당 이차원 배열에 값을 저장해준다.
if ((pt.x < BL_WIDTH * 13 && pt.x > 0) && (pt.y < BL_HEIGHT * 13 && pt.y > 0))
{
    g_map[pt.y / BL_HEIGHT][pt.x / BL_WIDTH] = cur_select;

    //다시 그려준다.
    InvalidateRect(hWnd, NULL, false);
}
```

WM_PAINT

*맵 블록을 그려준다.

```
//설정한 속성으로 생성해준다.(처음 생성시 기본값으로)
for (int i = 0; i < 13; i++)
{
    for (int j = 0; j < 13; j++)
    {
        if (g_map[i][j] == 0)
        {
            //사각형을 그린다.
            Rectangle(hdc, j * BL_WIDTH, i * BL_HEIGHT, (j + 1) * BL_WIDTH
                , (i + 1) * BL_HEIGHT);
        }
        else if (g_map[i][j] == 1)
        {
            //원을 그린다.
            Ellipse(hdc, j * BL_WIDTH, i * BL_HEIGHT, (j + 1) * BL_WIDTH
                , (i + 1) * BL_HEIGHT);
        }
```

```
else if (g_map[i][j] == 2)
{
    //원안의 검정색을 그린다.
    HBRUSH hBlack = (HBRUSH)GetStockObject(BLACK_BRUSH);
    HBRUSH old = (HBRUSH)SelectObject(hdc, hBlack);
    Ellipse(hdc, j * BL_WIDTH, i * BL_HEIGHT, (j + 1) * BL_WIDTH
            , (i + 1) * BL_HEIGHT);

    SelectObject(hdc, old);
```

WM_COMMNAD

*라디오 버튼 메시지 처리

```
case WM_COMMAND:
{
    switch (LOWORD(wParam))
    {
    case 0:
    case 1:
    case 2:
        //현재 선택한 오브젝트의 넘버
        cur_select = LOWORD(wParam);
        break;
```

*FileSave 버튼

- OPENFILENAME 구조체를 설정한다.

```
case 100: //SAVE Button
    {
        //파일열기 다이얼로그 박스를 생성하기위한 설정을 한다.
        OPENFILENAME OFN;
        char str[300];
        char lpstrFile[MAX_PATH] = "";
        char lpstrPath[MAX_PATH] = "";

        memset(&OFN, 0, sizeof(OPENFILENAME));
        OFN.lStructSize = sizeof(OPENFILENAME);
        OFN.hwndOwner = hWnd;
        OFN.lpstrFilter = "Every File(*.*)\0*.*\0Text File\0*.txt;*.doc\0";
        OFN.lpstrFile = lpstrFile;
        OFN.nMaxFile = 256;
```

- 디렉토리 경로를 가져온다.

```
//파일경로를 지정해준다.
//SetCurrentDirectory(TEXT("c:\\"));
//현재 파일의 경로를 알아온다.(프로젝트 폴더가 된다.)
GetCurrentDirectory(MAX_PATH, lpstrPath);

OFN.lpstrInitialDir = lpstrPath;
```

- DialogBox를 열어준다.

```
//파일을 저장하기위해 다이얼로그 박스를 열어준다.
if (GetSaveFileName(&OFN) == 0)
{
    //오류체크 함수 (다이얼로그 박스 여는 동안)
    DWORD err = CommDlgExtendedError();
    break;
}
```

- File 작성을 위한 핸들러를 생성한다.

```
//파일 핸들러를 만들어 준다.
HANDLE hFile = CreateFile(OFN.lpstrFile,GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                          FILE_ATTRIBUTE_NORMAL, 0);
```

- 라디오 버튼에 의해 저장된 정보를 파일에 써준다.

```
//맵의 타일수 만큼 쓴다.
for (int i = 0; i < 13; i++)
{
    for (int j = 0; j < 13; j++)
    {
        DWORD writeB;

        //파일에 입력할 정보를 써준다.
        WriteFile(hFile, &g_map[i][j], sizeof(int), &writeB, NULL);
    }
}
```

- 핸들러를 닫고 다시 그려준다.

```
//사용후 핸들러를 닫아준다.
CloseHandle(hFile);
MessageBox(hWnd, "파일저장 완료", "저장", MB_OK);
//다시 그려준다.
InvalidateRect(hWnd, NULL, false);
```

- 파일 열기,(파일 저장과 과정은 똑같음.)

```
//파일 핸들러를 만든다.
HANDLE hFile = CreateFile(OFN.lpstrFile, GENERIC_READ, 0, NULL, OPEN_EXISTING
                        , FILE_ATTRIBUTE_NORMAL, 0);
```

- 타일수 만큼 맵을 읽는다

```
//맵의 타일수 만큼 읽는다.
for (int i = 0; i < 13; i++)
{
    for (int j = 0; j < 13; j++)
    {
        DWORD writeB;

        //파일 내용을 읽는다.
        ReadFile(hFile, &g_map[i][j], sizeof(int), &writeB, NULL);
    }
}
```