< 윈도우프로그래밍 6주차 과제 >

2017253041_홍성우

#1

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam);

HINSTANCE hInst;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
	LPSTR lpszCmdLine, int nCmdShow)
	// UNICODE 사용시 wWinMain() 형태
	// hPrevInstance 이전 인스턴스 항상 0값
	// lpszCmdLine > 외부에서 (내부로) 입력받는 변수
	// nCmdShow 윈도우 출력 형태에 관련한 값
{
	HWND	hwnd;
	MSG		msg;
	WNDCLASS WndClass;
	hInst = hInstance;
	HACCEL hAcc;
	hAcc = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDR_ACCELERATOR5));
	WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
	WndClass.lpfnWndProc = WndProc;			// Proc 설정
	WndClass.cbClsExtra = 0;
	WndClass.cbWndExtra = 0;
	WndClass.hInstance = hInstance;
	WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
	WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
	WndClass.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);	// 형변환
	WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5);
	WndClass.lpszClassName = _T("Window Class Name");
	RegisterClass(&WndClass);		// WndClass 등록
	hwnd = CreateWindow(_T("Window Class Name"),
		_T("2017253041_홍성우"),		// 타이틀바, 학번이름 체크
		WS_OVERLAPPEDWINDOW,		// 윈도우 스타일
		600, 400,					// 창출력좌표 x, y
		600, 400,					// 창크기 x, y축
		NULL,						// 부모 윈도우
		NULL,						// 메뉴바 핸들
		hInstance,					// 인스턴스
		NULL						// 여분, NULL
	);
	ShowWindow(hwnd, nCmdShow);		// 윈도우 출력, WM_PAINT 출력내용 가져옴
	UpdateWindow(hwnd);				// WM_PAINT 출력내용 발생해서
출력하도록
                                                // hwnd 핸들을 통해
보여주고 갱신
```

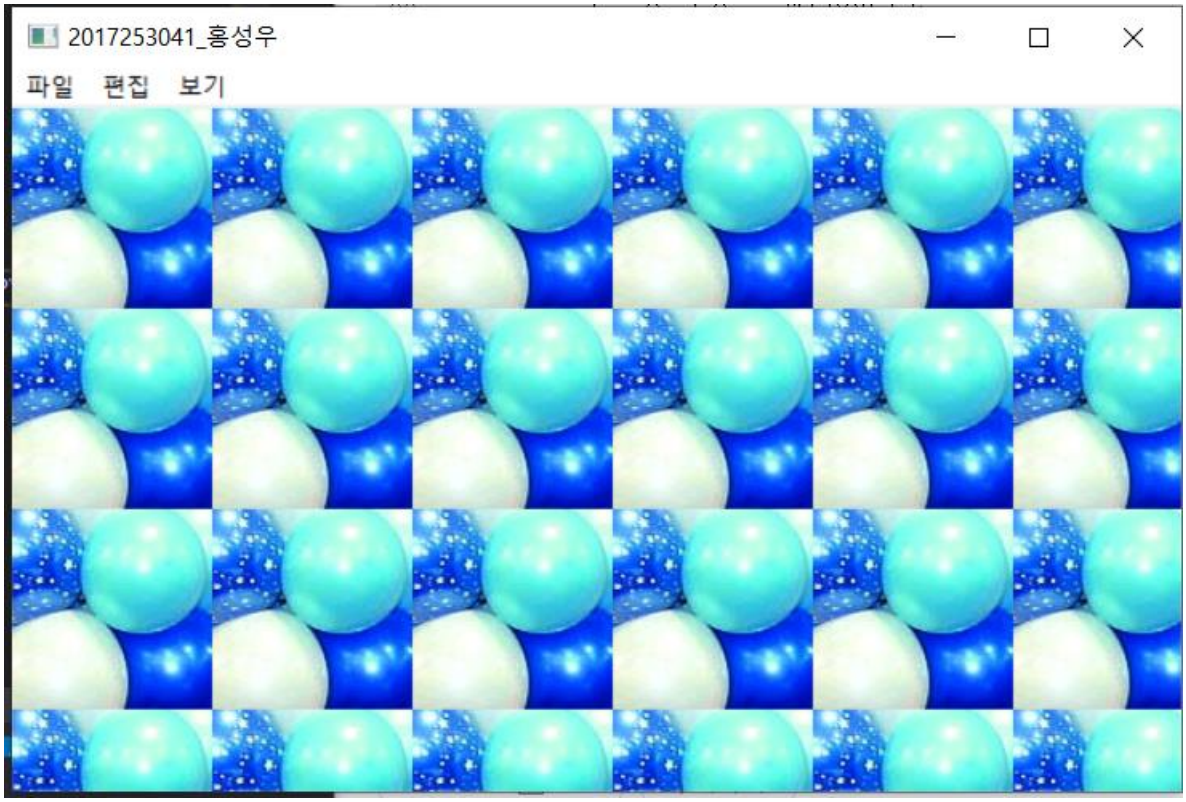
```
	//ShowWindow(hwnd, SW_SHOW);	// 위와 같음
	//UpdateWindow(hwnd);

	while (GetMessage(&msg, NULL, 0, 0))	// 메시지 큐의 메시지를 가져옴
	{
		if (!TranslateAccelerator(hwnd, hAcc, &msg)) {
			TranslateMessage(&msg);
			DispatchMessage(&msg);
		}
	}
	return (int)msg.wParam;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam)
	//	WinDef.h 에서 정의
	//	wPram > unsigned ptr, lParam > long ptr
{
	HDC hdc, memdc;
	PAINTSTRUCT ps;
	static HBITMAP hBit, oldBit;
	static RECT rectView;

	switch (iMsg)
	{
	case WM_CREATE:
		hBit = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP1));
		break;
	case WM_PAINT:
		hdc = BeginPaint(hwnd, &ps);
		GetClientRect(hwnd, &rectView);
		memdc = CreateCompatibleDC(hdc);
		oldBit = (HBITMAP)SelectObject(memdc, hBit);
		StretchBlt(hdc, 0, 0, rectView.right, rectView.bottom, memdc, 0, 0, 400, 400,
SRCCOPY);
		SelectObject(memdc, oldBit);
		EndPaint(hwnd, &ps);
		break;
	case WM_DESTROY:
		PostQuitMessage(0);
		break;
	}
	return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```

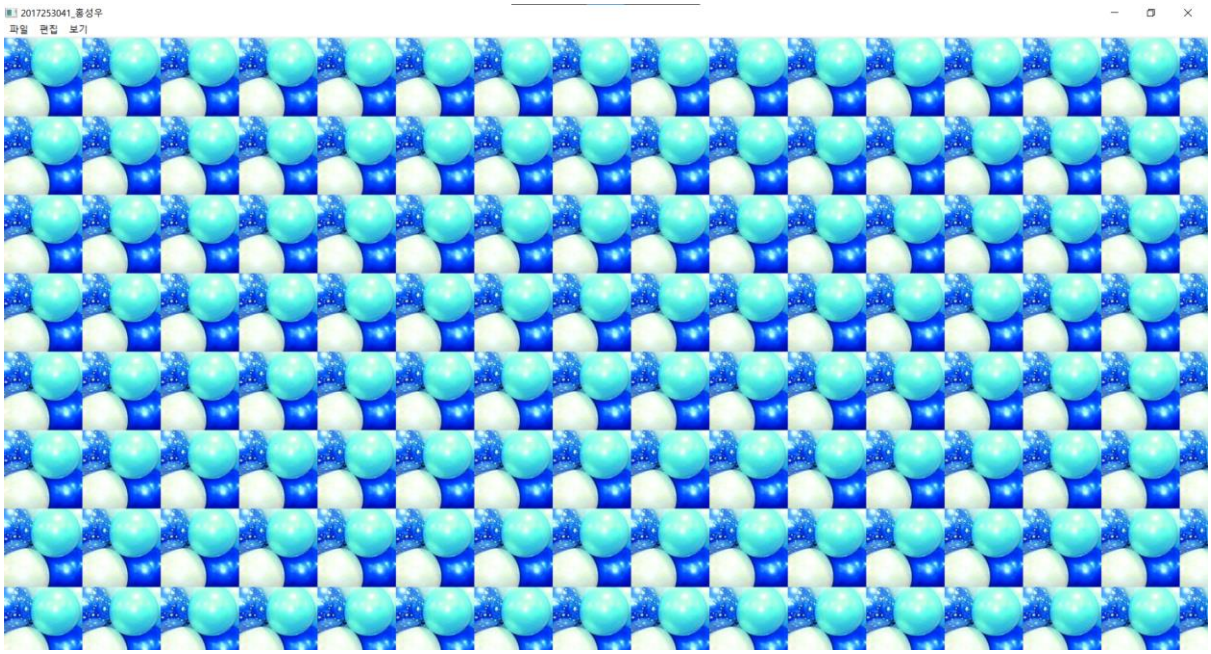
#2

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);

HINSTANCE hInst;
```

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpszCmdLine, int nCmdShow)
        // UNICODE 사용시 wWinMain() 형태
        // hPrevInstance 이전 인스턴스 항상 0값
        // lpszCmdLine > 외부에서 (내부로) 입력받는 변수
        // nCmdShow 윈도우 출력 형태에 관련한 값
{
        HWND     hwnd;
        MSG               msg;
        WNDCLASS WndClass;
        hInst = hInstance;
        HACCEL hAcc;
        hAcc = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDR_ACCELERATOR5));
        WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
        WndClass.lpfnWndProc = WndProc;           // Proc 설정
        WndClass.cbClsExtra = 0;
        WndClass.cbWndExtra = 0;
        WndClass.hInstance = hInstance;
        WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
        WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
        WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);      // 형변환
        WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5);
        WndClass.lpszClassName = _T("Window Class Name");
        RegisterClass(&WndClass);         // WndClass 등록
        hwnd = CreateWindow(_T("Window Class Name"),
                _T("2017253041_홍성우"),          // 타이틀바, 학번이름 체크
                WS_OVERLAPPEDWINDOW,              // 윈도우 스타일
                600, 400,                                   // 창출력좌표 x, y
                600, 400,                                   // 창크기 x, y축
                NULL,                                       // 부모 윈도우
                NULL,                                       // 메뉴바 핸들
                hInstance,                                  // 인스턴스
                NULL                                        // 여분, NULL
        );
        ShowWindow(hwnd, nCmdShow);               // 윈도우 출력, WM_PAINT 출력내용 가져옴
        UpdateWindow(hwnd);                               // WM_PAINT 출력내용 발생해서
출력하도록
                                                                    // hwnd 핸들을 통해
보여주고 갱신

        //ShowWindow(hwnd, SW_SHOW);      // 위와 같음
        //UpdateWindow(hwnd);

        while (GetMessage(&msg, NULL, 0, 0))      // 메시지 큐의 메시지를 가져옴
        {
                if (!TranslateAccelerator(hwnd, hAcc, &msg)) {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                }
        }                                                           // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                   // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}
```

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
	HDC hdc, memdc;
	PAINTSTRUCT ps;
	static HBITMAP hBit, oldBit;
	static RECT rectView;

	switch (iMsg)
	{
	case WM_CREATE:
		hBit = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP1));
		break;
	case WM_PAINT:
		hdc = BeginPaint(hwnd, &ps);
		GetClientRect(hwnd, &rectView);
		memdc = CreateCompatibleDC(hdc);
		oldBit = (HBITMAP)SelectObject(memdc, hBit);
		for (int j = 0; j < rectView.bottom; j += 100)
		{
			for (int i = 0; i < rectView.right; i += 100)
			{
				StretchBlt(hdc, i, j, 100, 100, memdc, 100, 100, 100, 100,
SRCCOPY);
			}
		}
		SelectObject(memdc, oldBit);
		EndPaint(hwnd, &ps);
		break;
	case WM_DESTROY:
		PostQuitMessage(0);
		break;

	}
	return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```





#3

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);

HINSTANCE hInst;
```

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpszCmdLine, int nCmdShow)
        // UNICODE 사용시 wWinMain() 형태
        // hPrevInstance 이전 인스턴스 항상 0값
        // lpszCmdLine > 외부에서 (내부로) 입력받는 변수
        // nCmdShow 윈도우 출력 형태에 관련한 값
{
        HWND     hwnd;
        MSG               msg;
        WNDCLASS WndClass;
        hInst = hInstance;
        HACCEL hAcc;
        hAcc = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDR_ACCELERATOR5));
        WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
        WndClass.lpfnWndProc = WndProc;           // Proc 설정
        WndClass.cbClsExtra = 0;
        WndClass.cbWndExtra = 0;
        WndClass.hInstance = hInstance;
        WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
        WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
        WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);      // 형변환
        WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5);
        WndClass.lpszClassName = _T("Window Class Name");
        RegisterClass(&WndClass);         // WndClass 등록
        hwnd = CreateWindow(_T("Window Class Name"),
                _T("2017253041_홍성우"),          // 타이틀바, 학번이름 체크
                WS_OVERLAPPEDWINDOW,              // 윈도우 스타일
                600, 400,                                   // 창출력좌표 x, y
                600, 400,                                   // 창크기 x, y축
                NULL,                                       // 부모 윈도우
                NULL,                                       // 메뉴바 핸들
                hInstance,                                  // 인스턴스
                NULL                                        // 여분, NULL
        );
        ShowWindow(hwnd, nCmdShow);               // 윈도우 출력, WM_PAINT 출력내용 가져옴
        UpdateWindow(hwnd);                               // WM_PAINT 출력내용 발생해서
출력하도록
                                                                    // hwnd 핸들을 통해
보여주고 갱신

        //ShowWindow(hwnd, SW_SHOW);      // 위와 같음
        //UpdateWindow(hwnd);

        while (GetMessage(&msg, NULL, 0, 0))      // 메시지 큐의 메시지를 가져옴
        {
                if (!TranslateAccelerator(hwnd, hAcc, &msg)) {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                }
        }                                                           // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                   // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}
```

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    static HDC hdc, memdc;
    PAINTSTRUCT ps;
    static HBITMAP hBit, oldBit;
    static int x, y;
    static RECT rect;
    static bool xdir = true, ydir = false;

    switch (iMsg)
    {
    case WM_CREATE:
        x = 150;
        y = 200;
        GetClientRect(hwnd, &rect);
        break;
    case WM_KEYDOWN:
        SetTimer(hwnd, 1, 100, NULL);
        break;
    case WM_TIMER:
        hdc = GetDC(hwnd);
        if (xdir == true && ydir == false) {
            x += 20;
            y -= 20;
        } else if (xdir == true && ydir == true) {
            x += 20;
            y += 20;
        } else if (xdir == false && ydir == true) {
            x -= 20;
            y += 20;
        } else if (xdir == false && ydir == false) {
            x -= 20;
            y -= 20;
        }
        BitBlt(hdc, x, y, 100, 100, memdc, 100, 150, SRCCOPY);
        if (x + 100 >= rect.right || x <= rect.left)
                xdir = xdir ? false : true;
        if (y + 100 >= rect.bottom || y <= rect.top)
            ydir = ydir ? false : true;
        ReleaseDC(hwnd, hdc);
        break;
    case WM_PAINT:
        hdc = GetDC(hwnd);
        hBit = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP1));
        memdc = CreateCompatibleDC(hdc);
        oldBit = (HBITMAP)SelectObject(memdc, hBit);
        BitBlt(hdc, x, y, 100, 100, memdc, 100, 150, SRCCOPY);
        ReleaseDC(hwnd, hdc);
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    }
```

```
    return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```