< 윈도우프로그래밍 4주차 과제 >

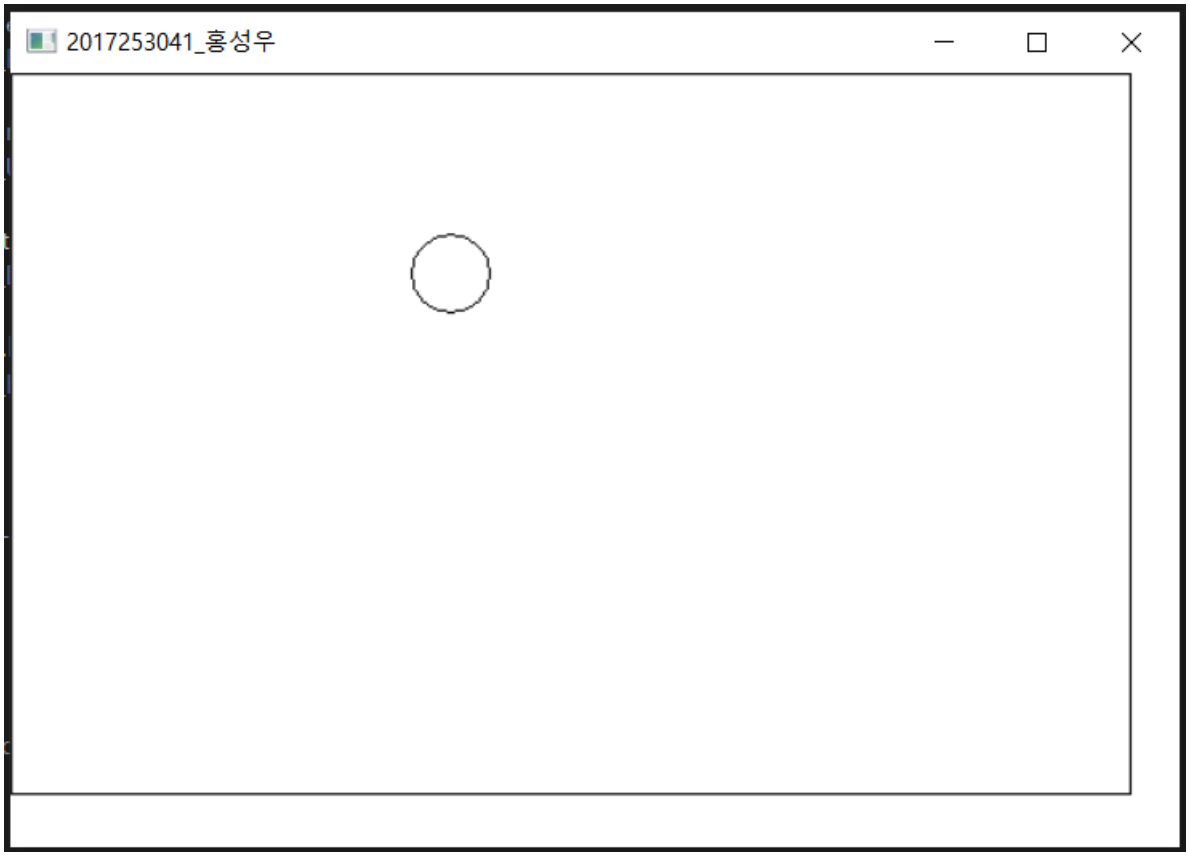
2017253041_홍성우

#1

```
#include <windows.h>
#include <TCHAR.H>
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam);
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;
	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 = NULL;
	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
	{
		TranslateMessage(&msg);		// 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
```

```
                DispatchMessage(&msg);          // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                               // 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)
        //      WinDef.h 에서 정의
        //      wPram > unsigned ptr, lParam > long ptr
{
        HDC hdc;
        PAINTSTRUCT ps;
        static int x, y;
        static RECT rectView;

        switch (iMsg)
        {
        case WM_CREATE:
                GetClientRect(hwnd, &rectView);
                x = 20;
                y = 20;
                break;
        case WM_PAINT:
                hdc = BeginPaint(hwnd, &ps);
                Rectangle(hdc, 0, 0, 40 * (int)(rectView.right / 40), 40 *
(int)(rectView.bottom / 40));
                Ellipse(hdc, x - 20, y - 20, x + 20, y + 20);
                EndPaint(hwnd, &ps);
                break;
        case WM_KEYDOWN:
                if (wParam == VK_LEFT) {
                        x -= 40;
                        if (x - 20 < rectView.left) x += 40;
                } else if (wParam == VK_RIGHT) {
                        x += 40;
                        if (x + 20 > rectView.right) x -= 40;
                } else if (wParam == VK_UP) {
                        y -= 40;
                        if (y - 20 < rectView.top) y += 40;
                } else if (wParam == VK_DOWN) {
                        y += 40;
                        if (y + 20 > rectView.bottom) y -= 40;
                } else if (wParam == VK_HOME) {
                        x = 20;
                        y = 20;
                }
                InvalidateRgn(hwnd, NULL, TRUE);
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
```

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



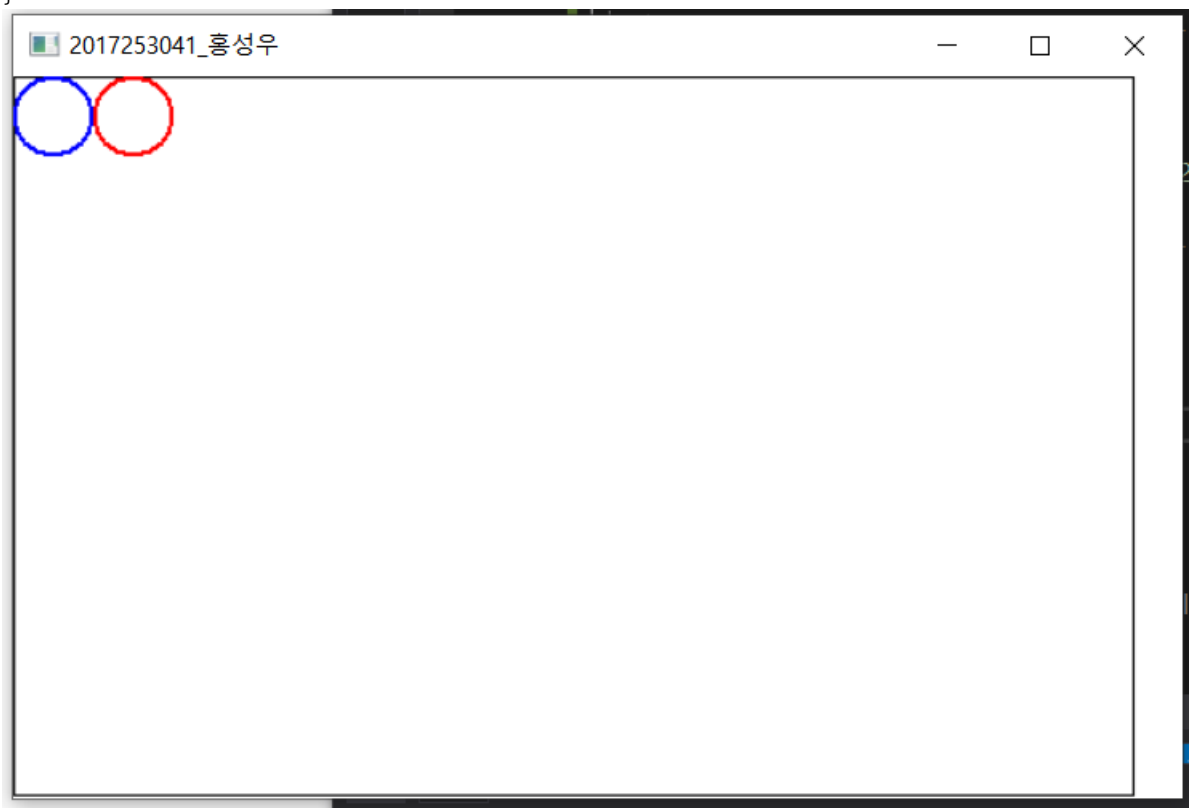
#2

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

```
	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
	{
		TranslateMessage(&msg);		// 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
		DispatchMessage(&msg);		// WndProc() 함수 호출과 WndProc()으로
메세지 전달
	}                                                               // 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)
	//	WinDef.h 에서 정의
	//	wPram > unsigned ptr, lParam > long ptr
{
	HDC hdc;
	PAINTSTRUCT ps;
	HPEN hPen, oldPen;
	static int x1, y1;
	static int x2, y2;
	static bool flag;
	static int direction;
	static RECT rectView;

	switch (iMsg)
	{
	case WM_CREATE:
		GetClientRect(hwnd, &rectView);
		x1 = 60;
		y1 = 20;
		x2 = 20;
```

```
		y2 = 20;
		flag = false;
		break;
	case WM_KEYDOWN:
		if (wParam == VK_LEFT && (x2 - 60 >= rectView.left)) {
			x1 = x2 - 40;
			y1 = y2;
			direction = 1;
		} else if (wParam == VK_RIGHT && (x2 + 60 <= rectView.right)) {
			x1 = x2 + 40;
			y1 = y2;
			direction = 2;
		} else if (wParam == VK_UP && (y2 - 60 >= rectView.top)) {
			x1 = x2;
			y1 = y2 - 40;
			direction = 3;
		} else if (wParam == VK_DOWN && (y2 + 60 <= rectView.bottom)) {
			x1 = x2;
			y1 = y2 + 40;
			direction = 4;
		} else if (wParam == VK_RETURN) {
			if (flag) {
				flag = false;
			} else {
				flag = true;
			}
			if (flag) {
				SetTimer(hwnd, 1, 100, NULL);
			} else {
				KillTimer(hwnd, 1);
			}
			InvalidateRgn(hwnd, NULL, TRUE);
			break;
		}
		InvalidateRgn(hwnd, NULL, TRUE);
		break;
	case WM_TIMER:
		if (direction == 1 && (x1 - 20 > rectView.left)) {
			x1 -= 40;
			x2 = x1 + 40;
		} else if (direction == 2 && (x1 + 20 < rectView.right)) {
			x1 += 40;
			x2 = x1 - 40;
		} else if (direction == 3 && (y1 - 20 > rectView.top)) {
			y1 -= 40;
			y2 = y1 + 40;
		} else if (direction == 4 && (y1 + 20 < rectView.bottom)) {
			y1 += 40;
			y2 = y1 - 40;
		}
		InvalidateRgn(hwnd, NULL, TRUE);
		break;
	case WM_PAINT:
		hdc = BeginPaint(hwnd, &ps);
		Rectangle(hdc, 0, 0, 40 * (int)(rectView.right / 40), 40 *
```

```
(int)(rectView.bottom / 40));

                hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
                oldPen = (HPEN)SelectObject(hdc, hPen);
                Ellipse(hdc, x1 - 20, y1 - 20, x1 + 20, y1 + 20);
                SelectObject(hdc, oldPen);
                DeleteObject(hPen);

                hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
                oldPen = (HPEN)SelectObject(hdc, hPen);
                Ellipse(hdc, x2 - 20, y2 - 20, x2 + 20, y2 + 20);
                SelectObject(hdc, oldPen);
                DeleteObject(hPen);

                EndPaint(hwnd, &ps);
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        }
        return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```

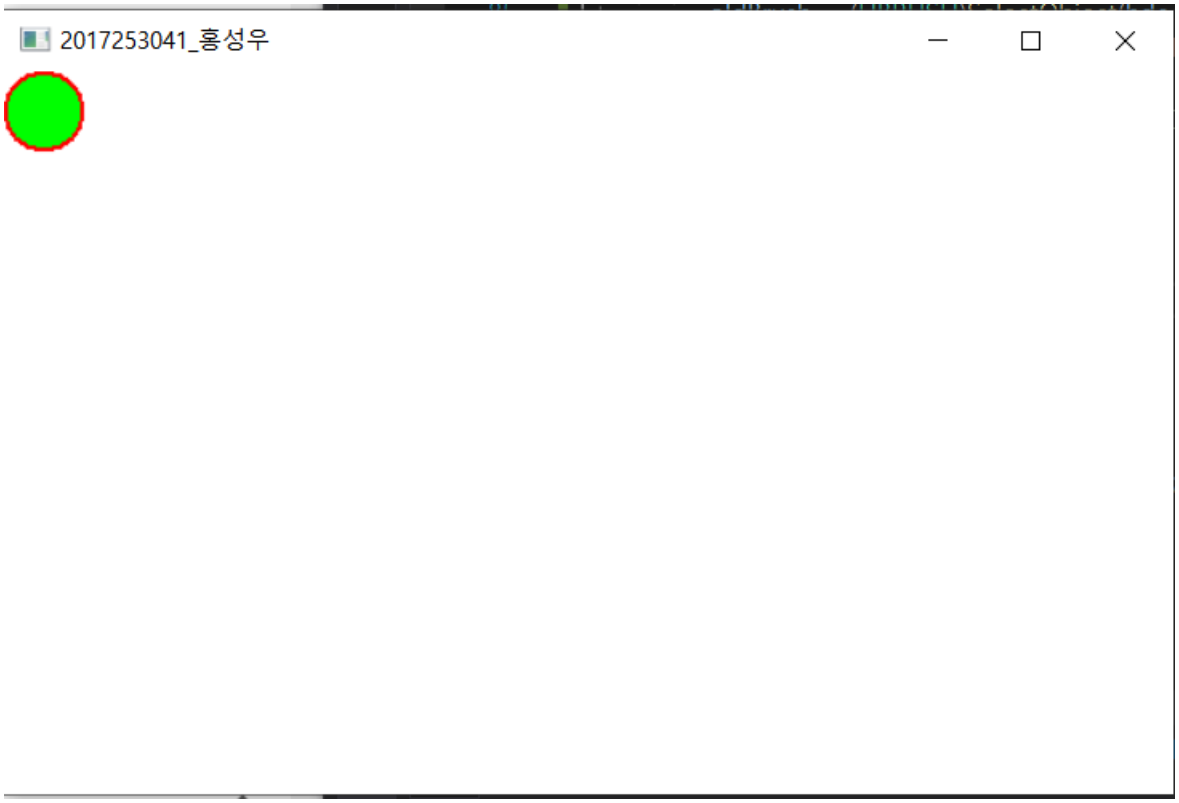


#3

```
#include <windows.h>
#include <TCHAR.H>
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);
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;
        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 = NULL;
        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
        {
                TranslateMessage(&msg);          // 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
                DispatchMessage(&msg);           // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                           // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                  // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}
#include <math.h>
#define BSIZE 20
float LengthPts(int x1, int y1, int x2, int y2) {
        return(sqrt((float)((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1))));
}
```

```
BOOL InCircle(int x, int y, int mx, int my) {
	if (LengthPts(x, y, mx, my) < BSIZE) return TRUE;
	else return FALSE;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam)
	//	WinDef.h 에서 정의
	//	wPram > unsigned ptr, lParam > long ptr
{
	HDC hdc;
	PAINTSTRUCT ps;
	static int x, y;
	static BOOL Selection;
	int mx, my;
	HPEN hPen, oldPen;
	HBRUSH hBrush, oldBrush;

	switch (iMsg)
	{
	case WM_CREATE:
		x = 20;
		y = 20;
		Selection = FALSE;
		break;
	case WM_PAINT:
		hdc = BeginPaint(hwnd, &ps);
		if (Selection) {
			hBrush = CreateSolidBrush(RGB(0, 255, 0));
			oldBrush = (HBRUSH)SelectObject(hdc, hBrush);
			Ellipse(hdc, x - BSIZE, y - BSIZE, x + BSIZE, y + BSIZE);
		}
		hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
		oldPen = (HPEN)SelectObject(hdc, hPen);
		Ellipse(hdc, x - BSIZE, y - BSIZE, x + BSIZE, y + BSIZE);
		SelectObject(hdc, oldPen);
		DeleteObject(hPen);
		EndPaint(hwnd, &ps);
		break;
	case WM_LBUTTONDOWN:
		mx = LOWORD(lParam);
		my = HIWORD(lParam);
		if (InCircle(x, y, mx, my)) Selection = TRUE;
		InvalidateRgn(hwnd, NULL, TRUE);
		break;
	case WM_LBUTTONUP:
		Selection = FALSE;
		InvalidateRgn(hwnd, NULL, TRUE);
		break;
	case WM_DESTROY:
		PostQuitMessage(0);
		break;
	}
	return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```



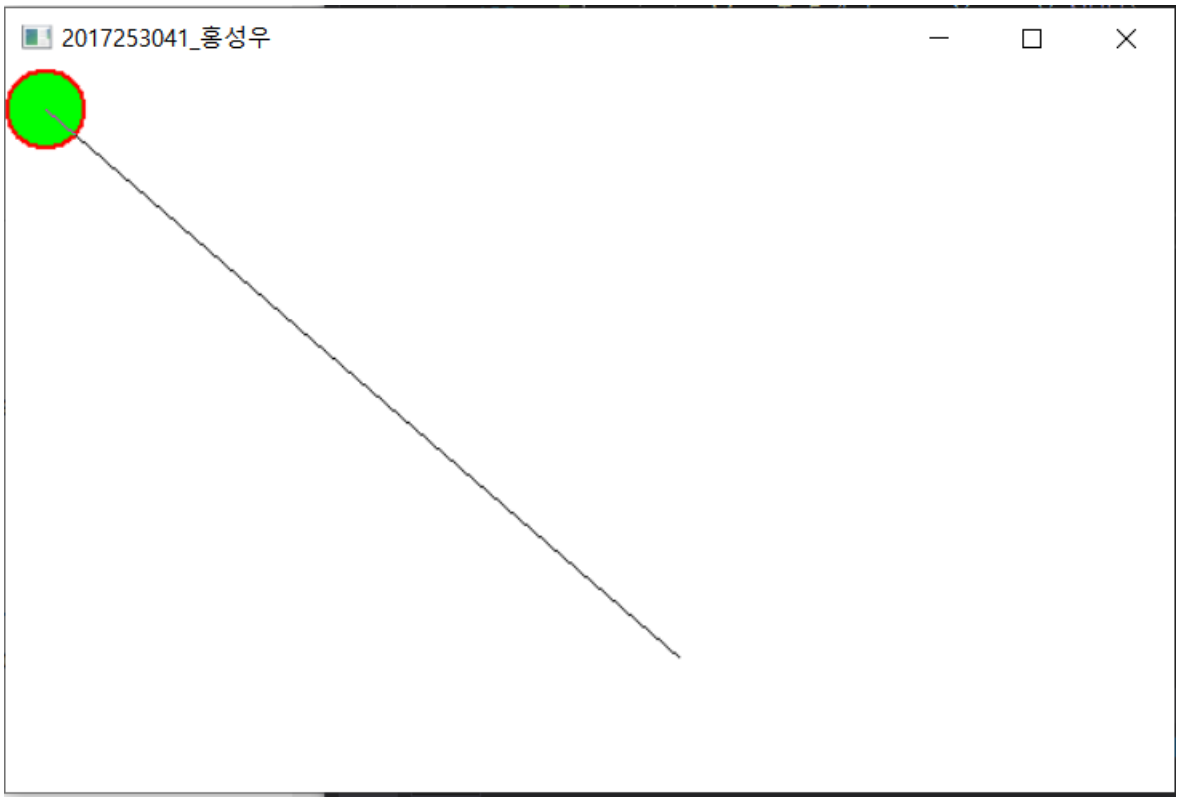
#4

```
#include <windows.h>
#include <TCHAR.H>
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);
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;
        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 = NULL;
        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
        {
                TranslateMessage(&msg);         // 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
                DispatchMessage(&msg);          // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                       // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                 // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}
#include <math.h>
#define BSIZE 20
float LengthPts(int x1, int y1, int x2, int y2) {
        return(sqrt((float)((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1))));
}
BOOL InCircle(int x, int y, int mx, int my) {
        if (LengthPts(x, y, mx, my) < BSIZE) return TRUE;
        else return FALSE;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam)
        //      WinDef.h 에서 정의
        //      wPram > unsigned ptr, lParam > long ptr
{
        HDC hdc;
        PAINTSTRUCT ps;
        static int x, y;
        static int startX, startY, oldX, oldY;
        static BOOL Drag;
        static BOOL Selection;
        int endX, endY;
        int mx, my;
        HPEN hPen, oldPen;
        HBRUSH hBrush, oldBrush;

        switch (iMsg)
```

```
{
case WM_CREATE:
        x = 20;
        y = 20;
        startX = oldX = 20;
        startY = oldY = 20;
        Drag = false;
        Selection = FALSE;
        break;
case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        MoveToEx(hdc, startX, startY, NULL);
        LineTo(hdc, oldX, oldY);
        if (Selection) {
                hBrush = CreateSolidBrush(RGB(0, 255, 0));
                oldBrush = (HBRUSH)SelectObject(hdc, hBrush);
                Ellipse(hdc, x - BSIZE, y - BSIZE, x + BSIZE, y + BSIZE);
        }
        hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
        oldPen = (HPEN)SelectObject(hdc, hPen);
        Ellipse(hdc, x - BSIZE, y - BSIZE, x + BSIZE, y + BSIZE);
        SelectObject(hdc, oldPen);
        DeleteObject(hPen);
        EndPaint(hwnd, &ps);
        break;
case WM_LBUTTONDOWN:
        Drag = TRUE;
        mx = LOWORD(lParam);
        my = HIWORD(lParam);
        if (InCircle(x, y, mx, my)) Selection = TRUE;
        InvalidateRgn(hwnd, NULL, TRUE);
        break;
case WM_LBUTTONUP:
        Drag = FALSE;
        Selection = FALSE;
        InvalidateRgn(hwnd, NULL, TRUE);
        break;
case WM_MOUSEMOVE:
        hdc = GetDC(hwnd);
        if (Drag) {
                SetROP2(hdc, R2_XORPEN);
                SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN));
                endX = LOWORD(lParam);
                endY = HIWORD(lParam);
                MoveToEx(hdc, startX, startY, NULL);
                LineTo(hdc, oldX, oldY);
                MoveToEx(hdc, startX, startY, NULL);
                LineTo(hdc, endX, endY);
                oldX = endX;
                oldY = endY;
        }
        ReleaseDC(hwnd, hdc);
        break;
case WM_DESTROY:
        PostQuitMessage(0);
```

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

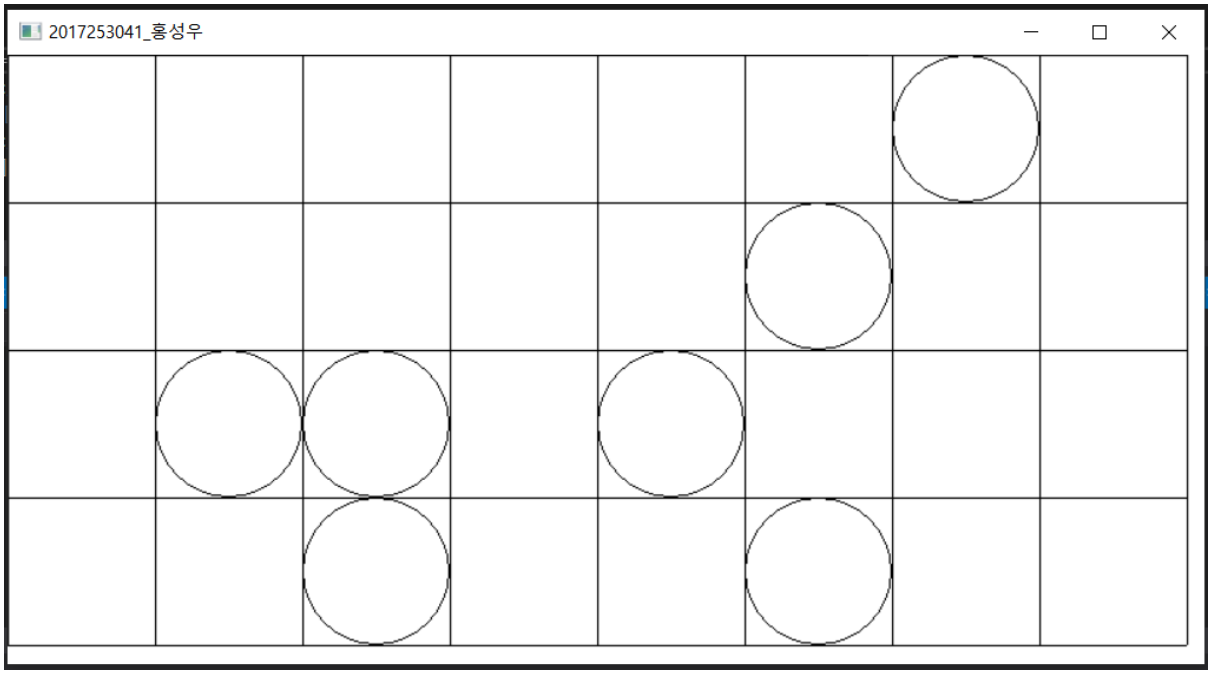


#5

```
#include <windows.h>
#include <TCHAR.H>
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);
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;
        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 = NULL;
	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
	{
		TranslateMessage(&msg);		// 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
		DispatchMessage(&msg);		// WndProc() 함수 호출과 WndProc()으로
메세지 전달
	}						// 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)
	//	WinDef.h 에서 정의
	//	wPram > unsigned ptr, lParam > long ptr
{
    HDC hdc;
    PAINTSTRUCT ps;
    static int  mX, mY;
    static int circleX[4][8], circleY[4][8];
    static int circleColor[4][8];
    static int circleCount = 0;
    switch (iMsg)
    {
    case WM_CREATE:
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 8; j++)
            {
                circleX[i][j] = -100;
                circleY[i][j] = -100;
            }
```

```
        InvalidateRgn(hwnd, NULL, TRUE);
        break;
    case WM_LBUTTONDOWN:
        hdc = GetDC(hwnd);
        for (int i = 0; i < 5; i++)
        {
            MoveToEx(hdc, 0, i * 100, NULL);
            LineTo(hdc, 800, i * 100);
        }
        for (int i = 0; i < 9; i++)
        {
            MoveToEx(hdc, i * 100, 0, NULL);
            LineTo(hdc, i * 100, 400);
        }
        mX = LOWORD(lParam);
        mY = HIWORD(lParam);
        circleX[mY / 100][mX / 100] = (mX / 100) * 100;
        circleY[mY / 100][mX / 100] = (mY / 100) * 100;
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 8; j++)
            {
                Ellipse(hdc, circleX[i][j], circleY[i][j], circleX[i][j] + 100, circleY[i][j] +
100);
            }
        ReleaseDC(hwnd, hdc);
        break;
    case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        for (int i = 0; i < 5; i++)
        {
            MoveToEx(hdc, 0, i * 100, NULL);
            LineTo(hdc, 800, i * 100);
        }
        for (int i = 0; i < 9; i++)
        {
            MoveToEx(hdc, i * 100, 0, NULL);
            LineTo(hdc, i * 100, 400);
        }
        EndPaint(hwnd, &ps);
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    }
    return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```


2017253041_홍성우