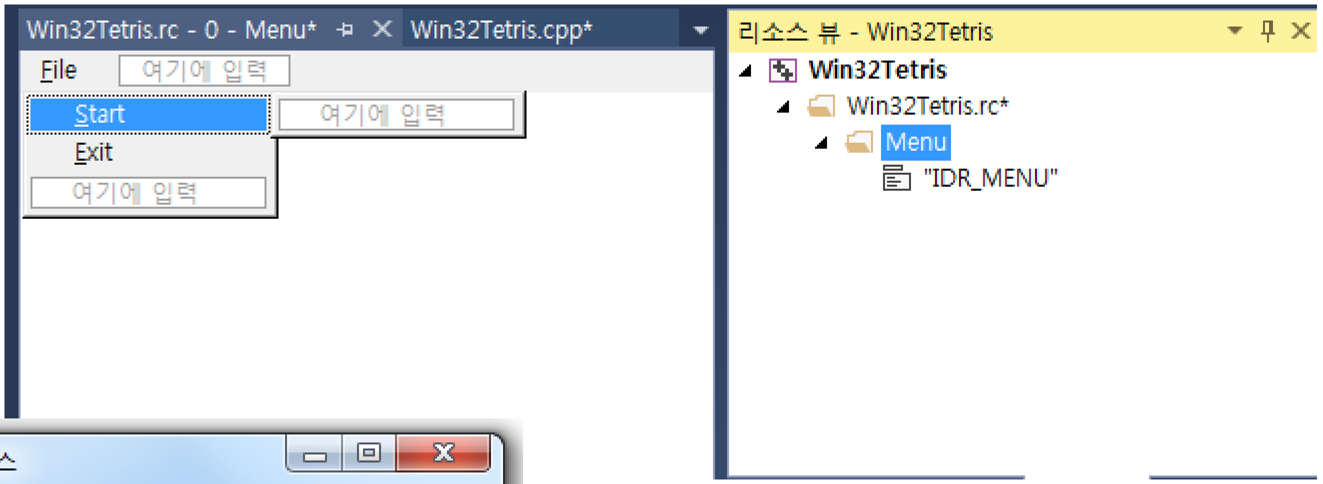
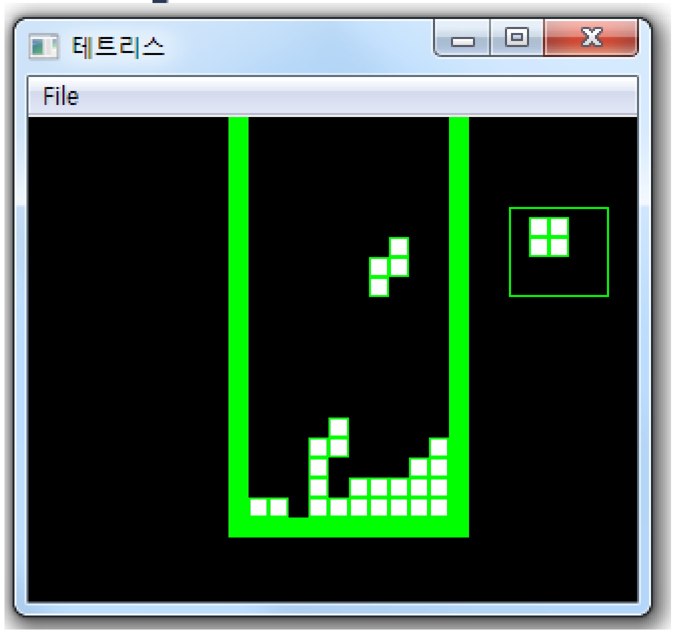
API Programming

Tetris Game

Project: Win32Tetris

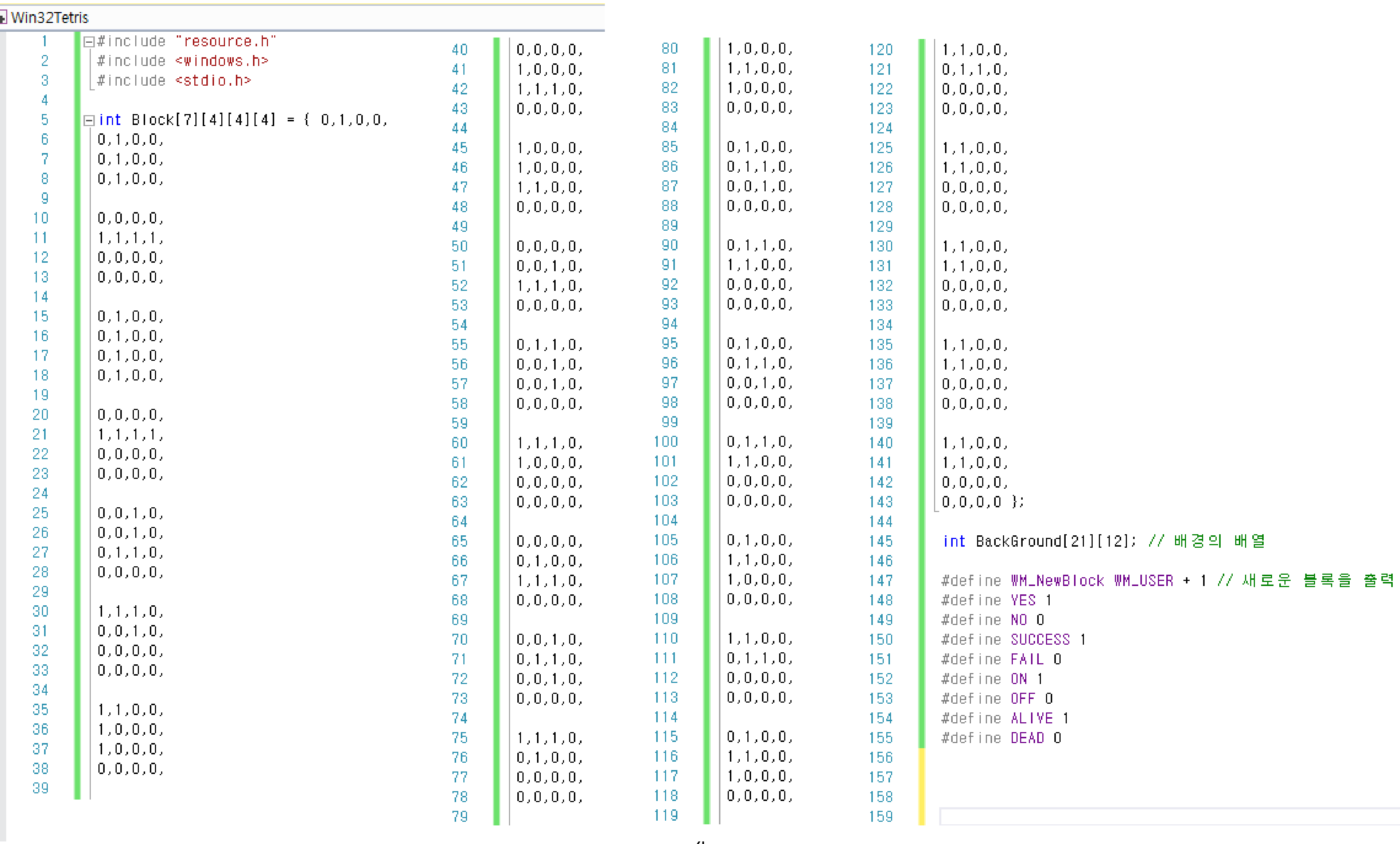
Menu Structure





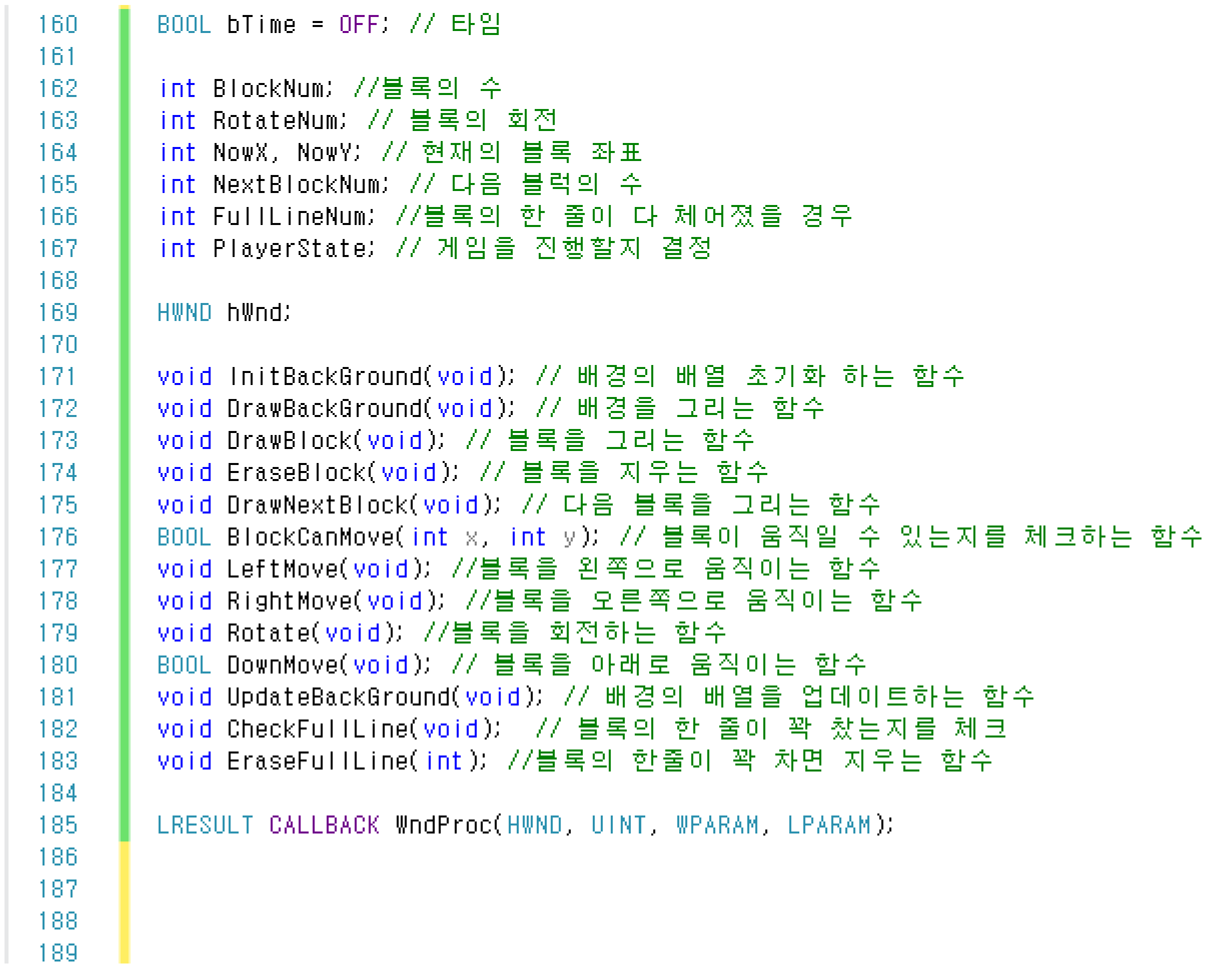
Project: Win32Tetris

Declare


Win32Tetris

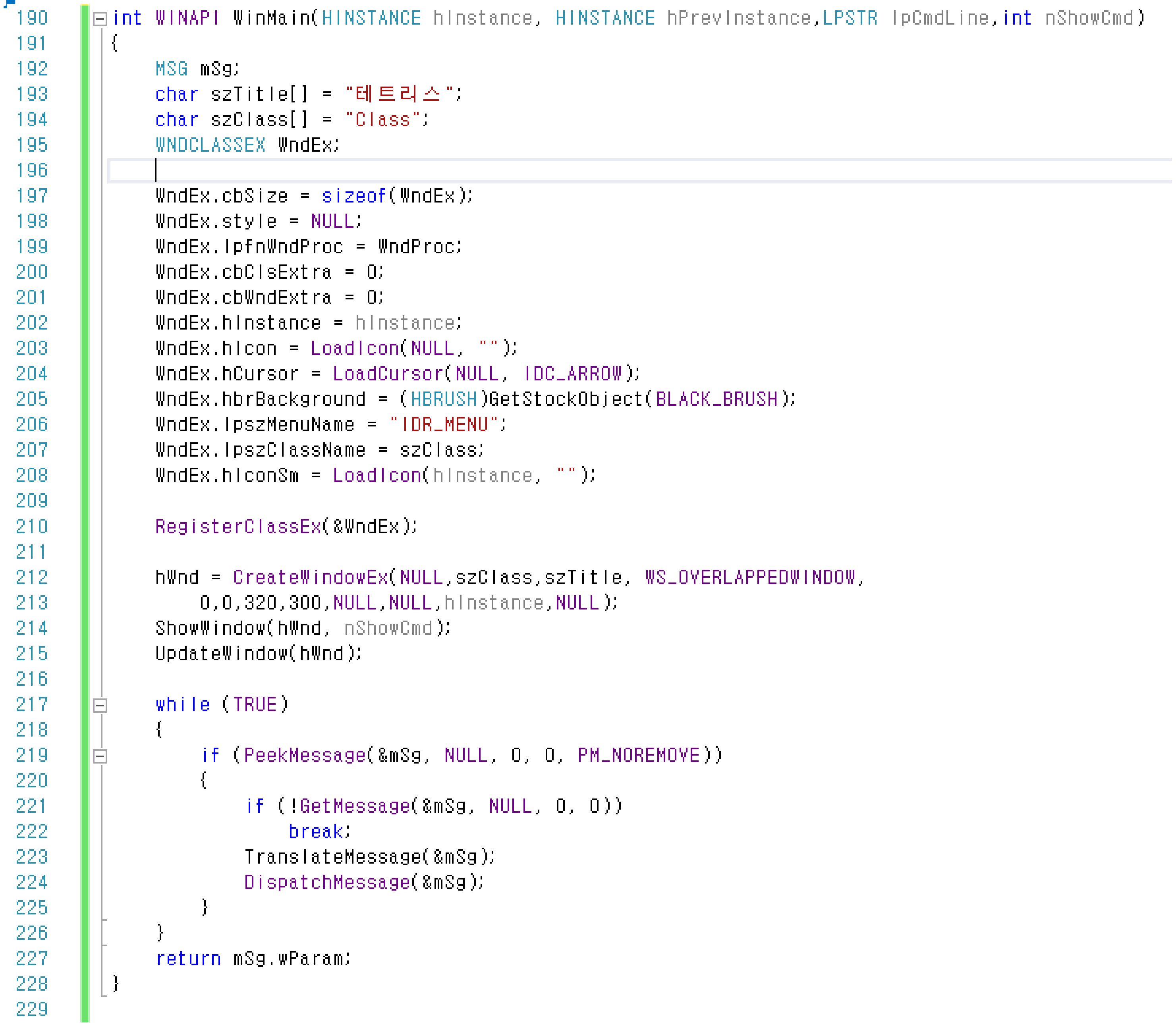
```
#include "resource.h"
#include <windows.h>
#include <stdio.h>

int Block[7][4][4][4] = { 0,1,0,0,
0,1,0,0,
0,1,0,0,
0,1,0,0,

0,0,0,0,
1,1,1,1,
0,0,0,0,
0,0,0,0,

0,1,0,0,
0,1,0,0,
0,1,0,0,
0,1,0,0,

0,0,0,0,
1,1,1,1,
0,0,0,0,
0,0,0,0,

0,0,1,0,
0,0,1,0,
0,1,1,0,
0,0,0,0,

1,1,1,0,
0,0,1,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,0,0,0,
1,0,0,0,
0,0,0,0,

0,0,0,0,
1,0,0,0,
1,1,1,0,
0,0,0,0,

1,0,0,0,
1,0,0,0,
1,1,0,0,
0,0,0,0,

0,0,0,0,
0,0,1,0,
1,1,1,0,
0,0,0,0,

0,1,1,0,
0,0,1,0,
0,0,1,0,
0,0,0,0,

1,1,1,0,
1,0,0,0,
0,0,0,0,
0,0,0,0,

0,0,0,0,
0,1,0,0,
1,1,1,0,
0,0,0,0,

0,0,1,0,
0,1,1,0,
0,0,1,0,
0,0,0,0,

1,1,1,0,
0,1,0,0,
0,0,0,0,
0,0,0,0,

1,0,0,0,
1,1,0,0,
1,0,0,0,
0,0,0,0,

0,1,0,0,
0,1,1,0,
0,0,1,0,
0,0,0,0,

0,1,1,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

0,1,0,0,
0,1,1,0,
0,0,1,0,
0,0,0,0,

0,1,1,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

0,1,0,0,
1,1,0,0,
1,0,0,0,
0,0,0,0,

1,1,0,0,
0,1,1,0,
0,0,0,0,
0,0,0,0,

0,1,0,0,
1,1,0,0,
1,0,0,0,
0,0,0,0,

1,1,0,0,
0,1,1,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0 };

int BackGround[21][12]; // 배경의 배열

#define WM_NewBlock WM_USER + 1 // 새로운 블록을 출력
#define YES 1
#define NO 0
#define SUCCESS 1
#define FAIL 0
#define ON 1
#define OFF 0
#define ALIVE 1
#define DEAD 0
```

Declare



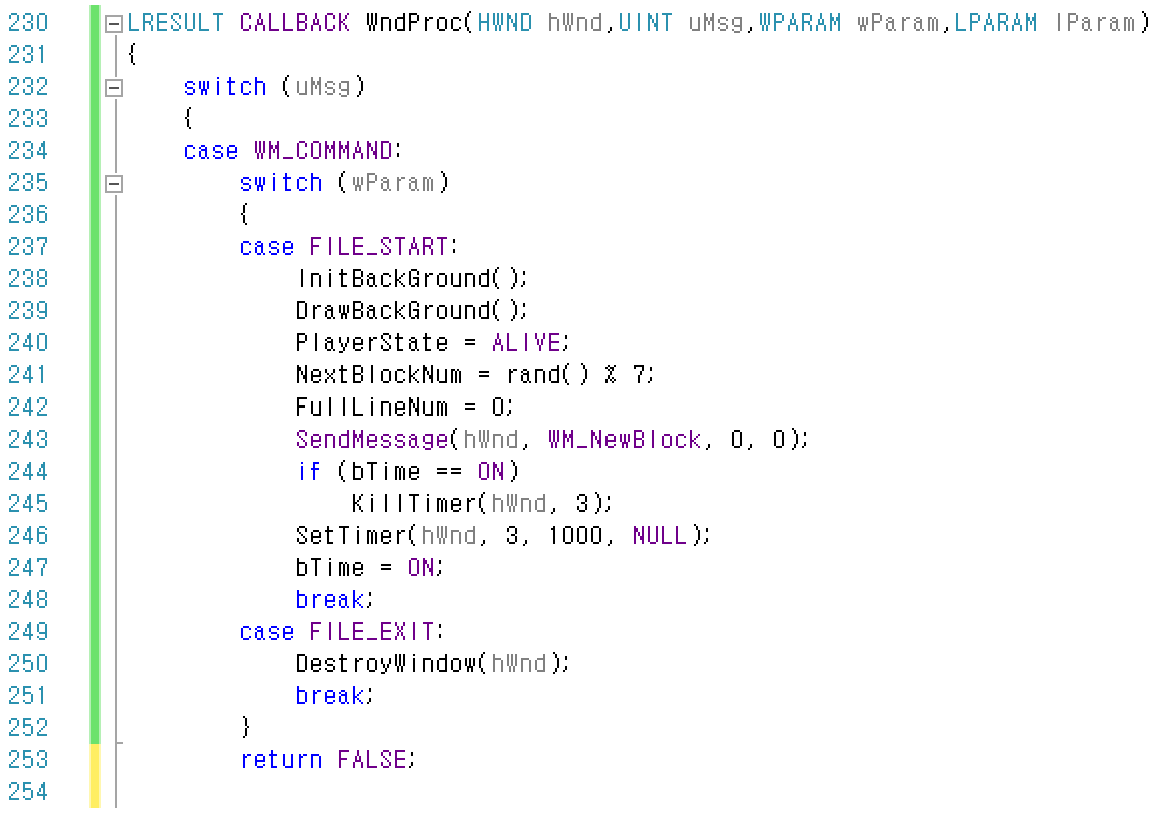
```c
BOOL bTime = OFF; // 타임

int BlockNum; //블록의 수
int RotateNum; // 블록의 회전
int NowX, NowY; // 현재의 블록 좌표
int NextBlockNum; // 다음 블럭의 수
int FullLineNum; //블록의 한 줄이 다 체어졌을 경우
int PlayerState; // 게임을 진행할지 결정

HWND hWnd;

void InitBackGround(void); // 배경의 배열 초기화 하는 함수
void DrawBackGround(void); // 배경을 그리는 함수
void DrawBlock(void); // 블록을 그리는 함수
void EraseBlock(void); // 블록을 지우는 함수
void DrawNextBlock(void); // 다음 블록을 그리는 함수
BOOL BlockCanMove(int x, int y); // 블록이 움직일 수 있는지를 체크하는 함수
void LeftMove(void); //블록을 왼쪽으로 움직이는 함수
void RightMove(void); //블록을 오른쪽으로 움직이는 함수
void Rotate(void); //블록을 회전하는 함수
BOOL DownMove(void); // 블록을 아래로 움직이는 함수
void UpdateBackGround(void); // 배경의 배열을 업데이트하는 함수
void CheckFullLine(void);  // 블록의 한 줄이 꽉 찼는지를 체크
void EraseFullLine(int); //블록의 한줄이 꽉 차면 지우는 함수

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
```

WinMain



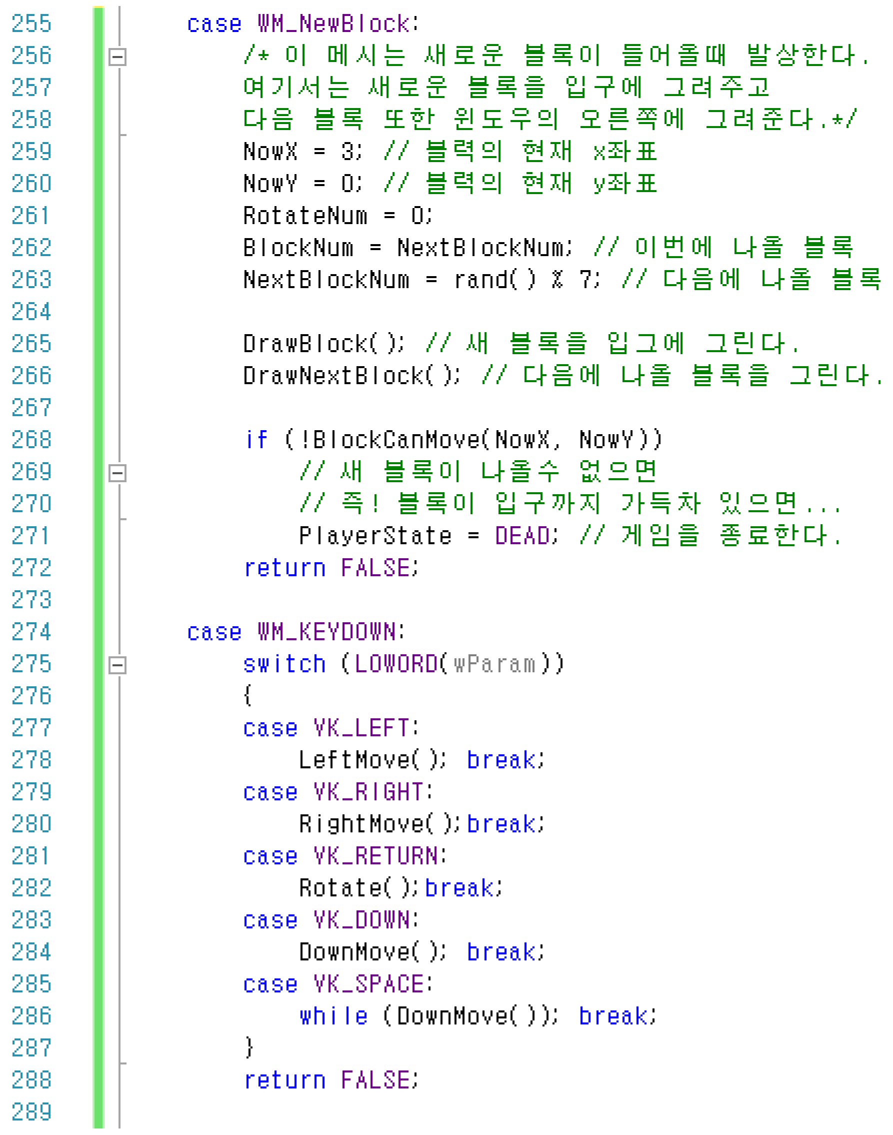
```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR lpCmdLine,int nShowCmd)
{
    MSG mSg;
    char szTitle[] = "테트리스";
    char szClass[] = "Class";
    WNDCLASSEX WndEx;

    WndEx.cbSize = sizeof(WndEx);
    WndEx.style = NULL;
    WndEx.lpfnWndProc = WndProc;
    WndEx.cbClsExtra = 0;
    WndEx.cbWndExtra = 0;
    WndEx.hInstance = hInstance;
    WndEx.hIcon = LoadIcon(NULL, "");
    WndEx.hCursor = LoadCursor(NULL, IDC_ARROW);
    WndEx.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);
    WndEx.lpszMenuName = "IDR_MENU";
    WndEx.lpszClassName = szClass;
    WndEx.hIconSm = LoadIcon(hInstance, "");

    RegisterClassEx(&WndEx);

    hWnd = CreateWindowEx(NULL,szClass,szTitle, WS_OVERLAPPEDWINDOW,
        0,0,320,300,NULL,NULL,hInstance,NULL);
    ShowWindow(hWnd, nShowCmd);
    UpdateWindow(hWnd);

    while (TRUE)
    {
        if (PeekMessage(&mSg, NULL, 0, 0, PM_NOREMOVE))
        {
            if (!GetMessage(&mSg, NULL, 0, 0))
                break;
            TranslateMessage(&mSg);
            DispatchMessage(&mSg);
        }
    }
    return mSg.wParam;
}
```

Callback



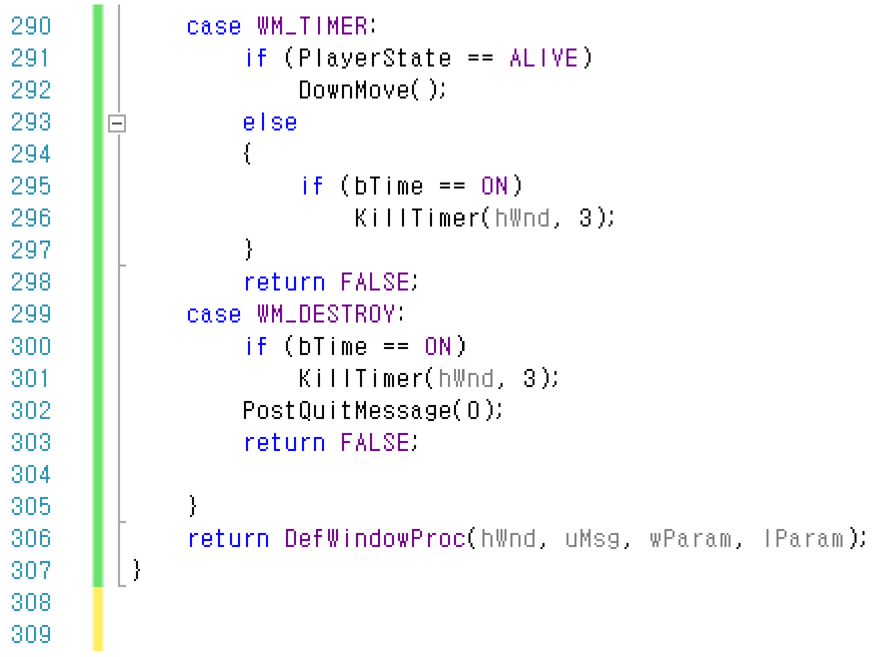
```
LRESULT CALLBACK WndProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch (uMsg)
    {
    case WM_COMMAND:
        switch (wParam)
        {
        case FILE_START:
            InitBackGround( );
            DrawBackGround( );
            PlayerState = ALIVE;
            NextBlockNum = rand( ) % 7;
            FullLineNum = 0;
            SendMessage(hWnd, WM_NewBlock, 0, 0);
            if (bTime == ON)
                KillTimer(hWnd, 3);
            SetTimer(hWnd, 3, 1000, NULL);
            bTime = ON;
            break;
        case FILE_EXIT:
            DestroyWindow(hWnd);
            break;
        }
        return FALSE;

```

Callback



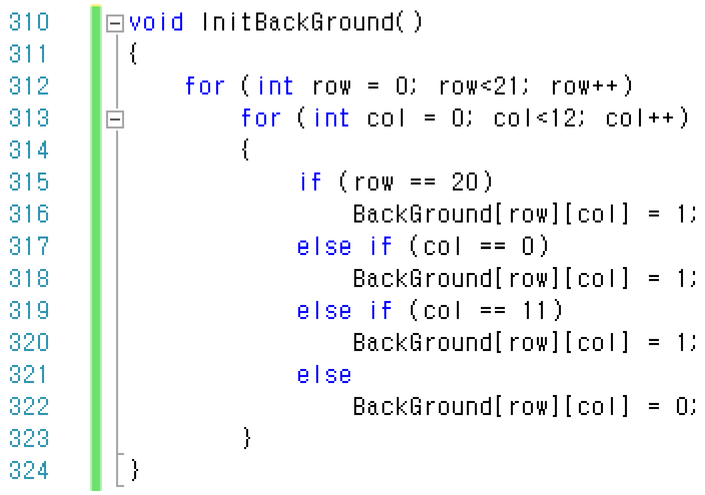
```
case WM_NewBlock:
    /* 이 메시는 새로운 블록이 들어올때 발상한다.
    여기서는 새로운 블록을 입구에 그려주고
    다음 블록 또한 윈도우의 오른쪽에 그려준다.*/
    NowX = 3; // 블력의 현재 x좌표
    NowY = 0; // 블력의 현재 y좌표
    RotateNum = 0;
    BlockNum = NextBlockNum; // 이번에 나올 블록
    NextBlockNum = rand( ) % 7; // 다음에 나올 블록

    DrawBlock( ); // 새 블록을 입그에 그린다.
    DrawNextBlock( ); // 다음에 나올 블록을 그린다.

    if (!BlockCanMove(NowX, NowY))
        // 새 블록이 나올수 없으면
        // 즉! 블록이 입구까지 가득차 있으면...
        PlayerState = DEAD; // 게임을 종료한다.
    return FALSE;

case WM_KEYDOWN:
    switch (LOWORD(wParam))
    {
    case VK_LEFT:
        LeftMove( ); break;
    case VK_RIGHT:
        RightMove( );break;
    case VK_RETURN:
        Rotate( );break;
    case VK_DOWN:
        DownMove( ); break;
    case VK_SPACE:
        while (DownMove( )); break;
    }
    return FALSE;
```

Callback



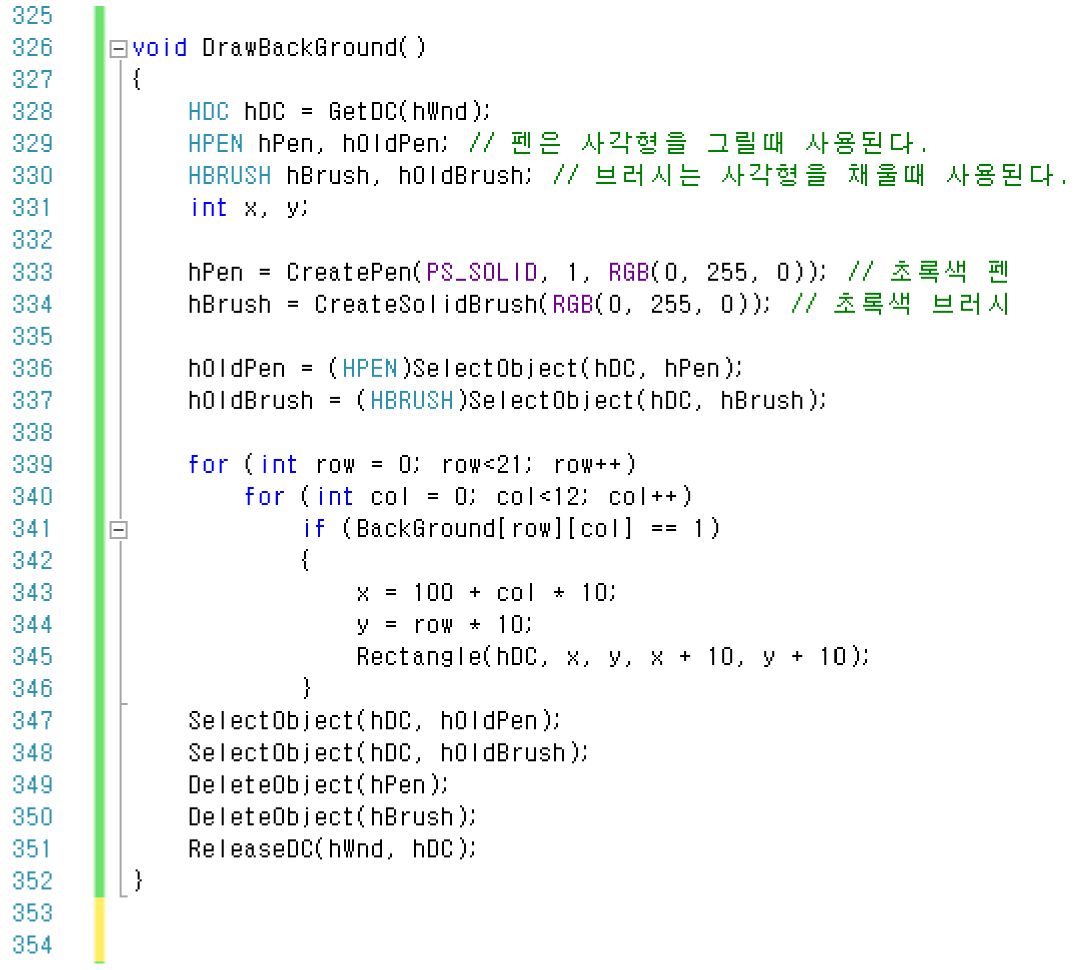
```
        case WM_TIMER:
            if (PlayerState == ALIVE)
                DownMove();
            else
            {
                if (bTime == ON)
                    KillTimer(hWnd, 3);
            }
            return FALSE;
        case WM_DESTROY:
            if (bTime == ON)
                KillTimer(hWnd, 3);
            PostQuitMessage(0);
            return FALSE;

        }
        return DefWindowProc(hWnd, uMsg, wParam, lParam);
    }
```

InitBackGround



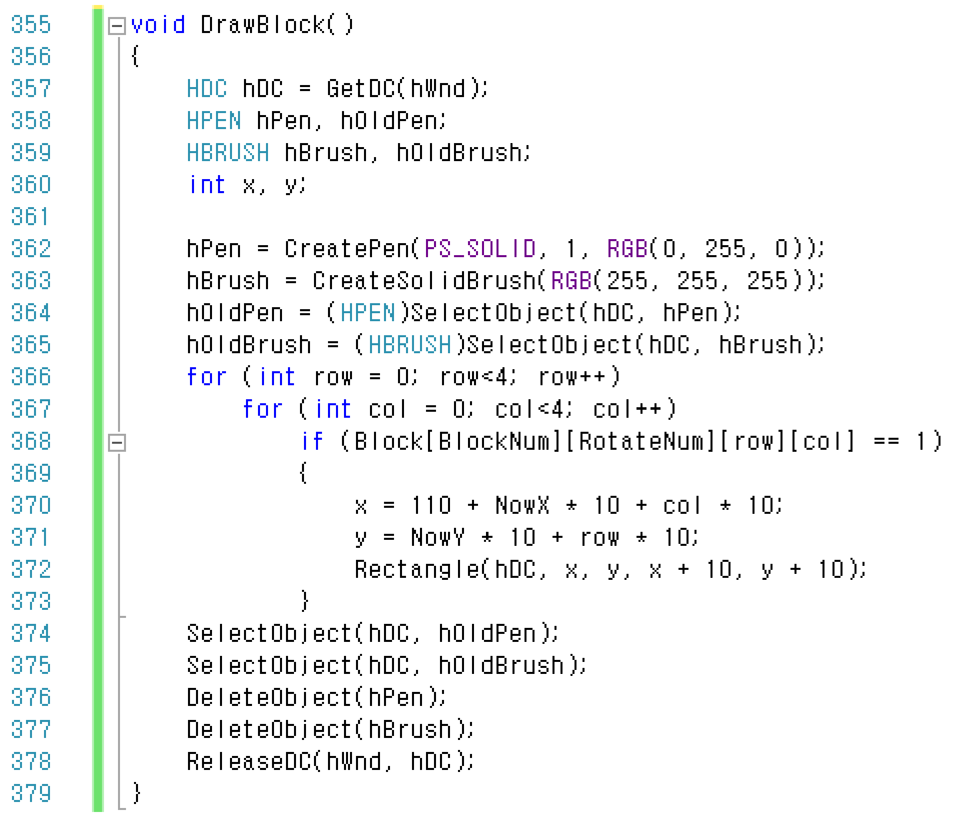
```
void InitBackGround( )
{
	for (int row = 0; row<21; row++)
		for (int col = 0; col<12; col++)
		{
			if (row == 20)
				BackGround[row][col] = 1;
			else if (col == 0)
				BackGround[row][col] = 1;
			else if (col == 11)
				BackGround[row][col] = 1;
			else
				BackGround[row][col] = 0;
		}
}
```

DrawBackGround



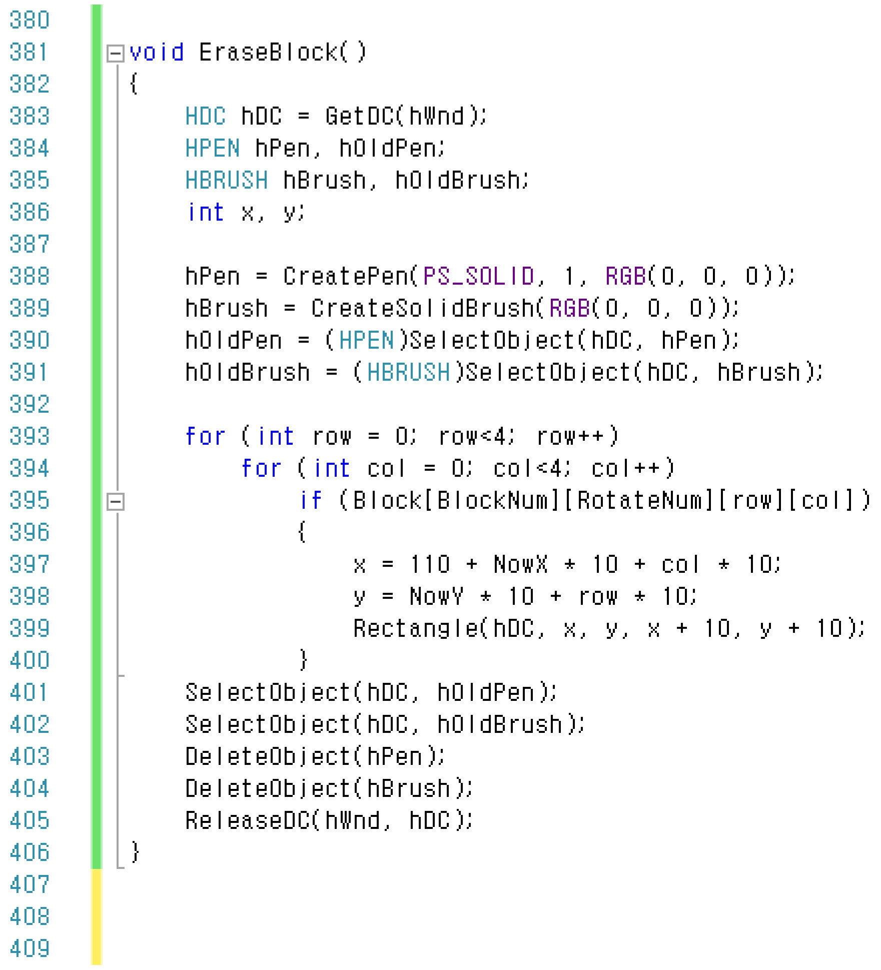
```
void DrawBackGround( )
{
    HDC hDC = GetDC(hWnd);
    HPEN hPen, hOldPen; // 펜은 사각형을 그릴때 사용된다.
    HBRUSH hBrush, hOldBrush; // 브러시는 사각형을 채울때 사용된다.
    int x, y;

    hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0)); // 초록색 펜
    hBrush = CreateSolidBrush(RGB(0, 255, 0)); // 초록색 브러시

    hOldPen = (HPEN)SelectObject(hDC, hPen);
    hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);

    for (int row = 0; row<21; row++)
        for (int col = 0; col<12; col++)
            if (BackGround[row][col] == 1)
            {
                x = 100 + col * 10;
                y = row * 10;
                Rectangle(hDC, x, y, x + 10, y + 10);
            }
    SelectObject(hDC, hOldPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hPen);
    DeleteObject(hBrush);
    ReleaseDC(hWnd, hDC);
}
```

DrawBlock



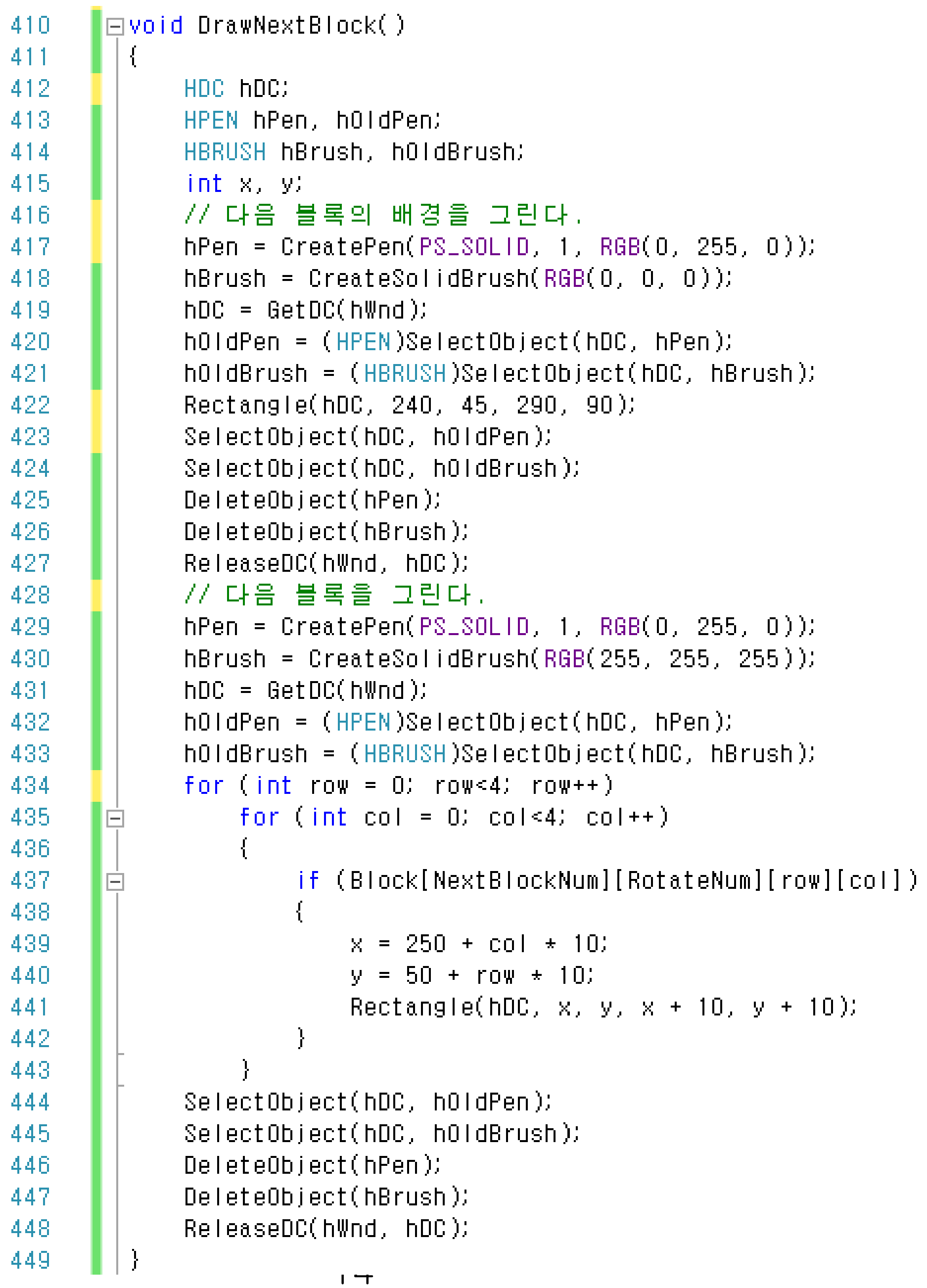
```
void DrawBlock( )
{
    HDC hDC = GetDC(hWnd);
    HPEN hPen, hOldPen;
    HBRUSH hBrush, hOldBrush;
    int x, y;

    hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0));
    hBrush = CreateSolidBrush(RGB(255, 255, 255));
    hOldPen = (HPEN)SelectObject(hDC, hPen);
    hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
    for (int row = 0; row<4; row++)
        for (int col = 0; col<4; col++)
            if (Block[BlockNum][RotateNum][row][col] == 1)
            {
                x = 110 + NowX * 10 + col * 10;
                y = NowY * 10 + row * 10;
                Rectangle(hDC, x, y, x + 10, y + 10);
            }
    SelectObject(hDC, hOldPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hPen);
    DeleteObject(hBrush);
    ReleaseDC(hWnd, hDC);
}
```

EraseBlock



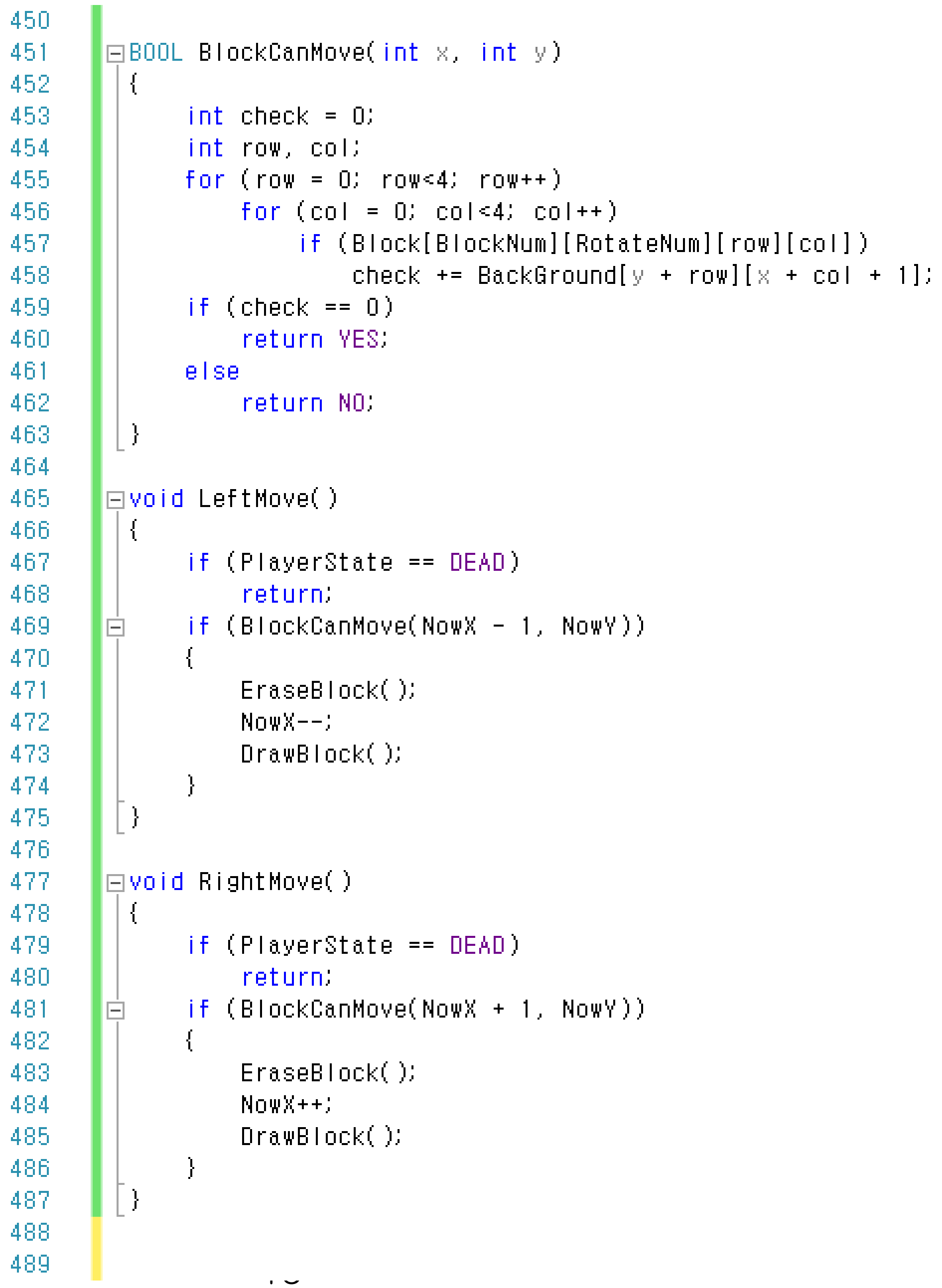
```
void EraseBlock( )
{
    HDC hDC = GetDC(hWnd);
    HPEN hPen, hOldPen;
    HBRUSH hBrush, hOldBrush;
    int x, y;

    hPen = CreatePen(PS_SOLID, 1, RGB(0, 0, 0));
    hBrush = CreateSolidBrush(RGB(0, 0, 0));
    hOldPen = (HPEN)SelectObject(hDC, hPen);
    hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);

    for (int row = 0; row<4; row++)
        for (int col = 0; col<4; col++)
            if (Block[BlockNum][RotateNum][row][col])
            {
                x = 110 + NowX * 10 + col * 10;
                y = NowY * 10 + row * 10;
                Rectangle(hDC, x, y, x + 10, y + 10);
            }
    SelectObject(hDC, hOldPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hPen);
    DeleteObject(hBrush);
    ReleaseDC(hWnd, hDC);
}
```

DrawNextBlock



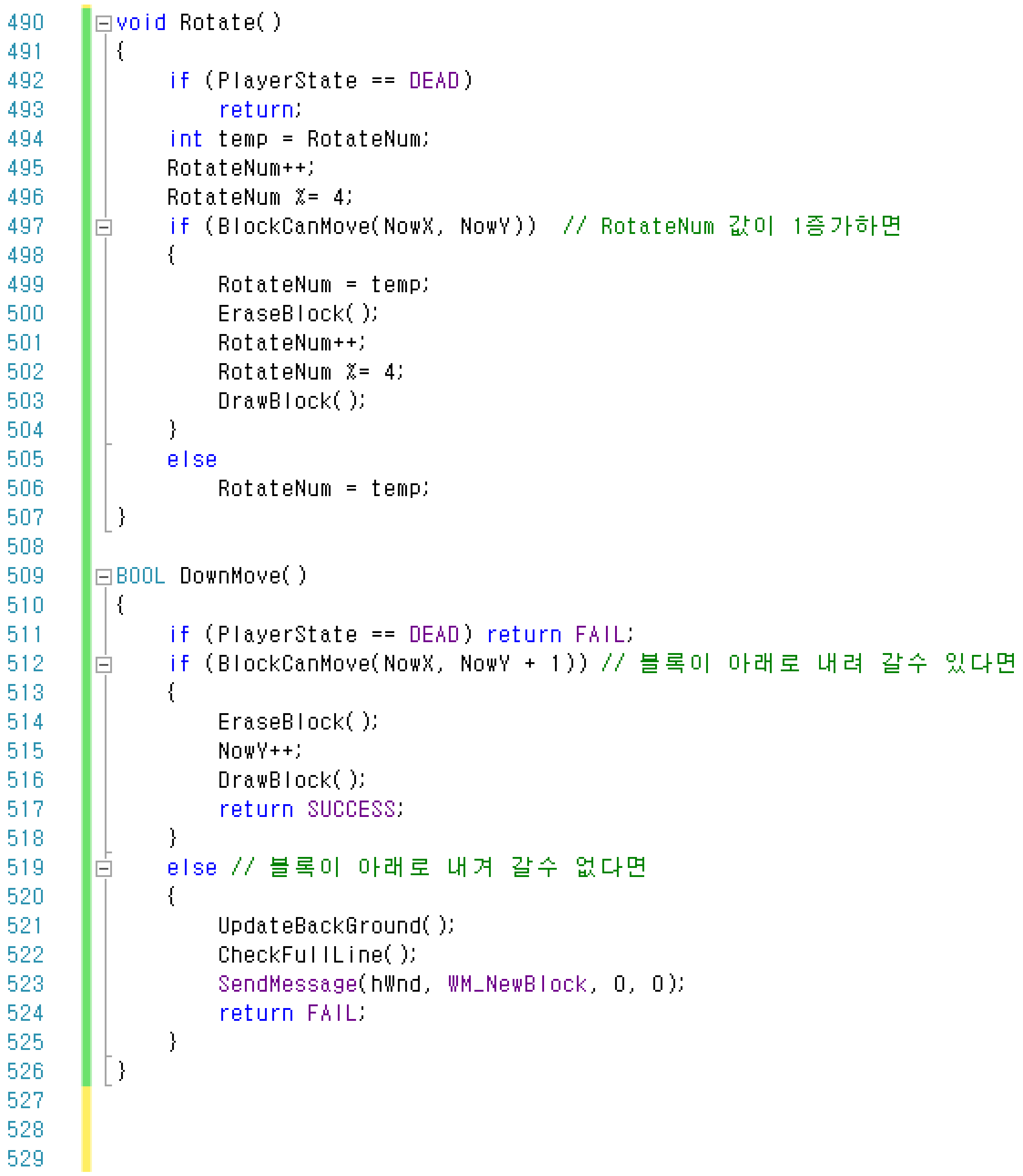
```
void DrawNextBlock()
{
    HDC hDC;
    HPEN hPen, hOldPen;
    HBRUSH hBrush, hOldBrush;
    int x, y;
    // 다음 블록의 배경을 그린다.
    hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0));
    hBrush = CreateSolidBrush(RGB(0, 0, 0));
    hDC = GetDC(hWnd);
    hOldPen = (HPEN)SelectObject(hDC, hPen);
    hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
    Rectangle(hDC, 240, 45, 290, 90);
    SelectObject(hDC, hOldPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hPen);
    DeleteObject(hBrush);
    ReleaseDC(hWnd, hDC);
    // 다음 블록을 그린다.
    hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0));
    hBrush = CreateSolidBrush(RGB(255, 255, 255));
    hDC = GetDC(hWnd);
    hOldPen = (HPEN)SelectObject(hDC, hPen);
    hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
    for (int row = 0; row<4; row++)
        for (int col = 0; col<4; col++)
        {
            if (Block[NextBlockNum][RotateNum][row][col])
            {
                x = 250 + col * 10;
                y = 50 + row * 10;
                Rectangle(hDC, x, y, x + 10, y + 10);
            }
        }
    SelectObject(hDC, hOldPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hPen);
    DeleteObject(hBrush);
    ReleaseDC(hWnd, hDC);
}
```

Block-Move



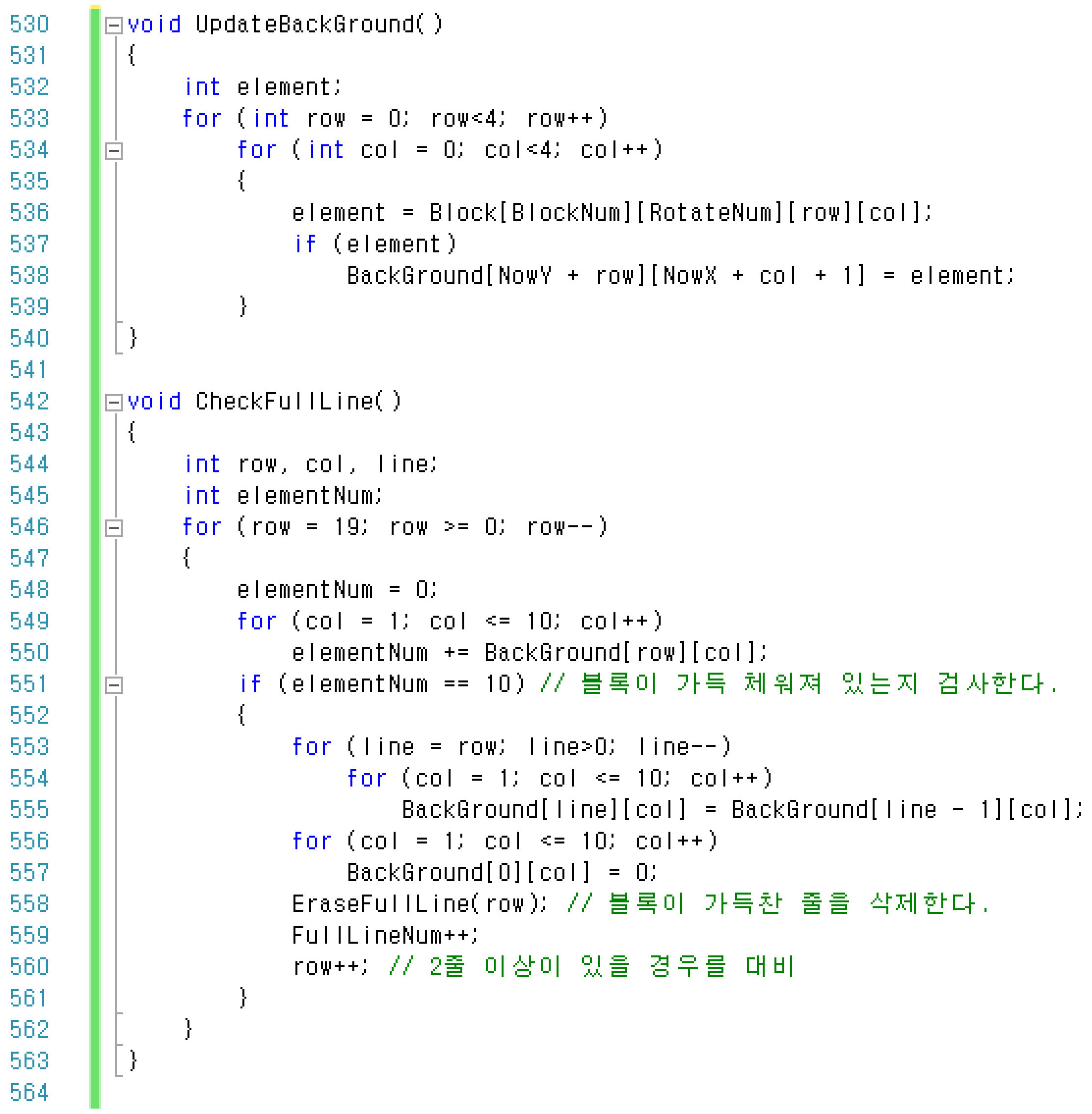
```
BOOL BlockCanMove(int x, int y)
{
    int check = 0;
    int row, col;
    for (row = 0; row<4; row++)
        for (col = 0; col<4; col++)
            if (Block[BlockNum][RotateNum][row][col])
                check += BackGround[y + row][x + col + 1];
    if (check == 0)
        return YES;
    else
        return NO;
}

void LeftMove()
{
    if (PlayerState == DEAD)
        return;
    if (BlockCanMove(NowX - 1, NowY))
    {
        EraseBlock();
        NowX--;
        DrawBlock();
    }
}

void RightMove()
{
    if (PlayerState == DEAD)
        return;
    if (BlockCanMove(NowX + 1, NowY))
    {
        EraseBlock();
        NowX++;
        DrawBlock();
    }
}
```

Block-Move



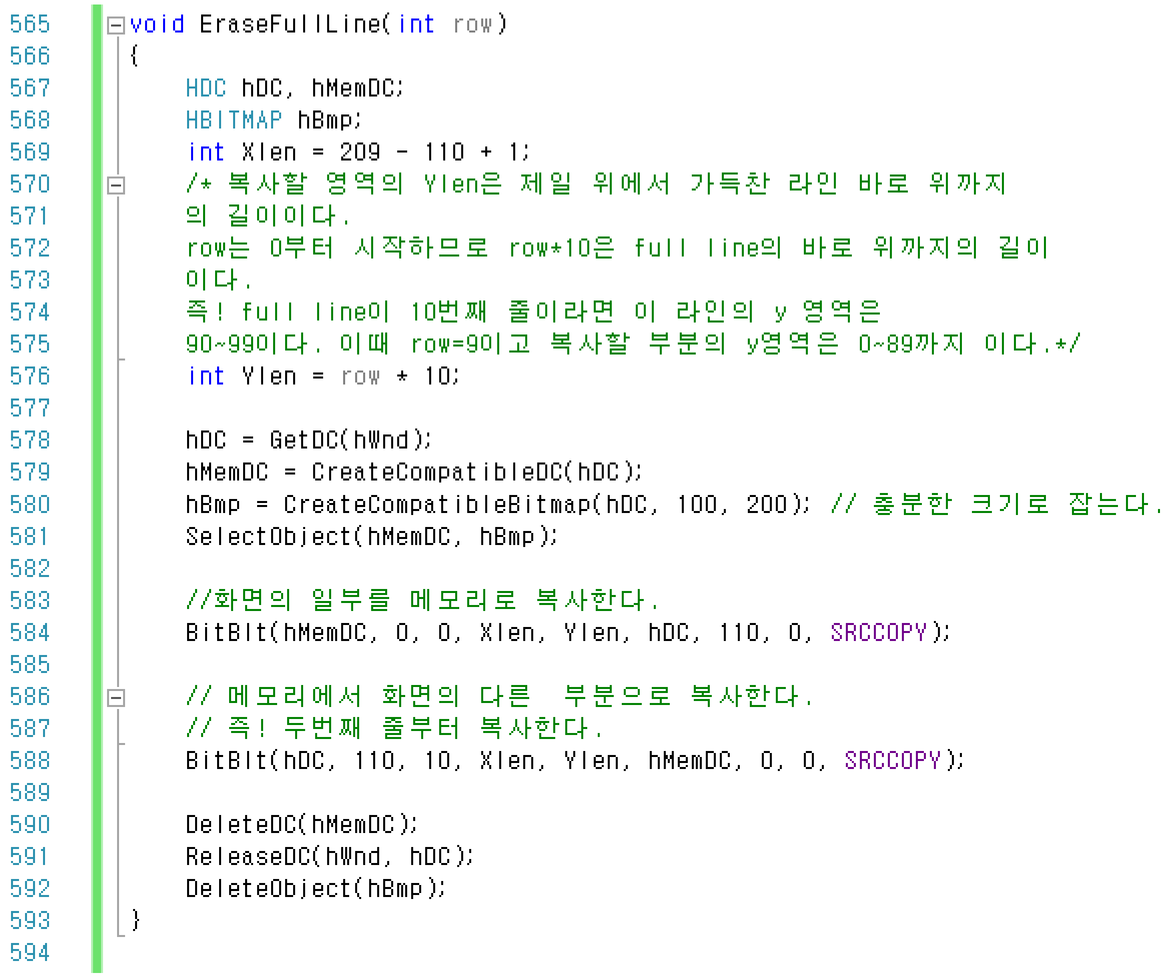
```
void Rotate( )
{
    if (PlayerState == DEAD)
        return;
    int temp = RotateNum;
    RotateNum++;
    RotateNum %= 4;
    if (BlockCanMove(NowX, NowY))  // RotateNum 값이 1증가하면
    {
        RotateNum = temp;
        EraseBlock( );
        RotateNum++;
        RotateNum %= 4;
        DrawBlock( );
    }
    else
        RotateNum = temp;
}

BOOL DownMove( )
{
    if (PlayerState == DEAD) return FAIL;
    if (BlockCanMove(NowX, NowY + 1)) // 블록이 아래로 내려 갈수 있다면
    {
        EraseBlock( );
        NowY++;
        DrawBlock( );
        return SUCCESS;
    }
    else // 블록이 아래로 내려 갈수 없다면
    {
        UpdateBackGround( );
        CheckFullLine( );
        SendMessage(hWnd, WM_NewBlock, 0, 0);
        return FAIL;
    }
}
```

Line



```
void UpdateBackGround()
{
    int element;
    for (int row = 0; row<4; row++)
        for (int col = 0; col<4; col++)
        {
            element = Block[BlockNum][RotateNum][row][col];
            if (element)
                BackGround[NowY + row][NowX + col + 1] = element;
        }
}

void CheckFullLine()
{
    int row, col, line;
    int elementNum;
    for (row = 19; row >= 0; row--)
    {
        elementNum = 0;
        for (col = 1; col <= 10; col++)
            elementNum += BackGround[row][col];
        if (elementNum == 10) // 블록이 가득 채워져 있는지 검사한다.
        {
            for (line = row; line>0; line--)
                for (col = 1; col <= 10; col++)
                    BackGround[line][col] = BackGround[line - 1][col];
            for (col = 1; col <= 10; col++)
                BackGround[0][col] = 0;
            EraseFullLine(row); // 블록이 가득찬 줄을 삭제한다.
            FullLineNum++;
            row++; // 2줄 이상이 있을 경우를 대비
        }
    }
}
```

Line



```
void EraseFullLine( int row)
{
	HDC hDC, hMemDC;
	HBITMAP hBmp;
	int Xlen = 209 - 110 + 1;
	/* 복사할 영역의 Ylen은 제일 위에서 가득찬 라인 바로 위까지
	의 길이이다.
	row는 0부터 시작하므로 row*10은 full line의 바로 위까지의 길이
	이다.
	즉! full line이 10번째 줄이라면 이 라인의 y 영역은
	90~99이다. 이때 row=9이고 복사할 부분의 y영역은 0~89까지 이다.*/
	int Ylen = row * 10;

	hDC = GetDC(hWnd);
	hMemDC = CreateCompatibleDC(hDC);
	hBmp = CreateCompatibleBitmap(hDC, 100, 200); // 충분한 크기로 잡는다.
	SelectObject(hMemDC, hBmp);

	//화면의 일부를 메모리로 복사한다.
	BitBlt(hMemDC, 0, 0, Xlen, Ylen, hDC, 110, 0, SRCCOPY);

	// 메모리에서 화면의 다른  부분으로 복사한다.
	// 즉! 두번째 줄부터 복사한다.
	BitBlt(hDC, 110, 10, Xlen, Ylen, hMemDC, 0, 0, SRCCOPY);

	DeleteDC(hMemDC);
	ReleaseDC(hWnd, hDC);
	DeleteObject(hBmp);
}
```