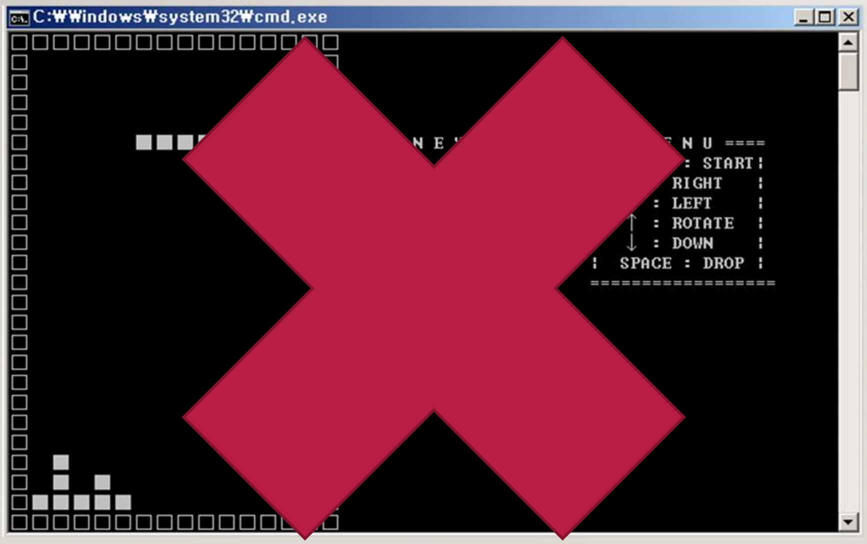
TETRIS WORLD

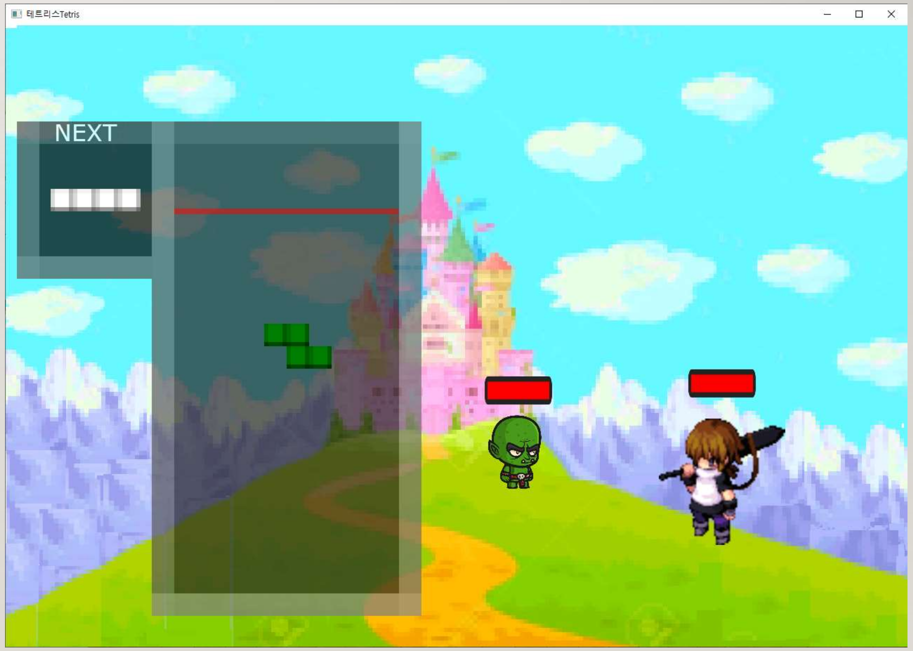
▶ PLAY

? HELP

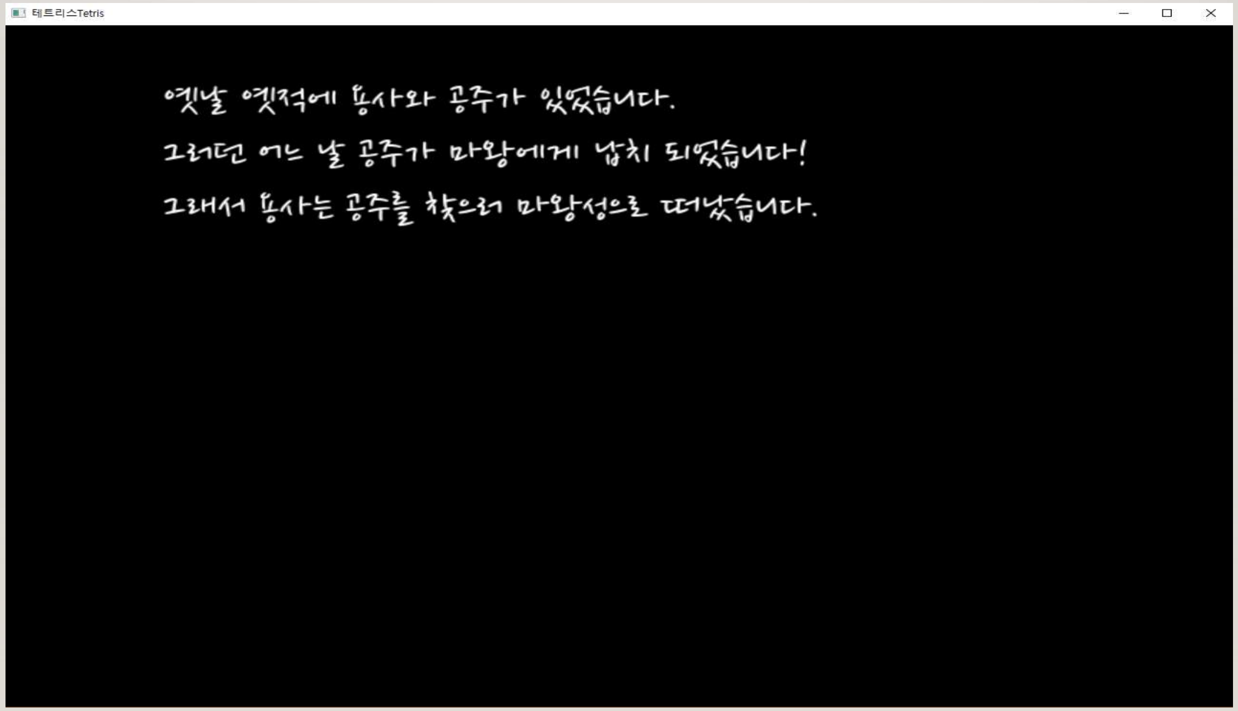
What We Make



```
C:\Windows\system32\cmd.exe
N E          N U ====
            : START¦
              RIGHT ¦
            : LEFT  ¦
          ↑ : ROTATE ¦
          ↓ : DOWN   ¦
       ¦  SPACE : DROP ¦
       =================
```



Story System

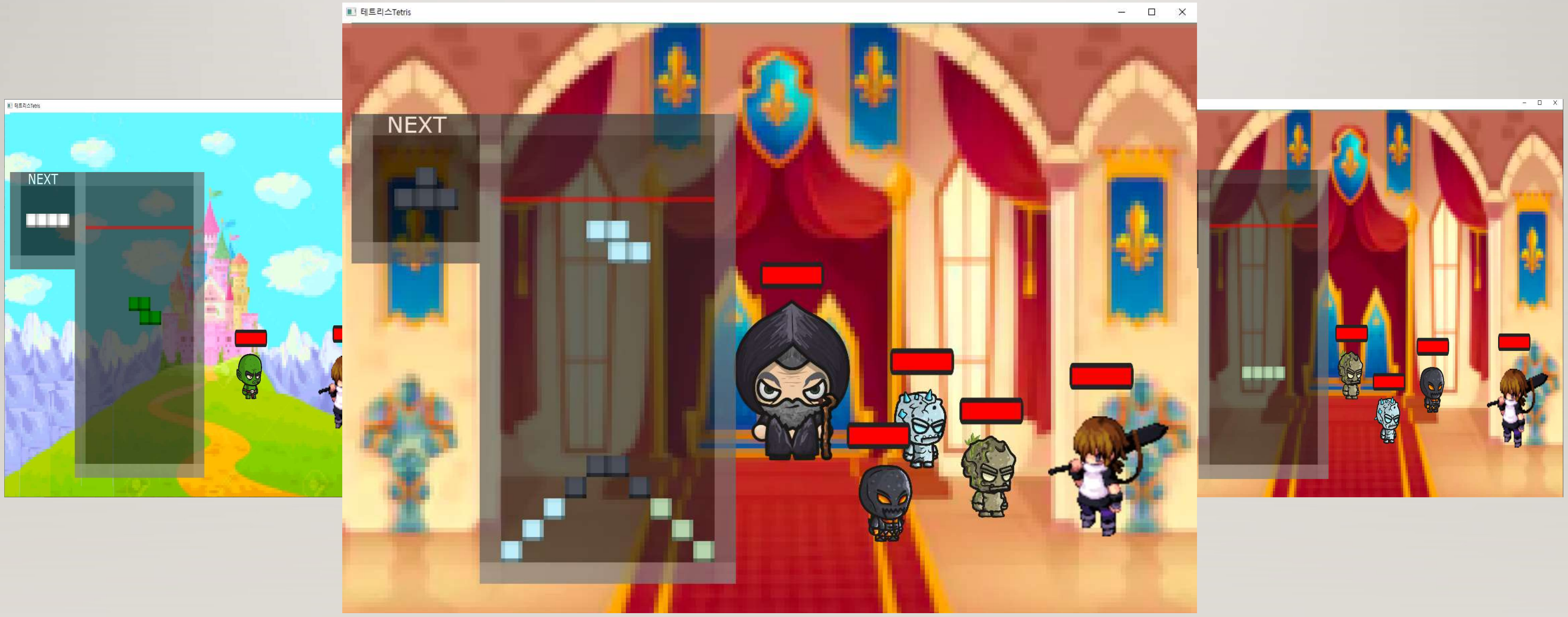


옛날 옛적에 용사와 공주가 있었습니다.

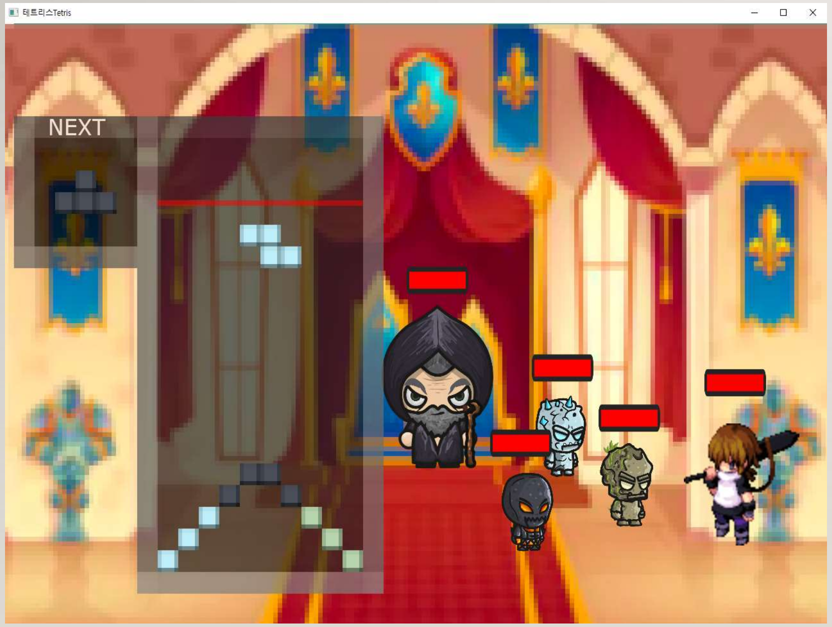
그러던 어느 날 공주가 마왕에게 납치 되었습니다!

그래서 용사는 공주를 찾으러 마왕성으로 떠났습니다.

Stage System



Battle System



Each monster has each HP

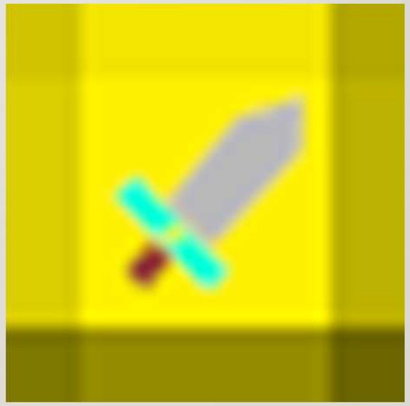
Each monster has unique color too

When we make a line, program checks the block color.

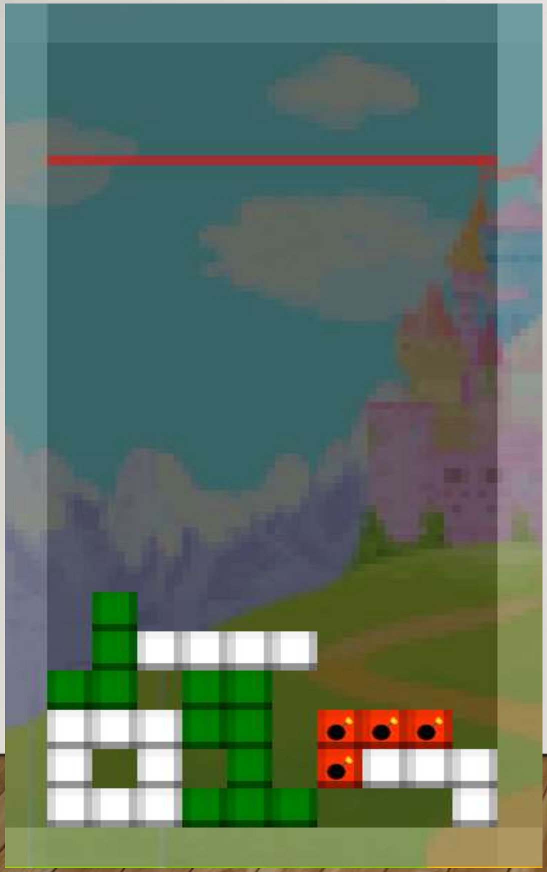
And monster's HP decrease

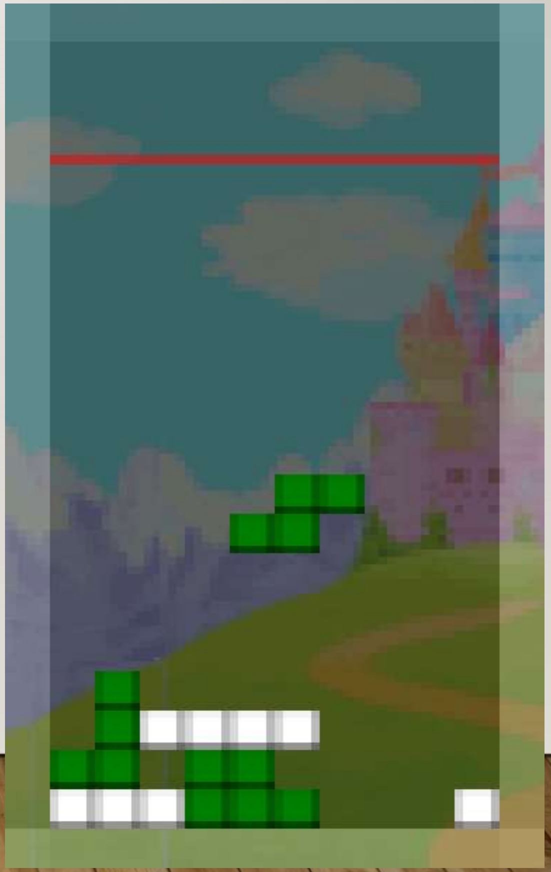
Item

Bomb

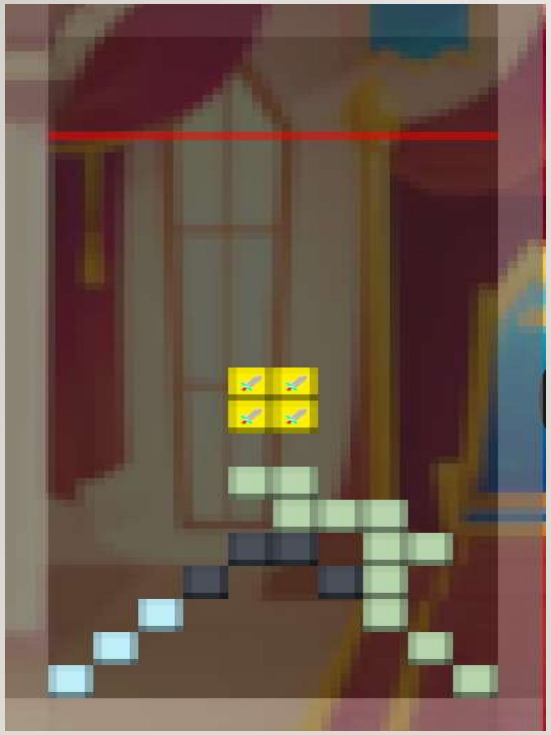

Sword

Bomb Item



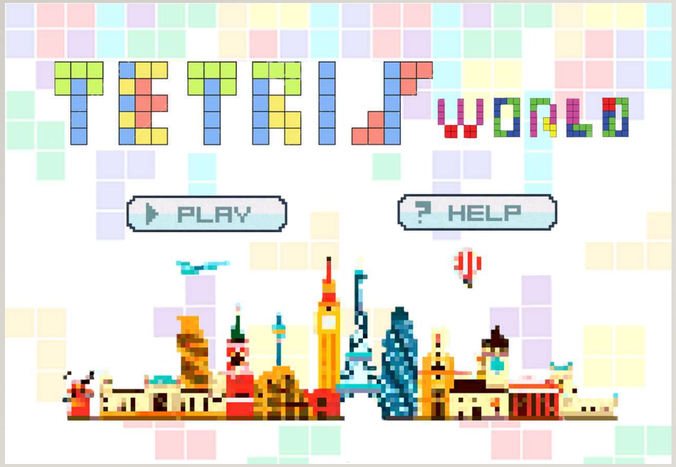


Sword Item

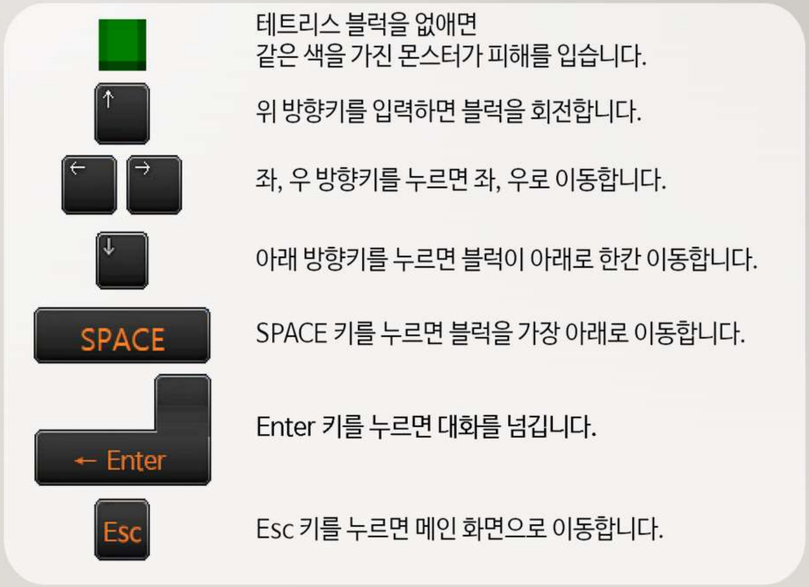


When you make the line,
if the line has that block,
damage will be double.

Manual



CLICK HELP

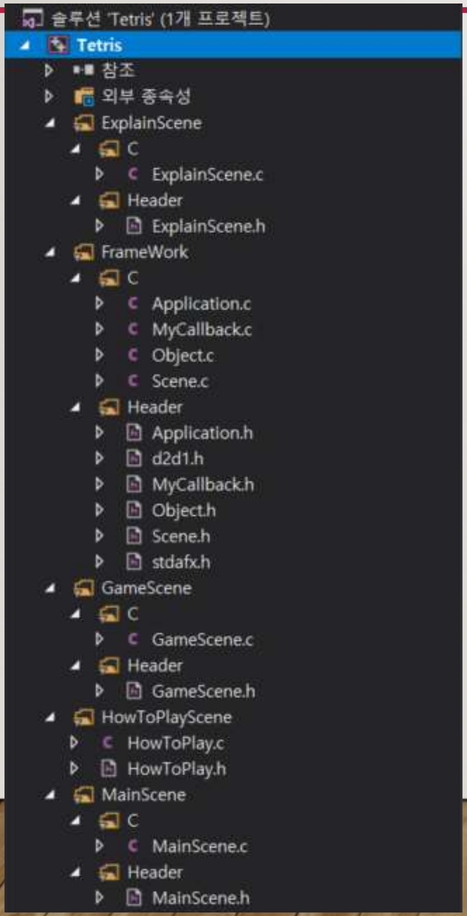


🟩	테트리스 블럭을 없애면 같은 색을 가진 몬스터가 피해를 입습니다.
↑	위 방향키를 입력하면 블럭을 회전합니다.
← →	좌, 우 방향키를 누르면 좌, 우로 이동합니다.
↓	아래 방향키를 누르면 블럭이 아래로 한칸 이동합니다.
SPACE	SPACE 키를 누르면 블럭을 가장 아래로 이동합니다.
← Enter	Enter 키를 누르면 대화를 넘깁니다.
Esc	Esc 키를 누르면 메인 화면으로 이동합니다.

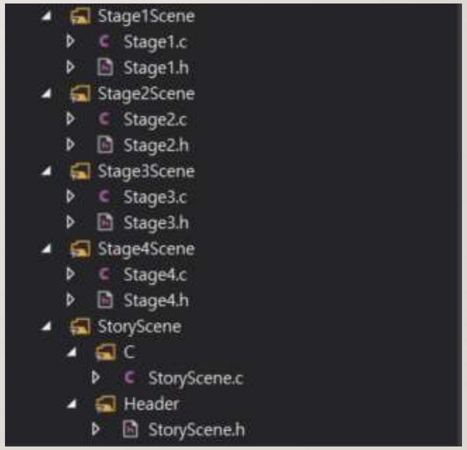
How to Make

Code
Explain

Code

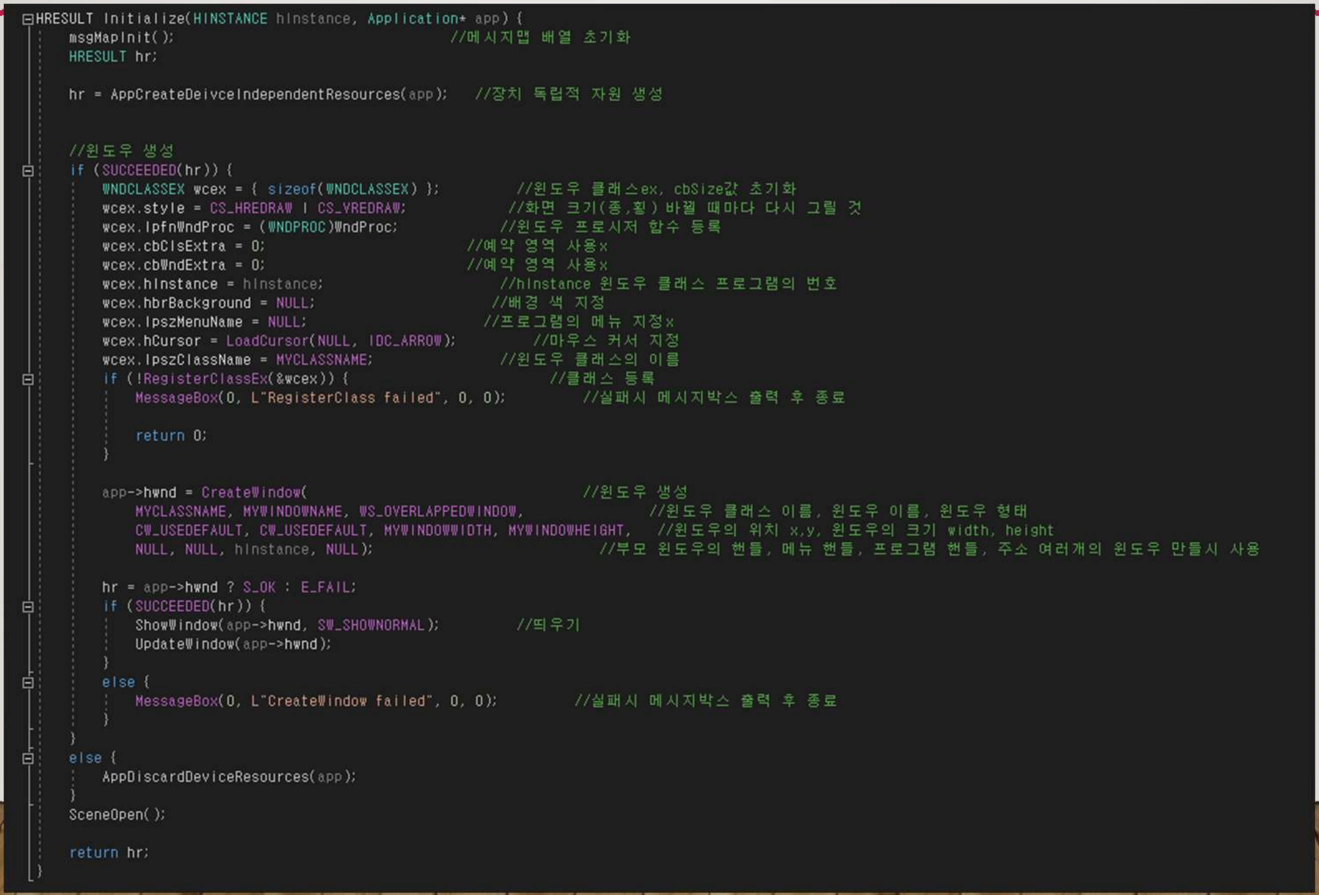


```
솔루션 'Tetris' (1개 프로젝트)
◢ Tetris
  ▷ 참조
  ▷ 외부 종속성
  ◢ ExplainScene
    ◢ C
      ▷ ExplainScene.c
    ◢ Header
      ▷ ExplainScene.h
  ◢ FrameWork
    ◢ C
      ▷ Application.c
      ▷ MyCallback.c
      ▷ Object.c
      ▷ Scene.c
    ◢ Header
      ▷ Application.h
      ▷ d2d1.h
      ▷ MyCallback.h
      ▷ Object.h
      ▷ Scene.h
      ▷ stdafx.h
  ◢ GameScene
    ◢ C
      ▷ GameScene.c
    ◢ Header
      ▷ GameScene.h
  ◢ HowToPlayScene
    ▷ HowToPlay.c
    ▷ HowToPlay.h
  ◢ MainScene
    ◢ C
      ▷ MainScene.c
    ◢ Header
      ▷ MainScene.h
```



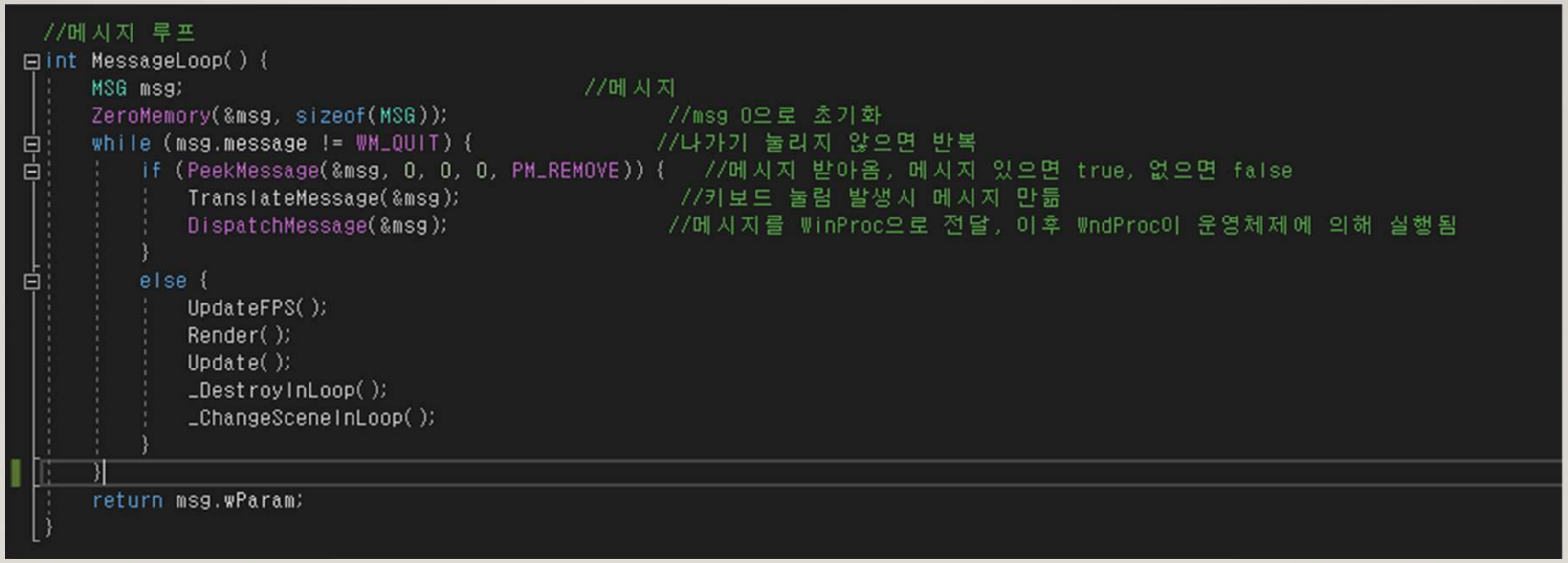
```
◢ Stage1Scene
  ▷ Stage1.c
  ▷ Stage1.h
◢ Stage2Scene
  ▷ Stage2.c
  ▷ Stage2.h
◢ Stage3Scene
  ▷ Stage3.c
  ▷ Stage3.h
◢ Stage4Scene
  ▷ Stage4.c
  ▷ Stage4.h
◢ StoryScene
  ◢ C
    ▷ StoryScene.c
  ◢ Header
    ▷ StoryScene.h
```

WINAPI



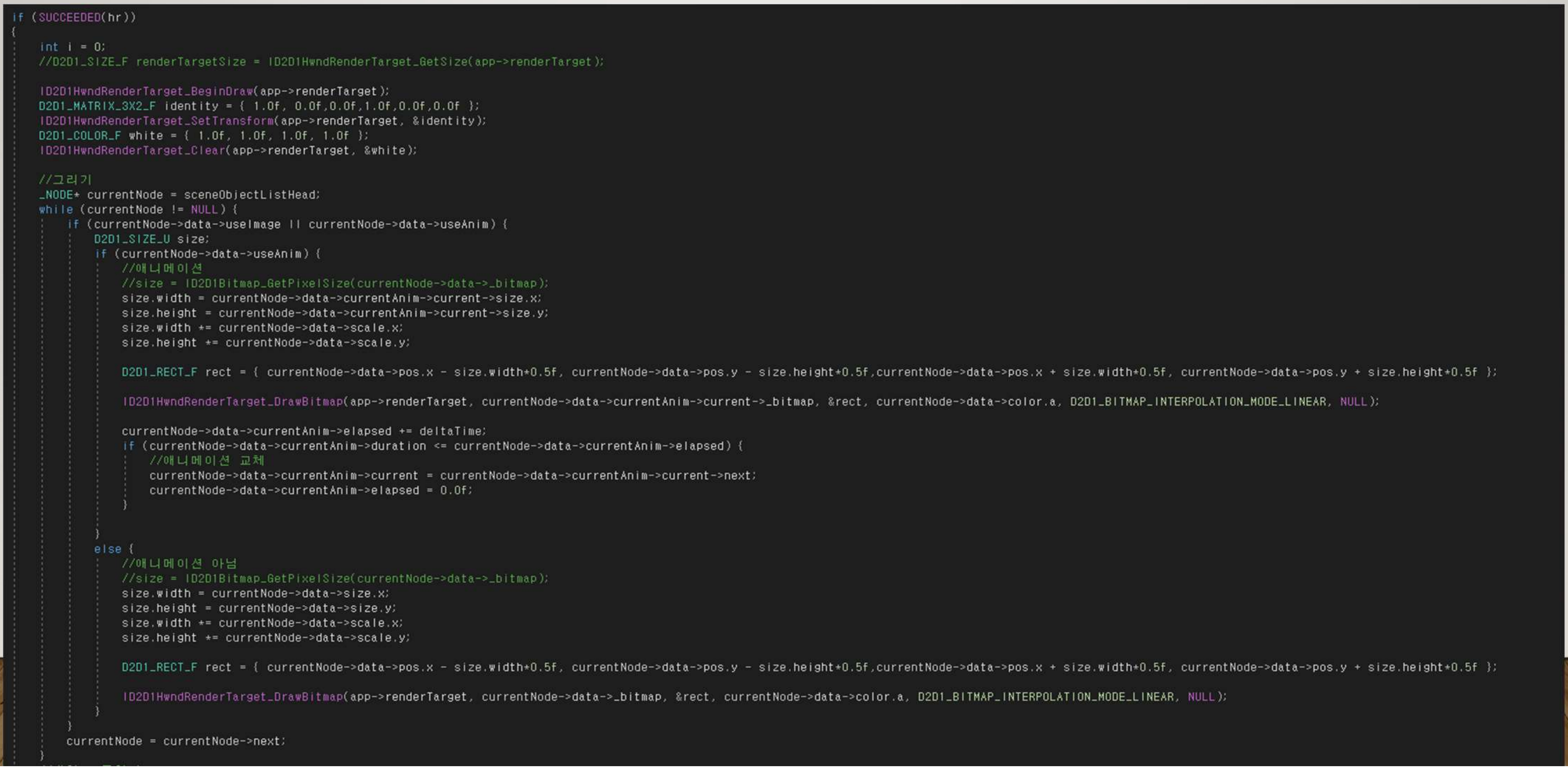
```
HRESULT Initialize(HINSTANCE hInstance, Application* app) {
    msgMapInit();                                           //메시지맵 배열 초기화
    HRESULT hr;

    hr = AppCreateDeivceIndependentResources(app);   //장치 독립적 자원 생성


    //윈도우 생성
    if (SUCCEEDED(hr)) {
        WNDCLASSEX wcex = { sizeof(WNDCLASSEX) };          //윈도우 클래스ex, cbSize값 초기화
        wcex.style = CS_HREDRAW | CS_VREDRAW;             //화면 크기(종,횡) 바뀔 때마다 다시 그릴 것
        wcex.lpfnWndProc = (WNDPROC)WndProc;             //윈도우 프로시저 함수 등록
        wcex.cbClsExtra = 0;                         //예약 영역 사용x
        wcex.cbWndExtra = 0;                         //예약 영역 사용x
        wcex.hInstance = hInstance;                      //hInstance 윈도우 클래스 프로그램의 번호
        wcex.hbrBackground = NULL;                      //배경 색 지정
        wcex.lpszMenuName = NULL;                      //프로그램의 메뉴 지정x
        wcex.hCursor = LoadCursor(NULL, IDC_ARROW);          //마우스 커서 지정
        wcex.lpszClassName = MYCLASSNAME;               //윈도우 클래스의 이름
        if (!RegisterClassEx(&wcex)) {                        //클래스 등록
            MessageBox(0, L"RegisterClass failed", 0, 0);         //실패시 메시지박스 출력 후 종료

            return 0;
        }

        app->hwnd = CreateWindow(                            //윈도우 생성
            MYCLASSNAME, MYWINDOWNAME, WS_OVERLAPPEDWINDOW,              //윈도우 클래스 이름, 윈도우 이름, 윈도우 형태
            CW_USEDEFAULT, CW_USEDEFAULT, MYWINDOWWIDTH, MYWINDOWHEIGHT,   //윈도우의 위치 x,y, 윈도우의 크기 width, height
            NULL, NULL, hInstance, NULL);                               //부모 윈도우의 핸들, 메뉴 핸들, 프로그램 핸들, 주소 여러개의 윈도우 만들시 사용

        hr = app->hwnd ? S_OK : E_FAIL;
        if (SUCCEEDED(hr)) {
            ShowWindow(app->hwnd, SW_SHOWNORMAL);         //띄우기
            UpdateWindow(app->hwnd);
        }
        else {
            MessageBox(0, L"CreateWindow failed", 0, 0);         //실패시 메시지박스 출력 후 종료
        }
    }
    else {
        AppDiscardDeviceResources(app);
    }
    SceneOpen();

    return hr;
}
```

WINAPI



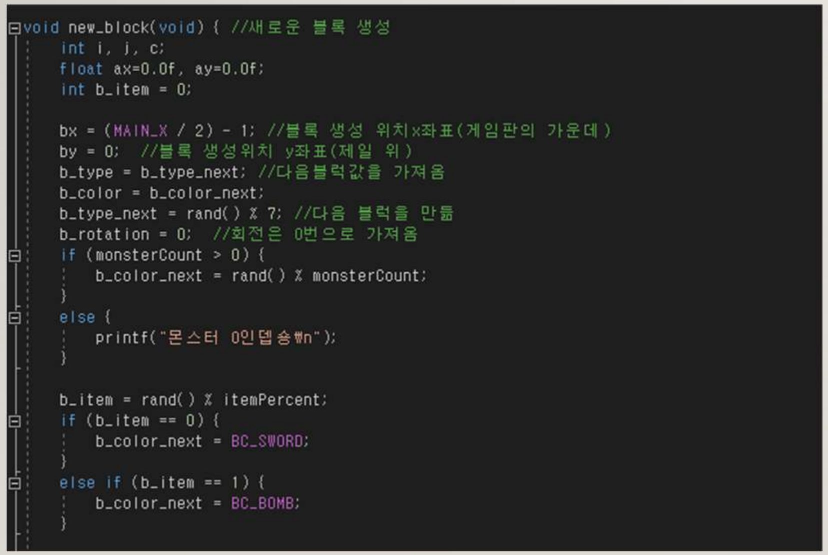
```
//메시지 루프
int MessageLoop( ) {
    MSG msg;                                  //메시지
    ZeroMemory(&msg, sizeof(MSG));                    //msg 0으로 초기화
    while (msg.message != WM_QUIT) {                 //나가기 눌리지 않으면 반복
        if (PeekMessage(&msg, 0, 0, 0, PM_REMOVE)) {   //메시지 받아옴, 메시지 있으면 true, 없으면 false
            TranslateMessage(&msg);                    //키보드 눌림 발생시 메시지 만듦
            DispatchMessage(&msg);                    //메시지를 WinProc으로 전달, 이후 WndProc이 운영체제에 의해 실행됨
        }
        else {
            UpdateFPS( );
            Render( );
            Update( );
            _DestroyInLoop( );
            _ChangeSceneInLoop( );
        }
    }
    return msg.wParam;
}
```

DirectX

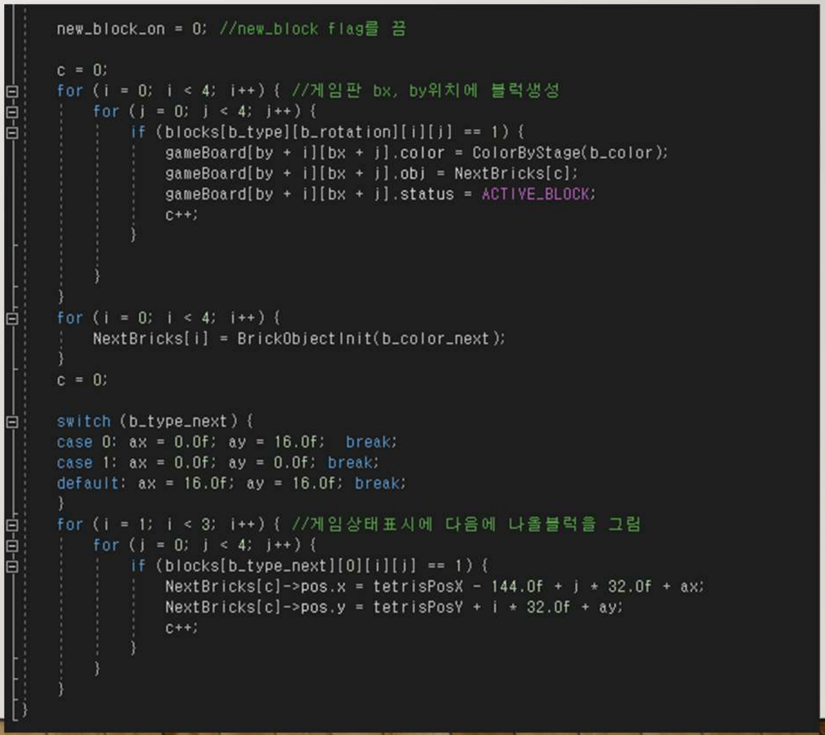


```c
if (SUCCEEDED(hr))
{
    int i = 0;
    //D2D1_SIZE_F renderTargetSize = ID2D1HwndRenderTarget_GetSize(app->renderTarget);

    ID2D1HwndRenderTarget_BeginDraw(app->renderTarget);
    D2D1_MATRIX_3X2_F identity = { 1.0f, 0.0f,0.0f,1.0f,0.0f,0.0f };
    ID2D1HwndRenderTarget_SetTransform(app->renderTarget, &identity);
    D2D1_COLOR_F white = { 1.0f, 1.0f, 1.0f, 1.0f };
    ID2D1HwndRenderTarget_Clear(app->renderTarget, &white);

    //그리기
    _NODE* currentNode = sceneObjectListHead;
    while (currentNode != NULL) {
        if (currentNode->data->useImage || currentNode->data->useAnim) {
            D2D1_SIZE_U size;
            if (currentNode->data->useAnim) {
                //애니메이션
                //size = ID2D1Bitmap_GetPixelSize(currentNode->data->_bitmap);
                size.width = currentNode->data->currentAnim->current->size.x;
                size.height = currentNode->data->currentAnim->current->size.y;
                size.width += currentNode->data->scale.x;
                size.height += currentNode->data->scale.y;

                D2D1_RECT_F rect = { currentNode->data->pos.x - size.width*0.5f, currentNode->data->pos.y - size.height*0.5f,currentNode->data->pos.x + size.width*0.5f, currentNode->data->pos.y + size.height*0.5f };

                ID2D1HwndRenderTarget_DrawBitmap(app->renderTarget, currentNode->data->currentAnim->current->_bitmap, &rect, currentNode->data->color.a, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR, NULL);

                currentNode->data->currentAnim->elapsed += deltaTime;
                if (currentNode->data->currentAnim->duration <= currentNode->data->currentAnim->elapsed) {
                    //애니메이션 교체
                    currentNode->data->currentAnim->current = currentNode->data->currentAnim->current->next;
                    currentNode->data->currentAnim->elapsed = 0.0f;
                }

            }
            else {
                //애니메이션 아님
                //size = ID2D1Bitmap_GetPixelSize(currentNode->data->_bitmap);
                size.width = currentNode->data->size.x;
                size.height = currentNode->data->size.y;
                size.width += currentNode->data->scale.x;
                size.height += currentNode->data->scale.y;

                D2D1_RECT_F rect = { currentNode->data->pos.x - size.width*0.5f, currentNode->data->pos.y - size.height*0.5f,currentNode->data->pos.x + size.width*0.5f, currentNode->data->pos.y + size.height*0.5f };

                ID2D1HwndRenderTarget_DrawBitmap(app->renderTarget, currentNode->data->_bitmap, &rect, currentNode->data->color.a, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR, NULL);
            }
        }
        currentNode = currentNode->next;
    }
```

new_block

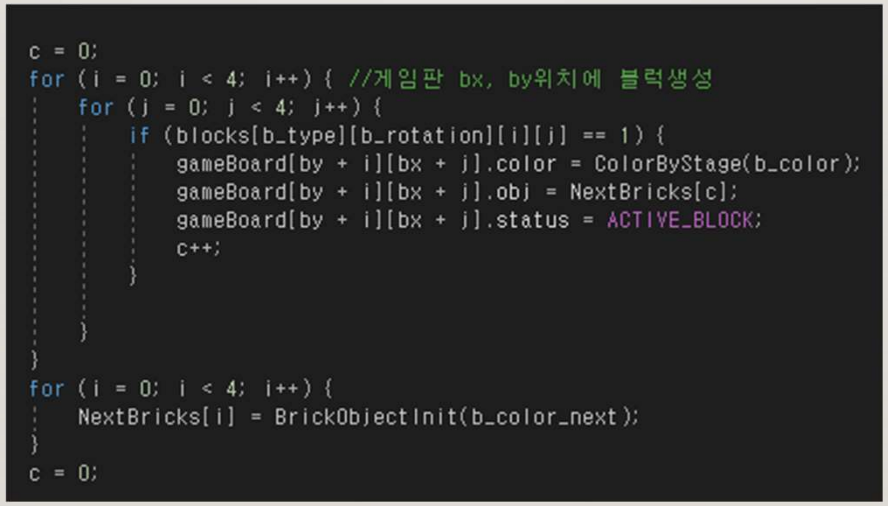


```c
void new_block(void) { //새로운 블록 생성
    int i, j, c;
    float ax=0.0f, ay=0.0f;
    int b_item = 0;

    bx = (MAIN_X / 2) - 1; //블록 생성 위치x좌표(게임판의 가운데)
    by = 0;  //블록 생성위치 y좌표(제일 위)
    b_type = b_type_next; //다음블럭값을 가져옴
    b_color = b_color_next;
    b_type_next = rand() % 7; //다음 블럭을 만듦
    b_rotation = 0;  //회전은 0번으로 가져옴
    if (monsterCount > 0) {
        b_color_next = rand() % monsterCount;
    }
    else {
        printf("몬스터 0인뎁송₩n");
    }

    b_item = rand() % itemPercent;
    if (b_item == 0) {
        b_color_next = BC_SWORD;
    }
    else if (b_item == 1) {
        b_color_next = BC_BOMB;
    }
```



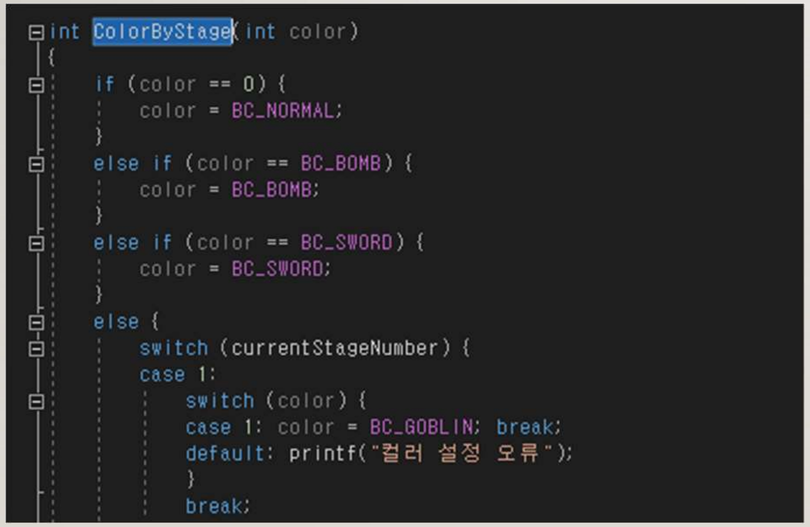
```c
    new_block_on = 0; //new_block flag를 끔

    c = 0;
    for (i = 0; i < 4; i++) { //게임판 bx, by위치에 블럭생성
        for (j = 0; j < 4; j++) {
            if (blocks[b_type][b_rotation][i][j] == 1) {
                gameBoard[by + i][bx + j].color = ColorByStage(b_color);
                gameBoard[by + i][bx + j].obj = NextBricks[c];
                gameBoard[by + i][bx + j].status = ACTIVE_BLOCK;
                c++;
            }


        }
    }
    for (i = 0; i < 4; i++) {
        NextBricks[i] = BrickObjectInit(b_color_next);
    }
    c = 0;

    switch (b_type_next) {
    case 0: ax = 0.0f; ay = 16.0f;  break;
    case 1: ax = 0.0f; ay = 0.0f; break;
    default: ax = 16.0f; ay = 16.0f; break;
    }
    for (i = 1; i < 3; i++) { //게임상태표시에 다음에 나올블럭을 그림
        for (j = 0; j < 4; j++) {
            if (blocks[b_type_next][0][i][j] == 1) {
                NextBricks[c]->pos.x = tetrisPosX - 144.0f + j * 32.0f + ax;
                NextBricks[c]->pos.y = tetrisPosY + i * 32.0f + ay;
                c++;
            }
        }
    }
}
```

new_block

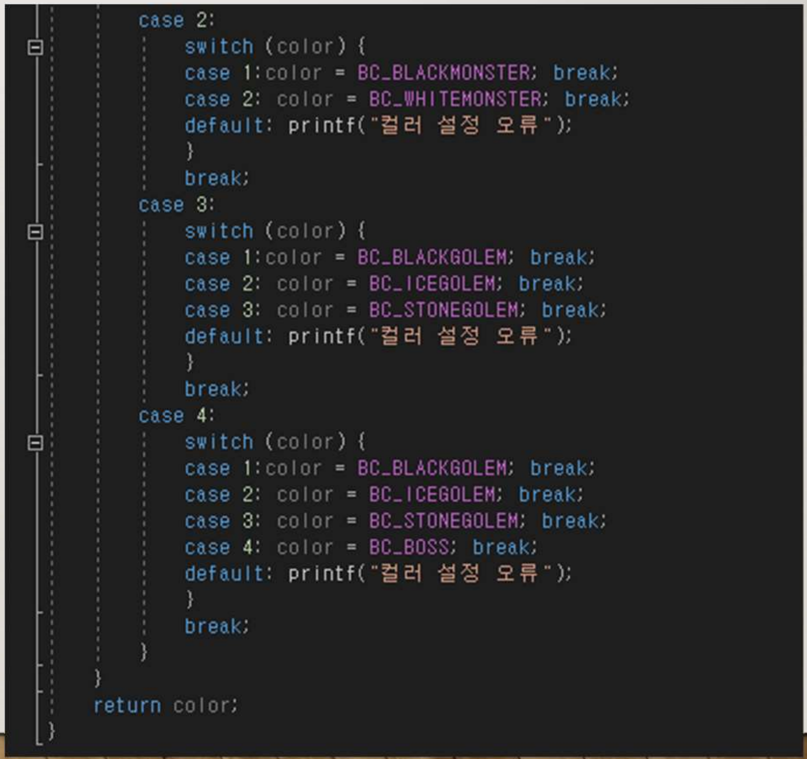


```
c = 0;
for (i = 0; i < 4; i++) { //게임판 bx, by위치에 블럭생성
    for (j = 0; j < 4; j++) {
        if (blocks[b_type][b_rotation][i][j] == 1) {
            gameBoard[by + i][bx + j].color = ColorByStage(b_color);
            gameBoard[by + i][bx + j].obj = NextBricks[c];
            gameBoard[by + i][bx + j].status = ACTIVE_BLOCK;
            c++;
        }


    }
}
for (i = 0; i < 4; i++) {
    NextBricks[i] = BrickObjectInit(b_color_next);
}
c = 0;
```

new_block

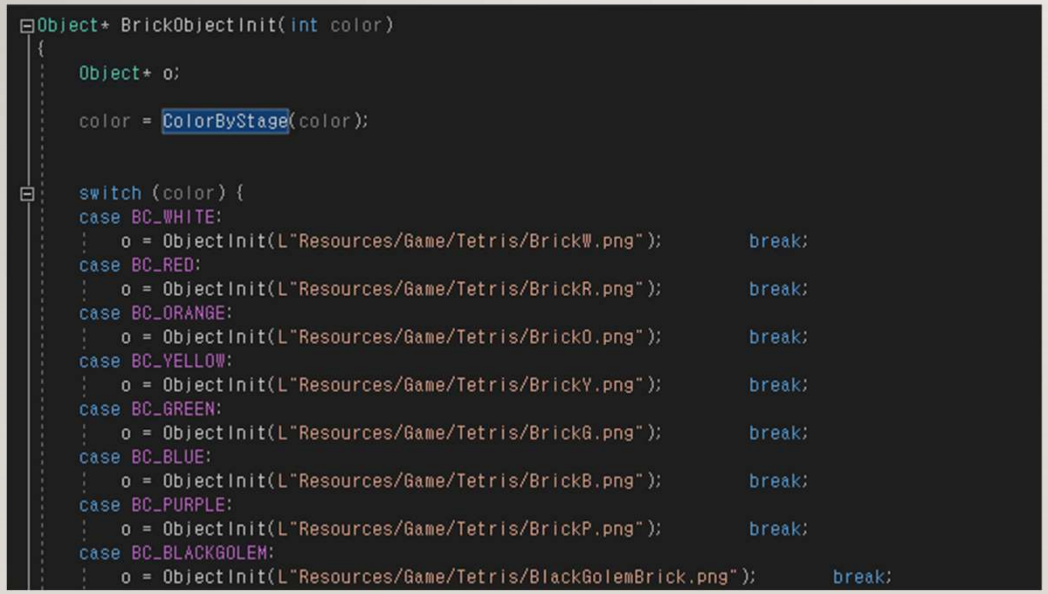


```
int ColorByStage(int color)
{
    if (color == 0) {
        color = BC_NORMAL;
    }
    else if (color == BC_BOMB) {
        color = BC_BOMB;
    }
    else if (color == BC_SWORD) {
        color = BC_SWORD;
    }
    else {
        switch (currentStageNumber) {
        case 1:
            switch (color) {
            case 1: color = BC_GOBLIN; break;
            default: printf("컬러 설정 오류");
            }
            break;
```

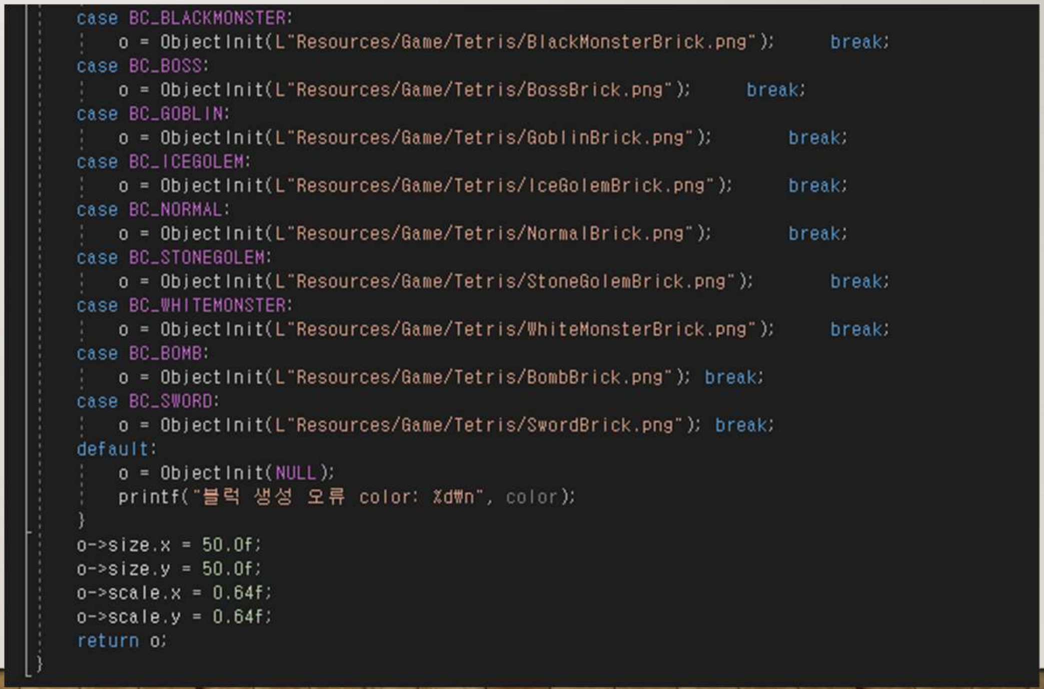


```
        case 2:
            switch (color) {
            case 1:color = BC_BLACKMONSTER; break;
            case 2: color = BC_WHITEMONSTER; break;
            default: printf("컬러 설정 오류");
            }
            break;
        case 3:
            switch (color) {
            case 1:color = BC_BLACKGOLEM; break;
            case 2: color = BC_ICEGOLEM; break;
            case 3: color = BC_STONEGOLEM; break;
            default: printf("컬러 설정 오류");
            }
            break;
        case 4:
            switch (color) {
            case 1:color = BC_BLACKGOLEM; break;
            case 2: color = BC_ICEGOLEM; break;
            case 3: color = BC_STONEGOLEM; break;
            case 4: color = BC_BOSS; break;
            default: printf("컬러 설정 오류");
            }
            break;
        }
    }
    return color;
}
```

new_block

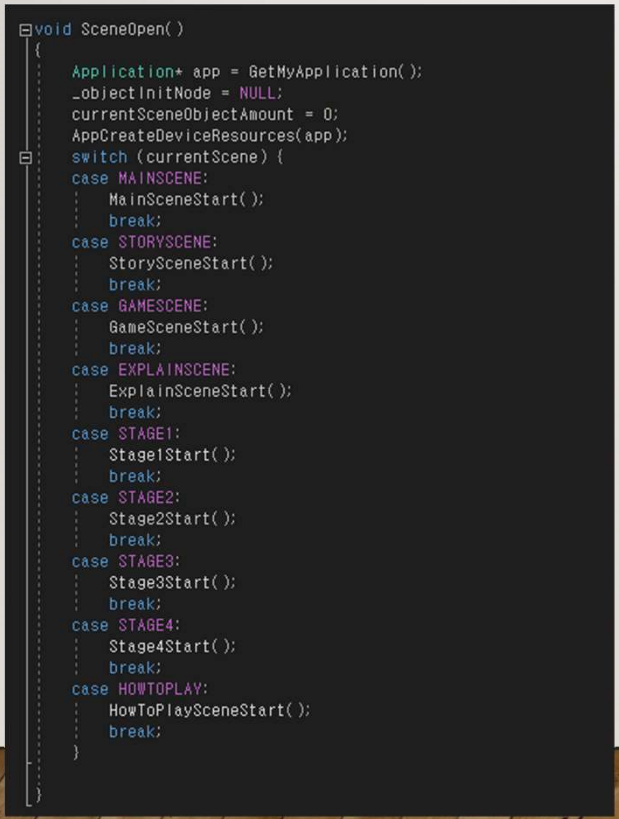


```
Object* BrickObjectInit(int color)
{
    Object* o;

    color = ColorByStage(color);


    switch (color) {
    case BC_WHITE:
        o = ObjectInit(L"Resources/Game/Tetris/BrickW.png");        break;
    case BC_RED:
        o = ObjectInit(L"Resources/Game/Tetris/BrickR.png");        break;
    case BC_ORANGE:
        o = ObjectInit(L"Resources/Game/Tetris/BrickO.png");        break;
    case BC_YELLOW:
        o = ObjectInit(L"Resources/Game/Tetris/BrickY.png");        break;
    case BC_GREEN:
        o = ObjectInit(L"Resources/Game/Tetris/BrickG.png");        break;
    case BC_BLUE:
        o = ObjectInit(L"Resources/Game/Tetris/BrickB.png");        break;
    case BC_PURPLE:
        o = ObjectInit(L"Resources/Game/Tetris/BrickP.png");        break;
    case BC_BLACKGOLEM:
        o = ObjectInit(L"Resources/Game/Tetris/BlackGolemBrick.png");        break;
```

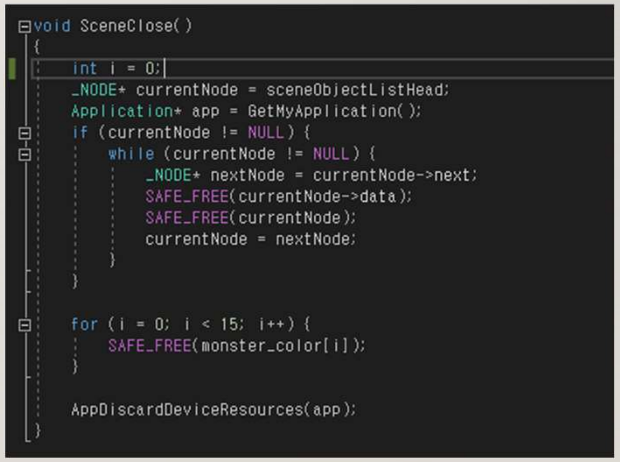


```
    case BC_BLACKMONSTER:
        o = ObjectInit(L"Resources/Game/Tetris/BlackMonsterBrick.png");     break;
    case BC_BOSS:
        o = ObjectInit(L"Resources/Game/Tetris/BossBrick.png");     break;
    case BC_GOBLIN:
        o = ObjectInit(L"Resources/Game/Tetris/GoblinBrick.png");       break;
    case BC_ICEGOLEM:
        o = ObjectInit(L"Resources/Game/Tetris/IceGolemBrick.png");     break;
    case BC_NORMAL:
        o = ObjectInit(L"Resources/Game/Tetris/NormalBrick.png");       break;
    case BC_STONEGOLEM:
        o = ObjectInit(L"Resources/Game/Tetris/StoneGolemBrick.png");       break;
    case BC_WHITEMONSTER:
        o = ObjectInit(L"Resources/Game/Tetris/WhiteMonsterBrick.png");     break;
    case BC_BOMB:
        o = ObjectInit(L"Resources/Game/Tetris/BombBrick.png"); break;
    case BC_SWORD:
        o = ObjectInit(L"Resources/Game/Tetris/SwordBrick.png"); break;
    default:
        o = ObjectInit(NULL);
        printf("블럭 생성 오류 color: %d\n", color);
    }
    o->size.x = 50.0f;
    o->size.y = 50.0f;
    o->scale.x = 0.64f;
    o->scale.y = 0.64f;
    return o;
}
```

Scene

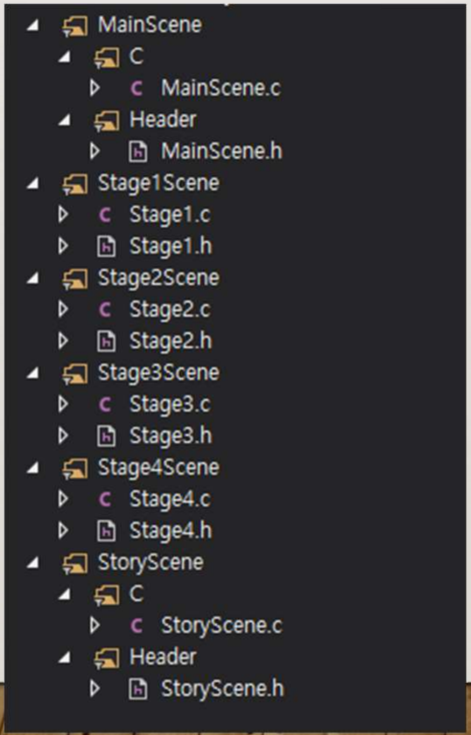


```
void SceneOpen()
{
    Application* app = GetMyApplication();
    _objectInitNode = NULL;
    currentSceneObjectAmount = 0;
    AppCreateDeviceResources(app);
    switch (currentScene) {
    case MAINSCENE:
        MainSceneStart();
        break;
    case STORYSCENE:
        StorySceneStart();
        break;
    case GAMESCENE:
        GameSceneStart();
        break;
    case EXPLAINSCENE:
        ExplainSceneStart();
        break;
    case STAGE1:
        Stage1Start();
        break;
    case STAGE2:
        Stage2Start();
        break;
    case STAGE3:
        Stage3Start();
        break;
    case STAGE4:
        Stage4Start();
        break;
    case HOWTOPLAY:
        HowToPlaySceneStart();
        break;
    }

}
```



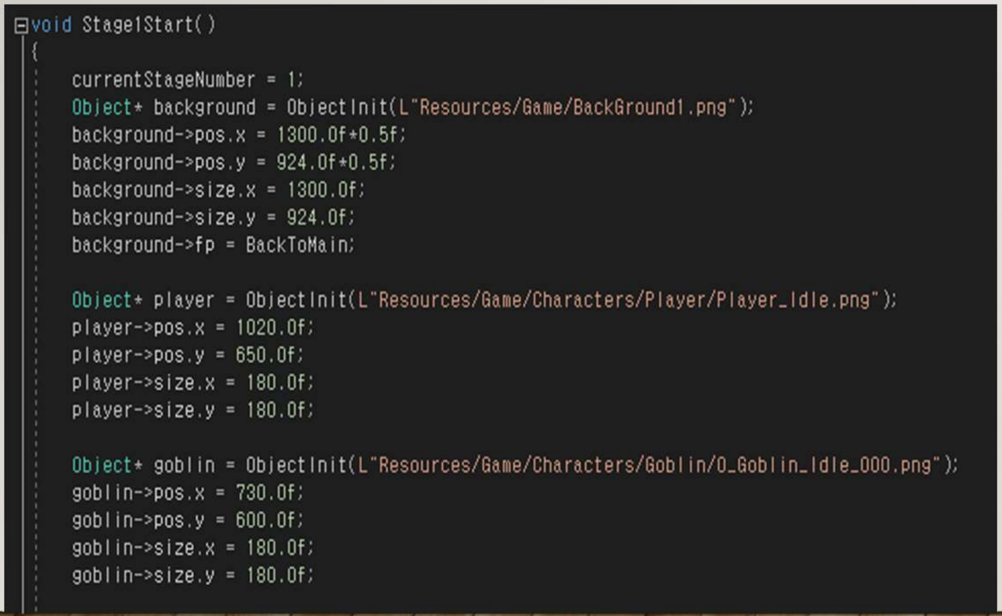
```
void SceneClose()
{
    int i = 0;
    _NODE* currentNode = sceneObjectListHead;
    Application* app = GetMyApplication();
    if (currentNode != NULL) {
        while (currentNode != NULL) {
            _NODE* nextNode = currentNode->next;
            SAFE_FREE(currentNode->data);
            SAFE_FREE(currentNode);
            currentNode = nextNode;
        }
    }

    for (i = 0; i < 15; i++) {
        SAFE_FREE(monster_color[i]);
    }

    AppDiscardDeviceResources(app);
}
```

Scenes

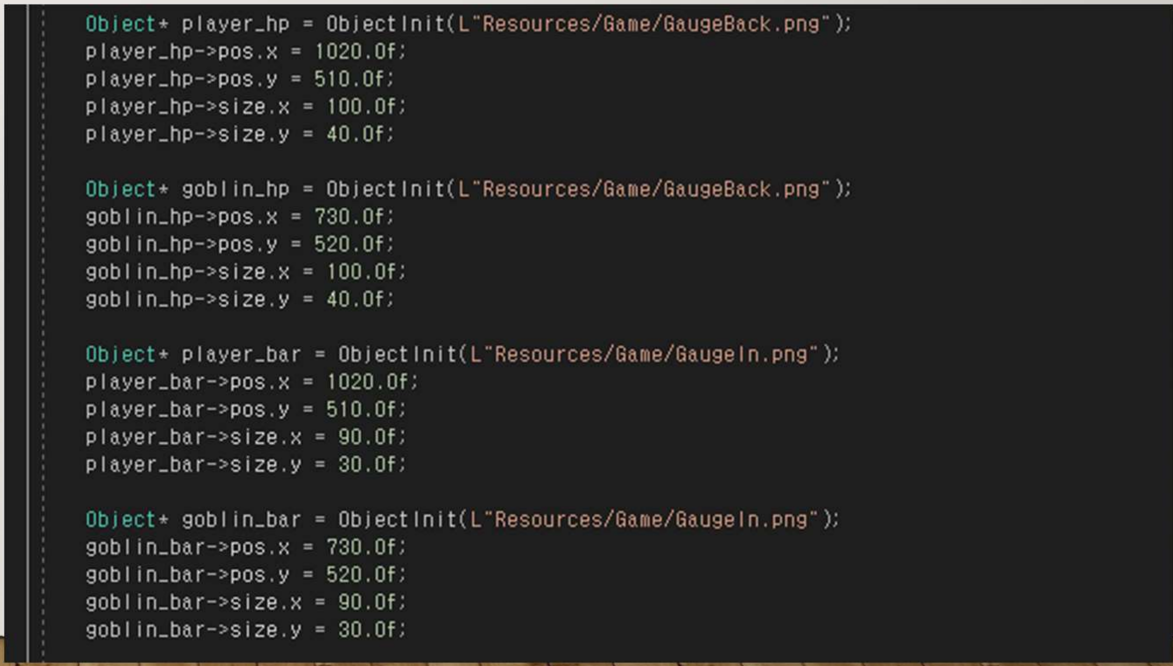


```
◢ MainScene
  ◢ C
    ▷ MainScene.c
  ◢ Header
    ▷ MainScene.h
◢ Stage1Scene
  ▷ Stage1.c
  ▷ Stage1.h
◢ Stage2Scene
  ▷ Stage2.c
  ▷ Stage2.h
◢ Stage3Scene
  ▷ Stage3.c
  ▷ Stage3.h
◢ Stage4Scene
  ▷ Stage4.c
  ▷ Stage4.h
◢ StoryScene
  ◢ C
    ▷ StoryScene.c
  ◢ Header
    ▷ StoryScene.h
```

Stage

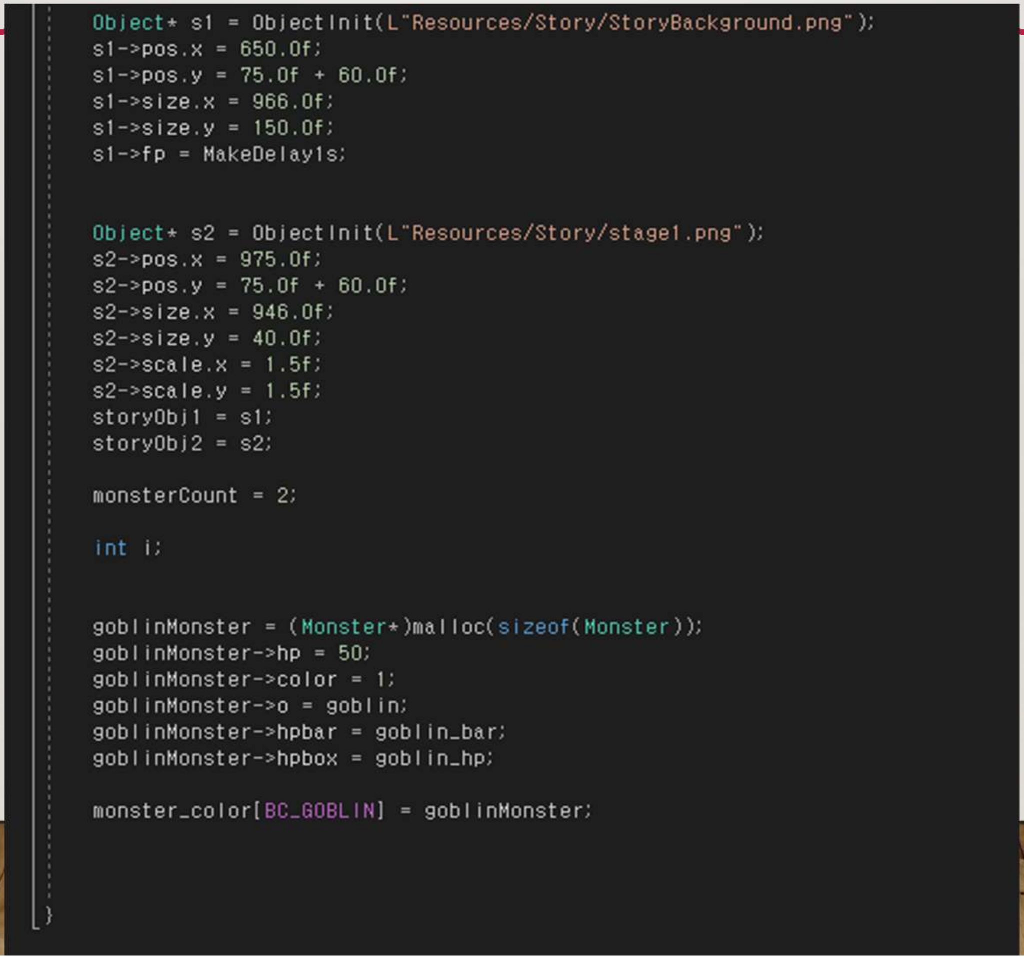


```
void Stage1Start()
{
    currentStageNumber = 1;
    Object* background = ObjectInit(L"Resources/Game/BackGround1.png");
    background->pos.x = 1300.0f*0.5f;
    background->pos.y = 924.0f*0.5f;
    background->size.x = 1300.0f;
    background->size.y = 924.0f;
    background->fp = BackToMain;

    Object* player = ObjectInit(L"Resources/Game/Characters/Player/Player_Idle.png");
    player->pos.x = 1020.0f;
    player->pos.y = 650.0f;
    player->size.x = 180.0f;
    player->size.y = 180.0f;

    Object* goblin = ObjectInit(L"Resources/Game/Characters/Goblin/0_Goblin_Idle_000.png");
    goblin->pos.x = 730.0f;
    goblin->pos.y = 600.0f;
    goblin->size.x = 180.0f;
    goblin->size.y = 180.0f;
```



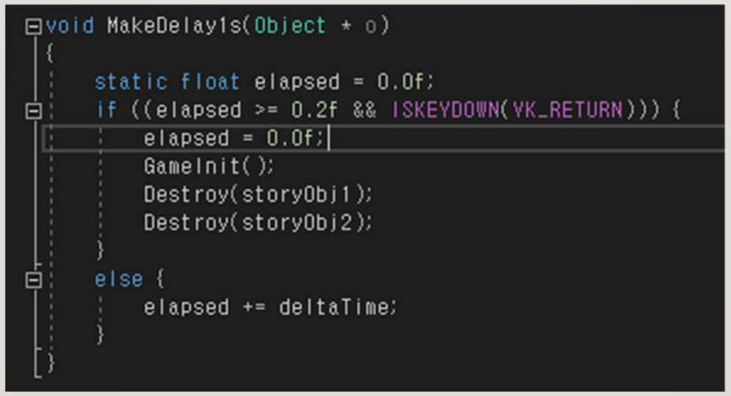
```
    Object* player_hp = ObjectInit(L"Resources/Game/GaugeBack.png");
    player_hp->pos.x = 1020.0f;
    player_hp->pos.y = 510.0f;
    player_hp->size.x = 100.0f;
    player_hp->size.y = 40.0f;

    Object* goblin_hp = ObjectInit(L"Resources/Game/GaugeBack.png");
    goblin_hp->pos.x = 730.0f;
    goblin_hp->pos.y = 520.0f;
    goblin_hp->size.x = 100.0f;
    goblin_hp->size.y = 40.0f;

    Object* player_bar = ObjectInit(L"Resources/Game/GaugeIn.png");
    player_bar->pos.x = 1020.0f;
    player_bar->pos.y = 510.0f;
    player_bar->size.x = 90.0f;
    player_bar->size.y = 30.0f;

    Object* goblin_bar = ObjectInit(L"Resources/Game/GaugeIn.png");
    goblin_bar->pos.x = 730.0f;
    goblin_bar->pos.y = 520.0f;
    goblin_bar->size.x = 90.0f;
    goblin_bar->size.y = 30.0f;
```

Stage



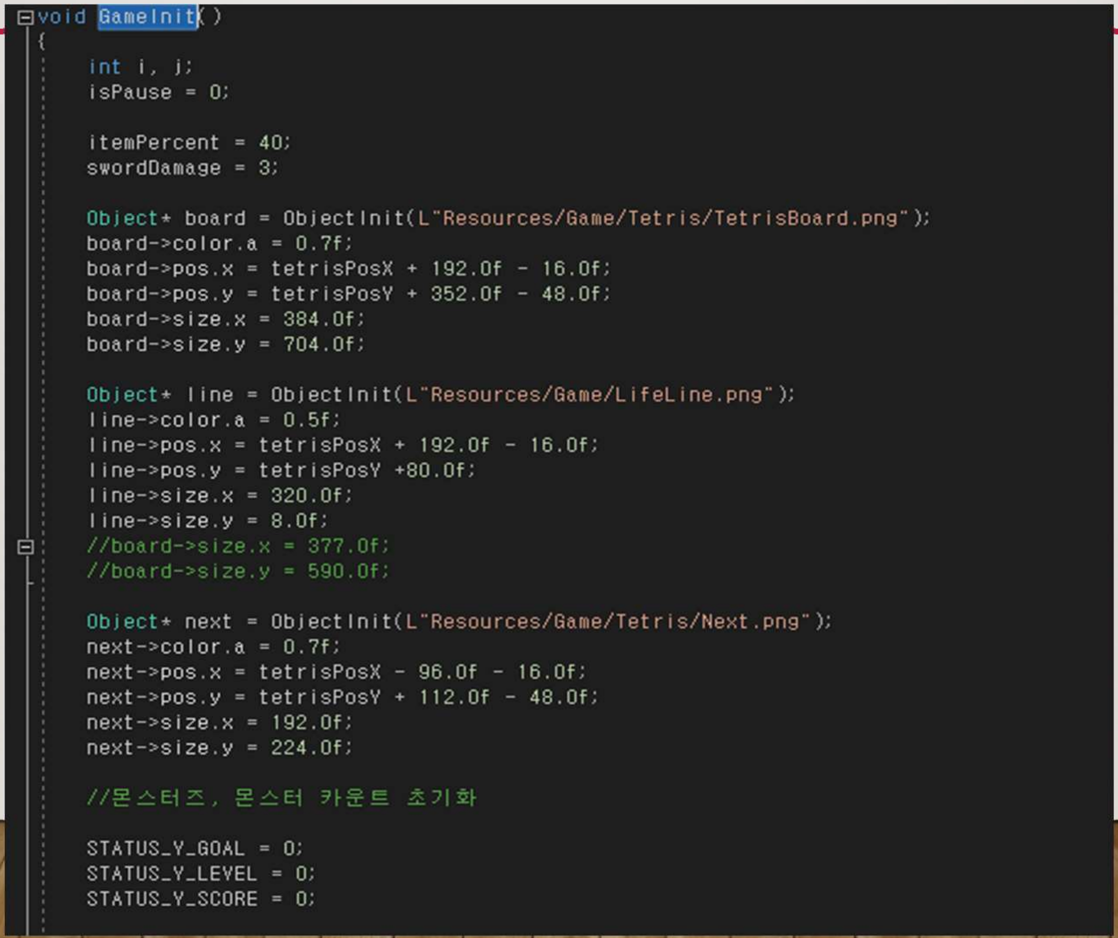
```
    Object* s1 = ObjectInit(L"Resources/Story/StoryBackground.png");
    s1->pos.x = 650.0f;
    s1->pos.y = 75.0f + 60.0f;
    s1->size.x = 966.0f;
    s1->size.y = 150.0f;
    s1->fp = MakeDelay1s;


    Object* s2 = ObjectInit(L"Resources/Story/stage1.png");
    s2->pos.x = 975.0f;
    s2->pos.y = 75.0f + 60.0f;
    s2->size.x = 946.0f;
    s2->size.y = 40.0f;
    s2->scale.x = 1.5f;
    s2->scale.y = 1.5f;
    storyObj1 = s1;
    storyObj2 = s2;

    monsterCount = 2;

    int i;


    goblinMonster = (Monster*)malloc(sizeof(Monster));
    goblinMonster->hp = 50;
    goblinMonster->color = 1;
    goblinMonster->o = goblin;
    goblinMonster->hpbar = goblin_bar;
    goblinMonster->hpbox = goblin_hp;

    monster_color[BC_GOBLIN] = goblinMonster;




}
```

Stage



```
void MakeDelay1s(Object * o)
{
    static float elapsed = 0.0f;
    if ((elapsed >= 0.2f && ISKEYDOWN(VK_RETURN))) {
        elapsed = 0.0f;
        GameInit();
        Destroy(storyObj1);
        Destroy(storyObj2);
    }
    else {
        elapsed += deltaTime;
    }
}
```

Stage



```
void GameInit( )
{
    int i, j;
    isPause = 0;

    itemPercent = 40;
    swordDamage = 3;

    Object* board = ObjectInit(L"Resources/Game/Tetris/TetrisBoard.png");
    board->color.a = 0.7f;
    board->pos.x = tetrisPosX + 192.0f - 16.0f;
    board->pos.y = tetrisPosY + 352.0f - 48.0f;
    board->size.x = 384.0f;
    board->size.y = 704.0f;

    Object* line = ObjectInit(L"Resources/Game/LifeLine.png");
    line->color.a = 0.5f;
    line->pos.x = tetrisPosX + 192.0f - 16.0f;
    line->pos.y = tetrisPosY +80.0f;
    line->size.x = 320.0f;
    line->size.y = 8.0f;
    //board->size.x = 377.0f;
    //board->size.y = 590.0f;

    Object* next = ObjectInit(L"Resources/Game/Tetris/Next.png");
    next->color.a = 0.7f;
    next->pos.x = tetrisPosX - 96.0f - 16.0f;
    next->pos.y = tetrisPosY + 112.0f - 48.0f;
    next->size.x = 192.0f;
    next->size.y = 224.0f;

    //몬스터즈, 몬스터 카운트 초기화

    STATUS_Y_GOAL = 0;
    STATUS_Y_LEVEL = 0;
    STATUS_Y_SCORE = 0;
```

Thank you
For listening