REPORT

광운대학교
KwangWoon University

담당교수: 신영주 교수님

학과: 컴퓨터정보공학부

학번: 2019202009

이름: 서여지

Introduction

❖ 문제 설명

3번과제에서 구현했던 2048게임을 MFC를 이용하여 제작하는 프로젝트이다. 프로그램은 2048게임의 각 블록을 총 16개의 사각형으로 출력하고, 블록 내부에 숫자를 출력한다.

2048의 기본 규칙은 다음과 같다.

1. 16개의 블록 중 무작위로 정해진 한 칸에 수가 추가된다. 문제에서는 새로 생성되는 수가 2로 고정되어있다.
2. 블록을 상, 하, 좌, 우 네 방향 중 한 쪽으로 밀어서 같은 수가 쓰인 블록이 이웃해서 밀어졌을 때 두 수가 더해진다.
3. 수를 더해서 2048을 완성하면 게임에서 승리한다.
4. 2048을 완성하지 못한 채로 16개의 칸이 모두 채워지고, 더 이상 수를 더할 수 없는 경우 게임에서 패배한다.

이 문제에서 프로그램은 사용자에게 두 종류의 입력을 받는다.

1. 키보드의 방향키를 입력한 경우: 해당하는 방향으로 블록을 민다.
2. 판 위에서 마우스를 좌우로 드래그 한 경우: 시계방향, 혹은 시계 반대방향으로 판을 회전시킨다.

프로그램의 메뉴창에는 두 개의 메뉴가 생성되어있다.

1. 출력메뉴: 블록에 숫자만을 출력할 것인지, 숫자와 색을 모두 출력할 것인지 정한다.
2. 기록메뉴: 사용자의 입력과 그에 따른 결과값을 저장한다.

판을 구성하는 16개의 블록은 각각 상, 하, 좌, 우에 이웃한 블록의 주소값을 저장하는 포인터변수를 가지고 있고, 해당 방향에 블록이 없는 경우 nullptr값을 갖는다. 각 블록의 크기는 64*64로 제시되었다. 판은 왼쪽 가장 위 블록의 주소값을 저장하는 포인터변수를 가진다.
출력메뉴에 의해 출력되는 블록의 색은 블록에 저장된 수에 따라 다르고, 문제에서 제시되었다.
기록메뉴에 의해 저장되는 정보는 log.txt 파일에 저장된다. 게임이 종료되었을 때 메시지박스로 게임을 계속할 것인지, 종료할 것인지 입력 받는다. 이때 승리한 경우와 패배한 경우에 출력되는 메시지박스의 종류는 다르지만 기능은 같다. 계속하기를 선택하는 경우 게임을 처음부터 실행시키고, 종료를 선택한 경우 프로그램을 종료한다.

❖ 작성한 프로그램 설명

작성한 프로그램에서 각각의 블록은 3번과제에서의 요구사항과 일치하는 (x,y) 좌표값을 가지고 있다. 가장 왼쪽의 x좌표가 0, 가장 오른쪽의 x좌표는 3이고, 가장 위쪽의 y좌표가 0, 가장 아래쪽의 y좌표는 3이다. 이 좌표값은 사용자에게 출력되지 않고, 특정 위치에 존재하는 블록의 내용을 이용할 때 사용되었다. 2048 게임을 진행하는 부분은 3-5과제를 일부 수정하여 작성하였다.

각각의 블록을 의미하는 MyBlock 클래스에는 블록의 수와 x,y 좌표를 저장할 정수형 변수가 모두 3개 선언되어있고, 이웃한 상, 하, 좌, 우 네 방향의 블록에 대한 주소값을 저장할 MyBlock형 포인터변수가 4개 선언되어있다. 클래스의 public영역에는 private영역의 내용을 반환하거나 값을 수정하는 함수가 선언되어있다.

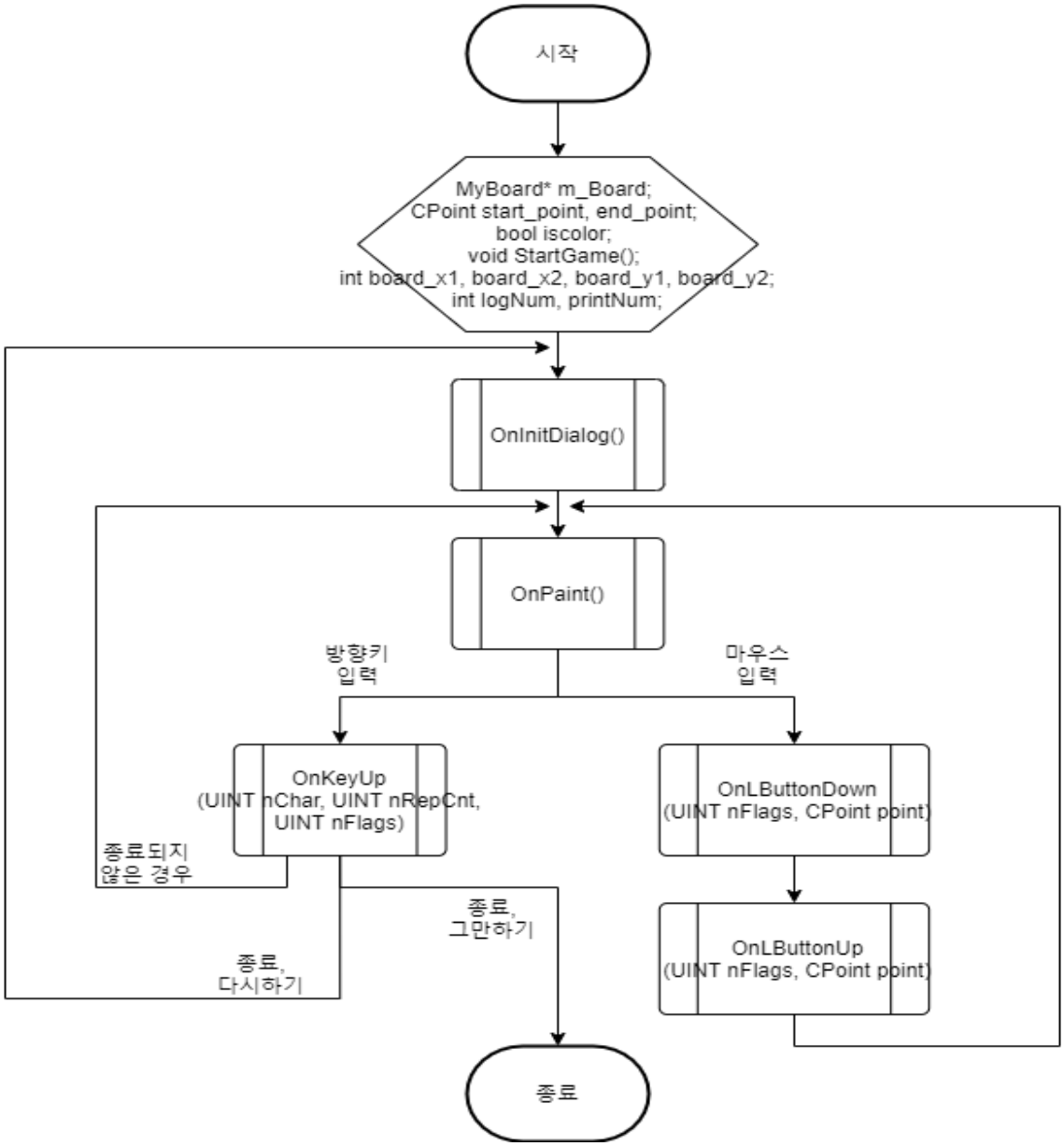
MyBoard 클래스는 판의 (0,0)좌표를 갖는 가장 왼쪽의 가장 위 칸의 블록 주소를 저장할 MyBlock형 포인터 변수 m_pHead를 갖는다. 이 포인터변수는 3-5문제에서는 수정할 일이

없었지만, 이번 문제에서는 마우스를 드래그했을 때 판을 회전시키는 동작에서 값이 변경되어야 한다. MyBoard의 멤버함수로는 특정 좌표에 위치하는 블록의 주소값을 반환하는 gotoxy함수, (0,0)에 위치하는 블록의 주소값을 반환하는 함수와 새로 저장하는 함수, log.txt파일에 실행 결과를 출력하는 함수가 있으며, 게임을 진행하기 위한 무작위로 2를 생성하는 함수, 종료조건을 확인하는 함수, 상, 하, 좌, 우 방향으로 블록을 미는 동작을 수행하는 4개의 함수와 각각 시계방향과 시계반대방향으로 판을 회전시키는 2개의 함수가 선언되었다. 3-5문제에서 수정 된 것은 print함수가 콘솔창이 아닌 log.txt파일에 출력을 하게 된 것과, m_pHead의 값을 수정하는 함수와 회전 동작을 실행하는 TurnCW함수, TurnCCW함수가 새로 정의된 것이다.

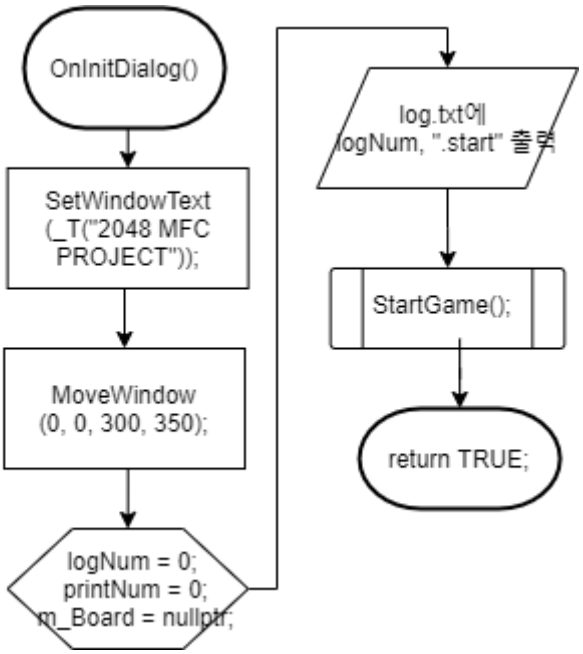
TurnCW함수와 TurnCCW함수는 게임판을 삭제하는 소멸자와 마찬가지로 판의 끝인 (3,3) 칸부터 값을 수정한다. 각 블록에 대해 이웃한 블록의 주소값을 변경하여 판을 회전시키고, 모든 블록에 대해 값이 수정되었다면 새로 (0,0)에 위치한 블록의 주소값을 m_pHead에 저장한다.

Flowchart

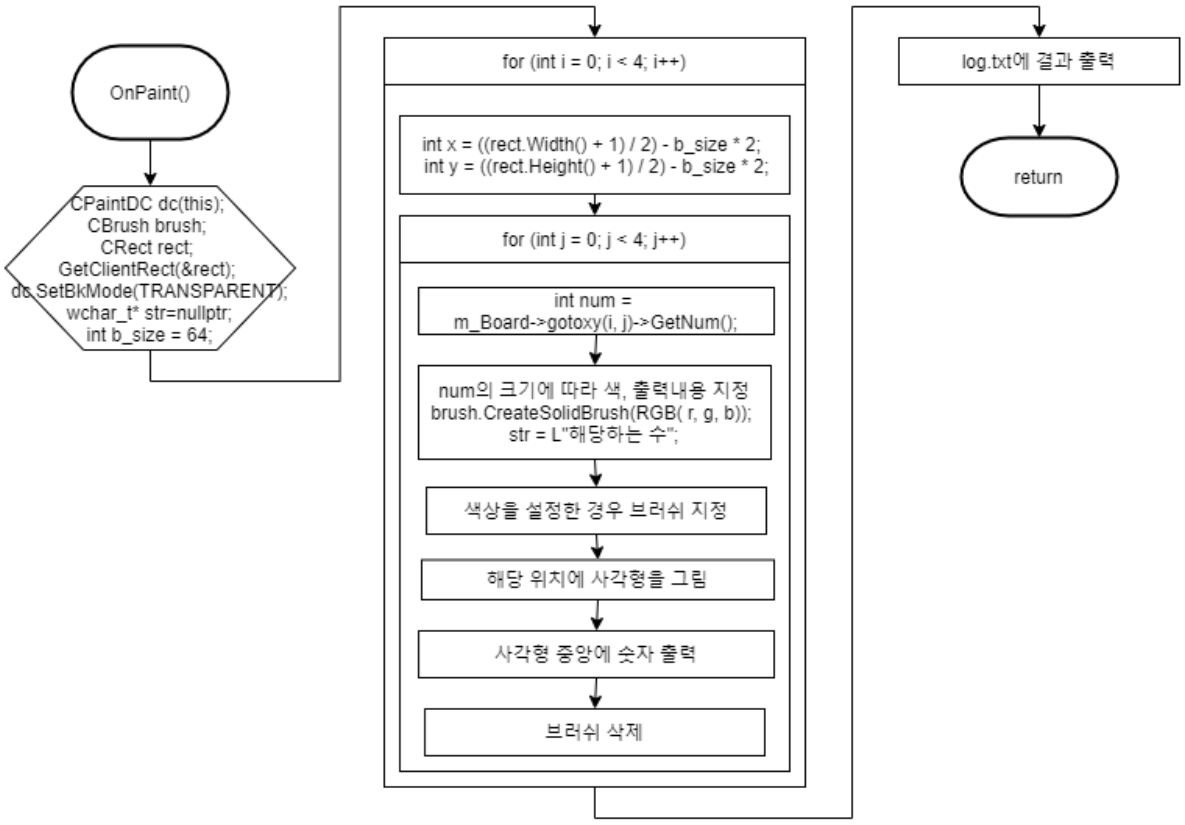
1. 전체 순서도



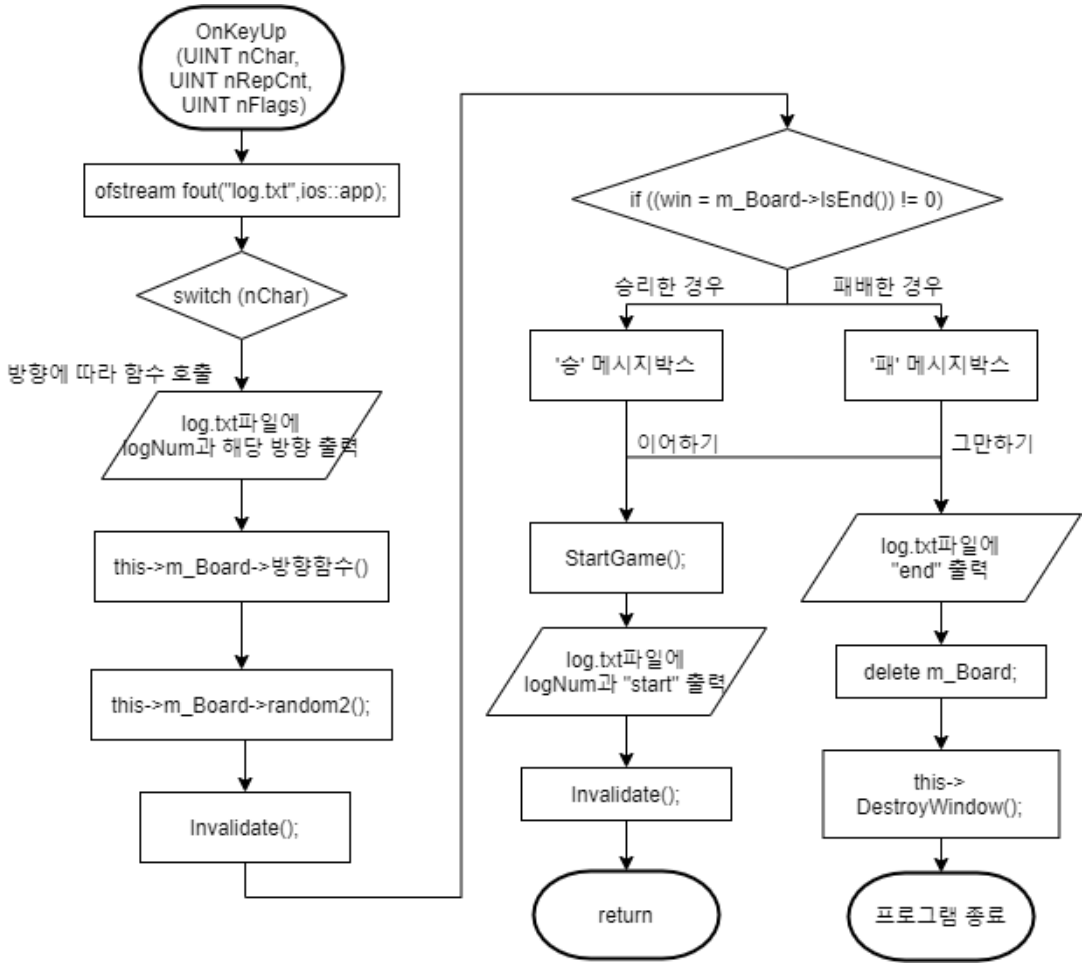
2. OnInitDialog 순서도



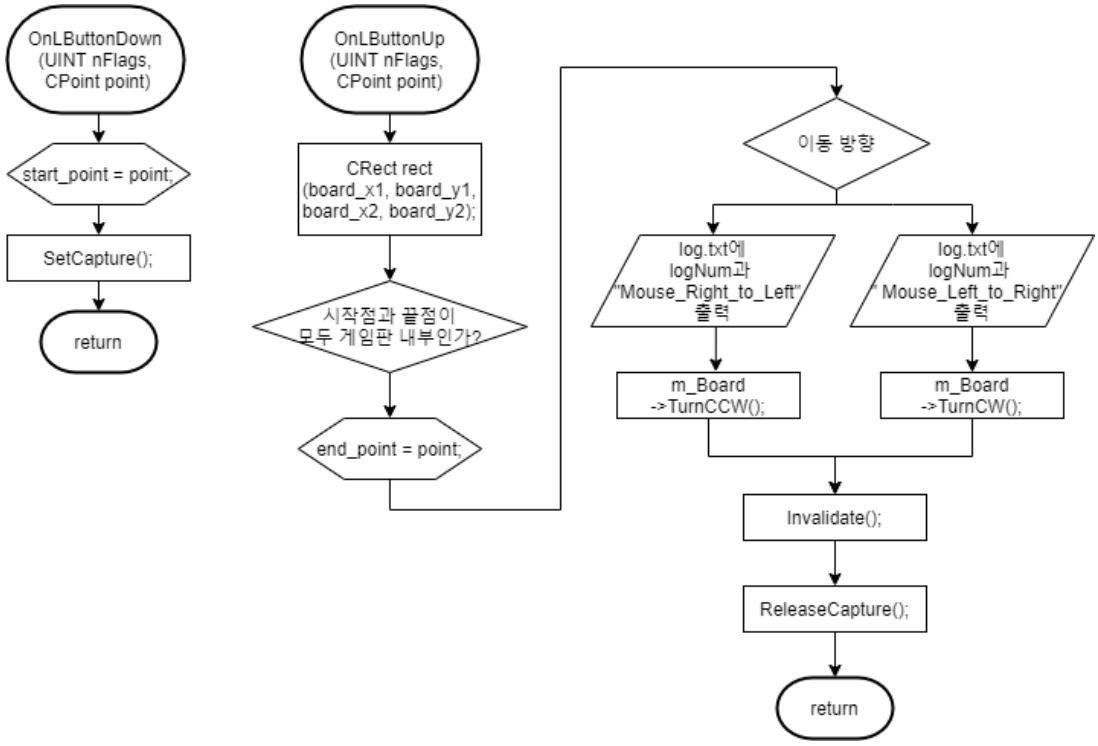
3. OnPaint 순서도



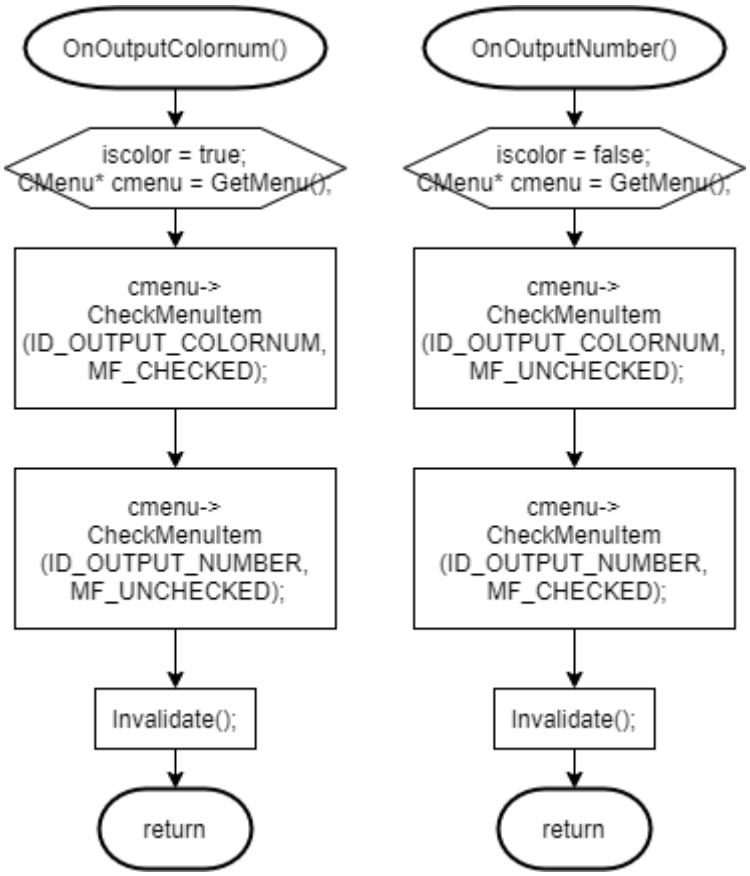
4. OnKeyUp 순서도



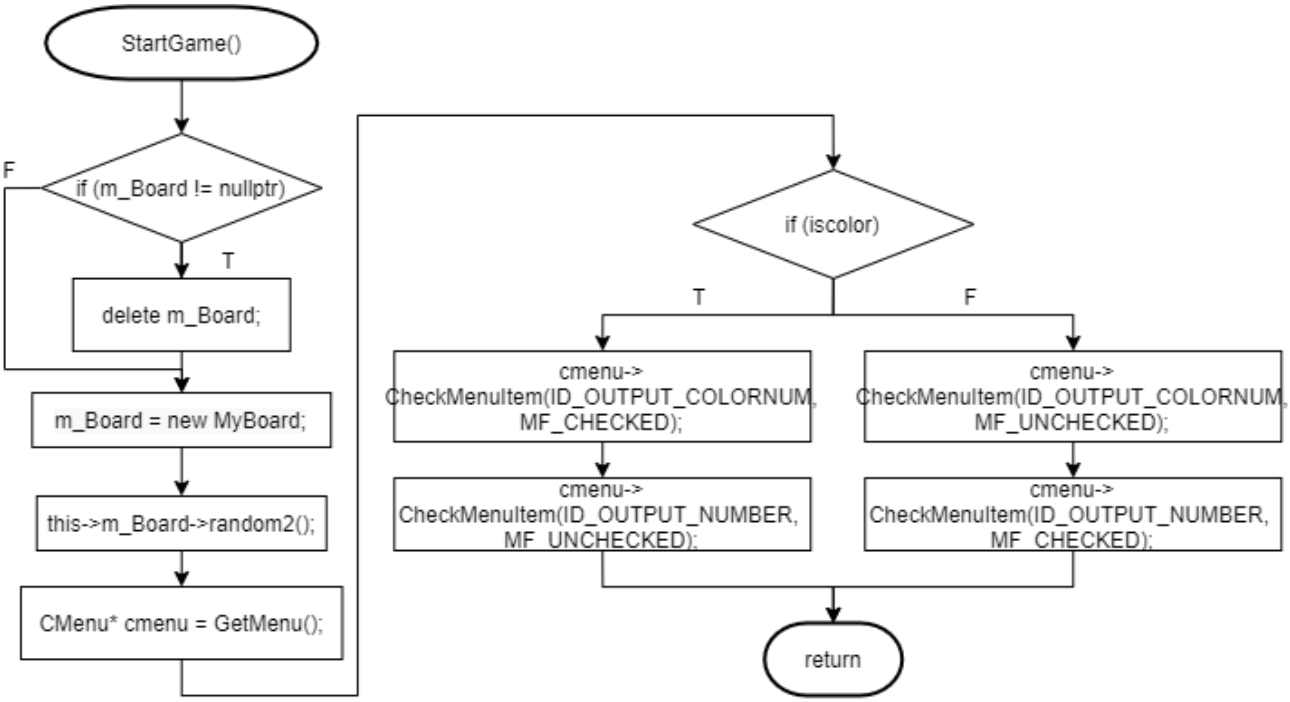
5. OnLButtonDown, OnLButtonUp 순서도



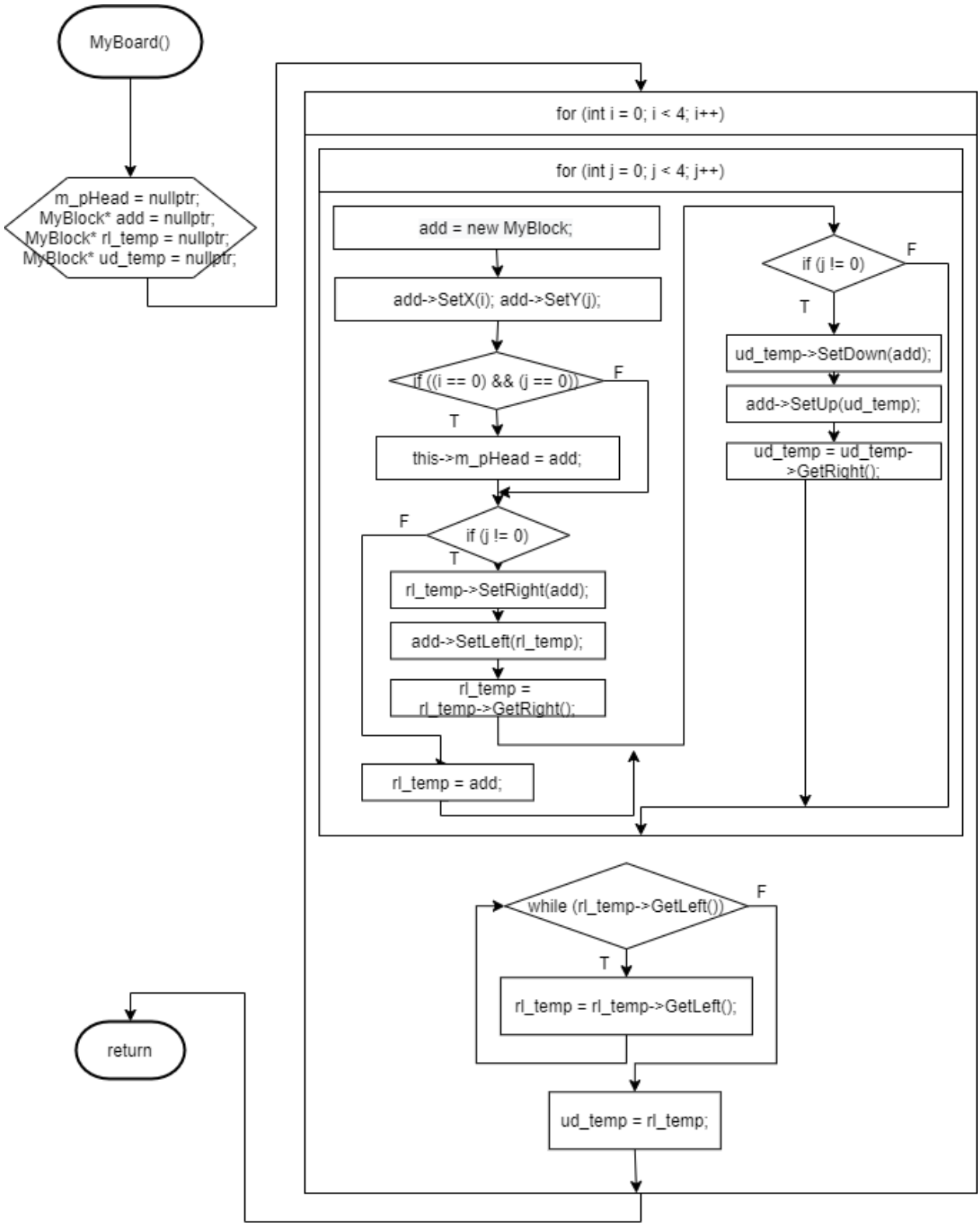
6. OnOutputColornum, OnOutputNumber 순서도



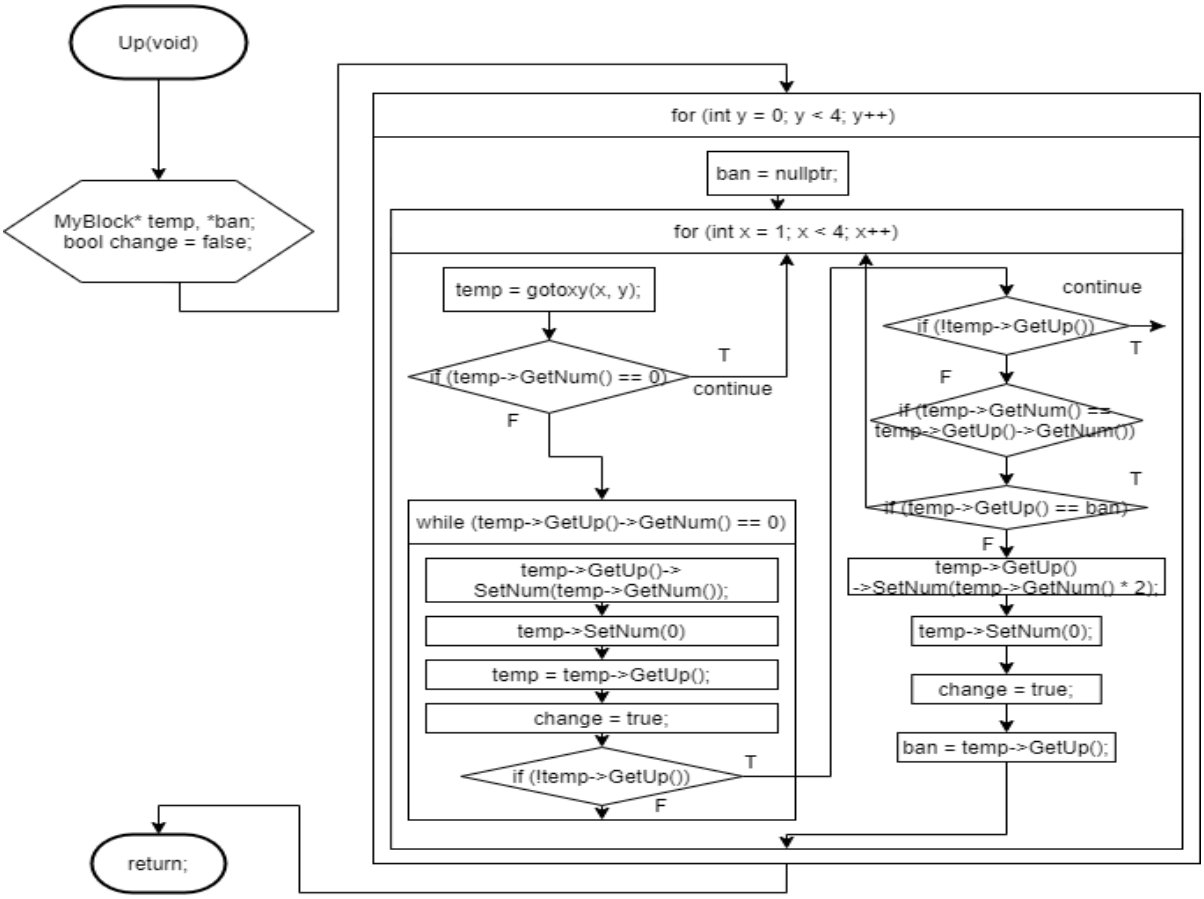
7. StartGame 순서도



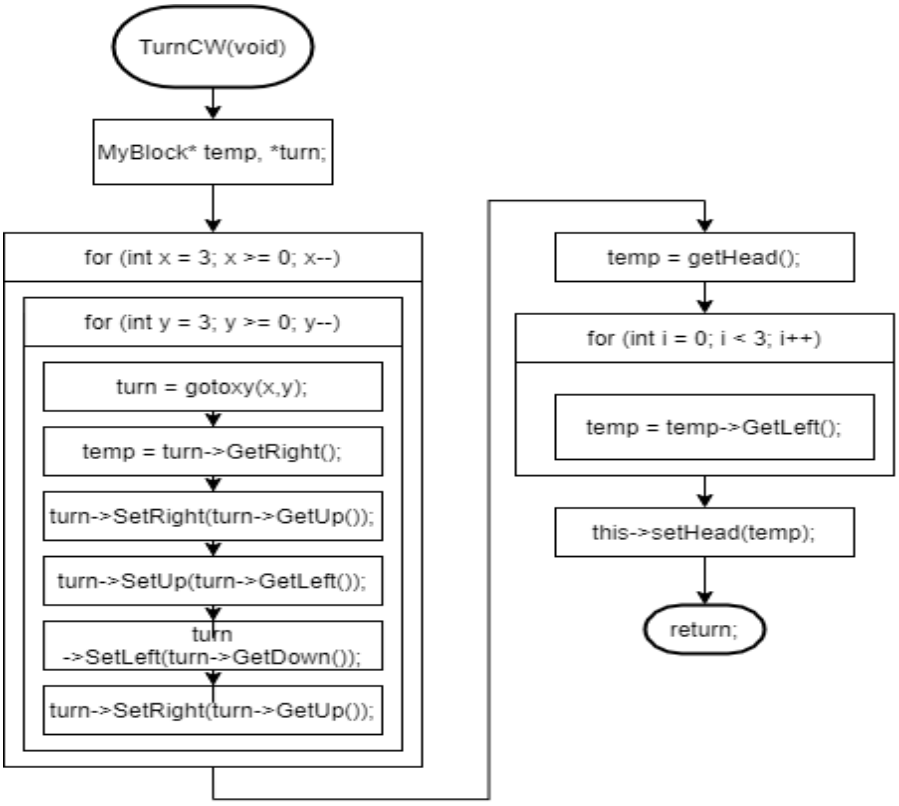
8. MyBoard 순서도



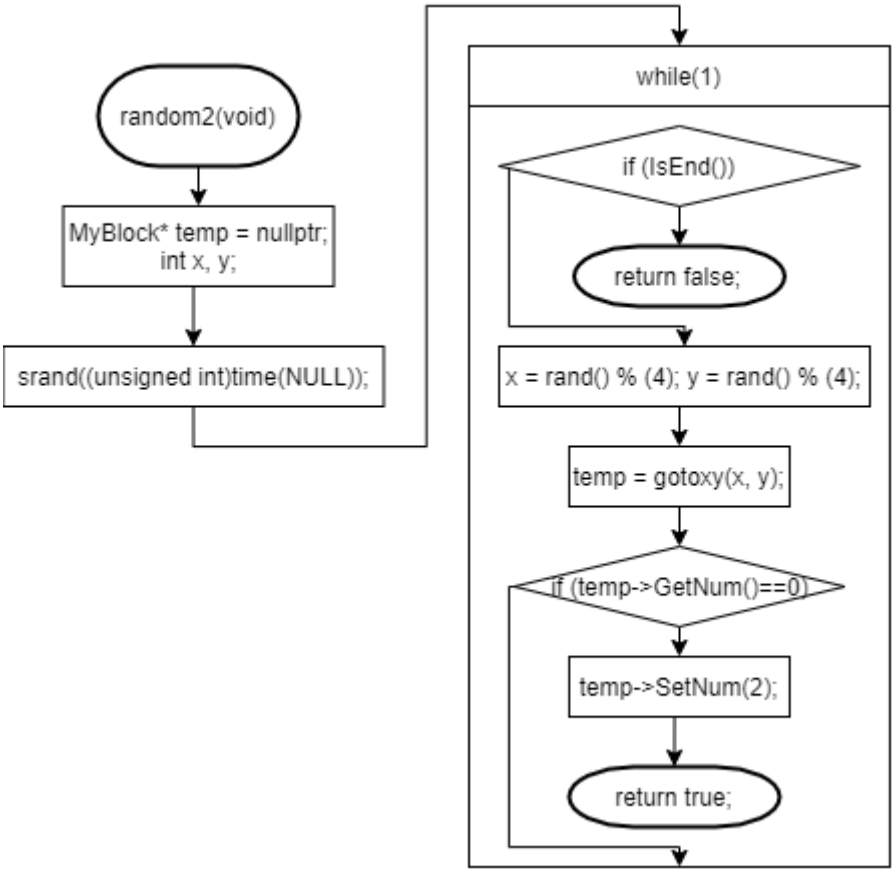
9. Up 순서도 (Down, Left, Right 함수는 구조가 유사함)



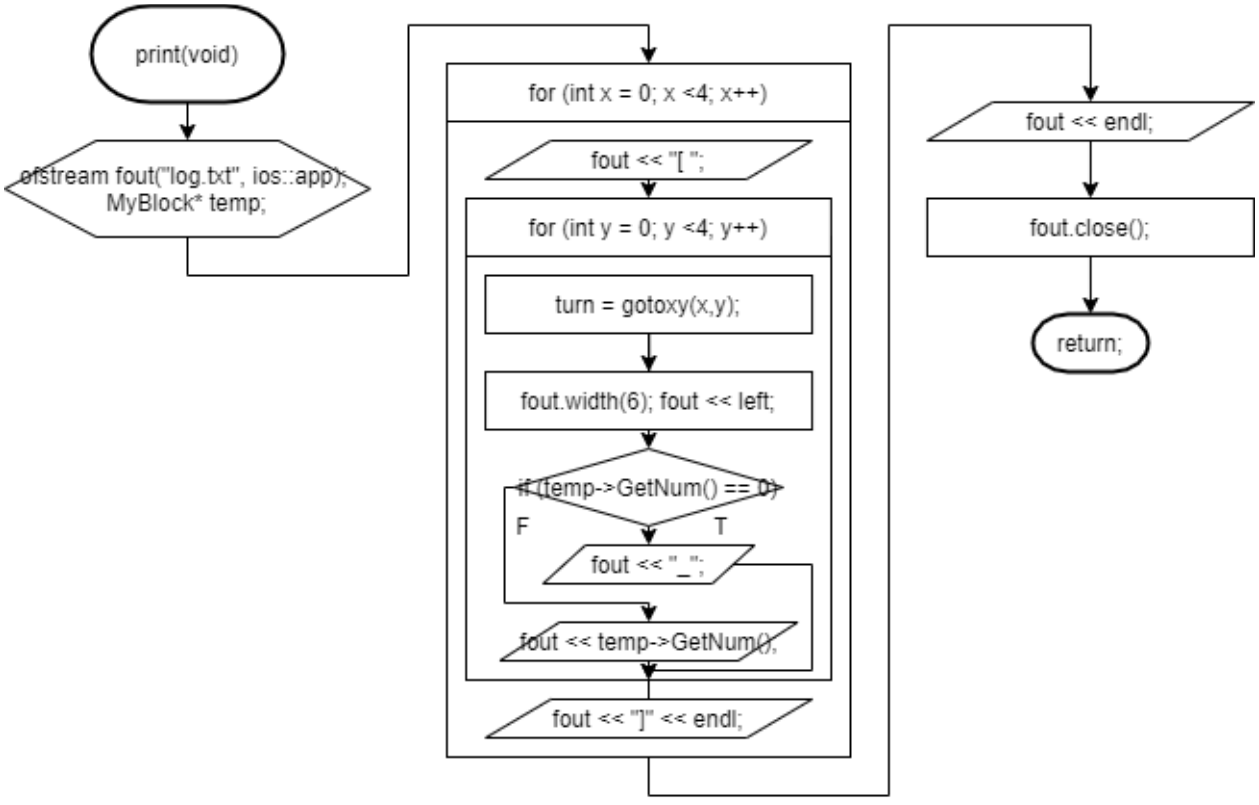
10. TurnCW 순서도 (TurnCCW 함수는 구조가 유사함)



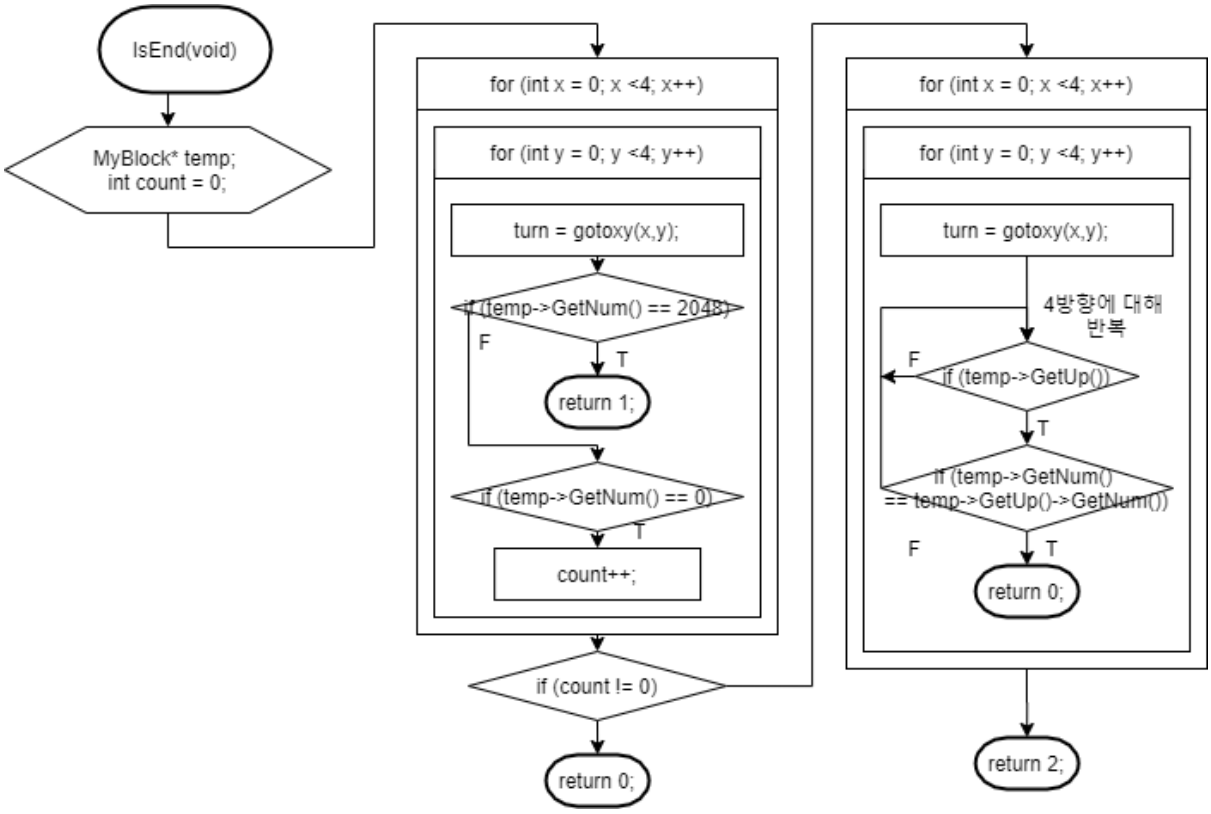
11. random2 순서도



12. print 순서도



13. IsEnd 순서도



Algorithm // Pseudo code

```
OnInitDialog()
        // 창 설정
        Set WindowText to "2048 MFC PROJECT";
        Set Window size 300*350;
        // 변수 초기값
        logNum = 0; printNum = 0;
        m_Board = nullptr;
        iscolor = true;
        // 파일 출력
        "log.txt" 쓰기모드로 열기 //없는 경우 생성
        print (logNum++,". start");
        file close;
        StartGame(); // 게임판 생성, 출력메뉴 설정
        return TRUE;
```

OnInitDialog()에서 실행되는 창의 이름과 크기를 설정하고, log.txt파일을 생성하여 게임이 시작했다는 메시지를 출력한다. 게임판을 생성하는 함수를 호출한다.

```
OnPaint()
        // 화면 출력을 위한 변수
        CPaintDC dc(this);
        CBrush brush;
        CRect rect;
        GetClientRect(&rect);
```

```
        dc.SetBkMode(TRANSPARENT);
        // 변수 초기값
        wchar_t* str=nullptr;
        int b_size = 64;
        // 게임판 출력
        for (int i = 0; i < 4; i++)
                int x = (윈도우의 가로 중간지점 - (사각형의 크기*2));
                int y = (윈도우의 세로 중간지점 - (사각형의 크기*2));
                for (int j = 0; j < 4; j++)
                        int num = (i,j)좌표에 위치한 블록이 갖는 수;
                        if num == 특정 수:
                                brush.CreateSolidBrush(RGB(지정된 색));
                                str = 특정 수(문자열)
                        if(iscolor) dc.SelectObject(brush); //색을 사용함
                        dc.Rectangle 좌표에 맞는 위치에 사각형을 그림
                        dc.TextOutW 사각형에 숫자를 표시함
                        // 전체 판의 영역을 기록
                        if ((i == 0) && (j == 0))
                                board_x1 = x + b_size * j;
                                board_y1 = y + b_size * i;
                        else if ((i == 3) && (j == 3))
                                board_x2 = x + b_size * (j+1);
                                board_y2 = y + b_size * (i+1);
                        brush.DeleteObject();
    return;
```

OnPaint()에서는 16개의 블록으로 구성된 판이 실행창의 중앙에 오도록 판을 출력한다. 이때 설정값에 따라 숫자만 출력하거나, 숫자와 색을 모두 출력한다. 실행창에 출력이 종료되면 log.txt파일을 열어 현재 상황을 출력한다.

```
OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
        "log.txt" 이어서 작성하는 모드로 열기
        //네 가지 방향에 대해 동작
        switch (nChar){
        case VK_UP: //UP
                print (logNum, ". Key_Up") //파일에 출력
                m_Board->Up()과 m_Board->IsFull()실행.
                if 실행결과 변한 값이 없고 판이 가득 찬 경우
                        break;
                else
                        m_Board->random2() 실행
                        break;
        Down, Right, Left 모든 방향에 대해 위와 같은 동작
        }
        Invalidate(); //화면 다시그리기
        win = m_Board->IsEnd() //종료판별
        if win ==1
                승리 메시지박스
        else if win ==2
                패배 메시지박스
```

```
        if 다시하기 선택한 경우
                StartGame();
                print (logNum,". start") // 파일에 출력
                Invalidate(); // 화면 다시그리기
        else
                print ("end.") //파일에 출력
                fout.close(); //파일 닫기
                delete m_Board;
                DestroyWindow(); //프로그램 종료
```

OnKeyUp 함수는 사용자가 키보드를 눌렀다가 손을 떼었을 때 실행되어, 방향키가 입력되면 해당 방향에 대해 블록 밀기 동작을 실행하고, 결과를 출력한다. 출력이 종료되면 게임의 종료여부를 판별하고, 상황에 따라 메시지 박스를 띄우거나 다시 사용자로부터 입력 받기를 기다린다.

```
OnLButtonDown(UINT nFlags, CPoint point)
        start_point = point; //드래그 시작 위치 저장
        SetCapture(); //마우스 위치 추적 시작
```

```
OnLButtonUp(UINT nFlags, CPoint point)
        CRect rect는 게임판 영역
        if (드래그 시작점과 종료점이 모두 rect 내부인 경우)
                "log.txt"를 이어쓰기 모드로 열기
                end_point = point; //끝점 저장
                if (시작점이 끝점보다 오른쪽에 위치)  // <-
                        print( logNum,". Mouse_Right_to_Left");
                        m_Board->TurnCCW(); //회전함수 호출
                else if (시작점이 끝점보다 왼쪽에 위치)  // ->
                        print(logNum, ". Mouse_Left_to_Right");
                        m_Board->TurnCW(); //회전함수 호출
                Invalidate(); //화면 다시그리기
                ReleaseCapture(); //마우스 추적 종료
                fout.close(); //파일닫기
```

OnLButtonDown과 OnLButtonUp함수는 사용자로부터 마우스 입력을 받아 동작한다. 사용자가 마우스 왼쪽 버튼을 눌렀을 때 커서의 위치와 놓았을 때의 위치를 비교하여 두 위치가 모두 게임판의 내부인 경우 주 위치의 x값 차이에 따라서 게임판을 지정된 방향으로 회전시키고 결과를 출력한다.

```
OnOutputColornum()
        iscolor = true;
        //메뉴의 체크표시를 옮김
        CMenu* cmenu = GetMenu();
        COLORNUM 설정에 체크, NUMBER 에 체크 해제
        Invalidate(); // 화면 다시 그리기
```

```
void CMFCApplicationDlg::OnOutputNumber()
        iscolor = false;
        //메뉴의 체크표시를 옮김
        CMenu* cmenu = GetMenu();
        COLORNUM 설정에 체크 해제, NUMBER 에 체크
        Invalidate(); // 화면 다시 그리기
```

OnOutputColornum()과 OnOutputNumber()는 출력메뉴의 해당 항목이 선택되었을 때 호출되어 설정을 변경한 후 결과를 출력한다.

```
StartGame()
        if (m_Board 가 존재하는 경우)
                delete m_Board;
        m_Board = new MyBoard //새로운 보드 만들기
        m_Board->random2();
        // 출력메뉴 체크
        CMenu* cmenu = GetMenu();
        if (iscolor)
                COLORNUM 설정에 체크, NUMBER 에 체크 해제
        else
                COLORNUM 설정에 체크 해제, NUMBER 에 체크
```

StartGame 함수에서는 게임을 진행할 판을 생성하고, 무작위 위치에 2를 배치한다. 이후 색상을 출력하는 설정에 따라 메뉴창의 해당 메뉴에 체크표시한다.

Down, Right, Left함수 유사함

```
bool MyBoard::Up(void)
        MyBlock* temp, *ban;
        bool change = false; // 변한 값이 있으면 true 없으면 false
        for (int y = 0; y < 4; y++) {
                ban = nullptr;
                for (int x = 1; x < 4; x++)  //(0,y) 제외
                        temp = x,y 좌표의 블록;
                        if temp 의 수가 0 인 경우 continue; // 빈 블럭은 이동하지 않음
                        while (temp 의 바로 위 블럭이 빈 경우)
                             위 블럭에 temp 의 수를 저장
                             기존 블록에 0 을 저장
                             temp = temp->GetUp(); //temp 이동
                             change = true;
                             if 위에 블록이 존재하지 않으면 break;
                        if 위에 블록이 존재하지 않으면 continue; //다음 행에 대해 계산
                        if (temp 의 바로 위 블럭이 temp 의 수와 같은 수를 가진 경우)
                            if (temp 의 바로 위 블럭이 이미 더해진 적 있다면) continue;
                             temp->GetUp()->SetNum(temp 의 수의 2 배);
                             기존 블록에 0 을 저장
                             change = true;
                             ban = temp->GetUp(); // 이번 실행에서 다시 더하지 않음
        return change;
```

블록 밀기 동작을 시행하는 함수이다. 다른 방향의 함수도 동작하는 방법은 같다. 위쪽 방향으로 밀기 함수로 예를 들면, 각 행이 독립하여 작동하고, 위에서부터 시작하여 위의 칸이 빈 경우 한 칸 이동, 위의 칸과 이동하려는 칸의 수가 같은 경우 두 수를 더하여 저장한다.

TurnCCW함수 유사함

```
void MyBoard::TurnCW(void)
        MyBlock* temp, *turn;
        for (int x = 3; x >= 0; x--)
                for (int y = 3; y >= 0; y--)
                turn = (x,y)좌표의 블록;
```

```
                temp = turn 의 오른쪽에 이웃한 블록;
                turn 의 오른쪽에 이웃한 블록 -> 위에 이웃한 블록으로 변경;
                turn 의 위에 이웃한 블록 -> 왼쪽에 이웃한 블록으로 변경;
                turn 의 왼쪽에 이웃한 블록 -> 위에 이웃한 블록으로 변경;
                turn 의 아래에 이웃한 블록 -> 변경 전 오른쪽에 이웃했던 블록으로 변경
        //새로운 Head 를 저장
        temp = getHead();
        for (int i = 0; i < 3; i++)
                temp = temp->GetLeft();
        this->setHead(temp);
        return;
```

게임판을 회전시키는 동작을 수행하는 함수이다. 중앙의 블록을 기준으로 주변에 이웃한 블록의 포인터 값을 변경한다. 16 개의 블록에 대해 모두 시행했다면 마지막으로 head 값을 바꾼다.

```
bool MyBoard::random2(void)
        MyBlock* temp = nullptr;
        int x, y;
        srand(time(NULL)); //시간을 이용한 난수발생
        while(1)
                if 판에 빈칸이 없다면 return false;
                x = 0~3 사이 난수; y = 0~3 사이 난수;
                temp = (x,y)좌표의 블록;
                if (temp 가 비어있는 경우)
                        temp 에 2 저장
                        return true;
```

난수를 발생시켜 빈칸에 2 를 무작위로 생성하는 함수이다.

```
void MyBoard::print(void)
        "log.txt" 이어서 쓰기 모드로 열기
        MyBlock* temp;
        print("[result]") //파일 출력
        for (int x = 0; x <4; x++)
                print ("[   ") //파일출력
                for (int y = 0; y <4; y++)
                        temp = (x,y)위치 블록
                        width(6) 6 칸 확보; 왼쪽 정렬;
                        if (temp 의 수가 0 인경우) print("_"); //파일출력
                        else print( temp->GetNum()); // 파일출력
                print ("]") //파일출력
        파일 닫기
        return;
```

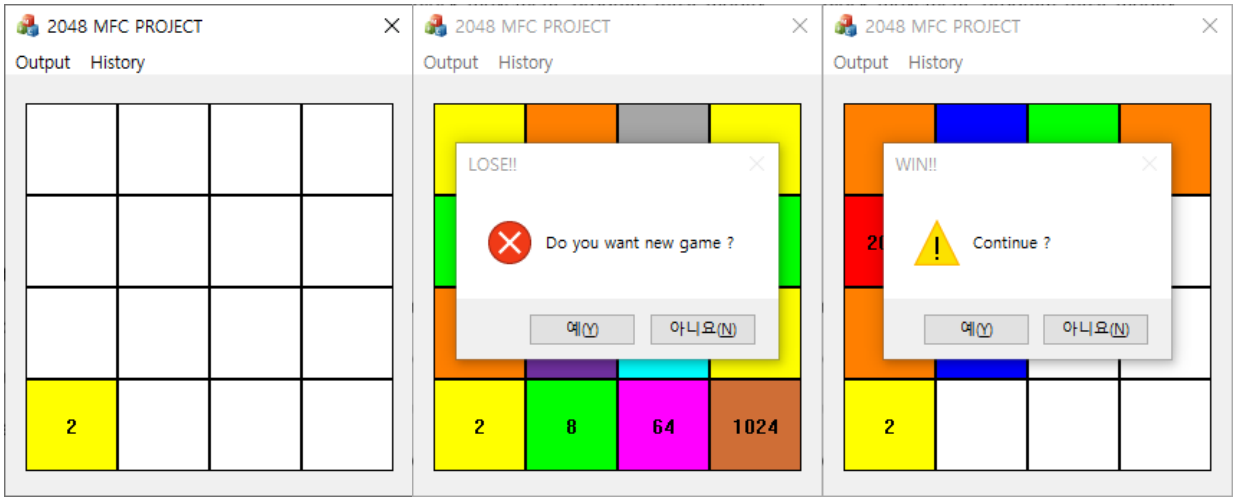
log.txt 파일에 현재 게임판의 상태를 출력하는 함수이다. 판의 모든 블록에 출력할 순서대로 접근하여 값을 출력한다.

```
int MyBoard::IsEnd(void)
        MyBlock* temp;
        int count = 0;
        for (int x = 0; x < 4; x++)
                for (int y = 0; y < 4; y++)
                        temp = (x, y) 위치 블록
                        if temp 에 2048 이 저장된 경우 return 1; // 승
                        if temp 가 빈 칸인 경우 count++; //빈칸의 개수
        if (빈칸이 있는 경우) return 0; // 이어서
        // 같은 수가 연달아 존재하는 경우 계속
        for (int x = 0; x < 4; x++)
                for (int y = 0; y < 4; y++)
```

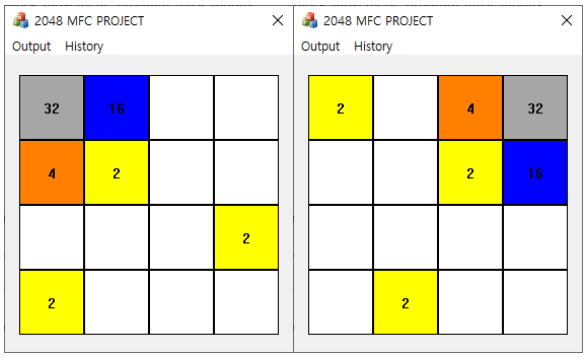
```
temp = (x, y) 위치 블록;
if temp 의 위칸이 존재할 때
        if (temp 의 수 == temp 의 위칸의 수) return 0; //이어서
if temp 의 아래칸이 존재할 때
        if (temp 의 수 == temp 의 아래칸의 수) return 0;//이어서
if temp 의 왼쪽칸이 존재할 때
        if (temp 의 수 == temp 의 왼쪽칸의 수) return 0;//이어서
if temp 의 오른쪽칸이 존재할 때
        if (temp 의 수 == temp 의 오른쪽칸의 수) return 0;//이어서
    return 2;// 패
```

게임의 종료여부를 확인하는 함수이다. 모든 칸을 조회하여 2048 이 있는지, 빈 칸의 개수가 몇 개인지 확인한다. 2048 이 있는 경우 게임은 승리로 종료되고, 빈칸이 있는 경우 게임은 종료되지 않는다. 빈칸이 존재하지 않지만 더할 수 있는 방향이 남아있는 경우에도 게임은 종료되지 않는다. 모든 경우에 해당하지 않으면 게임은 패배로 종료된다.

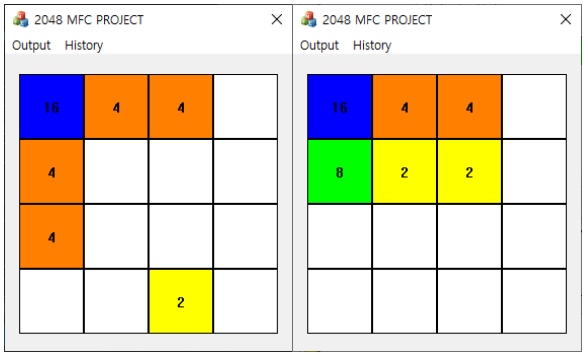
결과화면



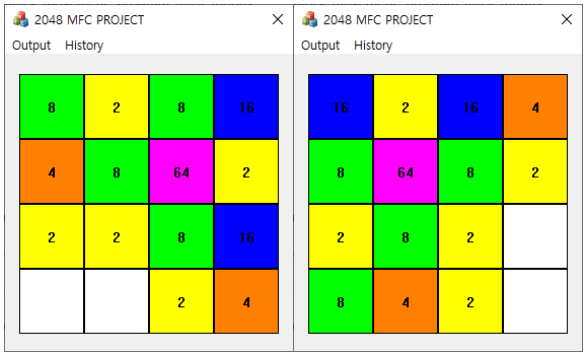
초기화면 패 승



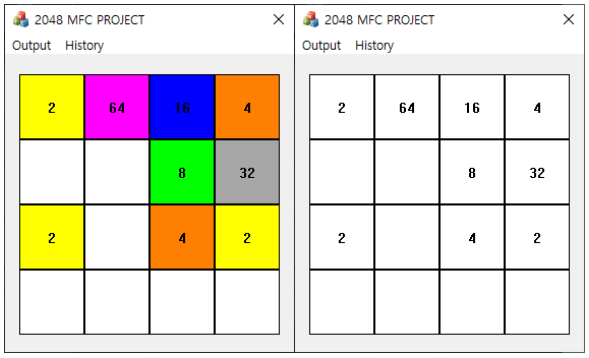
왼쪽에서 오른쪽으로 드래그



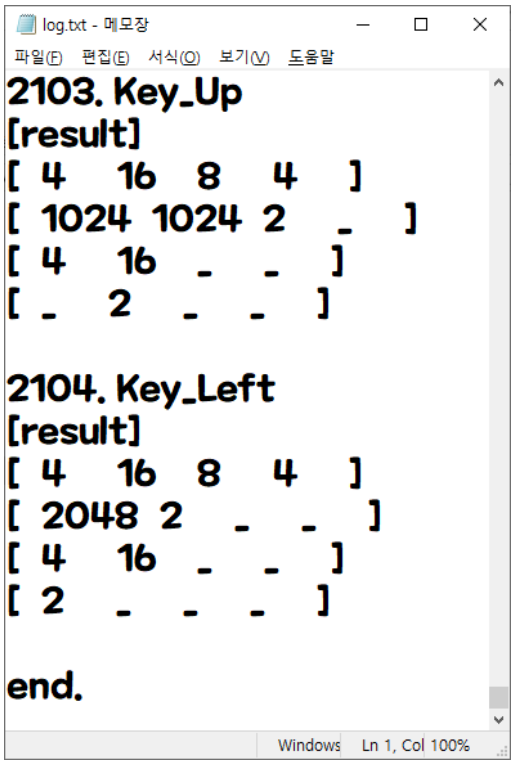
위쪽 방향키 입력



오른쪽에서 왼쪽으로 드래그



Output 메뉴 설정변경



log.txt파일 출력결과

고찰

OnKeyDown를 이용하여 프로그램을 작성하였지만 방향키에서 제대로 동작하지 않아 비슷한 기능을 하는 OnKeyUp함수로 대체하였다. OnKeyDown함수는 w,a,s,d등의 키에서는 제대로 작동하였지만 방향키에서는 작동하지 않는 이유를 찾아내지 못해서 아쉬움이 남는다. MFC프로그램을 처음 제작해보아서 몇 번의 시행착오를 겪었다. 특히 화면에 직접 나타나는 게임판 부분에 관련하여 문제가 발생하는 경우가 많았다. 마우스 드래그는 게임 판위에서 나타나야 하는데, 게임 판은 창의 크기에 맞추어 창의 중간에 나타나도록 만들어서 마우스의 인식 범위 또한 중간에 변화할 수 있게 만들어야 했던 문제가 있었다. 이것은 게임판을 출력할 때 판의 왼쪽 위 점의 좌표와 오른쪽 아래의 좌표를 저장해서 사용하는 것으로 해결했다. 메뉴의 이벤트를 잘못 적용하여 제대로 동작하지 않거나, 게임판의 출력 순서를 잘못 지정해두고 키보드 입력이 잘못된 것으로 판단하는 실수를 하기도 했다. MFC프로그램을 생성했을 때 많은 헤더파일과 cpp파일, 처음 접해보는 리소스 파일이 생성되어 당황하기도 했지만, 실제로 내용을 수정하여 사용한 부분은 적었다. 지금까지 작성했던 프로그램과는 디버깅과정에서 나타나는 것이 좀 다른 것 같다. 사용자의 입력을 받을 때 까지는 다음 동작을 하지 않는 것이 지금까지 접했던 프로그램의 실행과정이었는데, MFC프로그램은 여러 동작을 수행하고 있는 것 같았다. 이와 관련해서 MFC의 구조에 대해 더 자세히 알아보고 싶다. 동작에 필요한 함수를 제작하고 메뉴버튼을 만들어 연결하는 과정이 장난감 블록을 조립하는 것을 연상시켜서 재미있었다. 메뉴 이벤트와 메시지 처리의 다른 다양한 기능도 사용해 보고싶다.