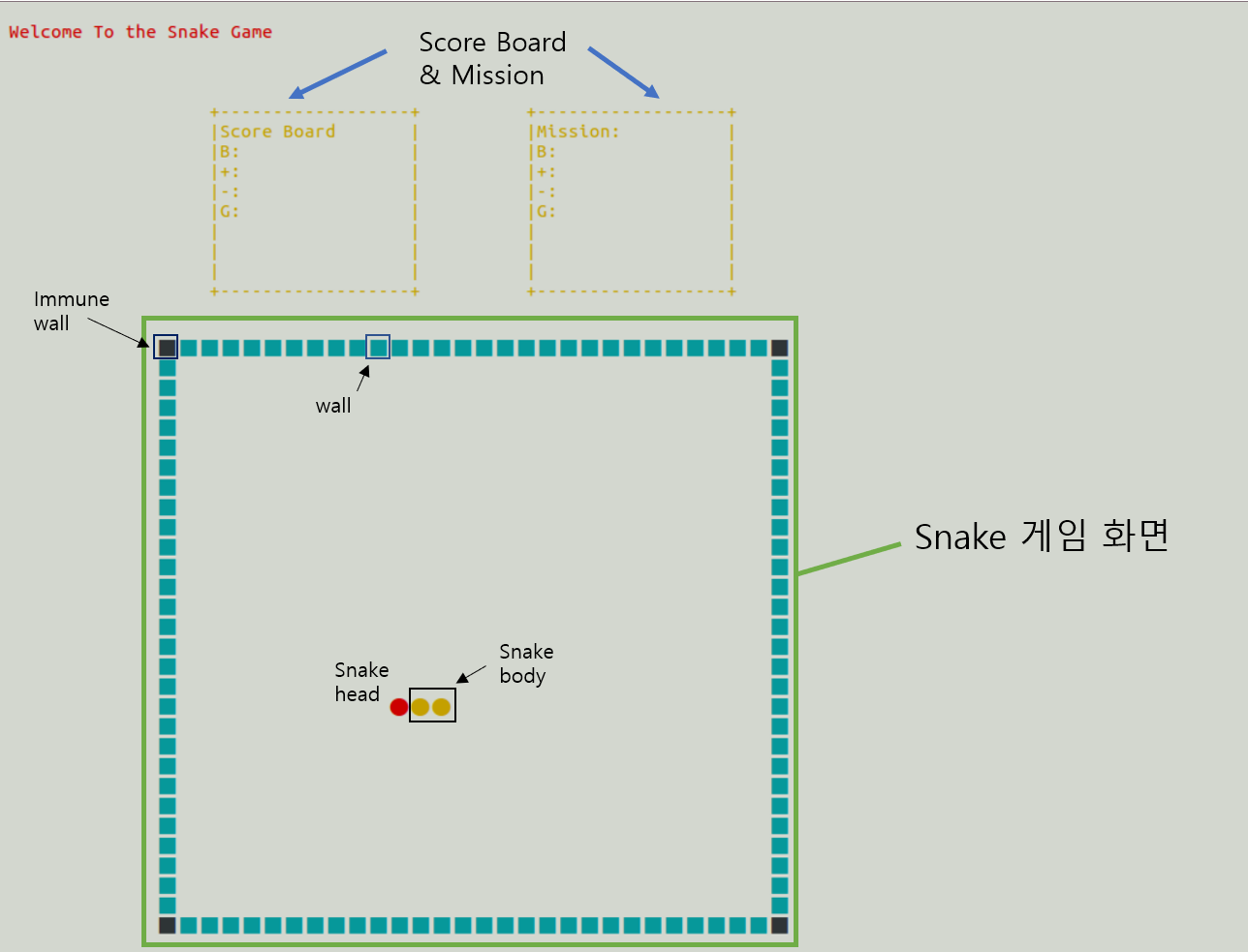
Snake stage2 보고서

결과물



1. 소스코드 구조
2. initScreen 함수

keypad와 관련된 설정을 마무리하고 색상 팔레트를 지정한 뒤 “Welcome To the Snake Game”을 출력하고 이를 refresh()로 화면에 나타낸다. Main 함수의 맨 처음에 실행된다.

1. Map 클래스

<vector<vector<int>>, 즉, int의 vector의 vector 타입의 데이터인 mapData를 선언한다. display 멤버 함수를 통해 화면을 각 색상(바탕색, 변색, 모서리 색)에 맞춰 출력한다. 출력하려는 snake를 구성하는 문자 유니코드가 비 유니코드 문자의 두 칸을 차지해서, snake의 지나간 자국을 지우기 위해서는 두 칸을 모두 바탕으로 칠해줘야 한다. 그리고 공백 문자를 mvwprintw 함수를 이용해서 출력했을 때 snake의 지나간 자국이 잘 지워지지 않는 문제가 발생하여 M 문자를 써주기로 했다.

1. Snake 클래스

Snake에 대한 정보를 담고 있다. snake가 충돌하는 등 게임 오버하게 되면 bool 타입의 데이터인 Fail 변수를 true를 만들어준다. 이 Fail 변수는 false로 초기화된다.

positionR과 positionC는 게임 시작 전의 snake의 Head가 위치하는 좌표로서 각각 Row(행 좌표)와 Column(열 좌표)을 의미한다. snakeData도 mapData와 마찬가지로 <vector<vector<int>>으로 선언된다. positionR과 positionC를 이용해 Head의 좌표와 두 개의 Body의 좌표를 초기화한다. int 타입의 key, key1은 각각 현재 입력 받은 키 값과 snake의 방향이 변경되기 전의 키 값이다.

1. tick 멤버함수

t1과 t2는 int 타입의 변수로서 각각 snake의 Head의 행좌표와 열좌표를 나타낸다. nodelay 함수로 stdscr 윈도우에 대해 getch 메서드를 이용해 문자를 입력받을 때 다른 코드들이 동작을 멈추지 않도록 해준다.

현재 키 값인 key의 값과 예전 키 값인 key1의 값에 따라 snake의 Head의 좌표를 변경시킨다.

* key의 값이 -1인 경우

새로운 key 입력이 없는 경우이다. 이 경우, 예전 key 값인 key1의 값에 따라 snake의 Head 좌표를 바꾼다. key1의 값이 KEY\_UP인 경우 위 방향 화살표를 누른 경우이므로 Head좌표를 나타내는 snakeData[0]의 0번째 index에 있는 값(Head의 행 좌표)을 1 감소시킨다. 윈도우의 좌표는 오른쪽이 +, 아래가 +이기 때문이다. 나머지의 경우(KEY\_LEFT, KEY\_DOWN, KEY\_RIGHT)도 마찬가지로 한다.

* key의 값이 KEY\_UP인 경우

위 방향 화살표를 누른 경우이다. Head의 방향에 영향을 미치는 마지막 키인 key1의 값이 KEY\_UP의 반대인 KEY\_DOWN인 경우 게임 오버를 발생시키기 위해 Fail 값을 true로 해준다. snake의 Head 좌표를 나타내는 snakeData를 변경시켜주고 key1에 Head의 방향에 영향을 미치는 마지막 키가 된 key를 대입한다.

1. getFail 멤버함수

private으로 선언된 Fail 변수의 값을 알아내어 return한다.

1. display 멤버함수
2. main 함수 내의 게임 루프

Snake의 객체인 snake의 Fail 멤버 변수가 false가 아닌 동안 게임 루프를 돈다.

    while (true) {

        //getch();

        if (snake.getFail()) {

            break;

        }

        snake.tick();

        map.display(win1);

        wrefresh(win1);

        snake.display(win1);

        wrefresh(win1);

        std::this\_thread::sleep\_for(std::chrono::milliseconds(200));

    }

1. 실행과정
2. 스크린 초기화 및 keypad 설정

initScreen 함수로 구현

1. Map 클래스

// Map 클래스

class Map

{

    vector<vector<int>> mapData;

public:

    Map(const vector<vector<int>> &mapData) : mapData(mapData)

    {

    }

    int getRowSize() {

        return mapData.size();

    }

    int getColSize() {

        return mapData[0].size();

    }

    void display(WINDOW \*win)

    {

        int nrow = mapData.size();

        int ncol = mapData[0].size();

        for (int i = 0; i < nrow; i++)

        {

            for (int j = 0; j < ncol; j++)

            {

                if (mapData[i][j] == 0)

                {

                    wattron(win, COLOR\_PAIR(5)); // 바탕색

                    mvwprintw(win, i, j\*2, "M");

                    mvwprintw(win, i, j\*2+1, "M");

                    wattroff(win, COLOR\_PAIR(5));

                }

                if (mapData[i][j] == 1)

                {

                    wattron(win, COLOR\_PAIR(6)); // 변 색

                    mvwprintw(win, i, j \* 2, "\u25A0");

                    //mvwprintw(win, i, j+1, "\u25A0");

                    wattroff(win, COLOR\_PAIR(6));

                }

                if (mapData[i][j] == 2)

                {

                    wattron(win, COLOR\_PAIR(7)); // 모서리 색

                    mvwprintw(win, i, j \* 2, "\u25A0");

                    //mvwprintw(win, i, j+1, "\u25A0");

                    wattroff(win, COLOR\_PAIR(7));

                }

            }

        }

        wrefresh(win);

    }

};

1. Snake 클래스

    // Snake 클래스

class Snake {

    bool Fail = false;

    int positionR = 7, positionC = 7;

    vector<vector<int>> snakeData = vector<vector<int>>({ { positionR, positionC }, {positionR, positionC+1}, {positionR, positionC+2} });

public:

    Snake() : positionR(7), positionC(7) {

    }

    int key, key1;

    void tick()

    {

        int t1 = snakeData[0][0];

        int t2 = snakeData[0][1];

        nodelay(stdscr, true);

        key = getch();

        // 새로운 key 입력이 없으면 HEAD는 예전 key 입력에 대한 동작을 반복함

        if (key == -1) {

            if (key1 == KEY\_UP) {

                snakeData[0][0]--;

            }

            if (key1 == KEY\_LEFT) {

                snakeData[0][1]--;

            }

            if (key1 == KEY\_DOWN) {

                snakeData[0][0]++;

            }

            if (key1 == KEY\_RIGHT) {

                snakeData[0][1]++;

            }

        }

        //새로운 key 입력에 따라 HEAD를 이동함

        if (key == KEY\_UP) {

            if (key1 == KEY\_DOWN) {

                Fail = true;

            }

            snakeData[0][0]--;

            key1 = key;

        }

        if (key == KEY\_LEFT) {

            if (key1 == KEY\_RIGHT) {

                Fail = true;

            }

            snakeData[0][1]--;

            key1 = key;

        }

        if (key == KEY\_DOWN) {

            if (key1 == KEY\_UP) {

                Fail = true;

            }

            snakeData[0][0]++;

            key1 = key;

        }

        if (key == KEY\_RIGHT) {

            if (key1 == KEY\_LEFT) {

                Fail = true;

            }

            snakeData[0][1]++;

            key1 = key;

        }

        // 자취 따라가기. BODY는 자신의 앞에 있는 BODY나 HEAD를 따라간다.

        for (int i  = snakeData.size()-1; i > 1; i--) {

            snakeData[i][0] = snakeData[i-1][0];

            snakeData[i][1] = snakeData[i-1][1];

        }

        snakeData[1][0] = t1;

        snakeData[1][1] = t2;

    }

    bool getFail() {

        return Fail;

    }

    void display(WINDOW \*win) {

        wattron(win, COLOR\_PAIR(5));

        mvwprintw(win, snakeData[0][0], snakeData[0][1]\*2, " ");

        mvwprintw(win, snakeData[0][0], snakeData[0][1]\*2+1, " ");

        wattroff(win, COLOR\_PAIR(5));

        wattron(win, COLOR\_PAIR(8)); // HEAD 색

        mvwprintw(win, snakeData[0][0], snakeData[0][1]\*2, "\u25CF");

        wattroff(win, COLOR\_PAIR(8));

        for (int i = 1; i < snakeData.size(); i++) {

            wattron(win, COLOR\_PAIR(5));

            mvwprintw(win, snakeData[i][0], snakeData[i][1]\*2, " ");

            mvwprintw(win, snakeData[i][0], snakeData[i][1]\*2+1, " ");

            wattroff(win, COLOR\_PAIR(5));

            wattron(win, COLOR\_PAIR(9)); // BODY 색

            mvwprintw(win, snakeData[i][0], snakeData[i][1]\*2, "\u25CF");

            wattroff(win, COLOR\_PAIR(9));

        }

    }

};

1. main 함수

    int main()

{

    initScreen();

    Map map = vector<vector<int>>(

        {

            {2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

            {2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2},

        }

    );

    WINDOW \*win1 = newwin(map.getRowSize(), map.getColSize() \* 2, 18, 16);

    wbkgd(win1, COLOR\_PAIR(1));

    map.display(win1);

    // score board

    WINDOW \*win2 = newwin(10, 20, 5, 20);

    int B = 0, plus = 0, minus = 0, G = 0;

    wbkgd(win2, COLOR\_PAIR(1));

    wattron(win2, COLOR\_PAIR(4));

    mvwprintw(win2, 1, 1, "Score Board");

    mvwprintw(win2, 2, 1, "B: ");

    mvwprintw(win2, 3, 1, "+: ");

    mvwprintw(win2, 4, 1, "-: ");

    mvwprintw(win2, 5, 1, "G: ");

    wborder(win2, '|', '|', '-', '-', '+', '+', '+', '+');

    wattroff(win2, COLOR\_PAIR(4));

    wrefresh(win2);

    // mission board

    WINDOW \*win3 = newwin(10, 20, 5, 50);

    wbkgd(win3, COLOR\_PAIR(1));

    wattron(win3, COLOR\_PAIR(4));

    mvwprintw(win3, 1, 1, "Mission: ");

    mvwprintw(win3, 2, 1, "B: ");

    mvwprintw(win3, 3, 1, "+: ");

    mvwprintw(win3, 4, 1, "-: ");

    mvwprintw(win3, 5, 1, "G: ");

    wborder(win3, '|', '|', '-', '-', '+', '+', '+', '+');

    wattroff(win3, COLOR\_PAIR(4));

    wrefresh(win3);

    //bool gameOn = true;

    Snake snake;

    // Game loop

    int i = 0;

    while (true) {

        //getch();

        if (snake.getFail()) {

            break;

        }

        snake.tick();

        map.display(win1);

        wrefresh(win1);

        snake.display(win1);

        wrefresh(win1);

        std::this\_thread::sleep\_for(std::chrono::milliseconds(200));

    }

    nodelay(stdscr, false);

    attron(COLOR\_PAIR(8));

    mvprintw(16, 30, "YOU LOSE...");

    attroff(COLOR\_PAIR(8));

    refresh();

    getch();

    delwin(win1);

    delwin(win2);

    delwin(win3);

    endwin();

    return 0;

}

1. 종료

    delwin(win1);

    delwin(win2);

    delwin(win3);

    endwin();

    return 0;

1. 문제점

키보드 입력을 통한 snake 방향전환이 안 될 때가 가끔 있다. 필자의 키보드 문제일 가능성이 있음.