**1942 프로젝트**

**최종 보고서**

2학년 3반

1705095 이창민

2022. 06. 14 제출

**목차**

|  |  |  |  |
| --- | --- | --- | --- |
| 파일 이름 | 분류 | 페이지 | 이동 |
| common.h | 프로그램 공통 헤더파일 | 5~6 | [→](#common) |
| cursor.c | 프로그램 공통 기능 | 7 | [→](#cursor) |
| gotoxy.c | 프로그램 공통 기능 | 8 | [→](#gotoxy) |
| move.c | 프로그램 공통 기능 | 9~10 | [→](#move) |
| set\_color.c | 프로그램 공통 기능 | 11 | [→](#set_color) |
| draw\_error.c | DB – 에러 UI | 12 | [→](#draw_error) |
| draw\_content.c | 프로그램 공통 UI | 13~25 | [→](#draw_content) |
| main.c | 메인화면 | 26~27 | [→](#main) |
| menu.c | 메인화면 메뉴 | 28 | [→](#menu) |
| login.c | 회원가입, 로그인 | 29 | [→](#login) |
| login\_db.c | 회원가입, 로그인 - DB | 30~34 | [→](#login_db) |
| ranking.c | 랭킹 | 35 | [→](#ranking) |
| ranking\_db.c | 랭킹 – DB | 36~45 | [→](#ranking_db) |
| shop.c | 상점 | 46~52 | [→](#shop) |
| shop\_db.c | 상점 – DB | 53~54 | [→](#shop_db) |
| shop\_preview.c | 상점 – UI | 55~59 | [→](#shop_preview) |
| draw\_owned\_item.c | 상점 – UI | 60 | [→](#draw_owned_item) |
|  | | | |
| game.c | 게임화면 | 61~70 | [→](#game) |
| game.h | 게임 – 헤더파일 | 71~75 | [→](#game_h) |
| init\_variables | 게임 – 변수 초기 설정 | 76~78 | [→](#init_variables) |
| load\_data | 게임 – DB 불러오기 | 79~81 | [→](#load_data) |
| save\_data | 게임 – DB 저장 | 82~85 | [→](#save_data) |
| draw\_game | 게임 UI | 86~90 | [→](#draw_game) |
| draw\_status | 게임 – 스탯 UI | 91~92 | [→](#draw_stat) |
| draw\_symbol | 게임 – 스탯 – 아이콘 | 93 | [→](#draw_symbol) |
| draw\_help | 게임 – 도움말 UI | 94 | [→](#draw_help) |
| draw\_end\_game | 게임 – 게임 종료 UI | 95~97 | [→](#draw_end_game) |
| draw\_bomb | 게임 – 폭탄 기능 UI | 98 | [→](#draw_bomb) |
| cannon\_move | 게임 – 캐논 이동, 출력 | 99 | [→](#cannon_move) |
| cannon\_status | 게임 – 캐논 상태 관리 | 100 | [→](#cannon_status) |
| bomb\_move | 게임 – 폭탄 기능 이동, 출력 | 101~102 | [→](#bomb_move) |
| bomb\_status | 게임 – 폭탄 상태 관리 | 103~105 | [→](#bomb_status) |
| bCannon\_status | 게임 – 폭탄 캐논 상태 관리 | 106~107 | [→](#bCannon_status) |
| bCannon\_move | 게임 – 폭탄 캐논 이동, 출력 | 108 | [→](#bCannon_move) |
| item\_gen | 게임 – 드롭 아이템 생성 | 109~112 | [→](#item_gen) |
| item\_move | 게임 – 드롭 아이템 이동, 출력 | 113~114 | [→](#item_move) |
| item\_status | 게임 – 드롭 아이템 상태 관리 | 115~118 | [→](#item_status) |
| enemy\_gen | 게임 – 적군 생성 | 119~124 | [→](#enemy_gen) |
| enemy\_move | 게임 – 적군 이동, 출력 | 125~131 | [→](#enemy_move) |
| enemy\_status | 게임 – 적군 상태 관리 | 132~138 | [→](#enemy_status) |
| eCannon\_status | 게임 – 적군 캐논 상태 관리 | 139~146 | [→](#eCannon_status) |
| eCannon\_move | 게임 – 적군 캐논 이동, 출력 | 147~149 | [→](#eCannon_move) |

MYSQL 데이터 베이스 구조

|  |  |  |  |
| --- | --- | --- | --- |
| user table | | | |
| Field | Type | Null | Key |
| id | varchar(50) | NO | 기본키 |
| passwd | varchar(50) | NO |  |
| score | int(11) | YES |  |
| point | int(11) | YES |  |

|  |  |  |  |
| --- | --- | --- | --- |
| item\_list table | | | |
| Field | Type | Null | Key |
| id | varchar(50) | NO | 외부키 (user) |
| item1 | int(11) | YES |  |
| item2 | int(11) | YES |  |
| item3 | int(11) | YES |  |
| item4 | int(11) | YES |  |
| item5 | int(11) | YES |  |
| item6 | int(11) | YES |  |
| item7 | int(11) | YES |  |
| item8 | int(11) | YES |  |
| item9 | int(11) | YES |  |
| item10 | int(11) | YES |  |

|  |  |  |  |
| --- | --- | --- | --- |
| ranking table | | | |
| Field | Type | Null | Key |
| id | varchar(50) | NO | 외부키 (user) |
| score | int(11) | YES |  |
| item1 | int(11) | YES |  |
| item2 | int(11) | YES |  |
| item3 | int(11) | YES |  |
| item4 | int(11) | YES |  |

|  |  |  |
| --- | --- | --- |
| common.h | [돌아가기](#moc) | 5~6 PAGE |

// 컴파일러에게 해당 헤더 파일이 한번만 빌드되도록 하게 한다.

#pragma once

// bool 타입을 흉내내는 변수에 사용하기 위한 매크로 상수들

#define TRUE 1

#define SUCCESS 1

#define FALSE 0

#define FAIL 0

// set\_color 함수에서 사용하기 위한 매크로 상수들

#define WHITE 15

#define RED 12

#define YELLOW 14

#define BLUE 9

// 키보드 입력을 구분하기 위한 매크로 상수들

#define UP 72

#define DOWN 80

#define ENTER 13

#define LEFT 75

#define RIGHT 77

#define BACKSPACE 8

#define ESC 27

// DB에서 공통적으로 사용하는 매크로 상수들

#define DB\_HOST "127.0.0.1"

#define DB\_USER "root"

#define DB\_PASS "abc123"

#define DB\_NAME "game\_db"

#define CHOP(x) x[strlen(x) - 1] = ' '

// 로그인 시점부터 프로그램이 종료될때까지 끝까지 살아있는 구조체

// 로그인할때 데이터를 입력받아서 데이터를 갱신하는 과정에서 기준점으로 사용함

// 구조체를 좀더 편하게 사용하기 위해 typedef를 사용

typedef struct User

{

    char id[20]; // 유저 아이디

    int point; // 유저 포인트

    int score; // 유저 최고 점수

    int item[10]; // 유저가 보유한 아이템

} USER;

// 프로그램 전체에서 공통적으로 사용하는 기능들

void cursor(int); // 커서를 지우거나 보여주는 역할을 함

void gotoxy(int, int); // 커맨드창에서 커서의 위치 변경을 위해 사용

void set\_color(unsigned short); // 출력 색상 변경을 위해 사용

int move(int, int, int, int); // 키보드를 입력받아 x,y 값을 변경해줌

// 각각의 화면

void game(void); // 게임이 돌아가는 화면

void shop(void); // 상점 화면

void ranking(int); // 랭킹 화면

int menu(int); // 메인화면의 메뉴

int login(int); // 회원가입, 로그인 화면

// UI 출력

void draw\_content(int); // 프로그램 전체적으로 UI를 그려줌

void shop\_preview(int); // 상점의 아이템 설명을 출력해줌

void draw\_owned\_item(void); // 유저가 이미 보유중인 항목을 출력해줌

void draw\_error(int); // DB 에러 화면 UI를 출력해줌

// DB 관련

void shop\_db(void); // 상점이용시 데이터 갱신을 위해 사용

int login\_db(int); // 가입, 로그인 이후 데이터 생성, 불러오기를 담당

void ranking\_db(int); // 랭킹 화면에 뿌려줄 데이터를 불러옴

// 전역 변수

extern USER user; // user 구조체 변수는 프로그램이 종료되기 직전까지 살아있어야 하고 여기저기서 사용하기 때문에 부득이하게 전역변수로 구현

|  |  |  |
| --- | --- | --- |
| cursor.c | [돌아가기](#moc) | 7 PAGE |

// 커서를 다루는 SetConsoleCursorPosition 함수가 포함되어 있다.

#include <windows.h>

// 함수 호출시 0 또는 1을 전달받아 커서를 보여주거나 가려준다.

void cursor(int n)

{

// CONSOLE\_CURSOR\_INFO라는 구조체를 선언

    CONSOLE\_CURSOR\_INFO ConsoleCursor;

// consoleCursor 구조체 안의 bVisible 값을 매개변수로 전달받은 n을 대입, TRUE는 보임, FALSE는 숨김이다.

    ConsoleCursor.bVisible = n;

// 커서의 굵기를 1~100으로 조절할 수 있다.

    ConsoleCursor.dwSize = 1;

// SetConsoleCursorPositoin 함수에, 콘솔의 출력 핸들을 주고, ConsoleCursor 구조체의 주소를 넘겨주어 동작을 한다.

    SetConsoleCursorInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &ConsoleCursor);

}

|  |  |  |
| --- | --- | --- |
| gotoxy.c | [돌아가기](#moc) | 8 PAGE |

#include <windows.h> // 커서를 다루는 SetConsoleCursorPosition 함수가 포함되어 있다.

// X 좌표, Y 좌표 값을 입력받아 커서의 위치를 변경시켜 준다.

void gotoxy(int x, int y)

{

 // COORD는 X 값과 Y값을 멤버로 가지는 구조체 타입이라고 한다.

   // 단순하게 X 값과 Y값을 저장한다고 생각하면 된다고 한다.

    COORD pos = { x,y };

    SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), pos);

// GetStdHandle 함수는 실제 핸들을 반환하는 함수이며 그안에 파라미터로 종류를 정해주면 핸들을 리턴한다.

// 운영체제는 자신의 리소스를 안전하게 관리하기 위해 주소를 사용하는 포인터 대신에 ‘핸들’ 이라는 개념을 사용하는데, ‘핸들’ 은 운영체제 내부에 있는 어떤 리소스의 주소를 정수로 치환한 값이라고 한다.

// STD\_OUTPUT\_HANDLE은 정의를 보면 (unsigned long)-11로 정의되어 있다.

// 즉 GetStdHandle(STD\_OUTPUT\_HANDLE) 한문장은 표준콘솔출력의 핸들을 반환해주는것이다.

// 정리하자면 SetConsoleCursorPositoin 함수에, 콘솔의 출력 핸들을 주고, 위치(pos)를 정해주면 커서가 이동한다는 것이다.

}

|  |  |  |
| --- | --- | --- |
| move.c | [돌아가기](#moc) | 9~10 PAGE |

// 표준 라이브러리 함수들을 포함하는 헤더, prinf 사용

#include <stdio.h>

// 키보드 처리 관련 매크로 상수들을 사용하기 위해 포함

#include "common.h"

// 화면 번호, 사용자가 입력한 키 값, X 좌표, Y좌표를 전달받아 좌표를 변경해준다.

// 화면에 따라 변경되는 좌표가 다르다.

int move(int screen\_num, int key, int POS\_X, int POS\_Y)

{

// 화면 번호를 구분한다.

    switch (screen\_num)

    {

    // 메인화면, 위 아래 동작만 있음

    case 0:

        switch (key)

        {

// 매크로 상수의 UP

        case UP:

// if문보다 보기 편해서 삼항 연산자 사용

// 1. 커서가 이미 첫번째 항목을 가리키고 있다면 값을 변경하지 않음

// 2. N번째 항목이라면 Y 좌표를 2 빼준다.

// 2-1. 2를 빼는 이유는 항목간의 간격 차이가 2라서

            POS\_Y = (POS\_Y <= 19) ? 19 : POS\_Y - 2;

            break; // switch문 탈출

// 매크로 상수의 DOWN

        case DOWN:

// 1. 커서가 이미 마지막 항목을 가리키고 있다면 값을 변경하지 않음

// 2. N번째 항목이라면 Y 좌표를 2 더해준다.

            POS\_Y = (POS\_Y >= 25) ? 25 : POS\_Y + 2;

            break;

        } // 메인화면은 Y 값만 움직이기 때문에 Y 좌표 리턴

        return POS\_Y;

    // PAUSE 화면, 좌 우 동작만 있음

    case 1:

        switch (key)

        {

  // PAUSE 화면은 예, 아니오 2개밖에 없다.

        case LEFT:

            POS\_X = (POS\_X > 23) ? 23 : POS\_X;

            break;

        case RIGHT:

            POS\_X = (POS\_X < 39) ? 39 : POS\_X;

            break;

        }

        return POS\_X;

    // 상점 화면, 위 아래 동작만 있음

    case 2:

        switch (key)

        {

// 좌표값만 다르고 메인화면과 작동 방식 동일

        case UP:

            POS\_Y = (POS\_Y <= 4) ? 4 : POS\_Y - 2;

            break;

        case DOWN:

            POS\_Y = (POS\_Y >= 22) ? 22 : POS\_Y + 2;

            break;

        }

        return POS\_Y;

    }

}

|  |  |  |
| --- | --- | --- |
| set\_color.c | [돌아가기](#moc) | 11 PAGE |

// 콘솔창의 색깔을 조정하는 SetConsoleTextAttribute 함수를 포함하고 있다.

#include <Windows.h>

// 값을 전달받아 콘솔창 출력 색깔을 변경한다.

void set\_color(unsigned short text)

{

// SetConsoleTextAttribute 함수에 콘솔의 출력 핸들을 주고, 넘겨받은 text 값을 주면 콘솔창 출력 색깔이 변한다.

    SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), text);

}

|  |  |  |
| --- | --- | --- |
| draw\_error.c | [돌아가기](#moc) | 12 PAGE |

// printf, puts 사용

#include <stdio.h>

// system 사용

#include <stdlib.h>

// gotoxy, set\_color 사용을 위해 포함

#include "common.h"

// 에러 값을 전달받아 상황에 맞게 출력해준다.

void draw\_error(int error\_code)

{

// 화면을 지우고 에러 메시지를 출력해준다.

    system("cls");

    set\_color(RED); // 색상 변경

    gotoxy(22, 10);

    puts("※ DB 접속 불가");

    gotoxy(22, 12);

    puts("프로그램 종료후 코드를 다시 확인 하십시오.");

    set\_color(WHITE);

    gotoxy(22, 14);

    switch (error\_code) // 에러 상황에 맞는 문구 출력

    {

    case 0:

        puts("DB 접속 에러 코드");

        break;

    case 1:

        puts("DB TABLE 접속 에러 코드");

        break;

    case 2  :

        puts("로그인 조회 쿼리문 에러 코드, 다른 아이디를 사용하십시오");

        break;

    }

// \_db.c 파일로 돌아갔을 때 다음 출력을 위해 빈공간으로 만들어줌

    gotoxy(22, 16);

    printf("                                                             ");

    gotoxy(22, 16); // 다음 출력의 위치를 지정

}

|  |  |  |
| --- | --- | --- |
| draw\_content.c | [돌아가기](#moc) | 13~25 PAGE |

#include <stdio.h> // printf, puts 사용

#include <stdlib.h> // system 사용

#include "common.h" // gotoxy, set\_color 사용

// 화면 번호 값을 전달받아 상황에 맞는 출력을 해줌

void draw\_content(int screen\_num)

{

    switch (screen\_num)

    {

    // 메인화면 - 도트 출력

    case 0:

        system("cls");

        for (int i = 0; i < 15; i++)

        {

            for (int j = 0; j < 100; j += 2)

            {

                gotoxy(j, i);

                printf("ㆍ");

            }

        }

        break;

    // 메인화면 - 1942 숫자 출력

    case 1:

    {

        // 1

        gotoxy(24, 3);

        printf("●●");

        gotoxy(22, 4);

        printf("●●●");

        gotoxy(20, 5);

        printf("●●●●");

        gotoxy(24, 6);

        printf("●●");

        gotoxy(24, 7);

        printf("●●");

        gotoxy(24, 8);

        printf("●●");

        gotoxy(24, 9);

        printf("●●");

        gotoxy(20, 10);

        printf("●●●●●●");

        gotoxy(20, 11);

        printf("●●●●●●");

        // 9

        gotoxy(38, 3);

        printf("●●●");

        gotoxy(36, 4);

        printf("●");

        gotoxy(44, 4);

        printf("●");

        gotoxy(36, 5);

        printf("●");

        gotoxy(44, 5);

        printf("●");

        gotoxy(36, 6);

        printf("●");

        gotoxy(44, 6);

        printf("●");

        gotoxy(38, 7);

        printf("●●●●");

        gotoxy(42, 8);

        printf("●●");

        gotoxy(42, 9);

        printf("●●");

        gotoxy(42, 10);

        printf("●●");

        gotoxy(42, 11);

        printf("●●");

        // 4

        gotoxy(60, 3);

        printf("●");

        gotoxy(58, 4);

        printf("●●");

        gotoxy(56, 5);

        printf("●");

        gotoxy(60, 5);

        printf("●");

        gotoxy(54, 6);

        printf("●");

        gotoxy(60, 6);

        printf("●");

        gotoxy(52, 7);

        printf("●");

        gotoxy(60, 7);

        printf("●");

        gotoxy(50, 8);

        printf("●●●●●●●");

        gotoxy(60, 9);

        printf("●");

        gotoxy(60, 10);

        printf("●");

        gotoxy(60, 11);

        printf("●");

        // 2

        gotoxy(70, 3);

        printf("●●●●");

        gotoxy(68, 4);

        printf("●");

        gotoxy(78, 4);

        printf("●");

        gotoxy(68, 5);

        printf("●");

        gotoxy(78, 5);

        printf("●");

        gotoxy(76, 6);

        printf("●");

        gotoxy(74, 7);

        printf("●");

        gotoxy(72, 8);

        printf("●");

        gotoxy(70, 9);

        printf("●");

        gotoxy(68, 10);

        printf("●");

        gotoxy(68, 11);

        printf("●●●●●●");

        }

        break;

    // 메인화면 - 조작키 프레임, 텍스트 출력

    case 2:

    {

        gotoxy(18, 17);

        printf("┏━━━━━━━━━━━━━━━━━━━━━━━━━━┓");

        gotoxy(18, 18);

        printf("┃                          ┃");

        gotoxy(18, 19);

        printf("┃                          ┃");

        gotoxy(18, 20);

        printf("┃                          ┃");

        gotoxy(18, 21);

        printf("┃                          ┃");

        gotoxy(18, 22);

        printf("┃                          ┃");

        gotoxy(18, 23);

        printf("┃                          ┃");

        gotoxy(18, 24);

        printf("┃                          ┃");

        gotoxy(18, 25);

        printf("┃                          ┃");

        gotoxy(18, 26);

        printf("┃                          ┃");

        gotoxy(18, 27);

        printf("┗━━━━━━━━━━━━━━━━━━━━━━━━━━┛");

        gotoxy(21, 18);

        printf("메인화면 조작키");

        gotoxy(18, 19);

        printf("┣━━━━━━━━━━━━━━━━━━━━━━━━━━┫");

        gotoxy(21, 21);

        printf("위   | 키보드 방향키 ↑");

        gotoxy(21, 23);

        printf("아래 | 키보드 방향키 ↓");

        gotoxy(21, 25);

        printf("확인 | 키보드 엔터키");

    }

        break;

    // 메인화면 - 로그인전 텍스트 출력

    case 3:

    {

        gotoxy(56, 19);

        printf("로그인");

        gotoxy(56, 21);

        printf("회원가입");

        gotoxy(56, 23);

        printf("랭킹화면");

        gotoxy(56, 25);

        printf("게임종료");

    }

        break;

    // 메인화면 - 로그인후 텍스트 출력

    case 4:

    {

        gotoxy(56, 19);

        printf("게임시작");

        gotoxy(56, 21);

        printf("상점");

        gotoxy(56, 23);

        printf("랭킹확인");

        gotoxy(56, 25);

        printf("게임종료");

        gotoxy(0, 29);

        printf("로그인 유저: %s", user.id);

    }

        break;

    // 로그인 화면 - 화면 지우기를 위한 공백 출력

    case 5:

    {

        gotoxy(34, 18);

        printf("                                 ");

        gotoxy(34, 20);

        printf("                                 ");

        gotoxy(34, 22);

        printf("                                 ");

        gotoxy(34, 24);

        printf("                                 ");

        cursor(1);

    }

        break;

    // 랭킹화면 - 비로그인 상태 프레임 출력

    case 6:

    {

        gotoxy(2, 1);

        printf("┏━━━━━┳━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓");

        for (int i = 2; i < 27; i++)

        {

            gotoxy(2, i);

            printf("┃");

            gotoxy(8, i);

            printf("┃");

            gotoxy(30, i);

            printf("┃");

            gotoxy(50, i);

            printf("┃");

            gotoxy(96, i);

            printf("┃");

        }

        for (int i = 3; i < 27; i += 2)

        {

            gotoxy(3, i);

            printf("━━━━━╋");

            gotoxy(9, i);

            printf("━━━━━━━━━━━━━━━━━━━━━╋");

            gotoxy(31, i);

            printf("━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━");

        }

        gotoxy(2, 27);

        printf("┗━━━━━┻━━━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛");

    }

        break;

    // 랭킹화면 - 비로그인 상태 텍스트 출력

    case 7:

    {

        gotoxy(4, 2);

        printf("순위");

        int colum = 4;

        for (int i = 1; i < 13; i++)

        {

            gotoxy(5, colum);

            printf("%2d", i);

            colum += 2;

        }

        gotoxy(18, 2);

        printf("이름");

        gotoxy(39, 2);

        printf("점수");

        gotoxy(68, 2);

        printf("사용 아이템");

        gotoxy(74, 28);

        printf("랭킹화면 나가기: ESC 키");

    }

        break;

    // 랭킹화면 - 로그인 상태 프레임 출력

    case 8:

    {

        gotoxy(2, 1);

        printf("┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓");

        for (int i = 2; i < 9; i++)

        {

            gotoxy(2, i);

            printf("┃");

            gotoxy(96, i);

            printf("┃");

        }

        for (int i = 2; i < 9; i++)

        {

            gotoxy(30, i);

            printf("┃");

        }

        for (int i = 9; i < 27; i++)

        {

            gotoxy(2, i);

            printf("┃");

            gotoxy(8, i);

            printf("┃");

            gotoxy(30, i);

            printf("┃");

            gotoxy(50, i);

            printf("┃");

            gotoxy(96, i);

            printf("┃");

        }

        gotoxy(2, 9);

        printf("┣━━━━━┳━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫");

        for (int i = 11; i < 27; i += 2)

        {

            gotoxy(3, i);

            printf("━━━━━╋");

            gotoxy(9, i);

            printf("━━━━━━━━━━━━━━━━━━━━━╋");

            gotoxy(31, i);

            printf("━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━");

        }

        gotoxy(2, 27);

        printf("┗━━━━━┻━━━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛");

    }

        break;

    // 랭킹화면 - 로그인 상태 텍스트 출력

    case 9:

    {

        // 유저 정보

        gotoxy(5, 3);

        printf("● 전투기: ");

        gotoxy(5, 5);

        printf("캐논 모양: ");

        gotoxy(5, 7);

        printf("고유 색상: ");

        gotoxy(33, 3);

        printf("이름: ");

        gotoxy(33, 5);

        printf("포인트");

        gotoxy(64, 3);

        printf("최고점수");

        gotoxy(33, 7);

        printf("사용한 아이템");

        // 순위 리스트

        gotoxy(4, 10);

        printf("순위");

        int colum = 12;

        for (int i = 1; i < 9; i++)

        {

            gotoxy(5, colum);

            printf("%d", i);

            colum += 2;

        }

        gotoxy(18, 10);

        printf("이름");

        gotoxy(39, 10);

        printf("점수");

        gotoxy(68, 10);

        printf("사용 아이템");

        gotoxy(74, 28);

        printf("랭킹화면 나가기: ESC 키");

    }

        break;

    // 상점화면 - 프레임 출력

    case 10:

    {

        gotoxy(2, 1);

        printf("┏━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┳━━━━━━━━━┳━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓");

        for (int i = 2; i < 24; i++)

        {

            gotoxy(2, i);

            printf("┃");

            gotoxy(22, i);

            printf("┃");

            gotoxy(36, i);

            printf("┃");

            gotoxy(46, i);

            printf("┃");

            gotoxy(54, i);

            printf("┃");

            gotoxy(96, i);

            printf("┃");

        }

        for (int i = 3; i < 24; i += 2)

        {

            gotoxy(2, i);

            printf("┣━━━━━━━━━━━━━━━━━━━");

            gotoxy(22, i);

            printf("╋━━━━━━━━━━━━━");

            gotoxy(36, i);

            printf("╋━━━━━━━━━");

            gotoxy(46, i);

            printf("╋━━━━━━━");

            gotoxy(54, i);

            printf("┫");

        }

        gotoxy(36, 23);

        printf("┻━━━━━━━━━");

        gotoxy(46, 23);

        printf("┻━━━━━━━");

        gotoxy(2, 24);

        printf("┃");

        gotoxy(22, 24);

        printf("┃");

        gotoxy(54, 24);

        printf("┃");

        gotoxy(96, 24);

        printf("┃");

        gotoxy(2, 25);

        printf("┗━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛");

    }

        break;

    // 상점화면 - 텍스트 출력

    case 11:

    {

        gotoxy(3, 26);

        printf("치장 아이템은 하나씩만 소지 가능 합니다. / 기존 치장 아이템은 60%% 가격에 자동 판매 됩니다.");

        gotoxy(3, 28);

        printf("목록 이동: 방향키 ↑ ↓  ||  구매: 엔터키                                  상점 나가기: ESC 키");

        gotoxy(11, 2);

        printf("항목");

        gotoxy(28, 2);

        printf("가격");

        gotoxy(39, 2);

        printf("보유수");

        gotoxy(49, 2);

        printf("선택");

        gotoxy(5, 4);

        printf("생명 추가");

        gotoxy(5, 6);

        printf("HP 회복");

        gotoxy(5, 8);

        printf("추가 점수");

        gotoxy(5, 10);

        printf("빨강 페인트");

        gotoxy(5, 12);

        printf("파랑 페인트");

        gotoxy(5, 14);

        printf("노랑 페인트");

        gotoxy(5, 16);

        printf("빨간 캐논");

        gotoxy(5, 18);

        printf("파란 캐논");

        gotoxy(5, 20);

        printf("노란 캐논");

        gotoxy(5, 22);

        printf("랭킹창 꾸미기");

        gotoxy(5, 24);

        printf("보유 포인트");

    }

        break;

    case 12:

    {

        gotoxy(29, 4);

        printf("1000 pt");

        gotoxy(30, 6);

        printf("500 pt");

        gotoxy(29, 8);

        printf("2000 pt");

        gotoxy(29, 10);

        printf("3500 pt");

        gotoxy(29, 12);

        printf("4500 pt");

        gotoxy(29, 14);

        printf("5500 pt");

        gotoxy(29, 16);

        printf("3500 pt");

        gotoxy(29, 18);

        printf("4500 pt");

        gotoxy(29, 20);

        printf("5500 pt");

        gotoxy(27, 22);

        printf("100000 pt");

    }

        break;

    case 13:

    {

        for (int i = 4; i < 24; i++)

        {

            gotoxy(59, i);

            printf("                                     ");

        }

        gotoxy(59, 5);

        printf("구매 불가");

        set\_color(12);

        gotoxy(59, 8);

        printf("포인트가 부족합니다.");

        set\_color(15);

    }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| main.c | [돌아가기](#moc) | 26~27 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // system 사용

#include "common.h" // gotoxy, move등 공통적인 기능 사용

// 프로그램 전체적으로 사용하는 구조체 변수

// 이곳저곳에서 쓰이며 프로그램이 종료될때까지 살아있어야 하기에 전역 사용

USER user = { NULL };

int main(void)

{

    // 커맨드창 설정

    system("mode con: cols=100 lines=30");

    system("title 1942 프로젝트 - 1705095 이창민");

    cursor(0); // 0: 커서 가림, 1: 커서 보이기

    // main에서만 사용하는 변수

// 로그인 유무를 검사하기 위해 bool 타입을 흉내내는 변수

    int is\_logined = FALSE;

// 유저가 선택한 메뉴 Y값을 저장하는 변수

    int choose\_menu = 19;

    // 유저가 게임종료를 선택할때까지 무한반복

    while (choose\_menu != 24)

    {

        system("cls");

        draw\_content(0); // 도트 출력

        draw\_content(1); // 1942 그림 출력

        draw\_content(2); // 조작키 설명 프레임, 텍스트 출력

        // draw\_content -> 3 비로그인 | 4 로그인

// 로그인을 했으면 1이고 안했으면 0이다. 보기 좋게 삼항으로 처리

        is\_logined % 2 == 1 ? draw\_content(4, user) : draw\_content(3);

        choose\_menu = menu(choose\_menu); // 키보드를 입력할때마다 Y값 갱신

// 아래 코드는 전부 로그인 유무에 따라 각각 화면을 호출해주는 코드

        if (choose\_menu == 19)

        {

            is\_logined % 2 == 0 ? is\_logined = login(1) : game();

            system("mode con: cols=100 lines=30");

        }

        else if (choose\_menu == 21)

        {

            is\_logined % 2 == 0 ? is\_logined = login(0) : shop(user);

        }

        else if (choose\_menu == 23)

        {

            is\_logined % 2 == 0 ? ranking(0, user) : ranking(1);

        }

        else

        {

            break;

        }

        choose\_menu = 19;

    }

    return 0;

}

|  |  |  |
| --- | --- | --- |
| menu.c | [돌아가기](#moc) | 28 PAGE |

#include <stdio.h> // pust 사용

#include <stdlib.h> // getch 사용

#include "common.h" // gotoxy 사용

#define MENU\_POS\_X 69 // 메뉴 항목의 X값은 고정이라서 상수로 선언

// 유저가 엔터키를 입력하고 나면 변경된 위치값을 돌려준다.

int menu(int choose\_menu)

{

// 유저의 입력을 받기 위한 변수

    int user\_input = 0;

// 엔터를 입력할때까지 반복

    while (user\_input != ENTER)

    {

// 현재 선택중인 메뉴를 보여준다.

        gotoxy(MENU\_POS\_X, choose\_menu);

        puts("ㆍㆍ ◀");

        user\_input = getch(); // 키 입력을 받는다.

      // y값을 변경하기 전에 자리를 지워준다.

gotoxy(MENU\_POS\_X, choose\_menu);

        puts("       ");

        // 화면번호, 입력받은 키 값, X 값, Y 값을 넘겨주어 선택한 메뉴의 Y값을 받는다.

        choose\_menu = move(0, user\_input, MENU\_POS\_X, choose\_menu);

    }

// 선택한 메뉴의 Y값을 리턴해준다.

    return choose\_menu;

}

|  |  |  |
| --- | --- | --- |
| login.c | [돌아가기](#moc) | 29 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // system 사용

#include "common.h" // gotoxy 사용

int login(int is\_logined)

{

    system("cls");

    draw\_content(0); // 도트 출력

    draw\_content(1); // 1942 그림 출력

    int is\_checked = FALSE; // 로그인 성공 여부를 검사

    while (is\_checked == FALSE) // 성공할때까지 반복

    {

        switch (is\_logined)

        {

        // 회원가입

        case FALSE:

// 회원가입을 위해 login\_db 호출

// 매개변수 0은 회원가입, 1은 로그인

            is\_checked = login\_db(0);

            if (is\_checked != TRUE) // 회원가입에 성공했는지 검사

            {

                break; // 실패했다면 while문 탈출

            }

// 프로그램의 구조가 회원 가입을 할 때 데이터를 생성하고

            // 로그인을 할때 게임에 필요한 유저 정보를 입력 받기 때문에

            // 회원가입후 로그인하는 절차가 필요하여 break; 없음

            is\_logined = TRUE; // 로그인 성공

        case TRUE:

            is\_checked = FALSE; // 회원가입 이후 로그인 하는 경우를 위해 FALSE로 초기화

// 로그인 절차에 들어감

            is\_checked = login\_db(1);

            break;

        }

    }

    return 1; // 로그인에 성공할때까지 못 빠져나가서 1만 리턴함

}

|  |  |  |
| --- | --- | --- |
| login\_db.c | [돌아가기](#moc) | 30~34 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // system 사용

#include <string.h> // strcpy 사용

#include <mysql.h> // mysql 사용

#include "common.h" // gotoxy 사용

#pragma commnet(lib, "libmysql.lib") // 프로젝트 설정에서 설정했으나 혹시나 모를 오류를 방지하기 위해 적어둠

int login\_db(int sign)

{

    MYSQL\* connection = NULL; // 연결 상태 저장

    MYSQL conn = { 0, }; // 연결을 위해 사용

    MYSQL\_RES\* result = NULL; // 결과를 받음

    MYSQL\_ROW row = { 0, }; // 행 단위로 넘어오는 값을 받기 위해 사용

    int query\_stat = 0; // 오류 검사를 위해 사용

    int is\_successed = FAIL; // db데이터를 받았는지 검사 기준

    int join\_success = FAIL; // 회원가입 성공했는지 검사 기준

    char id[20] = { 0, }; // 아이디를 담는 공간

    char passwd[20] = { 0, }; // 패스워드를 담는 공간

    char query[255] = { 0, }; // 쿼리문을 담는 공간

    mysql\_init(&conn); // 초기화

    connection = mysql\_real\_connect(&conn, DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME, 3306, (char\*)NULL, 0); // 연결

// 에러나는지 검사

    if (connection == NULL)

    {

// 에러가 발생하면 에러 UI 출력

        draw\_error(0);

        fprintf(stderr, "%s", mysql\_error(&conn));

        gotoxy(22, 17);

        system("pause");

// 사용자가 제어 불가능한 문제이기에 강제 종료

        exit(0);

    }

// 로그인인지 회원가입인지 나눔

    switch (sign)

    {

    // 회원가입

    case FALSE:

// 회원가입이 성공할때까지 반복

        while (join\_success == FAIL)

        {

// 회원가입 UI 출력

            draw\_content(5);

            gotoxy(34, 18);

            printf("회원가입");

            gotoxy(34, 20);

            printf("20자 이내에 영문으로 입력하십시오");

            gotoxy(34, 22);

            printf("아이디: ");

            fgets(id, 20, stdin);

            CHOP(id); // 엔터키 삭제

            gotoxy(34, 24);

            printf("비밀번호: ");

            fgets(passwd, 20, stdin);

            CHOP(passwd);

            cursor(0);

            // 입력받은 아이디, 비밀번호 DB 등록

            sprintf(query, "insert into user values " "('%s', '%s', '%d', '%d')", id, passwd, 0, 0);

// 쿼리를 날림

            query\_stat = mysql\_query(connection, query);

// 오류 검사

            if (query\_stat != 0)

            {

// 에러 UI 출력해줌

                draw\_error(2);

                fprintf(stderr, "%s", mysql\_error(&conn));

                Sleep(3500);

                system("cls");

                draw\_content(0); // 도트 출력

                draw\_content(1); // 1942 그림 출력

                continue; // 다시 입력하게 위로 올려보냄

            }

            // 입력받은 아이디를 기준으로 item\_list db에 초기값 등록

            sprintf(query, "insert into item\_list values " "('%s', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d')", id, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15);

            query\_stat = mysql\_query(connection, query);

            if (query\_stat != 0)

            {

                draw\_error(2);

                fprintf(stderr, "%s", mysql\_error(&conn));

                gotoxy(22, 17);

                system("pause");

                exit(0);

            }

            // 입력받은 아이디를 기준으로 ranking db에 초기값 등록

            sprintf(query, "insert into ranking values " "('%s', '%d', '%d', '%d', '%d', '%d')", id, 0, 0, 0, 0, 15);

            query\_stat = mysql\_query(connection, query);

            if (query\_stat != 0)

            {

                draw\_error(2);

                fprintf(stderr, "%s", mysql\_error(&conn));

                gotoxy(22, 17);

                system("pause");

                exit(0);

            }

            join\_success = TRUE; // 회원가입 여부를 TRUE로 변경

        }

        return SUCCESS; // login.c switch문에서 로그인 과정으로 보내기 위해 성공을 리턴

    // 로그인

    case TRUE:

// 로그인 UI 출력

        draw\_content(5);

        gotoxy(34, 18);

        printf("로그인");

        gotoxy(34, 20);

        printf("아이디: ");

        fgets(id, 20, stdin);

        CHOP(id);

        gotoxy(34, 22);

        printf("비밀번호: ");

        fgets(passwd, 20, stdin);

        CHOP(passwd);

        cursor(0);

// 입력받은 아이디 패스워드로 조회

        sprintf(query, "select \* from user where id = '%s' and passwd = '%s'", id, passwd);

        query\_stat = mysql\_query(connection, query);

        if (query\_stat != 0)

        {

// 오류가 발생하면 UI 출력

            draw\_error(2);

            fprintf(stderr, "%s", mysql\_error(&conn));

            Sleep(3500);

            system("cls");

            draw\_content(0); // 도트 출력

            draw\_content(1); // 1942 그림 출력

            return FAIL; // 실패를 리턴

        }

// userㅌ 테이블에서 값을 받아옴

        result = mysql\_store\_result(connection);

// 행 단위로 나눠서 받음

        while ((row = mysql\_fetch\_row(result)) != NULL)

        {

// 입력받은 아이디는 user 구조체 변수에 복사

            // 점수, 포인트를 불러와서 user 구조체 변수에 저장

            strcpy(user.id, id);

            user.score = atoi(row[2]);

            user.point = atoi(row[3]);

            is\_successed = TRUE; // 성공했다고 알림

        }

        mysql\_free\_result(result); // 보유 아이템 항목을 불러와야하기 때문에 result를 비워줌

        if (is\_successed == TRUE)

        {

            // 소지 아이템 불러오기위해 조회

            sprintf(query, "select \* from item\_list where id = '%s'", id);

            query\_stat = mysql\_query(connection, query);

            if (query\_stat != 0)

            {

                draw\_error(2);

                fprintf(stderr, "%s", mysql\_error(&conn));

                Sleep(3500);

                return FAIL;

            }

// 마찬가지로 행단위로 입력받아 저장

            result = mysql\_store\_result(connection);

            while ((row = mysql\_fetch\_row(result)) != NULL)

            {

                // 불러오기 성공

                for (int i = 0; i < 10; i++)

                {

                    user.item[i] = atoi(row[i+1]);

                }

                is\_successed = TRUE;

            }

            mysql\_free\_result(result);

            mysql\_close(connection); // mysql 연결 종료

            return SUCCESS; // 성공을 리턴

        }

        break;

    }

    mysql\_close(connection);

    return FAIL; // 로그인 실패를 리턴

}

|  |  |  |
| --- | --- | --- |
| ranking.c | [돌아가기](#moc) | 35 PAGE |

#include <stdio.h> // 표준 헤더

#include <stdlib.h> // system 사용

#include "common.h" // 다른 함수들 사용

void ranking(int is\_logined)

{

    system("cls");

    int key = 0; // 키를 입력받기 위해 사용

    switch (is\_logined)

    {

    // 비로그인

    case FALSE:

// 비로그인용 랭킹 UI 출력

        draw\_content(6);

        draw\_content(7);

        ranking\_db(0); // 랭킹 db에 접근

        while (key != ESC)

        {

            key = getch(); // ESC키를 입력하면 탈출

        }

        break;

    // 로그인

    case TRUE:

// 로그인용 UI 출력

        draw\_content(8);

        draw\_content(9);

        ranking\_db(1); // 랭킹 db에 접근

        while (key != ESC)

        {

            key = getch(); // ESC키를 입력하면 탈출

        }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| ranking\_db.c | [돌아가기](#moc) | 36~45 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // system 사용

#include <string.h> // strlen 사용

#include <mysql.h> // mysql 사용

#include "common.h" // 공통 기능들 사용

#pragma commnet(lib, "libmysql.lib") // 프로젝트 설정에서 설정했으나 혹시나 모를 오류를 방지하기 위해 적어둠

// 전달받은 값에 따라 출력하는게 다름

void ranking\_db(int is\_logined)

{

    MYSQL\* connection = NULL; // 연결 상태 저장

    MYSQL conn = { 0, }; // 연결을 위해 사용

    MYSQL\_RES\* result = NULL; // 결과를 받음

    MYSQL\_ROW row = { 0, }; // 행 단위로 값을 받음

    int query\_stat = 0; // 오류 검사를 위해 사용

    char query[255] = { 0, }; // 쿼리문을 담는 공간

    int POS\_Y = 4; // Y 좌표 값

    int count = 1; // 등수 출력

    mysql\_init(&conn); // 초기화

    connection = mysql\_real\_connect(&conn, DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME, 3306, (char\*)NULL, 0); // 연결

// 오류검사를 해서 오류가 발생하면 UI 출력해주고 프로그램 종료

    if (connection == NULL)

    {

        draw\_error(0);

        fprintf(stderr, "%s", mysql\_error(&conn));

        gotoxy(22, 17);

        system("pause");

        exit(0);

    }

    switch (is\_logined)

    {

    // 비로그인

    case FALSE:

// ranking 테이블의 데이터들을 score 칼럼을 기준으로 순서대로 조회

        mysql\_query(connection, "select \* from ranking order by score desc");

// 결과를 받아서 행 단위로 나눈다.

        result = mysql\_store\_result(connection);

        while ((row = mysql\_fetch\_row(result)) != NULL)

        {

            // 넘어오는 정수값은 문자로 넘어오기 때문에 정수로 바꿔주고 랭킹 테이블의 마지막 아이템은 랭킹창 유저 이름의 색깔을 저장하기 때문에 set\_color 사용 가능

            set\_color(atoi(row[5]));

// 좌표로 이동해서 출력

            gotoxy(10, POS\_Y);

            int indent = (20 - strlen(row[0])) / 2; // 왼쪽 여백 구하기

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

            {

                putchar(' '); // 왼쪽 여백 채우기

            }

            printf("%s", row[0]);

            set\_color(15);

            // 점수 출력

            gotoxy(39, POS\_Y);

            printf("%11s", row[1]);

// 최고 점수를 달성하는데 사용한 아이템을 출력한다.

            if (atoi(row[2]) != 0)

            {

                gotoxy(53, POS\_Y);

                printf("생명추가");

            }

            if (atoi(row[2]) != 0 && atoi(row[3]) != 0)

            {

                gotoxy(61, POS\_Y);

                printf(", 체력회복");

            }

            else if (atoi(row[2]) == 0 && atoi(row[3]) != 0)

            {

                gotoxy(53, POS\_Y);

                printf("체력회복");

            }

            if ((atoi(row[2]) != 0 && atoi(row[3]) != 0) && atoi(row[4]) != 0)

            {

                gotoxy(72, POS\_Y);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) != 0) && atoi(row[4]) != 0)

            {

                gotoxy(62, POS\_Y);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) != 0 && atoi(row[3]) == 0) && atoi(row[4]) != 0)

            {

                gotoxy(61, POS\_Y);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) == 0) && atoi(row[4]) != 0)

            {

                gotoxy(53, POS\_Y);

                printf("추가점수");

            }

            if (count == 12)

            {

                break;

            }

            POS\_Y += 2;

            count++;

        }

        mysql\_free\_result(result);

        break;

    // 로그인

    case TRUE:

// 로그인 랭킹창은 유저의 정보도 출력한다.

        // 전투기 모양, 캐논, 고유색상 출력

        sprintf(query, "select \* from item\_list where id = '%s'", user.id);

        query\_stat = mysql\_query(connection, query);

        if (query\_stat != 0)

        {

            draw\_error(2);

            fprintf(stderr, "%s", mysql\_error(&conn));

            gotoxy(22, 17);

            system("pause");

            exit(0);

        }

// item\_list 테이블을 조회하여 출력

        mysql\_query(connection, "select \* from item\_list where id = '%s'", user.id);

        result = mysql\_store\_result(connection);

        while ((row = mysql\_fetch\_row(result)) != NULL)

        {

// 아이템 보유 여부에 따라 전투기 모양과 색상이 다름

            if (atoi(row[4]) == 1)

            {

                set\_color(RED);

                gotoxy(16, 3);

                printf("<-\*->");

                set\_color(WHITE);

            }

            else if (atoi(row[5]) == 1)

            {

                set\_color(BLUE);

                gotoxy(16, 3);

                printf("{:\*:}");

                set\_color(WHITE);

            }

            else if (atoi(row[6]) == 1)

            {

                set\_color(YELLOW);

                gotoxy(16, 3);

                printf("H:\*:H");

                set\_color(WHITE);

            }

            else // default 상태

            {

                gotoxy(16, 3);

                printf("[-\*-]");

            }

// 아이템 보유 여부에 따라 캐논(총알)의 모양과 색상이 다름

            if (atoi(row[7]) == 1)

            {

                set\_color(RED);

                gotoxy(16, 5);

                printf("γ ψ Ψ");

                set\_color(WHITE);

            }

            else if (atoi(row[8]) == 1)

            {

                set\_color(BLUE);

                gotoxy(16, 5);

                printf("＋ ± ÷");

                set\_color(WHITE);

            }

            else if (atoi(row[9]) == 1)

            {

                set\_color(YELLOW);

                gotoxy(16, 5);

                printf("★ ♠ ♣");

                set\_color(WHITE);

            }

            else // default 상태

            {

                gotoxy(16, 5);

                printf("ⅰ ⅱ ⅲ");

            }

// 유저의 고유 색상을 출력

            set\_color(atoi(row[10]));

            gotoxy(16, 7);

            switch (atoi(row[10]))

            {

            case 1:

                printf("BLUE");

                break;

            case 2:

                printf("GREEN");

                break;

            case 3:

                printf("CYAN");

                break;

            case 4:

                printf("RED");

                break;

            case 5:

                printf("MAGENTA");

                break;

            case 6:

                printf("BROWN");

                break;

            case 7:

                printf("LIGHT GRAY");

                break;

            case 8:

                printf("DARK GRAY");

                break;

            case 9:

                printf("LIGHT BLUE");

                break;

            case 10:

                printf("LIGHT GREEN");

                break;

            case 11:

                printf("LIGHT CYAN");

                break;

            case 12:

                printf("LIGHT RED");

                break;

            case 13:

                printf("LIGHT MAGENTA");

                break;

            case 14:

                printf("YELLOW");

                break;

            default:

                printf("DEAFULT");

                break;

            }

            set\_color(WHITE);

        }

        mysql\_free\_result(result);

        // 이름, 포인트, 사용한 아이템, 최고점수

        sprintf(query, "select \* from ranking where id = '%s'", user.id);

        query\_stat = mysql\_query(connection, query);

        if (query\_stat != 0)

        {

            draw\_error(2);

            fprintf(stderr, "%s", mysql\_error(&conn));

            gotoxy(22, 17);

            system("pause");

            exit(0);

        }

// 랭킹 DB의 데이터 조회

        mysql\_query(connection, "select \* from ranking where id = '%s'", user.id);

        result = mysql\_store\_result(connection);

        while ((row = mysql\_fetch\_row(result)) != NULL)

        {

// 유저 이름, 점수, 포인트 출력

            set\_color(atoi(row[5]));

            gotoxy(39, 3);

            printf("%s", row[0]);

            set\_color(15);

            gotoxy(72, 3);

            printf(": %s", row[1]);

            gotoxy(39, 5);

            printf(": %d", user.point);

            // 사용한 아이템 출력

            if (atoi(row[2]) != 0)

            {

                gotoxy(46, 7);

                printf(": 생명추가");

            }

            if (atoi(row[2]) != 0 && atoi(row[3]) != 0)

            {

                gotoxy(56, 7);

                printf(", 체력회복");

            }

            else if (atoi(row[2]) == 0 && atoi(row[3]) != 0)

            {

                gotoxy(46, 7);

                printf(": 체력회복");

            }

            if ((atoi(row[2]) != 0 && atoi(row[3]) != 0) && atoi(row[4]) != 0)

            {

                gotoxy(67, 7);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) != 0) && atoi(row[4]) != 0)

            {

                gotoxy(57, 7);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) != 0 && atoi(row[3]) == 0) && atoi(row[4]) != 0)

            {

                gotoxy(56, 7);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) == 0) && atoi(row[4]) != 0)

            {

                gotoxy(46, 7);

                printf(": 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) == 0) && atoi(row[4]) == 0)

            {

                gotoxy(46, 7);

                printf(": 없음");

            }

        }

        mysql\_free\_result(result);

        // 순위 출력

        mysql\_query(connection, "select \* from ranking order by score desc");

        result = mysql\_store\_result(connection);

        POS\_Y = 12;

        while ((row = mysql\_fetch\_row(result)) != NULL)

        {

            // 이름 출력

            gotoxy(10, POS\_Y);

            int indent = (20 - strlen(row[0])) / 2; // 왼쪽 여백 구하기

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

            {

                putchar(' '); // 왼쪽 여백 채우기

            }

            set\_color(atoi(row[5]));

            printf("%s", row[0]);

            set\_color(15);

            // 점수 출력

            gotoxy(39, POS\_Y);

            printf("%11s", row[1]);

            // 사용한 아이템 출력

            if (atoi(row[2]) != 0)

            {

                gotoxy(53, POS\_Y);

                printf("생명추가");

            }

            if (atoi(row[2]) != 0 && atoi(row[3]) != 0)

            {

                gotoxy(61, POS\_Y);

                printf(", 체력회복");

            }

            else if (atoi(row[2]) == 0 && atoi(row[3]) != 0)

            {

                gotoxy(53, POS\_Y);

                printf("체력회복");

            }

            if ((atoi(row[2]) != 0 && atoi(row[3]) != 0) && atoi(row[4]) != 0)

            {

                gotoxy(72, POS\_Y);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) != 0) && atoi(row[4]) != 0)

            {

                gotoxy(62, POS\_Y);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) != 0 && atoi(row[3]) == 0) && atoi(row[4]) != 0)

            {

                gotoxy(61, POS\_Y);

                printf(", 추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) == 0) && atoi(row[4]) != 0)

            {

                gotoxy(53, POS\_Y);

                printf("추가점수");

            }

            else if ((atoi(row[2]) == 0 && atoi(row[3]) == 0) && atoi(row[4]) == 0)

            {

                gotoxy(53, POS\_Y);

                printf("없음");

            }

            if (count == 8)

            {

                break;

            }

            POS\_Y += 2;

            count++;

        }

        mysql\_free\_result(result); // 사용하고 나면 result 비워줌

        break;

    }

    mysql\_close(connection); // db 연결 종료

}

|  |  |  |
| --- | --- | --- |
| shop.c | [돌아가기](#moc) | 46~52 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // srand 사용

#include <time.h> // time 사용

#include <windows.h> // Sleep 사용

#include "common.h" // 공통 함수 사용

#define SHOP\_POS\_X 50 // 위아래로만 움직이기 때문에 X 좌표 고정

void shop(void)

{

    system("cls");

    draw\_content(10); // 상점화면 프레임 출력

    draw\_content(11); // 상점화면 아이템 리스트 출력

    draw\_content(12); // 아이템 가격 출력

    draw\_owned\_item(); // 유저 정보 (포인트, 보유 아이템 출력하는 함수 필요)

    shop\_preview(4); // 상점화면 아이템 리스트 설명 출력, 처음에는 1번 항목 출력

    // shop에서 사용하는 변수

    int list\_num = 4; // 선택한 아이템 구별

    int choose\_item = 0; // ESC를 입력받음

    int price = 0; // 구매를 위해 제품 가격을 저장

    // ESC 키를 입력할때까지 무한 반복

    while (choose\_item != ESC)

    {

// UI 출력

// 선택하고 있는 아이템에 따라 미리보기 안내문을 출력

        gotoxy(SHOP\_POS\_X, list\_num);

        puts("○");

        draw\_owned\_item(user);

        shop\_preview(list\_num);

        choose\_item = getch(); // 엔터키 입력

// 선택한 아이템에 따라 가격 지정

        if (choose\_item == ENTER)

        {

            switch (list\_num)

            {

            case 4:

                price = 1000;

                break;

            case 6:

                price = 500;

                break;

            case 8:

                price = 2000;

                break;

            case 10:

                price = 3500;

                break;

            case 12:

                price = 4500;

                break;

            case 14:

                price = 5500;

                break;

            case 16:

                price = 3500;

                break;

            case 18:

                price = 4500;

                break;

            case 20:

                price = 5500;

                break;

            case 22:

                price = 100000;

                break;

            }

        // 유저가 보유중인 포인트가 가격보다 큰지 검사

            if (user.point >= price)

            {

                // 기능 아이템

                if (list\_num < 10)

                {

                    switch (list\_num)

                    {

                    // 생명 추가

                    case 4:

// 생명 추가 아이템은 3개까지만 소지 가능

                        if (user.item[0] < 3)

                        {

                            user.point -= 1000; // 포인트 차감

                            user.item[0]++; // 구매 성공

                        }

                        break;

                    // HP 회복

                    case 6:

// HP 회복은 5개까지 소지 가능

                        if (user.item[1] < 5)

                        {

                            user.point -= 500; // 포인트 차감

                            user.item[1]++; // 구매 성공

                        }

                        break;

                    // 추가 점수 아이템은 무한정 소지 가능

                    case 8:

                        user.point -= 2000; // 포인트 차감

                        user.item[2]++; // 구매 성공

                        break;

                    }

                }

                // 치장 아이템

                else

                {

                    switch (list\_num)

                    {

                    // 빨강 페인트

// 치장 아이템의 경우 타입별 하나만 구매 가능하기 때문에 다른 항목을 가지고 있는지 검사해서 있다면 판매를 해버리고 구매를 한다.

                    case 10:

                        if (user.item[3] == 0) // 가지고 있지 않다면 구매

                        {

                            user.point -= 3500; // 포인트 차감

                            user.item[3]++; // 구매 성공

                            if (user.item[4] == 1) // 다른 치장 아이템 소지시 자동 판매

                            {

                                user.item[4] = 0; // 소지 개수 초기화

                                user.point += 2700; // 환불

                            }

                            else if (user.item[5] == 1) // 위와 같음

                            {

                                user.item[5] = 0;

                                user.point += 3300;

                            }

                        }

                        else

                        {

                            user.point += 1700;

                            user.item[3]--;

                        }

                        break;

                    // 파랑 페인트

// 빨강 페인트와 작동 방식 동일

                    case 12:

                        if (user.item[4] == 0)

                        {

                            user.point -= 4500;

                            user.item[4]++;

                            if (user.item[3] == 1)

                            {

                                user.item[3] = 0;

                                user.point += 2100;

                            }

                            else if (user.item[5] == 1)

                            {

                                user.item[5] = 0;

                                user.point += 3300;

                            }

                        }

                        else

                        {

                            user.point += 2700;

                            user.item[4]--;

                        }

                        break;

                    // 노랑 페인트

// 빨강 페인트와 작동 방식 동일

                    case 14:

                        if (user.item[5] == 0)

                        {

                            user.point -= 5500;

                            user.item[5]++;

                            if (user.item[3] == 1)

                            {

                                user.item[3] = 0;

                                user.point += 2100;

                            }

                            else if (user.item[4] == 1)

                            {

                                user.item[4] = 0;

                                user.point += 2700;

                            }

                        }

                        else

                        {

                            user.point += 3300;

                            user.item[5]--;

                        }

                        break;

                    // 빨간 캐논

// 빨강 페인트와 작동 방식 동일

                    case 16:

                        if (user.item[6] == 0)

                        {

                            user.point -= 3500;

                            user.item[6]++;

                            if (user.item[7] == 1)

                            {

                                user.item[7] = 0;

                                user.point += 2700;

                            }

                            else if (user.item[8] == 1)

                            {

                                user.item[8] = 0;

                                user.point += 3300;

                            }

                        }

                        else

                        {

                            user.point += 1700;

                            user.item[6]--;

                        }

                        break;

                    // 파란 캐논

// 빨강 페인트와 작동 방식 동일

                    case 18:

                        if (user.item[7] == 0)

                        {

                            user.point -= 4500;

                            user.item[7]++;

                            if (user.item[6] == 1)

                            {

                                user.item[6] = 0;

                                user.point += 2100;

                            }

                            else if (user.item[8] == 1)

                            {

                                user.item[8] = 0;

                                user.point += 3300;

                            }

                        }

                        else

                        {

                            user.point += 2700;

                            user.item[7]--;

                        }

                        break;

                    // 노란 캐논

// 빨강 페인트와 작동 방식 동일

                    case 20:

                        if (user.item[8] == 0)

                        {

                            user.point -= 5500;

                            user.item[8]++;

                            if (user.item[6] == 1)

                            {

                                user.item[6] = 0;

                                user.point += 2100;

                            }

                            else if (user.item[7] == 1)

                            {

                                user.item[7] = 0;

                                user.point += 2700;

                            }

                        }

                        else

                        {

                            user.point += 3300;

                            user.item[8]--;

                        }

                        break;

// 랭킹창의 유저 이름 색깔을 바꿔주는 아이템

                    case 22:

                        user.point -= 100000;

                        srand(time(NULL));

                        user.item[9] = rand() % 14 + 1; // 랜덤으로 색상 지정

                        break;

                    }

                }

            }

            else // 잔액 부족으로 구매불가

            {

                draw\_content(13); // 구매불가 알림

                Sleep(800);

            }

            choose\_item = 0;

        }

        else // 입력한 키가 엔터키가 아니면 항목 이동

        {

// 원래 위치는 지워준다.

            gotoxy(SHOP\_POS\_X, list\_num);

            puts("  ");

            // 화면번호, 입력받은 키 값, X 값, Y 값

            list\_num = move(2, choose\_item, SHOP\_POS\_X, list\_num);

        }

    }

    shop\_db(user); // db 저장

}

|  |  |  |
| --- | --- | --- |
| shop\_db.c | [돌아가기](#moc) | 53~54 PAGE |

#include <stdio.h> // 표준함수

#include <stdlib.h> // system 사용

#include <mysql.h> // mysql 사용

#include "common.h" // 공통적인 함수 사용

#pragma commnet(lib, "libmysql.lib") // 프로젝트 설정에서 설정했으나 혹시나 모를 오류를 방지하기 위해 사용

void shop\_db(void)

{

    MYSQL\* connection = NULL; // 연결 상태 저장

    MYSQL conn = { 0, }; // 연결을 위해 사용

    MYSQL\_RES\* result = NULL; // 결과를 반영

    MYSQL\_ROW row = { 0, }; // 행 단위로 값을 넘겨 받음

    int query\_stat = 0; // 오류 검사를 위해 사용

    char query[255] = { 0, }; // 쿼리문을 담는 공간

    mysql\_init(&conn); // 초기화

    connection = mysql\_real\_connect(&conn, DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME, 3306, (char\*)NULL, 0); // 연결

    if (connection == NULL) // 오류 검사

    {

// 오류 UI 출력

        draw\_error(0);

        fprintf(stderr, "%s", mysql\_error(&conn));

        gotoxy(22, 17);

        system("pause");

        exit(0); // 사용자가 대처 불가능한 에러는 강제 종료

    }

    // c언어에서 \를 쓰면 줄넘김을 해서 작성할 수 있음

// item\_list 테이블 데이터를 전부 갱신

    sprintf(query, "update item\_list set item1 = %d, item2 = %d, item3 = %d, \

        item4 = %d, item5 = %d, item6 = %d, item7 = %d, item8 = %d, item9 = %d, \

        item10 = %d where id = '%s'", user.item[0], user.item[1], user.item[2], \

        user.item[3], user.item[4], user.item[5], user.item[6], user.item[7], \

        user.item[8], user.item[9], user.id);

    query\_stat = mysql\_query(connection, query);

    if (query\_stat != 0) // 오류 검사

    {

        system("cls");

        fprintf(stderr, "%s", mysql\_error(&conn));

        system("pause");

    }

// user 테이블의 포인트 갱신

    sprintf(query, "update user set point = %d where id = '%s'", user.point, user.id);

    query\_stat = mysql\_query(connection, query);

    if (query\_stat != 0)

    {

        system("cls");

        fprintf(stderr, "%s", mysql\_error(&conn));

        system("pause");

    }

// 랭킹 테이블의 고유 색상 아이템을 갱신

    sprintf(query, "update ranking set item4 = %d where id = '%s'", user.item[9], user.id);

    query\_stat = mysql\_query(connection, query);

    if (query\_stat != 0) // 오류 검사

    {

        system("cls");

        fprintf(stderr, "%s", mysql\_error(&conn));

        system("pause");

    }

    mysql\_close(connection); // 연결 종료

}

|  |  |  |
| --- | --- | --- |
| shop\_preview.c | [돌아가기](#moc) | 55~59 PAGE |

#include <stdio.h> // prinf 사용

#include "common.h" // 공통함수 사용

void shop\_preview(int list\_num) // 미리보기 출력을 해줌

{

// 항목이 바뀔때마다 미리보기 영역을 지워준다.

// system(cls)를 사용하지 않는 이유는 화면이 깜빡 거려서

    for (int i = 4; i < 24; i++)

    {

        gotoxy(59, i);

        printf("                                     ");

    }

    gotoxy(59, 4);

    printf("[아이템 이름]");

    gotoxy(59, 7);

    printf("[아이템 설명]");

    gotoxy(59, 10);

    printf("[조건]");

    switch (list\_num) // 항목에 따라 미리보기 출력

    {

    case 4:

        gotoxy(59, 5);

        printf("생명 추가");

        gotoxy(59, 8);

        printf("플레이어의 LIFE를 1 추가한다.");

        gotoxy(59, 11);

        printf("1. 플레이어의 체력이 3미만일때는");

        gotoxy(59, 12);

        printf("체력을 1 추가하고 폭탄 한발을");

        gotoxy(59, 13);

        printf("무료로 자동 발사 해준다.");

        gotoxy(59, 15);

        printf("2. 폭탄이 발사중인 상태에서는");

        gotoxy(59, 16);

        printf("폭탄의 개수를 1개 추가한다.");

        gotoxy(59, 18);

        printf("3. 플레이어의 체력이 3이라면");

        gotoxy(59, 19);

        printf("폭탄을 2개 추가한다.");

        gotoxy(59, 21);

        printf("4. 폭탄이 3개라면 추가 하지 않는다.");

        gotoxy(59, 23);

        printf("5. 최대 3개까지 보유 가능하다.");

        break;

    case 6:

        gotoxy(59, 5);

        printf("HP 회복");

        gotoxy(59, 8);

        printf("플레이어의 HP를 1 회복한다.");

        gotoxy(59, 11);

        printf("1. 연속 사용 가능");

        gotoxy(59, 13);

        printf("2. HP가 3일때는 사용 불가");

        gotoxy(59, 15);

        printf("3. 게임이 종료시 아이템 삭제");

        gotoxy(59, 17);

        printf("4. 최대 5개까지 보유 가능하다.");

        break;

    case 8:

        gotoxy(59, 5);

        printf("추가 점수");

        gotoxy(59, 8);

        printf("적 파괴시 추가 점수 획득 가능");

        gotoxy(59, 11);

        printf("1. 게임 시작시 자동 사용");

        gotoxy(59, 13);

        printf("2. 랭킹 확인시 아이템 사용 표시");

        gotoxy(59, 15);

        printf("3. 포인트 추가 획득 가능");

        gotoxy(59, 17);

        printf("4. 무제한 구매 가능");

        break;

    case 10:

        gotoxy(59, 5);

        printf("빨강 페인트");

        gotoxy(59, 8);

        printf("플레이어의 전투기 색상을 변경한다.");

        gotoxy(59, 11);

        printf("1. 다른 색상 구매시 자동 판매");

        gotoxy(59, 13);

        printf("2. 전투기 디자인 변경");

        gotoxy(59, 15);

        printf("3. 데미지 차이는 없다.");

        gotoxy(59, 18);

        printf("변경전\t\t변경후");

        gotoxy(59, 20);

        printf("[-\*-]");

        gotoxy(80, 20);

        set\_color(RED);

        printf("<-\*->");

        set\_color(WHITE);

        break;

    case 12:

        gotoxy(59, 5);

        printf("파랑 페인트");

        gotoxy(59, 8);

        printf("플레이어의 전투기 색상을 변경한다.");

        gotoxy(59, 11);

        printf("1. 다른 색상 구매시 자동 판매");

        gotoxy(59, 13);

        printf("2. 전투기 디자인 변경");

        gotoxy(59, 15);

        printf("3. 데미지 차이는 없다.");

        gotoxy(59, 18);

        printf("변경전\t\t변경후");

        gotoxy(59, 20);

        printf("[-\*-]");

        gotoxy(80, 20);

        set\_color(BLUE);

        printf("{:\*:}");

        set\_color(WHITE);

        break;

    case 14:

        gotoxy(59, 5);

        printf("노랑 페인트");

        gotoxy(59, 8);

        printf("플레이어의 전투기 색상을 변경한다.");

        gotoxy(59, 11);

        printf("1. 다른 색상 구매시 자동 판매");

        gotoxy(59, 13);

        printf("2. 전투기 디자인 변경");

        gotoxy(59, 15);

        printf("3. 데미지 차이는 없다.");

        gotoxy(59, 18);

        printf("변경전\t\t변경후");

        gotoxy(59, 20);

        printf("[-\*-]");

        gotoxy(80, 20);

        set\_color(YELLOW);

        printf("H:\*:H");

        set\_color(WHITE);

        break;

    case 16:

        gotoxy(59, 5);

        printf("빨간 캐논");

        gotoxy(59, 8);

        printf("플레이어의 캐논 색상을 변경한다.");

        gotoxy(59, 11);

        printf("1. 다른 색상 구매시 자동 판매");

        gotoxy(59, 13);

        printf("2. 캐논 디자인 변경");

        gotoxy(59, 15);

        printf("3. 데미지 차이는 없다.");

        gotoxy(59, 18);

        printf("변경전\t\t변경후");

        gotoxy(59, 20);

        printf("ⅰ ⅱ ⅲ");

        gotoxy(80, 20);

        set\_color(RED);

        printf("γ ψ Ψ");

        set\_color(WHITE);

        break;

    case 18:

        gotoxy(59, 5);

        printf("파란 캐논 ");

        gotoxy(59, 8);

        printf("플레이어의 캐논 색상을 변경한다.");

        gotoxy(59, 11);

        printf("1. 다른 색상 구매시 자동 판매");

        gotoxy(59, 13);

        printf("2. 캐논 디자인 변경");

        gotoxy(59, 15);

        printf("3. 데미지 차이는 없다.");

        gotoxy(59, 18);

        printf("변경전\t\t변경후");

        gotoxy(59, 20);

        printf("ⅰ ⅱ ⅲ");

        gotoxy(80, 20);

        set\_color(BLUE);

        printf("＋ ± ÷");

        set\_color(WHITE);

        break;

    case 20:

        gotoxy(59, 5);

        printf("노란 캐논 ");

        gotoxy(59, 8);

        printf("플레이어의 캐논 색상을 변경한다.");

        gotoxy(59, 11);

        printf("1. 다른 색상 구매시 자동 판매");

        gotoxy(59, 13);

        printf("2. 캐논 디자인 변경");

        gotoxy(59, 15);

        printf("3. 데미지 차이는 없다.");

        gotoxy(59, 18);

        printf("변경전\t\t변경후");

        gotoxy(59, 20);

        printf("ⅰ ⅱ ⅲ");

        gotoxy(80, 20);

        set\_color(YELLOW);

        printf("★ ♠ ♣");

        set\_color(WHITE);

        break;

    case 22:

        gotoxy(59, 5);

        printf("랭킹창 꾸미기");

        gotoxy(59, 8);

        printf("랭킹창의 유저명을 꾸며줍니다.");

        gotoxy(59, 11);

        printf("1. 랜덤 색상을 부여합니다.");

        gotoxy(59, 13);

        printf("2. 색상은 14가지 입니다.");

        gotoxy(59, 15);

        printf("2. 재구매시 색상이 변경 됩니다.");

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| draw\_owned\_item.c | [돌아가기](#moc) | 60 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

void draw\_owned\_item(void) // 소지 아이템 출력

{

    int count = 0; // 아이템 인텍스를 변경하기 위해 사용

// for문 기준은 아이템 리스트 항목

    for (int POS\_Y = 4; POS\_Y <= 20; POS\_Y += 2)

    {

// 좌표를 이동하면서 개수를 출력

        gotoxy(40, POS\_Y);

        printf("%d 개", user.item[count]);

        count++;

    }

// 랭킹창 색상 아이템은 개수가 아니라 값을 저장하기 때문에 값이 있을때만 1개 출력

    if (user.item[9] != 15)

    {

        gotoxy(40, 22);

        printf("1 개");

    }

    else

    {

        gotoxy(40, 22);

        printf("0 개");

    }

// 유저 포인트 출력

    gotoxy(40, 24);

    printf("%11d pt", user.point);

}

|  |  |  |
| --- | --- | --- |
| game.c | [돌아가기](#moc) | 61 ~ 70 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // system 사용

#include <Windows.h> // GetAsyncKeyState 사용

#include "common.h" // 공통 함수들 사용

#include "game.h" // 구조체, 함수들 사용

void game(void)

{

// 게임화면의 명령창 크기 변경

    system("mode con: cols=80 lines=30");

    system("cls");

    // 게임 시간 변수

    int frame\_time = 0;

    // 구조체 변수들

// 구조체 변수들은 전부 포인터 변수로 사용

    PLAYER\* player = malloc(sizeof(PLAYER));

    CANNON\* cannon = malloc(sizeof(CANNON));

    BOMB\* bomb = malloc(sizeof(BOMB));

    bomb->bomb\_cannon[0] = malloc(sizeof(BOMB\_CANNON));

    bomb->bomb\_cannon[1] = malloc(sizeof(BOMB\_CANNON));

    DROP\_ITEM\* drop\_item = malloc(sizeof(DROP\_ITEM));

    SHOP\_ITEM\* shop\_item = malloc(sizeof(SHOP\_ITEM));

    ENEMY\* enemy = malloc(sizeof(ENEMY));

    for (int i = 0; i < ENEMY\_SIZE; i++)

    {

        enemy->enemy\_cannon[i] = malloc(sizeof(ENEMY));

    }

    // 구조체 변수 초기화

    init\_variables(player, cannon, bomb, drop\_item, shop\_item, enemy);

    // DB에서 데이터를 읽어 초기 정보 세팅

    load\_data(shop\_item);

    // 추가 점수 아이템은 자동적용

    if (shop\_item->score\_buff > 0)

    {

        shop\_item->score\_buff--; // 추가 점수 아이템 보유수 차감

        shop\_item->buff\_on = TRUE; // 적용상태로 변경

    }

    // 게임 시작전 도움말 출력

    draw\_help();

// 게임 종료까지 무한 반복

    while (1)

    {

// 진행 시간 출력

        gotoxy(0, 0);

        printf("%d", frame\_time);

        draw\_game(0); // 게임 UI 출력

        draw\_status(player, shop\_item); // 스탯 출력

        // 전투기 출력

        {

            gotoxy(player->pos\_x, player->pos\_y);

            set\_color(shop\_item->combat\_color);

            printf(shop\_item->combat\_design);

            set\_color(WHITE);

        }

        // 키보드 입력 감지

        // 함수화, getch()로 감지 등등 전부 반응속도가 떨어지고 툭툭 끊김

        {

            // 왼쪽 이동

            if ((GetAsyncKeyState(VK\_LEFT) & 0x8000) && player->pos\_x > LEFT\_LIMIT)

            {

// 원래 위치를 지우고 전투기 색상, 모양에 따라 출력

                gotoxy(player->pos\_x, player->pos\_y);

                printf("     ");

                player->pos\_x--;

                gotoxy(player->pos\_x, player->pos\_y);

                set\_color(shop\_item->combat\_color);

                printf(shop\_item->combat\_design);

                set\_color(WHITE);

            }

            // 오른쪽 이동

            if ((GetAsyncKeyState(VK\_RIGHT) & 0x8000) && player->pos\_x < RIGHT\_LIMIT)

            {

                gotoxy(player->pos\_x, player->pos\_y);

                printf("     ");

                player->pos\_x++;

                gotoxy(player->pos\_x, player->pos\_y);

                set\_color(shop\_item->combat\_color);

                printf(shop\_item->combat\_design);

                set\_color(WHITE);

            }

            // 위로 이동

            if ((GetAsyncKeyState(VK\_UP) & 0x8000) && player->pos\_y > UP\_LIMIT)

            {

                gotoxy(player->pos\_x, player->pos\_y);

                printf("     ");

                player->pos\_y--;

                gotoxy(player->pos\_x, player->pos\_y);

                set\_color(shop\_item->combat\_color);

                printf(shop\_item->combat\_design);

                set\_color(WHITE);

            }

            // 아래로 이동

            if ((GetAsyncKeyState(VK\_DOWN) & 0x8000) && player->pos\_y < DOWN\_LIMIT)

            {

                gotoxy(player->pos\_x, player->pos\_y);

                printf("     ");

                player->pos\_y++;

                gotoxy(player->pos\_x, player->pos\_y);

                set\_color(shop\_item->combat\_color);

                printf(shop\_item->combat\_design);

                set\_color(WHITE);

            }

        }

        // 게임 일시 정지

        // 게임 시작전 눌린 키를 가려내기 위해 frame\_time을 검사

// 0x8000은 현재 키가 눌려진 상태, 0x0001은 지난번 호출과 이번 호출 사이에 키가 눌러진적이 있었음을 의미

// 0x0001을 사용하는 이유는 PAUSE를 걸고 아니요를 선택해도 다시 ESC가 눌리는 현상이 있어서

        if ((GetAsyncKeyState(VK\_ESCAPE) & 0x0001) && frame\_time > 0)

        {

            int key = 0; // 키값 저장

            int do\_exit = 23; // 나갈지 말지 저장

// 사용자에게 네/아니오 입력 받음

            while (key != ENTER)

            {

                // PAUSE 화면 출력

                draw\_game(1);

                gotoxy(do\_exit, 14);

                printf("◎");

                key = getch();

                gotoxy(do\_exit, 14);

                printf("  ");

                do\_exit = move(1, key, do\_exit, 14);

            }

            // 네 선택

            if (do\_exit == 23)

            {

// 게임 엔딩 화면 출력

                draw\_end\_game(player, shop\_item);

                draw\_game(3);

// 사용한 아이템 DB에 반영

                save\_data(player, shop\_item);

// 포인터 구조체 변수들 메모리 해제

                free(player);

                free(cannon);

                free(bomb->bomb\_cannon[0]);

                free(bomb->bomb\_cannon[1]);

                free(bomb);

                free(drop\_item);

                free(shop\_item);

                for (int i = 0; i < ENEMY\_SIZE; i++)

                {

                    free(enemy->enemy\_cannon[i]);

                }

                free(enemy);

                break;

            }

            // 아니요 선택

            else

            {

// 화면을 한번 지우고 다시 올려보냄

                system("cls");

                continue;

            }

        }

        // 플레이어 공격 - 스페이스바

        if ((GetAsyncKeyState(VK\_SPACE) & 0x8000) && frame\_time > 0)

        {

// 캐논 수량만큼 공격

            for (int i = 0; i < player->cannon\_limit; i++)

            {

// 캐논 각각의 상태가 FALSE 일때만 발사

                if (cannon->condition[i] == FALSE)

                {

// 좌표를 저장하고 상태를 TRUE로 변경

                    cannon->pos\_x[i] = player->pos\_x + 1;

                    cannon->pos\_y[i] = player->pos\_y - 1;

                    cannon->condition[i] = TRUE;

                    break;

                }

            }

        }

        // 폭탄 공격 - Z키

        if ((GetAsyncKeyState(0x5A) & 0x0001) && frame\_time > 0)

        {

// 폭탄 수량이 남아있고 발사 상태가 아닐 때

            if (player->bomb\_quantity > 0 && (bomb->condition[0] == FALSE && bomb->condition[1] == FALSE))

            {

// 수량을 차감하고 발사

                player->bomb\_quantity--;

                bomb->condition[0] = TRUE;

                bomb->condition[1] = TRUE;

            }

        }

        // 생명 추가 - X키

// 스킬 재사용 시간이 0이고 수량이 남아있을 때

        if (((GetAsyncKeyState(0x58) & 0x0001) && frame\_time > 1) && (shop\_item->cooldown\_time == 0 && shop\_item->life\_plus > 0))

        {

// 생명을 추가함

            player->life++;

// 점수 차감

            player->score -= 50;

// 아이템 개수 차감

            shop\_item->life\_plus--;

// 재사용 시간 설정

            shop\_item->cooldown\_time = 1000;

// 폭탄이 하나라도 있으면 차감하지 않고 발사

            if (player->bomb\_quantity > 0)

            {

                bomb->condition[0] = TRUE;

                bomb->condition[1] = TRUE;

            }

// 폭탄이 없으면 수량 추가

            else

            {

                player->bomb\_quantity++;

            }

        }

        // 체력 회복 - C키

// 플레이어 체력이 3 미만이고 아이템이 있을 때 사용

        if (((GetAsyncKeyState(0x43) & 0x0001) && frame\_time > 1) && (shop\_item->hp\_recover > 0 && player->hp < 3))

        {

// 아이템 차감

            shop\_item->hp\_recover--;

            player->hp++; // 체력 증가

            player->score -= 10; // 점수 차감

        }

        // 적군 생성

        enemy\_gen(enemy, frame\_time);

        // 아이템 생성

        item\_gen(drop\_item, frame\_time);

        // 적군 이동

        enemy\_move(enemy);

        // 폭탄 이동

        if (bomb->condition[0] == TRUE || bomb->condition[1] == TRUE)

        {

            bomb\_move(bomb);

        }

        // 캐논 이동

        cannon\_move(cannon, shop\_item);

        // 적군 캐논 발사

        eCannon\_status(enemy, player, bomb, cannon, 0);

        // 적 캐논 이동

        eCannon\_move(enemy);

        // 아이템 이동

        item\_move(drop\_item);

        // 아이템 유저 충돌

        item\_status(drop\_item, player, cannon, shop\_item, 0);

        // 아이템 바닥 충돌

        item\_status(drop\_item, player, cannon, shop\_item, 1);

        // 적 캐논 플레이어 충돌

        eCannon\_status(enemy, player, bomb, cannon, 1);

        // 적 캐논 바닥 도달

        eCannon\_status(enemy, player, bomb, cannon, 2);

        // 적 캐논 과 플레이어 캐논 충돌

        eCannon\_status(enemy, player, bomb, cannon, 3);

        // 폭탄 발사시 적 캐논 폭탄 앞에서 삭제

        eCannon\_status(enemy, player, bomb, cannon, 4);

        // 폭탄 - 적 충돌

        bomb\_status(bomb, enemy, player, shop\_item, 0);

        // 폭탄이 천장에 도달하면 삭제

        bomb\_status(bomb, enemy, player, shop\_item, 1);

        // 적군이 바닥에 도달하면 삭제

        enemy\_status(enemy, bomb, player, cannon, shop\_item, 0);

        // 플레이어 - 적 충돌

        enemy\_status(enemy, bomb, player, cannon, shop\_item, 1);

        // 캐논 - 적 충돌

        enemy\_status(enemy, bomb, player, cannon, shop\_item, 2);

        // 폭탄 캐논 - 적 충돌

        enemy\_status(enemy, bomb, player, cannon, shop\_item, 3);

        // 캐논 이 천장에 도달하면 삭제

        cannon\_status(cannon);

        // 폭탄 캐논 발사

        bCannon\_status(bomb, 0);

        // 폭탄 캐논 이동

        bCannon\_move(bomb);

        // 폭탄 캐논이 천장에 도달하면 삭제

        bCannon\_status(bomb, 1);

        // 스킬 재사용 대기

        if (shop\_item->cooldown\_time > 0)

        {

            shop\_item->cooldown\_time--;

        }

        // 게임오버 검사

        if (player->life == 0 && player->hp == 0)

        {

// 게임 엔딩 화면 출력

            draw\_end\_game(player, shop\_item);

            draw\_game(3);

            save\_data(player, shop\_item); // db에 데이터 저장

// 포인터 구조체 변수 메모리 해제

            free(player);

            free(cannon);

            free(bomb->bomb\_cannon[0]);

            free(bomb->bomb\_cannon[1]);

            free(bomb);

            free(drop\_item);

            free(shop\_item);

            for (int i = 0; i < ENEMY\_SIZE; i++)

            {

                free(enemy->enemy\_cannon[i]);

            }

            free(enemy);

            break;

        }

        // 게임 클리어 검사

// 시간이 다 지났던가 특정 시간 이후로 적이 없다면 클리어

        if (frame\_time == 3800 || frame\_time > 3100)

        {

            int is\_cleared = TRUE;

// 적이 남아있는지 검사

            for (int i = 0; i < ENEMY\_SIZE; i++)

            {

                if (enemy->condition[i] == TRUE)

                {

                    is\_cleared = FALSE;

                    break;

                }

            }

// 적이 하나도 없다면 클리어

            if (is\_cleared == TRUE)

            {

// 게임 엔딩창 출력

                draw\_end\_game(player, shop\_item);

                draw\_game(4);

                save\_data(player, shop\_item); // DB 저장

// 포인터 구조체 변수 메모리 해제

                free(player);

                free(cannon);

                free(bomb->bomb\_cannon[0]);

                free(bomb->bomb\_cannon[1]);

                free(bomb);

                free(drop\_item);

                free(shop\_item);

                for (int i = 0; i < ENEMY\_SIZE; i++)

                {

                    free(enemy->enemy\_cannon[i]);

                }

                free(enemy);

                break;

            }

        }

        frame\_time++; // 게임 시간 증가

        Sleep(18); // 게임 전체 속도 조절

    }

}

|  |  |  |
| --- | --- | --- |
| game.h | [돌아가기](#moc) | 71 ~ 75 PAGE |

// 컴파일러에게 해당 헤더 파일이 한번만 빌드되도록 시킨다.

#pragma once

// 키보드 입력 관련 매크로 상수들

#define LEFT\_LIMIT 5

#define RIGHT\_LIMIT 51

#define UP\_LIMIT 14

#define DOWN\_LIMIT 29

// 개체 레벨을 위해 사용하는 매크로 상수들

#define LEVEL0 0

#define LEVEL1 1

#define LEVEL2 2

// 플레이어 전투기 레벨에 따라 모양이 다른 매크로 상수들

#define COMBAT1 "[-\*-]"

#define COMBAT2 "<-\*->"

#define COMBAT3 "{:\*:}"

#define COMBAT4 "H:\*:H"

// 플레이어 캐논 매크로 상수들

// 레벨과 상점 아이템에 따라 모양과 색상이 다름

#define CANNON1 "ⅰ"

#define CANNON2 "γ"

#define CANNON3 "＋"

#define CANNON4 "★"

#define CANNON5 "ⅱ"

#define CANNON6 "ψ"

#define CANNON7 "±"

#define CANNON8 "♠"

#define CANNON9 "ⅲ"

#define CANNON10 "Ψ"

#define CANNON11 "÷"

#define CANNON12 "♣"

// 구조체 변수들 크기를 위한 매크로 상수들

#define CANNON\_SIZE 4

#define BOMB\_SIZE 2

#define BOMB\_CANNON\_SIZE 5

#define ITEM\_SIZE 5

#define ENEMY\_SIZE 30

#define ENEMY\_CANNON\_SIZE 10

// 플레이어 구조체

// X값, Y값, 생명, 체력, 폭탄 수량, 캐논 수량, 점수

typedef struct Player

{

    int pos\_x;

    int pos\_y;

    int life;

    int hp;

    int bomb\_quantity;

    int cannon\_limit;

    int score;

} PLAYER;

// 캐논 구조체

// x값, y값, 상태 관리, 레벨 관리

typedef struct Cannon

{

    int pos\_x[CANNON\_SIZE];

    int pos\_y[CANNON\_SIZE];

    int condition[CANNON\_SIZE];

    int upgrade;

} CANNON;

// 폭탄 구조체

// x값, y값, 상태 관리, 움직임 속도, 폭탄-캐논 구조체 변수

typedef struct Bomb

{

    int pos\_x[BOMB\_SIZE];

    int pos\_y[BOMB\_SIZE];

    int condition[BOMB\_SIZE];

    int move\_interval[BOMB\_SIZE];

    struct BombCannon\* bomb\_cannon[BOMB\_SIZE];

} BOMB;

// 폭탄 캐논 구조체

// x값 y값 상태 관리

typedef struct BombCannon

{

    int pos\_x[BOMB\_CANNON\_SIZE];

    int pos\_y[BOMB\_CANNON\_SIZE];

    int condition[BOMB\_CANNON\_SIZE];

} BOMB\_CANNON;

// 드롭 아이템 구조체

// x값 y값 상태 관리, 아이템 타입, 움직임 속도 조절용

typedef struct DropItem

{

    int pos\_x[ITEM\_SIZE];

    int pos\_y[ITEM\_SIZE];

    int condition[ITEM\_SIZE];

    int type[ITEM\_SIZE];

    int move\_interval[ITEM\_SIZE];

} DROP\_ITEM;

// 상점 아이템 구조체

// 생명 추가, 체력 회복, 추가 점수, 추가 점수 아이템 적용 여부

// 재사용 시간, 전투기 색상, 전투기 모양, 캐논 색상, 캐논 모양

typedef struct ShopItem

{

    int life\_plus;

    int hp\_recover;

    int score\_buff;

    int buff\_on;

    int cooldown\_time;

    int combat\_color;

    char\* combat\_design;

    int cannon\_color;

    char\* cannon\_design;

} SHOP\_ITEM;

// 적군 구조체

// x값, y값, 상태관리, 모양, 움직임과 공격 패턴

// 방향 전환용, 움직임 속도 조절용, 체력, 적군 캐논 구조체 변수

typedef struct Enemy

{

    int pos\_x[ENEMY\_SIZE];

    int pos\_y[ENEMY\_SIZE];

    int condition[ENEMY\_SIZE];

    int design[ENEMY\_SIZE];

    int pattern[ENEMY\_SIZE];

    int move\_count[ENEMY\_SIZE];

    int move\_interval[ENEMY\_SIZE];

    int hp[ENEMY\_SIZE];

    struct EnemyCannon\* enemy\_cannon[ENEMY\_SIZE];

} ENEMY;

// 적군 캐논 구조체

// x값 y값 상태 관리, 움직임 속도, 움직임 패턴

typedef struct EnemyCannon

{

    int pos\_x[ENEMY\_CANNON\_SIZE];

    int pos\_y[ENEMY\_CANNON\_SIZE];

    int condition[ENEMY\_CANNON\_SIZE];

    int move\_interval;

    int move\_pattern[ENEMY\_CANNON\_SIZE];

} ENEMY\_CANNON;

// 초기설정, DB 불러오기 저장하기

void init\_variables(PLAYER\*, CANNON\*, BOMB\*, DROP\_ITEM\*, SHOP\_ITEM\*, ENEMY\*);

void load\_data(SHOP\_ITEM\*);

void save\_data(PLAYER\*, SHOP\_ITEM\*);

// UI 출력

void draw\_status(PLAYER\*, SHOP\_ITEM\*);

void draw\_game(int);

void draw\_symbol(int, int, int);

void draw\_bomb(int, int, int);

void draw\_end\_game(PLAYER\*, SHOP\_ITEM\*);

void draw\_help(void);

// 캐논 관련

void cannon\_move(CANNON\*, SHOP\_ITEM\*);

void cannon\_status(CANNON\*);

// 폭탄 관련

void bomb\_move(BOMB\*);

void bomb\_status(BOMB\*, ENEMY\*, PLAYER\*, SHOP\_ITEM\* , int);

void bCannon\_status(BOMB\*, int);

void bCannon\_move(BOMB\*);

// 드랍 아이템 관련

void item\_gen(DROP\_ITEM\*, int);

void item\_move(DROP\_ITEM\*);

void item\_status(DROP\_ITEM\*, PLAYER\*, CANNON\*, SHOP\_ITEM\*, int);

// 적군 관련

void enemy\_gen(ENEMY\*, int);

void enemy\_move(ENEMY\*);

void enemy\_status(ENEMY\*, BOMB\*, PLAYER\*, CANNON\*, SHOP\_ITEM\*, int);

void eCannon\_status(ENEMY\*, PLAYER\*, BOMB\*, CANNON\*, int);

void eCannon\_move(ENEMY\*);

|  |  |  |
| --- | --- | --- |
| init\_variables.c | [돌아가기](#moc) | 76 ~ 78 PAGE |

#include <stdio.h> // 표준 함수

#include "game.h" // 구조체 변수를 사용하기 위해 포함

#include "common.h" // 매크로 상수를 사용하기 위해 포함

// 구조체 변수들을 전달받아 초기화 해줌

void init\_variables(PLAYER\* player, CANNON\* cannon, BOMB\* bomb, DROP\_ITEM\* drop\_item, SHOP\_ITEM\* shop\_item, ENEMY\* enemy)

{

    // player 구조체 초기화

    // 좌표 X, Y 값, 생명, 체력, 폭탄 잔량, 캐논 발사 가능 잔량, 점수

    player->pos\_x = 28;

    player->pos\_y = 28;

    player->life = 1;

    player->hp = 3;

    player->bomb\_quantity = 1;

    player->cannon\_limit = 1;

    player->score = 0;

    // cannon 구조체 초기화

    // 좌표 X, Y 값, 각각의 발사 확인용, 캐논 레벨 확인

    for (int i = 0; i < CANNON\_SIZE; i++)

    {

        cannon->pos\_x[i] = 0;

        cannon->pos\_y[i] = 0;

        cannon->condition[i] = FALSE;

    }

    cannon->upgrade = LEVEL0;

    // bomb 구조체 초기화

    // 좌표 X, Y 값, 발사 확인용, 이동 속도 조절용

    bomb->pos\_x[0] = 29;

    bomb->pos\_y[0] = 24;

    bomb->condition[0] = FALSE;

    bomb->move\_interval[0] = 5;

    bomb->pos\_x[1] = 6;

    bomb->pos\_y[1] = 24;

    bomb->condition[1] = FALSE;

    bomb->move\_interval[1] = 5;

    // bomb -> bomb\_cannon 구조체 초기화

    // 좌표 X, Y 값, 발사 확인용

    for (int i = 0; i < BOMB\_CANNON\_SIZE; i++)

    {

        bomb->bomb\_cannon[0]->pos\_x[i] = 0;

        bomb->bomb\_cannon[0]->pos\_y[i] = 0;

        bomb->bomb\_cannon[0]->condition[i] = FALSE;

        bomb->bomb\_cannon[1]->pos\_x[i] = 0;

        bomb->bomb\_cannon[1]->pos\_y[i] = 0;

        bomb->bomb\_cannon[1]->condition[i] = FALSE;

    }

    // drop\_item 구조체 초기화

    // 좌표 X, Y 값, 발사 확인용, 아이템 종류, 움직임 속도 조절용

    for (int i = 0; i < ITEM\_SIZE; i++)

    {

        drop\_item->pos\_x[i] = 0;

        drop\_item->pos\_y[i] = 0;

        drop\_item->condition[i] = FALSE;

        drop\_item->type[i] = 0;

        drop\_item->move\_interval[i] = 0;

    }

    // shop\_item 구조체 초기화

    // 생명, 체력, 추가점수, 추가점수 활성화 확인용, 스킬 재사용 시간, 전투기 색상, 전투기 디자인, 캐논 색상, 캐논 디자인

    shop\_item->life\_plus = 0;

    shop\_item->hp\_recover = 0;

    shop\_item->score\_buff = 0;

    shop\_item->score\_buff = FALSE;

    shop\_item->cooldown\_time = 0;

    shop\_item->combat\_color = WHITE;

    shop\_item->combat\_design = COMBAT1;

    shop\_item->cannon\_color = WHITE;

    shop\_item->cannon\_design = CANNON1;

    // enemy 구조체 초기화

    // 좌표 X, Y 값, 발사 확인용, 움직임 & 공격 패턴, 움직임 방향 전환용, 움직임 속도 조절용, 적군 체력

    for (int i = 0; i < ENEMY\_SIZE; i++)

    {

        enemy->pos\_x[i] = 0;

        enemy->pos\_y[i] = 0;

        enemy->condition[i] = FALSE;

        enemy->design[i] = 0;

        enemy->pattern[i] = 0;

        enemy->move\_count[i] = 0;

        enemy->move\_interval[i] = 0;

        enemy->hp[i] = 0;

        // enemy -> enemy\_cannon 구조체 초기화

        // 좌표 X, Y 값, 발사 확인용, 공격 패턴, 움직임 속도 조절용

        for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

        {

            enemy->enemy\_cannon[i]->pos\_x[i] = 0;

            enemy->enemy\_cannon[i]->pos\_y[i] = 0;

            enemy->enemy\_cannon[i]->condition[i] = 0;

            enemy->enemy\_cannon[i]->move\_pattern[i] = 0;

        }

        enemy->enemy\_cannon[i]->move\_interval = 0;

    }

}

|  |  |  |
| --- | --- | --- |
| load\_data.c | [돌아가기](#moc) | 79 ~ 81 PAGE |

#include <stdio.h> // 표준함수

#include <stdlib.h> // system 사용

#include <mysql.h> // mysql 사용

#include "common.h" // gotoxy, 공통 함수 사용

#include "game.h" // 구조체 사용

#pragma commnet(lib, "libmysql.lib") // 오류 방지를 위해 사용

// DB에서 데이터를 불러와서 저장한다.

void load\_data(SHOP\_ITEM\* shop\_item)

{

    MYSQL\* connection = NULL; // 연결 상태 저장

    MYSQL conn = { 0, }; // 연결을 위해 사용

    MYSQL\_RES\* result = NULL; // 결과를 반영

    MYSQL\_ROW row = { 0, }; // 행 단위로 값을 받음

    int query\_stat = 0; // 오류 검사용

    char query[255] = { 0, }; // 쿼리문 저장

    mysql\_init(&conn); // 초기화

    connection = mysql\_real\_connect(&conn, DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME, 3306, (char\*)NULL, 0); // 연결

    if (connection == NULL) // 에러 발생시 에러 UI 출력

    {

        draw\_error(0);

        fprintf(stderr, "%s", mysql\_error(&conn));

        gotoxy(22, 17);

        system("pause");

        exit(0); // 사용자가 해결 불가능한 에러는 강제 종료

    }

    // item\_list db에서 조회

    sprintf(query, "select \* from item\_list where id = '%s'", user.id);

    query\_stat = mysql\_query(connection, query);

    if (query\_stat != 0) // 에러 검사

    {

        draw\_error(2);

        fprintf(stderr, "%s", mysql\_error(&conn));

        gotoxy(22, 17);

        system("pause");

        exit(0);

    }

// 데이터를 넘겨 받음

    result = mysql\_store\_result(connection);

    while ((row = mysql\_fetch\_row(result)) != NULL)

    {

// 구조체에 저장

// 정수값을 받아도 문자열로 들어오기 때문에 정수로 변환함

        shop\_item->life\_plus = atoi(row[1]);

        shop\_item->hp\_recover = atoi(row[2]);

        shop\_item->score\_buff = atoi(row[3]);

        shop\_item->combat\_color = WHITE;

        shop\_item->cannon\_color = WHITE;

        // 전투기 색상, 모양 설정

        if (atoi(row[4]) != 0)

        {

            shop\_item->combat\_color = RED;

            shop\_item->combat\_design = COMBAT2;

        }

        else if (atoi(row[5]) != 0)

        {

            shop\_item->combat\_color = BLUE;

            shop\_item->combat\_design = COMBAT3;

        }

        else if (atoi(row[6]) != 0)

        {

            shop\_item->combat\_color = YELLOW;

            shop\_item->combat\_design = COMBAT4;

        }

        // 캐논 색상, 모양 설정

        if (atoi(row[7]) != 0)

        {

            shop\_item->cannon\_color = RED;

            shop\_item->cannon\_design = CANNON2;

        }

        else if (atoi(row[8]) != 0)

        {

            shop\_item->cannon\_color = BLUE;

            shop\_item->cannon\_design = CANNON3;

        }

        else if (atoi(row[9]) != 0)

        {

            shop\_item->cannon\_color = YELLOW;

            shop\_item->cannon\_design = CANNON4;

        }

    }

    mysql\_free\_result(result); // 사용하고 나면 비워줌

    mysql\_close(connection); // 연결 해제

}

|  |  |  |
| --- | --- | --- |
| save\_data.c | [돌아가기](#moc) | 82 ~ 85 PAGE |

#include <stdio.h> // 표준함수

#include <stdlib.h> // system 사용

#include <mysql.h> // mysql 사용

#include "common.h" // user 구조체 사용

#include "game.h" // 구조체 사용

#pragma commnet(lib, "libmysql.lib") // 에러 방지

// 게임 종료시 데이터 저장

void save\_data(PLAYER\* player, SHOP\_ITEM\* shop\_item)

{

    MYSQL\* connection = NULL; // 연결 상태 저장

    MYSQL conn = { 0, }; // 연결을 위해 사용

    MYSQL\_RES\* result = NULL; // 결과를 저장

    MYSQL\_ROW row = { 0, }; // 행단위로 값을 받음

    int query\_stat = 0; // 오류 검사용

    char query[255] = { 0, }; // 쿼리문 저장

    mysql\_init(&conn); // 초기화

    connection = mysql\_real\_connect(&conn, DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME, 3306, (char\*)NULL, 0); // 연결

    if (connection == NULL) // 에러 발생시 UI 출력

    {

        draw\_error(0);

        fprintf(stderr, "%s", mysql\_error(&conn));

        gotoxy(22, 17);

        system("pause");

        exit(0); // 강제 종료

    }

    // 사용한 아이템 DB에 반영

    // c언어에서 '\'나 괄호를 쓰면 줄넘김 할 수 있음

    sprintf(query, "update item\_list set item1 = %d, item2 = %d, item3 = %d" where id = ‘%s’, shop\_item->life\_plus, shop\_item->hp\_recover, shop\_item->score\_buff, user.id);

    query\_stat = mysql\_query(connection, query);

    if (query\_stat != 0)

    {

        system("cls");

        fprintf(stderr, "%s", mysql\_error(&conn));

        system("pause");

    }

    // user.score에는 최고 기록이 저장되어 있어서 최근 게임의 점수와 비교해서 저장한다.

    // 최고 기록 갱신시 랭킹 테이블 업데이트

    if (user.score < player->score)

    {

        // 기록 갱신전 초기화 해줌

        // 초기화를 하지않으면 이전 기록이 남아 있어서 따로 업데이트를 해줘야하기 때문에

        // 어차피 업데이트할거 일괄적으로 초기화 해줌

        sprintf(query, "update ranking set score = %d, item1 = %d, item2 = %d, item3 = %d where id = '%s'", 0, 0, 0, 0, user.id);

        query\_stat = mysql\_query(connection, query);

        if (query\_stat != 0)

        {

            system("cls");

            fprintf(stderr, "%s", mysql\_error(&conn));

            system("pause");

        }

        user.score = player->score; // 점수 업데이트 이후에 user 구조체 점수 변경

        // 추가 점수 아이템이 있을때는 자동 적용

        if (user.item[2] != shop\_item->score\_buff)

        {

// 아래 if-else if문들은 전부 사용한 아이템 경우라서 과정이 동일하다.

            if ((user.item[0] != shop\_item->life\_plus) && (user.item[1] != shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item1 = %d, item2 = %d, item3 = %d where id = '%s'", user.score, 1, 1, 1, user.id);

            }

            else if ((user.item[0] == shop\_item->life\_plus) && (user.item[1] != shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item2 = %d, item3 = %d where id = '%s'", user.score, 1, 1, user.id);

            }

            else if ((user.item[0] != shop\_item->life\_plus) && (user.item[1] == shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item1 = %d, item3 = %d where id = '%s'", user.score, 1, 1, user.id);

            }

            else if ((user.item[0] == shop\_item->life\_plus) && (user.item[1] == shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item3 = %d where id = '%s'", user.score, 1, user.id);

            }

        }

        else

        {

            if ((user.item[0] != shop\_item->life\_plus) && (user.item[1] != shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item1 = %d, item2 = %d where id = '%s'", user.score, 1, 1, user.id);

            }

            else if ((user.item[0] == shop\_item->life\_plus) && (user.item[1] != shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item2 = %d where id = '%s'", user.score, 1, user.id);

            }

            else if ((user.item[0] != shop\_item->life\_plus) && (user.item[1] == shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d, item1 = %d where id = '%s'", user.score, 1, user.id);

            }

            else if ((user.item[0] == shop\_item->life\_plus) && (user.item[1] == shop\_item->hp\_recover))

            {

                sprintf(query, "update ranking set score = %d where id = '%s'", user.score, user.id);

            }

        }

// 에러 검사

        query\_stat = mysql\_query(connection, query);

        if (query\_stat != 0)

        {

            system("cls");

            fprintf(stderr, "%s", mysql\_error(&conn));

            system("pause");

        }

    }

// 획득한 포인트 반영

    if ((player->score \* 0.5) > 0)

    {

        user.point = user.point + (player->score \* 0.5);

    }

    // 점수와 포인트 갱신

    sprintf(query, "update user set score = %d, point = %d where id = '%s'", user.score, user.point, user.id);

    query\_stat = mysql\_query(connection, query);

    if (query\_stat != 0) // 오류 검사

    {

        system("cls");

        fprintf(stderr, "%s", mysql\_error(&conn));

        system("pause");

    }

// 아이템 보유 개수 갱신

    user.item[0] = shop\_item->life\_plus;

    user.item[1] = shop\_item->hp\_recover;

    user.item[2] = shop\_item->score\_buff;

}

|  |  |  |
| --- | --- | --- |
| draw\_game.c | [돌아가기](#moc) | 86 ~ 90 PAGE |

#include <stdio.h> // printf 사용

#include <stdlib.h> // system 사용

#include "common.h" // gotoxy 사용

// 화면 번호에 맞게 출력

void draw\_game(int screen\_num)

{

    switch (screen\_num)

    {

    // 게임화면 - 조작 범위 프레임

    case 0:

        for (int i = 0; i < 30; i++)

        {

            gotoxy(4, i);

            printf(":");

            gotoxy(56, i);

            printf(":");

            // draw\_content(0) 보면 printf 사용한 이유 나옴

        }

        break;

    // 게임화면 - PAUSE 화면

    case 1:

        gotoxy(16, 10);

        puts("PAUSE");

        gotoxy(15, 11);

        puts("┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓");

        gotoxy(15, 12);

        printf("┃  게임을 종료 하시겠습니까?  ┃");

        gotoxy(15, 13);

        puts("┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫");

        gotoxy(15, 14);

        printf("┃   예 (  )  │  아니요 (  )   ┃");

        gotoxy(15, 15);

        puts("┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛");

        break;

    // 스탯창

    case 2:

        gotoxy(58, 1);

        printf("┏━━━━━━━━━━━━━━━━━━┓");

        gotoxy(58, 2);

        printf("┃                  ┃");

        gotoxy(58, 3);

        printf("┣━━━━━━━━━━━━━━━━━━┫");

        gotoxy(58, 4);

        printf("┃                  ┃");

        gotoxy(58, 5);

        printf("┗━━━━━━━━━━━━━━━━━━┛");

        gotoxy(59, 6);

        printf("ㆍㆍㆍㆍㆍㆍㆍㆍㆍㆍ");

        break;

    // 게임오버 글자 출력

    case 3:

        gotoxy(16, 3);

        printf("GAME OVER");

        gotoxy(16, 26);

        system("pause");

        break;

    // 게임클리어 글자 출력

    case 4:

        gotoxy(16, 3);

        printf("GAME CLEAR");

        gotoxy(16, 26);

        system("pause");

        break;

    // game\_help, 엔딩 화면

    case 5:

        gotoxy(15, 4);

        printf("┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓");

        gotoxy(15, 5);

        printf("┃                                              ┃");

        gotoxy(15, 6);

        printf("┃                                              ┃");

        gotoxy(15, 7);

        printf("┃                                              ┃");

        gotoxy(15, 8);

        printf("┃                                              ┃");

        gotoxy(15, 9);

        printf("┃                                              ┃");

        gotoxy(15, 10);

        printf("┃                                              ┃");

        gotoxy(15, 11);

        printf("┃                                              ┃");

        gotoxy(15, 12);

        printf("┃                                              ┃");

        gotoxy(15, 13);

        printf("┃                                              ┃");

        gotoxy(15, 14);

        printf("┃                                              ┃");

        gotoxy(15, 15);

        printf("┃                                              ┃");

        gotoxy(15, 16);

        printf("┃                                              ┃");

        gotoxy(15, 17);

        printf("┃                                              ┃");

        gotoxy(15, 18);

        printf("┃                                              ┃");

        gotoxy(15, 19);

        printf("┃                                              ┃");

        gotoxy(15, 20);

        printf("┃                                              ┃");

        gotoxy(15, 21);

        printf("┃                                              ┃");

        gotoxy(15, 22);

        printf("┃                                              ┃");

        gotoxy(15, 23);

        printf("┃                                              ┃");

        gotoxy(15, 24);

        printf("┃                                              ┃");

        gotoxy(15, 25);

        printf("┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛");

        break;

    // game\_help 화면

    case 6:

        gotoxy(20, 6);

        printf("게임 조작 방법");

        gotoxy(20, 8);

        printf("움직이기: 방향키");

        gotoxy(20, 10);

        printf("기본 공격: 스페이스바");

        gotoxy(20, 12);

        printf("폭탄 공격: Z 키");

        gotoxy(20, 14);

        printf("생명 추가 아이템: X 키");

        gotoxy(20, 16);

        printf("체력 회복 아이템: C 키");

        set\_color(RED);

        gotoxy(20, 18);

        printf("추가 점수 아이템은 자동 적용됩니다.");

        set\_color(WHITE);

        gotoxy(20, 21);

        printf("적 기체별 획득 점수가 다릅니다.");

        gotoxy(20, 23);

        printf("상점 아이템 이용시 점수가 차감됩니다.");

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| draw\_stat.c | [돌아가기](#moc) | 91 ~ 92 PAGE |

#include <stdio.h> // printf 사용

#include "game.h" // 구조체 사용

#include "common.h" // 공통 함수 사용

// 스탯창 출력

void draw\_status(PLAYER\* player, SHOP\_ITEM\* shop\_item)

{

    int POS\_Y = 0; // 좌표 Y 값

    // 스탯창 프레임 출력

    draw\_game(2);

    // 점수 출력

    gotoxy(66, 2);

    printf("SCORE");

    gotoxy(60, 4);

    printf("%16d", player->score);

    // 생명 출력

    POS\_Y = 8;

    gotoxy(59, POS\_Y);

    puts("생명");

    draw\_symbol(POS\_Y, 0, player->life); // 아이콘 출력을 위해 호출

    // 체력 출력

    POS\_Y = 11;

    gotoxy(59, POS\_Y);

    puts("체력: ");

    draw\_symbol(POS\_Y, 1, player->hp);

    // 폭탄 출력

    POS\_Y = 14;

    gotoxy(59, POS\_Y);

    puts("Z-폭탄: ");

    draw\_symbol(POS\_Y, 2, player->bomb\_quantity);

    // 스킬 - 생명 회복

    POS\_Y = 17;

    gotoxy(59, POS\_Y);

    puts("X-생명: ");

    draw\_symbol(POS\_Y, 3, shop\_item->life\_plus);

    // 스킬 - HP 회복

    POS\_Y = 20;

    gotoxy(59, POS\_Y);

    puts("C-체력: ");

    draw\_symbol(POS\_Y, 4, shop\_item->hp\_recover);

}

|  |  |  |
| --- | --- | --- |
| draw\_symbol.c | [돌아가기](#moc) | 93 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

// 스탯창 아이콘 출력

void draw\_symbol(int POS\_Y, int symbol, int count)

{

    int POS\_X = 67; // X좌표

// 출력전에 자리를 지워줌

    gotoxy(POS\_X, POS\_Y);

    printf("          ");

// 개수와 모양을 맞춰서 출력

    for (int i = 0; i < count; i++)

    {

        gotoxy(POS\_X, POS\_Y);

        switch (symbol)

        {

        case 0:

            printf("♥");

            break;

        case 1:

            printf("★");

            break;

        case 2:

            printf("⊙");

            break;

        case 3:

            printf("∀");

            break;

        case 4:

            printf("◈");

            break;

        }

        POS\_X += 2; // 가로로 달아서 출력하기 때문에 X 좌표 값 변경

    }

}

|  |  |  |
| --- | --- | --- |
| draw\_help.c | [돌아가기](#moc2) | 94 PAGE |

#include <stdio.h> // 표준함수

#include <stdlib.h> // system 사용

#include "common.h" // gotoxy 사용

#include "game.h" // draw\_game 사용

void draw\_help(void)

{

// 도움말 출력

    system("cls");

    draw\_game(5);

    draw\_game(6);

    gotoxy(16, 26);

    system("pause");

    system("cls");

}

|  |  |  |
| --- | --- | --- |
| draw\_end\_game.c | [돌아가기](#moc2) | 95~97 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 게임 엔딩창 출력

void draw\_end\_game(PLAYER\* player, SHOP\_ITEM\* shop\_item)

{

// 게임 결과 출력

    draw\_game(5);

    gotoxy(20, 6);

    printf("게임 결과");

    gotoxy(20, 8);

    printf("최고 점수: %d", user.score);

    gotoxy(20, 10);

    printf("이번판 점수: ");

    set\_color(RED);

    gotoxy(33, 10);

    printf("%d", player->score);

    set\_color(WHITE);

    gotoxy(20, 12);

    printf("기존 포인트: %d", user.point);

    gotoxy(20, 14);

    printf("획득 포인트: ");

    set\_color(RED);

    gotoxy(33, 14);

    // 점수가 -일때는 포인트 없음

    if ((player->score \* 0.5) < 0)

    {

        printf("0");

    }

    else

    {

        printf("%d", (int)(player->score \* 0.5));

    }

    set\_color(WHITE);

    gotoxy(20, 17);

    printf("아이템 사용 목록");

    gotoxy(20, 19);

    printf("생명 추가: ");

    gotoxy(20, 21);

    printf("체력 회복: ");

    gotoxy(20, 23);

    printf("추가 점수: ");

// 아이템 사용 여부 출력

    if (user.item[2] > 0)

    {

        set\_color(YELLOW);

        gotoxy(31, 23);

        printf("◎");

        set\_color(WHITE);

    }

    else

    {

        set\_color(WHITE);

        gotoxy(31, 23);

        printf("×");

    }

    if ((user.item[0] != shop\_item->life\_plus) && (user.item[1] != shop\_item->hp\_recover))

    {

        set\_color(YELLOW);

        gotoxy(31, 19); // 생명 추가

        printf("◎");

        gotoxy(31, 21); // 체력 회복

        printf("◎");

        set\_color(WHITE);

    }

    else if ((user.item[0] == shop\_item->life\_plus) && (user.item[1] != shop\_item->hp\_recover))

    {

        set\_color(WHITE);

        gotoxy(31, 19); // 생명 추가

        printf("×");

        set\_color(YELLOW);

        gotoxy(31, 21); // 체력 회복

        printf("◎");

        set\_color(WHITE);

    }

    else if ((user.item[0] != shop\_item->life\_plus) && (user.item[1] == shop\_item->hp\_recover))

    {

        set\_color(YELLOW);

        gotoxy(31, 19); // 생명 추가

        printf("◎");

        set\_color(WHITE);

        gotoxy(31, 21); // 체력 회복

        printf("×");

    }

    else if ((user.item[0] == shop\_item->life\_plus) && (user.item[1] == shop\_item->hp\_recover))

    {

        gotoxy(31, 19); // 생명 추가

        printf("×");

        gotoxy(31, 21); // 체력 회복

        printf("×");

    }

}

|  |  |  |
| --- | --- | --- |
| draw\_bomb.c | [돌아가기](#moc2) | 98 PAGE |

#include <stdio.h> // puts 사용

#include "common.h" // gotoxy 사용

// 폭탄 출력

void draw\_bomb(int state, int POS\_X, int POS\_Y)

{

// 지워질 차례인지 그려줄 차례인지를 나눠서 출력해줌

    switch (state)

    {

    case 0:

        gotoxy(POS\_X, POS\_Y);

        puts("                         ");

        gotoxy(POS\_X + 11, POS\_Y + 1);

        puts("   ");

        gotoxy(POS\_X + 11, POS\_Y + 2);

        puts("   ");

        gotoxy(POS\_X + 11, POS\_Y + 3);

        puts("   ");

        gotoxy(POS\_X + 7, POS\_Y + 4);

        puts("           ");

        break;

    case 1:

        gotoxy(POS\_X, POS\_Y);

        puts("<<[[\*]]:=:{-\*-}:=:[[\*]]>>");

        gotoxy(POS\_X + 11, POS\_Y + 1);

        puts("| |");

        gotoxy(POS\_X + 11, POS\_Y + 2);

        puts("| |");

        gotoxy(POS\_X + 11, POS\_Y + 3);

        puts("| |");

        gotoxy(POS\_X + 7, POS\_Y + 4);

        puts("<[[::=::]]>");

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| cannon\_move.c | [돌아가기](#moc2) | 99 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 캐논의 좌표를 변경해주고 출력해준다.

void cannon\_move(CANNON\* cannon, SHOP\_ITEM\* shop\_item)

{

// 캐논 최대 개수만큼 반복한다.

    for (int i = 0; i < CANNON\_SIZE; i++)

    {

// 캐논의 상태가 발사 상태라면

        if (cannon->condition[i] == TRUE)

        {

// 원래 자리는 지워준다.

            gotoxy(cannon->pos\_x[i], cannon->pos\_y[i]);

            printf("  ");

// 좌표 변경

            cannon->pos\_y[i]--;

// 새로 그려준다.

            gotoxy(cannon->pos\_x[i], cannon->pos\_y[i]);

            set\_color(shop\_item->cannon\_color);

            puts(shop\_item->cannon\_design);

            set\_color(WHITE);

        }

    }

}

|  |  |  |
| --- | --- | --- |
| cannon\_status.c | [돌아가기](#moc2) | 100 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구초제 사용

// 캐논의 상태를 관리

void cannon\_status(CANNON\* cannon)

{

// 캐논 최대 개수 만큼 반복

    for (int i = 0; i < CANNON\_SIZE; i++)

    {

// 캐논이 발사 상태이고 Y 좌표가 0일 때

        if (cannon->condition[i] == TRUE && cannon->pos\_y[i] == 0)

        {

// 자리를 지워주고 FALSE 상태로 바꾼다.

            gotoxy(cannon->pos\_x[i], cannon->pos\_y[i]);

            printf("  ");

            cannon->condition[i] = FALSE;

        }

    }

}

|  |  |  |
| --- | --- | --- |
| bomb\_move.c | [돌아가기](#moc2) | 101~102 PAGE |

#include <stdio.h> // 표준 함수

#include "common.h" // 공통함수 사용

#include "game.h" // 구조체, 함수 사용

void bomb\_move(BOMB\* bomb)

{

    int POS\_X = bomb->pos\_x[0]; // 폭탄 X 좌표 값

    int POS\_Y = bomb->pos\_y[0]; // 폭탄 Y 좌표 값

    // 폭탄 0번이 TRUE 상태인지 한번 더 검사하는 이유는

    // 폭탄 1번이 0번보다 느리게 끝나기 때문에

    // 한번 더 검사하지 않으면 폭탄 1번이 끝나는 시점에서

    // 하단에 폭탄 0번이 찍혀버리기 때문

    if (bomb->condition[0] == TRUE)

    {

        // 폭탄 0번 느리게 움직일때

        if ((POS\_Y >= 13 && POS\_Y <= 18) && bomb->move\_interval[0] < 1)

        {

            draw\_bomb(0, POS\_X, POS\_Y);

            bomb->pos\_y[0]--;

            POS\_Y = bomb->pos\_y[0];

            draw\_bomb(1, POS\_X, POS\_Y);

            bomb->move\_interval[0] = 20;

        }

        // 폭탄 0번 빠르게 움직일때

        if (POS\_Y > 18 || POS\_Y < 13)

        {

            draw\_bomb(0, POS\_X, POS\_Y);

            bomb->pos\_y[0]--;

            POS\_Y = bomb->pos\_y[0];

            draw\_bomb(1, POS\_X, POS\_Y);

        }

    }

    // 폭탄 0번이 특정 위치까지 갔을때 폭탄 1번 이동

    if (POS\_Y < 18 || (bomb->condition[0] == FALSE && bomb->condition[1] == TRUE))

    {

        POS\_X = bomb->pos\_x[1];

        POS\_Y = bomb->pos\_y[1];

        // 천천히 움직이는 위치

        if ((POS\_Y >= 14 && POS\_Y <= 19) && bomb->move\_interval[1] < 1)

        {

            draw\_bomb(0, POS\_X, POS\_Y);

            bomb->pos\_y[1]--;

            POS\_Y = bomb->pos\_y[1];

            draw\_bomb(1, POS\_X, POS\_Y);

            bomb->move\_interval[1] = 20;

        }

        // 빨리 움직이는 위치

        if (POS\_Y > 19 || POS\_Y < 14)

        {

            draw\_bomb(0, POS\_X, POS\_Y);

            bomb->pos\_y[1]--;

            POS\_Y = bomb->pos\_y[1];

            draw\_bomb(1, POS\_X, POS\_Y);

        }

        bomb->move\_interval[1]--; // 턴 감소

    }

    bomb->move\_interval[0]--; // 턴 감소

}

|  |  |  |
| --- | --- | --- |
| bomb\_status.c | [돌아가기](#moc2) | 103~105 PAGE |

#include <stdio.h> // printf 사용

#include "game.h" // 구조체 사용

#include "common.h" // gotoxy 사용

void bomb\_status(BOMB\* bomb, ENEMY\* enemy, PLAYER\* player, SHOP\_ITEM\* shop\_item, int state)

{

// 상황에 맞게 검사한다.

    switch (state)

    {

    // 폭탄 - 적 충돌

    case 0:

// 폭탄들이 발사 상태일 때

        if (bomb->condition[0] == TRUE || bomb->condition[1] == TRUE)

        {

// 폭탄 개수 만큼 반복

            for (int i = 0; i < BOMB\_SIZE; i++)

            {

// 적군 개수 만큼 반복

                for (int j = 0; j < ENEMY\_SIZE; j++)

                {

// 적군이 살아 있을 때

                    if (enemy->condition[j] == TRUE)

                    {

// 적군의 Y 좌표와 폭탄의 Y좌표가 같을 때

                        if (bomb->pos\_y[i] == enemy->pos\_y[j] || (enemy->pos\_y[j] - bomb->pos\_y[i] > 0))

                        {

// 적을 지워주고 초기화 한다.

                            gotoxy(enemy->pos\_x[j], enemy->pos\_y[j]);

                            printf("      ");

                            enemy->pos\_x[j] = 0;

                            enemy->pos\_y[j] = 0;

                            enemy->condition[j] = FALSE;

// 적군의 타입별 + 추가 점수 아이템 사용 여부에 따라 획득 점수가 다름

                            switch (enemy->design[j])

                            {

                            case 0:

                                if (shop\_item->buff\_on == TRUE)

                                {

                                    player->score += 100;

                                }

                                else

                                {

                                    player->score += 10;

                                }

                                break;

                            case 1:

                                if (shop\_item->buff\_on == TRUE)

                                {

                                    player->score += 200;

                                }

                                else

                                {

                                    player->score += 20;

                                }

                                break;

                            }

                        }

                    }

                }

            }

        }

        break;

    // 폭탄이 천장에 도달하면 삭제

    case 1:

// 폭탄 0이 살아있고 Y가 0일 때

        if (bomb->condition[0] == TRUE && bomb->pos\_y[0] == 0)

        {

// 꺼짐 상태로 만들고 초기화 한다.

            draw\_bomb(0, bomb->pos\_x[0], bomb->pos\_y[0]);

            bomb->pos\_y[0] = 24;

            bomb->condition[0] = FALSE;

            bomb->move\_interval[0] = 5;

        }

// 폭탄 1이 살아있고 Y가 0일 때

        if (bomb->condition[1] == TRUE && bomb->pos\_y[1] == 0)

        {

// 꺼짐 상태로 만들고 초기화 해준다.

            draw\_bomb(0, bomb->pos\_x[1], bomb->pos\_y[1]);

            bomb->pos\_y[1] = 24;

            bomb->condition[1] = FALSE;

            bomb->move\_interval[1] = 5;

// 폭탄이 꺼지면 화면 전체를 한번 지워준다.

            for (int i = 0; i < 30; i++)

            {

                gotoxy(4, i);

                printf("                                                    ");

            }

        }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| bCannon\_status.c | [돌아가기](#moc2) | 106~107 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 상황에 맞춰서 검사한다.

void bCannon\_status(BOMB\* bomb, int state)

{

    switch (state)

    {

    // 폭탄 캐논 발사

    case 0:

// 폭탄 캐논 수량만큼 반복

        for (int i = 0; i < BOMB\_CANNON\_SIZE; i++)

        {

// 폭탄의 위치가 13~18이고 폭탄 캐논 상태가 FALSE일 때 발사

            if ((bomb->pos\_y[0] >= 13 && bomb->pos\_y[0] <= 18) && bomb->bomb\_cannon[0]->condition[i] == FALSE)

            {

// 좌표를 설정해주고 TRUE 상태로 변경

                bomb->bomb\_cannon[0]->pos\_x[i] = bomb->pos\_x[0];

                bomb->bomb\_cannon[0]->pos\_y[i] = bomb->pos\_y[0] - 2;

                bomb->bomb\_cannon[0]->condition[i] = TRUE;

            }

// 위와 같은 방법

            if ((bomb->pos\_y[1] >= 14 && bomb->pos\_y[1] <= 19) && bomb->bomb\_cannon[1]->condition[i] == FALSE)

            {

                bomb->bomb\_cannon[1]->pos\_x[i] = bomb->pos\_x[1];

                bomb->bomb\_cannon[1]->pos\_y[i] = bomb->pos\_y[1] - 2;

                bomb->bomb\_cannon[1]->condition[i] = TRUE;

            }

        }

        break;

    // 폭탄 캐논 천장 도달

    case 1:

// 폭탄 캐논 수량 만큼 반복

        for (int i = 0; i < BOMB\_CANNON\_SIZE; i++)

        {

// 폭탄 캐논이 천장에 도달하면 삭제

            if (bomb->bomb\_cannon[0]->condition[i] == TRUE && bomb->bomb\_cannon[0]->pos\_y[i] == 0)

            {

                gotoxy(bomb->bomb\_cannon[0]->pos\_x[i], bomb->bomb\_cannon[0]->pos\_y[i]);

                printf("                          ");

                bomb->bomb\_cannon[0]->condition[i] = FALSE;

            }

// 폭탄 캐논이 천장에 도달하면 삭제

            if (bomb->bomb\_cannon[1]->condition[i] == TRUE && bomb->bomb\_cannon[1]->pos\_y[i] == 0)

            {

                gotoxy(bomb->bomb\_cannon[1]->pos\_x[i], bomb->bomb\_cannon[1]->pos\_y[i]);

                printf("                          ");

                bomb->bomb\_cannon[1]->condition[i] = FALSE;

            }

        }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| bCannon\_move.c | [돌아가기](#moc2) | 108 PAGE |

#include <stdio.h> // puts 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 폭탄 캐논을 이동시키고 출력해준다.

void bCannon\_move(BOMB\* bomb)

{

// 폭탄 캐논 수량 만큼 반복

    for (int i = 0; i < BOMB\_CANNON\_SIZE; i++)

    {

// 폭탄 0번의 상태가 발사 상태일 때

        if (bomb->bomb\_cannon[0]->condition[i] == TRUE)

        {

// 기존 위치를 지워주고

            gotoxy(bomb->bomb\_cannon[0]->pos\_x[i], bomb->bomb\_cannon[0]->pos\_y[i]);

            puts("                      ");

            bomb->bomb\_cannon[0]->pos\_y[i]--; // 좌표 변경

// 다시 출력해준다.

            gotoxy(bomb->bomb\_cannon[0]->pos\_x[i], bomb->bomb\_cannon[0]->pos\_y[i]);

            puts("    ㅆ              ㅆ");

        }

// 위와 같음

        if (bomb->bomb\_cannon[1]->condition[i] == TRUE)

        {

            gotoxy(bomb->bomb\_cannon[1]->pos\_x[i], bomb->bomb\_cannon[1]->pos\_y[i]);

            puts("                      ");

            bomb->bomb\_cannon[1]->pos\_y[i]--;

            gotoxy(bomb->bomb\_cannon[1]->pos\_x[i], bomb->bomb\_cannon[1]->pos\_y[i]);

            puts("    ㅆ              ㅆ");

        }

    }

}

|  |  |  |
| --- | --- | --- |
| item\_gen.c | [돌아가기](#moc2) | 109~112 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 드롭 아이템 생성

void item\_gen(DROP\_ITEM\* drop\_item, int frame\_time)

{

// 생성 여부 확인

    int isCreated = FALSE;

// 아이템 개수 만큼 반복

    for (int i = 0; i < ITEM\_SIZE; i++)

    {

        // 캐논 업그레이드 700초에 드랍

// 아래 코드들도 시간만 다르고 동작 방법은 같음

        if (frame\_time == 700 && drop\_item->condition[i] == FALSE)

        {

// 좌표 설정, 생성 TRUE, 아이템 타입 지정

            drop\_item->pos\_x[i] = 10;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 0;

            isCreated = TRUE;

        }

        // 캐논 수량 추가

        if (frame\_time == 900 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 30;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 1;

            isCreated = TRUE;

        }

        // 생명 추가

        if (frame\_time == 1300 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 49;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 2;

            isCreated = TRUE;

        }

        // 체력 회복

        if (frame\_time == 1400 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 21;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 3;

            isCreated = TRUE;

        }

        // 캐논 수량 추가

        if (frame\_time == 2000 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 30;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 1;

            isCreated = TRUE;

        }

        // 체력 회복 추가

        if (frame\_time == 2050 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 11;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 3;

            isCreated = TRUE;

        }

        // 캐논 업그레이드

        if (frame\_time == 2300 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 41;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 0;

            isCreated = TRUE;

        }

        // 캐논 수량 추가

        if (frame\_time == 2600 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 28;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 1;

            isCreated = TRUE;

        }

        // 체력 회복 추가

        if (frame\_time == 2900 && drop\_item->condition[i] == FALSE)

        {

            drop\_item->pos\_x[i] = 51;

            drop\_item->pos\_y[i] = 0;

            drop\_item->condition[i] = TRUE;

            drop\_item->type[i] = 3;

            isCreated = TRUE;

        }

// 생성여부 확인후

        if (isCreated == TRUE)

        {

// 좌표값으로 이동해서 출력

            gotoxy(drop\_item->pos\_x[i], drop\_item->pos\_y[i]);

            switch (drop\_item->type[i])

            {

            case 0:

                printf("[P]");

                break;

            case 1:

                printf("[ⅲ]");

                break;

            case 2:

                printf("[♥]");

                break;

            case 3:

                printf("[★]");

                break;

            }

            break;

        }

    }

}

|  |  |  |
| --- | --- | --- |
| item\_move.c | [돌아가기](#moc2) | 113~114 PAGE |

#include <stdio.h> // printf 사용

#include "game.h" // 구조체 사용

#include "common.h" // gotoxy 사용

// 드롭 아이템 좌표 이동, 출력

void item\_move(DROP\_ITEM\* drop\_item)

{

// 아이템 개수 만큼 반복

    for (int i = 0; i < ITEM\_SIZE; i++)

    {

// 아이템이 생성된 상태일 때

        if (drop\_item->condition[i] == TRUE)

        {

// 아이템이 움직일 턴일 때

            if (drop\_item->move\_interval[i] == 10)

            {

// 기존 위치를 지워주고

                drop\_item->move\_interval[i] = 0;

                gotoxy(drop\_item->pos\_x[i], drop\_item->pos\_y[i]);

                printf("     ");

                drop\_item->pos\_y[i]++; // 좌표값 변경

// 아이템을 그려준다.

                gotoxy(drop\_item->pos\_x[i], drop\_item->pos\_y[i]);

                switch (drop\_item->type[i])

                {

                case 0:

                    printf("[P]");

                    break;

                case 1:

                    printf("[ⅲ]");

                    break;

                case 2:

                    printf("[♥]");

                    break;

                case 3:

                    printf("[★]");

                    break;

                }

            }

// 아이템이 움직일 턴이 아닐때는 턴 값을 증가

            else

            {

                drop\_item->move\_interval[i]++;

            }

        }

    }

}

|  |  |  |
| --- | --- | --- |
| item\_status.c | [돌아가기](#moc2) | 115~118 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 아이템 상태를 관리하는 함수

// 바닥에 충돌하거나 플레이어와 충동할때로 나뉨

void item\_status(DROP\_ITEM\* drop\_item, PLAYER\* player, CANNON\* cannon, SHOP\_ITEM\* shop\_item, int state)

{

    switch (state)

    {

    // 아이템 - 플레이어 충돌

    case 0:

// 아이템 개수 만큼 반복

        for (int i = 0; i < ITEM\_SIZE; i++)

        {

// 아이템이 TRUE 상태일때

            if (drop\_item->condition[i] == TRUE)

            {

// 아이템의 Y 값과 플레이어의 Y값이 같을 때

                if (drop\_item->pos\_y[i] == player->pos\_y)

                {

// X값들도 비교해서 범위안에 들어갈 때

                    if (drop\_item->pos\_x[i] >= player->pos\_x && drop\_item->pos\_x[i] <= player->pos\_x + 5)

                    {

// 아이템을 지워주고

                        gotoxy(drop\_item->pos\_x[i], drop\_item->pos\_y[i]);

                        printf("   ");

// 아이템을 꺼버리고

                        drop\_item->condition[i] = FALSE;

// 추가 점수 아이템 여부에 따라 점수를 획득하고

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 50;

                        }

                        else

                        {

                            player->score += 10;

                        }

// 아이템 타입에 따라

                        switch (drop\_item->type[i])

                        {

                        case 0:

// 캐논을 업그레이드 한다.

// 캐논 색상에 따라 모양이 다름

                            if (cannon->upgrade == LEVEL0)

                            {

                                cannon->upgrade = LEVEL1;

                                switch (shop\_item->cannon\_color)

                                {

                                case WHITE:

                                    shop\_item->cannon\_design = CANNON5;

                                    break;

                                case RED:

                                    shop\_item->cannon\_design = CANNON6;

                                    break;

                                case BLUE:

                                    shop\_item->cannon\_design = CANNON7;

                                    break;

                                case YELLOW:

                                    shop\_item->cannon\_design = CANNON8;

                                    break;

                                }

                            }

// 캐논을 업그레이드 한다.

// 캐논 색상에 따라 모양이 다름

                            else if (cannon->upgrade == LEVEL1)

                            {

                                cannon->upgrade = LEVEL2;

                                switch (shop\_item->cannon\_color)

                                {

                                case WHITE:

                                    shop\_item->cannon\_design = CANNON9;

                                    break;

                                case RED:

                                    shop\_item->cannon\_design = CANNON10;

                                    break;

                                case BLUE:

                                    shop\_item->cannon\_design = CANNON11;

                                    break;

                                case YELLOW:

                                    shop\_item->cannon\_design = CANNON12;

                                    break;

                                }

                            }

                            break;

// 캐논 최대 수량을 늘린다.

                        case 1:

                            if (player->cannon\_limit < 5)

                            {

                                player->cannon\_limit++;

                            }

                            break;

// 플레이어의 생명 추가

                        case 2:

                            if (player->life < 3)

                            {

                                player->life++;

                            }

// 가득 찼다면 점수를 올려준다.

                            else

                            {

                                player->score += 100;

                            }

                            break;

// 플레이어 체력 회복

                        case 3:

                            if (player->hp < 3)

                            {

                                player->hp++;

                            }

// 가득 찼다면 점수를 올려준다.

                            else

                            {

                                player->score += 90;

                            }

                            break;

                        }

                    }

                }

            }

        }

        break;

    // 아이템 바닥 도달

    case 1:

// 아이템 사이즈 만큼 반복

        for (int i = 0; i < ITEM\_SIZE; i++)

        {

// 아이템이 생성된 상태라면

            if (drop\_item->condition[i] == TRUE)

            {

// 아이템 위치가 바닥일 때

                if (drop\_item->pos\_y[i] == 29)

                {

// 아이템을 지워주고 FALSE로 바꾼다.

                    gotoxy(drop\_item->pos\_x[i], drop\_item->pos\_y[i]);

                    printf("    ");

                    drop\_item->condition[i] = FALSE;

                }

            }

        }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| enemy\_gen.c | [돌아가기](#moc2) | 119~124 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 적군 생성 함수

void enemy\_gen(ENEMY\* enemy, int frame\_time)

{

// 생성 여부 판별을 위한 변수

    int is\_created = FALSE;

// 적의 최대 수량 만큼 반복

    for (int i = 0; i < ENEMY\_SIZE; i++)

    {

// i번째 적이 생성되지 않았다면

        if (enemy->condition[i] == FALSE)

        {

// 게임 시간 조건에 맞다면

            // 패턴1 – 왼쪽으로 이동을 생성한다.

// 아래 코드들도 이와 동일한 구조임

            if ((frame\_time >= 45 && frame\_time <= 315) && (frame\_time - 45) % 90 == 0)

            {

// 좌표 설정

                enemy->pos\_x[i] = 6;

                enemy->pos\_y[i] = 0;

                enemy->condition[i] = TRUE; // 생성상태로 변경

                enemy->design[i] = 0; // 적군 기체 모양 설정

                enemy->pattern[i] = 0; // 대각선 우하향 & 직선 공격

                enemy->move\_count[i] = 0; // 방향 전환 초기값

                enemy->move\_interval[i] = 0; // 움직임 속도 초기값

                enemy->hp[i] = 3; // 체력 설정

                is\_created = TRUE;

            }

            // 패턴1 – 오른쪽 이동 생성

            if ((frame\_time >= 90 && frame\_time <= 360) && (frame\_time - 90) % 90 == 0)

            {

                enemy->pos\_x[i] = 50;

                enemy->pos\_y[i] = 1;

                enemy->condition[i] = TRUE;

                enemy->design[i] = 0;

                enemy->pattern[i] = 1; // 대각선 좌하향 & 직선 공격

                enemy->move\_count[i] = 0;

                enemy->move\_interval[i] = 0;

                enemy->hp[i] = 3;

                is\_created = TRUE;

            }

            // 패턴2 왼쪽 등장 생성

            if (frame\_time == 500 || frame\_time == 2950)

            {

                enemy->pos\_x[i] = 18;

                enemy->pos\_y[i] = 1;

                enemy->pos\_x[i + 1] = 13;

                enemy->pos\_y[i + 1] = 0;

                enemy->pos\_x[i + 2] = 23;

                enemy->pos\_y[i + 2] = 0;

                for (int j = i; j < i+3; j++)

                {

                    enemy->condition[j] = TRUE;

                    enemy->design[j] = 1;

                    enemy->pattern[j] = 2; // 아래로 직진 & 우측 대각선 공격

                    enemy->move\_count[j] = 0;

                    enemy->move\_interval[j] = 0;

                    enemy->hp[j] = 5;

                }

                is\_created = TRUE;

            }

            // 패턴2 오른쪽 등장 생성

            if (frame\_time == 600 || frame\_time == 2620)

            {

                enemy->pos\_x[i] = 36;

                enemy->pos\_y[i] = 1;

                enemy->pos\_x[i + 1] = 31;

                enemy->pos\_y[i + 1] = 0;

                enemy->pos\_x[i + 2] = 41;

                enemy->pos\_y[i + 2] = 0;

                for (int j = i; j < i+3; j++)

                {

                    enemy->condition[j] = TRUE;

                    enemy->design[j] = 1;

                    enemy->pattern[j] = 3; // 아래로 직진 & 좌측 대각선 공격

                    enemy->move\_count[j] = 0;

                    enemy->move\_interval[j] = 0;

                    enemy->hp[j] = 5;

                }

                is\_created = TRUE;

            }

            // 패턴3 왼쪽 등장 생성

            if (frame\_time == 780 || frame\_time == 2290)

            {

                enemy->pos\_x[i] = 13;

                enemy->pos\_y[i] = 0;

                enemy->pos\_x[i + 1] = 25;

                enemy->pos\_y[i + 1] = 0;

                for (int j = i; j < i + 2; j++)

                {

                    enemy->condition[j] = TRUE;

                    enemy->design[j] = 2;

                    enemy->pattern[j] = 4; // 등장해서 얼마간 공격하고 다시 위로 올라가서 사라짐

                    enemy->move\_count[j] = 0;

                    enemy->move\_interval[j] = 0;

                    enemy->hp[j] = 5;

                }

                is\_created = TRUE;

            }

            // 패턴3 오른쪽 등장 생성

            if (frame\_time == 980 || frame\_time == 2400)

            {

                enemy->pos\_x[i] = 31;

                enemy->pos\_y[i] = 0;

                enemy->pos\_x[i + 1] = 43;

                enemy->pos\_y[i + 1] = 0;

                for (int j = i; j < i + 2; j++)

                {

                    enemy->condition[j] = TRUE;

                    enemy->design[j] = 2;

                    enemy->pattern[j] = 4; // 등장해서 얼마간 공격하고 다시 위로 올라가서 사라짐

                    enemy->move\_count[j] = 0;

                    enemy->move\_interval[j] = 0;

                    enemy->hp[j] = 5;

                }

                is\_created = TRUE;

            }

            // 패턴4 중앙 등장 생성

            if (frame\_time == 1225)

            {

                enemy->pos\_x[i] = 28;

                enemy->pos\_y[i] = 0;

                enemy->condition[i] = TRUE;

                enemy->design[i] = 3;

                enemy->pattern[i] = 5; // 직선으로 내려오면서 정면, 좌우로 공격

                enemy->move\_count[i] = 0;

                enemy->move\_interval[i] = 0;

                enemy->hp[i] = 10;

                is\_created = TRUE;

            }

            // 패턴4 왼쪽 등장 생성

            if (frame\_time == 1510 || frame\_time == 2680)

            {

                enemy->pos\_x[i] = 12;

                enemy->pos\_y[i] = 0;

                enemy->condition[i] = TRUE;

                enemy->design[i] = 3;

                enemy->pattern[i] = 5; // 직선으로 내려오면서 정면, 좌우로 공격

                enemy->move\_count[i] = 0;

                enemy->move\_interval[i] = 0;

                enemy->hp[i] = 10;

                is\_created = TRUE;

            }

            // 패턴4 오른쪽 등장 생성

            if (frame\_time == 1720 || frame\_time == 3100)

            {

                enemy->pos\_x[i] = 42;

                enemy->pos\_y[i] = 0;

                enemy->condition[i] = TRUE;

                enemy->design[i] = 3;

                enemy->pattern[i] = 5; // 직선으로 내려오면서 정면, 좌우로 공격

                enemy->move\_count[i] = 0;

                enemy->move\_interval[i] = 0;

                enemy->hp[i] = 10;

                is\_created = TRUE;

            }

            // 패턴1 – 왼쪽 이동 생성

            if ((frame\_time >= 1850 && frame\_time <= 2120) && (frame\_time - 45) % 90 == 0)

            {

                enemy->pos\_x[i] = 6;

                enemy->pos\_y[i] = 0;

                enemy->condition[i] = TRUE;

                enemy->design[i] = 0;

                enemy->pattern[i] = 0; // 대각선 우하향 & 직선 공격

                enemy->move\_count[i] = 0;

                enemy->move\_interval[i] = 0;

                enemy->hp[i] = 3;

                is\_created = TRUE;

            }

            // 패턴1 – 오른쪽 이동 생성

            if ((frame\_time >= 1895 && frame\_time <= 2165) && (frame\_time - 90) % 90 == 0)

            {

                enemy->pos\_x[i] = 50;

                enemy->pos\_y[i] = 1;

                enemy->condition[i] = TRUE;

                enemy->design[i] = 0;

                enemy->pattern[i] = 1; // 대각선 좌하향 & 직선 공격

                enemy->move\_count[i] = 0;

                enemy->move\_interval[i] = 0;

                enemy->hp[i] = 3;

                is\_created = TRUE;

            }

        }

// 적이 생성됬다면 출력

        if (is\_created == TRUE)

        {

            gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

            switch (enemy->design[i])

            {

            case 0:

                printf("<WvW>");

                break;

            case 1:

                printf("<XVX>");

                break;

            case 2:

                printf("[TWT]");

                break;

            case 3:

                printf("(pWq)");

                break;

            }

            break;

        }

    }

}

|  |  |  |
| --- | --- | --- |
| enemy\_move.c | [돌아가기](#moc2) | 125~131 PAGE |

#include <stdio.h> // printf 사용

#include "common.h" // gotoxy 사용

#include "game.h" // 구조체 사용

// 적군의 움직임 담당

void enemy\_move(ENEMY\* enemy)

{

// 적군 수량 만큼 반복

    for (int i = 0; i < ENEMY\_SIZE; i++)

    {

// 적이 생성된 상태라면

        if (enemy->condition[i] == TRUE)

        {

// 기존 위치를 지워주고

            gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

            printf("     ");

            // 오른쪽 대각선 이동 패턴

            if (enemy->pattern[i] == 0)

            {

// 움직일 턴이 왔다면

                if (enemy->move\_interval[i] == 6)

                {

                    // 아래로 내려갈 차례인지 옆으로 갈 차례인지 검사

                    if (enemy->move\_count[i] == 3)

                    {

// 좌표 변경, 방향 전환

                        enemy->pos\_y[i]++;

                        enemy->move\_count[i] = 0;

                    }

                    else

                    {

                        // 벽에 붙었는지 검사 & 붙었으면 move\_count는 올릴 필요 없음

                        if (enemy->pos\_x[i] > 48)

                        {

                            enemy->pos\_y[i]++;

                        }

                        else

                        {

// 벽에 안 붙었다면 좌표 변경, 방향 전환

                            enemy->pos\_x[i]++;

                            enemy->move\_count[i]++;

                        }

                    }

                    enemy->move\_interval[i] = 0;

                }

                else

                {

// 움직일 턴이 아니라면 턴 증가

                    enemy->move\_interval[i]++;

                }

            }

            // 왼쪽 대각선 이동 패턴

            if (enemy->pattern[i] == 1)

            {

// 움직일 턴이라면

                if (enemy->move\_interval[i] == 9)

                {

                    // 아래로 내려갈 차례인지 옆으로 갈 차례인지 검사

                    if (enemy->move\_count[i] == 2)

                    {

// 좌표 변경후 방향 전환

                        enemy->pos\_y[i]++;

                        enemy->move\_count[i] = 0;

                    }

                    else

                    {

                        // 벽에 붙었는지 검사 & 붙었으면 move\_count는 올릴 필요 없음

                        if (enemy->pos\_x[i] < 10)

                        {

                            enemy->pos\_y[i]++;

                        }

                        else

                        {

// 안 붙었으면 좌표 변경, 방향 전환

                            enemy->pos\_x[i] -= 2;

                            enemy->move\_count[i]++;

                        }

                    }

                    enemy->move\_interval[i] = 0; // 턴 초기화

                }

                else

                {

// 턴 증가

                    enemy->move\_interval[i]++;

                }

            }

            // 아래로 직진 패턴

            if (enemy->pattern[i] == 2 || enemy->pattern[i] == 3)

            {

// 특정 위치까지 왔는지 검사

                if (enemy->pos\_y[i] > 12)

                {

                    // 자기 턴이 오면 진행

                    if (enemy->move\_interval[i] == 2)

                    {

                        enemy->pos\_y[i]++; // Y값 변경

// 패턴 2,3은 특정 위치부터는 빨리 내려가는데 그때도 캐논을 발사하기 때문에 움직임 횟수 관리

                        if (enemy->move\_count[i] == 3)

                        {

                            enemy->move\_count[i] = 0;

                        }

                        else

                        {

                            enemy->move\_count[i]++;

                        }

                        // 턴 초기화

                        enemy->move\_interval[i] = 0;

                    }

                    else

                    {

// 턴 증가

                        enemy->move\_interval[i]++;

                    }

                }

                else // 느리게 움직일 때

                {

                    // 자기 턴이 오면 진행

                    if (enemy->move\_interval[i] == 20)

                    {

                        enemy->pos\_y[i]++; // Y값 증가

// 캐논 발사 관리를 위해 움직임 횟수 체크

                        if (enemy->move\_count[i] == 3)

                        {

                            enemy->move\_count[i] = 0;

                        }

                        else

                        {

                            enemy->move\_count[i]++;

                        }

                        // 턴 초기화

                        enemy->move\_interval[i] = 0;

                    }

                    else

                    { // 턴증가

                        enemy->move\_interval[i]++;

                    }

                }

            }

            // 내려갔다가 일정 시간 정지했다가 다시 돌아오는 패턴

            if (enemy->pattern[i] == 4)

            {

// 방향 전환 신호가 왔는지 검사

                if (enemy->move\_count[i] == 1)

                {

                    // 자기 턴이 오면 진행

                    if ((enemy->move\_interval[i] > 150) && (enemy->move\_interval[i] % 100 == 0))

                    {

                        // 다시 뒤로 돌아가는 패턴은 천장이 바닥이기 때문에 여기서 소멸을 관리

                        if (enemy->pos\_y[i] == 0)

                        {

// 위치를 공백으로 지우고 값 초기화

                            gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

                            printf("     ");

                            enemy->pos\_x[i] = 0;

                            enemy->pos\_y[i] = 0;

                            enemy->condition[i] = FALSE;

                        }

// 아직 천장이 아니라면 올라간다.

                        else

                        {

                            enemy->pos\_y[i]--;

                        }

                        // 다시 뒤로 돌아가야하기 때문에 턴수를 초기화 하지 않음

                    }

// 턴수 추가

                    else

                    {

                        enemy->move\_interval[i]++;

                    }

                }

                else

                {

                    // 자기 턴이 오면 진행

                    if (enemy->move\_interval[i] == 5)

                    {

// Y값을 증가시켜 내려간다.

                        enemy->pos\_y[i]++;

// 특정 위치까지 내려가면 방향 전환

                        if (enemy->pos\_y[i] > 4)

                        {

                            enemy->move\_count[i]++;

                        }

                        // 턴 초기화

                        enemy->move\_interval[i] = 0;

                    }

                    else

                    {

// 턴 증가

                        enemy->move\_interval[i]++;

                    }

                }

            }

            // 일직선으로 내려가는 패턴 – 정면, 좌우 공격

            if (enemy->pattern[i] == 5)

            {

// 특정 위치까지는 천천히 내려감

                if (enemy->pos\_y[i] > 16)

                {

// 움직일 턴이면

                    if (enemy->move\_interval[i] == 1)

                    {

// 한칸 내려가고 턴 초기화

                        enemy->pos\_y[i]++;

                        enemy->move\_interval[i] = 0;

                    }

                    else

                    {

// 턴 증가

                        enemy->move\_interval[i]++;

                    }

                }

// 천천히 내려갈 위치라면

                else

                {

// 움직일 턴에

                    if (enemy->move\_interval[i] == 35)

                    {

// 한칸 내려가고 턴 초기화

                        enemy->pos\_y[i]++;

                        enemy->move\_interval[i] = 0;

                    }

// 턴 증가

                    else

                    {

                        enemy->move\_interval[i]++;

                    }

                }

            }

// 좌표값 변경후에 적군을 그려준다.

            gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

            switch (enemy->design[i])

            {

            case 0:

                printf("<WvW>");

                break;

            case 1:

                printf("<XVX>");

                break;

            case 2:

                printf("[TWT]");

                break;

            case 3:

                printf("(pWq)");

                break;

            }

        }

    }

}

|  |  |  |
| --- | --- | --- |
| enemy\_status.c | [돌아가기](#moc2) | 132~138 PAGE |

#include <stdio.h> // printf 사용

#include "game.h" // 구조체 사용

#include "common.h" // gotoxy 사용

// 적군 상태를 관리하는 함수

void enemy\_status(ENEMY\* enemy, BOMB\* bomb, PLAYER\* player, CANNON\* cannon, SHOP\_ITEM\*  shop\_item, int state)

{

// 상황에 맞게 구분

    switch (state)

    {

    // 적군이 바닥에 도달하면 삭제

    case 0:

// 적군 최대 개수 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 적이 살아있고 바닥일 때

            if (enemy->condition[i] == TRUE && enemy->pos\_y[i] == 29)

            {

// 적을 지우고 초기화 한다.

                gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

                printf("     ");

                enemy->pos\_x[i] = 0;

                enemy->pos\_y[i] = 0;

                enemy->condition[i] = FALSE;

            }

        }

        break;

    // 플레이어 - 적 충돌

    case 1:

// 적 개수 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 적의 Y값과 플레이어의 Y값이 같을 때

            if (player->pos\_y == enemy->pos\_y[i])

            {

// 적이 플레이어안에 겹친다면

                if ((enemy->pos\_x[i] <= player->pos\_x && enemy->pos\_x[i] + 5 >= player->pos\_x) && enemy->pos\_x[i] - 4 <= player->pos\_x)

                {

                    // game.c -> while(1) 마지막에 게임 오버를 체크하며 속도가 빠르기 때문에 게임 오버 처리는 따로 하지 않음

// 플레리어의 생명이 남아 있다면

                    if (player->life > 0)

                    {

// 생명 감소 및 초기 시작점으로 이동

                        player->life--;

                        player->pos\_x = 26;

                        player->pos\_y = 26;

                        player->score -= 100; // 점수 차감

// 적군을 지우고 초기화한다.

                        gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

                        printf("      ");

                        enemy->pos\_x[i] = 0;

                        enemy->pos\_y[i] = 0;

                        enemy->condition[i] = FALSE;

// 폭탄을 자동 발사한다.

                        bomb->condition[0] = TRUE;

                        bomb->condition[1] = TRUE;

                        break;

                    }

                    break;

                }

            }

        }

        break;

    // 캐논 - 적 충돌

    case 2:

// 캐논 수량 만큼 반복

        for (int i = 0; i < CANNON\_SIZE; i++)

        {

// 적 수량 만큼 반복

            for (int j = 0; j < ENEMY\_SIZE; j++)

            {

// 캐논의 Y값과 적의 Y값이 같다면

                if ((cannon->condition[i] == TRUE && enemy->condition[j] == TRUE) && cannon->pos\_y[i] == enemy->pos\_y[j])

                {

// 캐논이 적과 겹친다면

                    if ((enemy->pos\_x[j] <= cannon->pos\_x[i] - 1) && (enemy->pos\_x[j] + 5) >= cannon->pos\_x[i] + 1)

                    {

// 적의 체력이 남아 있다면

                        if (enemy->hp[j] > 0)

                        {

// 캐논을 삭제하고 적의 체력을 감소 시킨다.

                            gotoxy(cannon->pos\_x[i], cannon->pos\_y[i]);

                            printf("  ");

                            cannon->condition[i] = FALSE;

                            enemy->hp[j]--;

                        }

// 적의 체력이 없다면

                        else

                        {

// 적을 지워주고

                            gotoxy(enemy->pos\_x[j], enemy->pos\_y[j]);

                            printf("      ");

// 적을 초기화하고 삭제한다.

                            enemy->pos\_x[j] = 0;

                            enemy->pos\_y[j] = 0;

                            enemy->condition[j] = FALSE;

// 적의 타입에 따라 점수 확복

                            switch (enemy->design[j])

                            {

                            case 0:

                                if (shop\_item->buff\_on == TRUE)

                                {

                                    player->score += 300;

                                }

                                else

                                {

                                    player->score += 100;

                                }

                                break;

                            case 1:

                                if (shop\_item->buff\_on == TRUE)

                                {

                                    player->score += 500;

                                }

                                else

                                {

                                    player->score += 200;

                                }

                                break;

                            case 2:

                                if (shop\_item->buff\_on == TRUE)

                                {

                                    player->score += 500;

                                }

                                else

                                {

                                    player->score += 200;

                                }

                                break;

                            }

// 캐논을 지워주고 삭제한다.

                            gotoxy(cannon->pos\_x[i], cannon->pos\_y[i]);

                            printf("  ");

                            cannon->condition[i] = FALSE;

                        }

                    }

                }

            }

        }

        break;

    // 폭탄 캐논 - 적 충돌

    case 3:

// 적 개수 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 폭탄 캐논 수량 만큼 반복

            for (int j = 0; j < BOMB\_CANNON\_SIZE; j++)

            {

// 폭탄 캐논의 Y 값과 적의 Y값이 겹친다면

                if ((bomb->bomb\_cannon[0]->condition[j] == TRUE && enemy->condition[i] == TRUE) && bomb->bomb\_cannon[0]->pos\_y[j] == enemy->pos\_y[i])

                {

// 적을 지워준다.

                    gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

                    printf("      ");

                    enemy->pos\_x[i] = 0;

                    enemy->pos\_y[i] = 0;

                    enemy->condition[i] = FALSE;

// 적의 타입에 따라 점수 확보

// 폭탄으로 제거 할때는 점수 감소

                    switch (enemy->design[i])

                    {

                    case 0:

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 150;

                        }

                        else

                        {

                            player->score += 50;

                        }

                        break;

                    case 1:

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 200;

                        }

                        else

                        {

                            player->score += 60;

                        }

                        break;

                    case 2:

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 170;

                        }

                        else

                        {

                            player->score += 53;

                        }

                        break;

                    }

                }

// 1번 폭탄 또한 0번 폭탄과 동일한 방법

// Y값이 겹친다면

                if ((bomb->bomb\_cannon[1]->condition[j] == TRUE && enemy->condition[i] == TRUE) && bomb->bomb\_cannon[1]->pos\_y[j] == enemy->pos\_y[i])

                {

// 적을 지워주고 초기화 한다.

                    gotoxy(enemy->pos\_x[i], enemy->pos\_y[i]);

                    printf("      ");

                    enemy->pos\_x[i] = 0;

                    enemy->pos\_y[i] = 0;

                    enemy->condition[i] = FALSE;

// 적의 타입에 따라 점수 지급

// 폭탄으로 제거 할때는 점수 감소

                    switch (enemy->design[i])

                    {

                    case 0:

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 150;

                        }

                        else

                        {

                            player->score += 50;

                        }

                        break;

                    case 1:

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 200;

                        }

                        else

                        {

                            player->score += 60;

                        }

                        break;

                    case 2:

                        if (shop\_item->buff\_on == TRUE)

                        {

                            player->score += 170;

                        }

                        else

                        {

                            player->score += 53;

                        }

                        break;

                    }

                }

            }

        }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| eCannon\_status.c | [돌아가기](#moc2) | 139~146 PAGE |

#include <stdio.h> // printf 사용

#include "game.h" // 구조체 사용

#include "common.h" // gotoxy 사용

// 적군 캐논 상태 관리 함수

void eCannon\_status(ENEMY\* enemy, PLAYER\* player, BOMB\* bomb, CANNON\* cannon, int state)

{

    switch (state)

    {

    // 적군 캐논 발사

    case 0:

// 적 개수 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 적이 살아 있다면

            if (enemy->condition[i] == TRUE)

            {

// 적이 아직 상단에 있다면

                if (enemy->pos\_y[i] < 25)

                {

// 적 캐논 수량 만큼 반복

                    for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

                    {

// 적 캐논이 발사되지 않은 상태라면

                        if (enemy->enemy\_cannon[i]->condition[j] == FALSE)

                        {

// 적이 패턴 1,0 이라면

                            if (enemy->pattern[i] == 1 || enemy->pattern[i] == 0)

                            {

// 적 캐논의 발사 빈도와 적의 움직임 횟수를 검사하여

                                if (enemy->enemy\_cannon[i]->move\_interval > 60 && (enemy->move\_count[i] == 1 || enemy->move\_count[i] == 3))

                                {

// 적의 캐논에 좌표 값을 저장하고

                                    enemy->enemy\_cannon[i]->pos\_x[j] = enemy->pos\_x[i] + 2;

                                    enemy->enemy\_cannon[i]->pos\_y[j] = enemy->pos\_y[i] + 1;

// 생성한다.

                                    enemy->enemy\_cannon[i]->condition[j] = TRUE;

                                    enemy->enemy\_cannon[i]->move\_interval = 0;

// 좌표에다가 캐을 출력해준다.

                                    gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                                    printf("\*");

                                }

                            }

// 패턴이 2,3 이라면

                            if (enemy->enemy\_cannon[i]->move\_interval > 60 && (enemy->pattern[i] == 2 || enemy->pattern[i] == 3))

                            {

// 움직임 횟수가 3일 때 발사

                                if (enemy->move\_count[i] == 3)

                                {

                                    enemy->enemy\_cannon[i]->pos\_x[j] = enemy->pos\_x[i] + 2;

                                    enemy->enemy\_cannon[i]->pos\_y[j] = enemy->pos\_y[i] + 1;

                                    enemy->enemy\_cannon[i]->condition[j] = TRUE;

                                    enemy->enemy\_cannon[i]->move\_interval = 0;

                                    gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                                    printf("¤");

                                }

                            }

// 패턴이 4이며 방향 전환을 했는데 움직임 턴이 아직 안 왔다면

                            if ((enemy->pattern[i] == 4 && enemy->move\_count[i] == 1) && (enemy->move\_interval[i] < 175))

                            {

// 좌표값과 초기값을 지정하고 생성

                                enemy->enemy\_cannon[i]->pos\_x[j] = enemy->pos\_x[i] + 2;

                                enemy->enemy\_cannon[i]->pos\_y[j] = enemy->pos\_y[i] + 1;

                                enemy->enemy\_cannon[i]->move\_interval = 0;

                                enemy->enemy\_cannon[i]->condition[j] = TRUE;

                                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                                printf("W");

                            }

// 패턴이 5이고 특정 위치 안이라면

                            if (enemy->enemy\_cannon[i]->move\_interval > 60 && (enemy->pattern[i] == 5 && (enemy->pos\_y[i] > 1 && enemy->pos\_y[i] < 18)))

                            {

// 전방, 좌우로 좌표값을 주고 생성한다.

                                enemy->enemy\_cannon[i]->pos\_x[j] = enemy->pos\_x[i] + 2;

                                enemy->enemy\_cannon[i]->pos\_y[j] = enemy->pos\_y[i] + 1;

                                enemy->enemy\_cannon[i]->pos\_x[j + 1] = enemy->pos\_x[i] + 6;

                                enemy->enemy\_cannon[i]->pos\_y[j + 1] = enemy->pos\_y[i];

                                enemy->enemy\_cannon[i]->pos\_x[j + 2] = enemy->pos\_x[i] - 3;

                                enemy->enemy\_cannon[i]->pos\_y[j + 2] = enemy->pos\_y[i];

                                enemy->enemy\_cannon[i]->condition[j] = TRUE;

                                enemy->enemy\_cannon[i]->condition[j+1] = TRUE;

                                enemy->enemy\_cannon[i]->condition[j+2] = TRUE;

                                enemy->enemy\_cannon[i]->move\_pattern[j + 1] = 1;

                                enemy->enemy\_cannon[i]->move\_pattern[j + 2] = 2;

                                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                                printf("\*");

                                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j+1], enemy->enemy\_cannon[i]->pos\_y[j + 1]);

                                printf("\*");

                                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j + 2], enemy->enemy\_cannon[i]->pos\_y[j + 2]);

                                printf("\*");

                                enemy->enemy\_cannon[i]->move\_interval = 0;

                            }

                        }

                    }

// 캐논의 발사 빈도를 조절

                    enemy->enemy\_cannon[i]->move\_interval++;

                }

            }

        }

        break;

    // 적 캐논 플레이어 충돌

    case 1:

// 적 개수 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 적 캐논 수량 만큼 반복

            for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

            {

// 적의 캐논이 플레이어의 Y값과 겹친다면

                if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->enemy\_cannon[i]->pos\_y[j] == player->pos\_y)

                {

// 적의 캐논이 플레이어의 X값과 겹친다면

                    if ((enemy->enemy\_cannon[i]->pos\_x[j] >= player->pos\_x) && (enemy->enemy\_cannon[i]->pos\_x[j] <= player->pos\_x + 5))

                    {

// 적의 캐논을 지워준다.

                        gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                        printf("  ");

                        enemy->enemy\_cannon[i]->condition[j] = FALSE;

// 플레이어의 체력이 남아있다면

                        if (player->hp > 0)

                        {

                            player->hp--; // 체력 감소

                        }

                        else // 체력이 없다면

                        {

// 생명이 없으면 game.c에서 게임 오버 검사할 때 걸린다.

// 생명이 남아 있다면

                            if (player->life > 0)

                            {

// 생명을 차감하고

// 체력을 회복하고

// 점수를 차감한다.

                                player->life--;

                                player->hp = 3;

                                player->score -= 100;

// 적 캐논을 지워주고

                                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                                printf("     ");

// 플레이어의 위치를 초기화하고

                                player->pos\_x = 26;

                                player->pos\_y = 26;

// 폭탄을 발사한다.

                                bomb->condition[0] = TRUE;

                                bomb->condition[1] = TRUE;

                            }

                        }

                    }

                }

            }

        }

        break;

    // 적 캐논 바닥 도달

    case 2:

// 적 수량 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 적 캐논 수량 만큼 반복

            for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

            {

// 적 캐논이 바닥에 도달하면

                if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->enemy\_cannon[i]->pos\_y[j] == 29)

                {

// 적 캐논을 지워주고

                    gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                    printf("    ");

                    enemy->enemy\_cannon[i]->condition[j] = FALSE;

                }

// 적 캐논이 왼쪽 벽에 부딪히면

                if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->enemy\_cannon[i]->pos\_x[j] == 5)

                {

// 적 캐논을 지워준다.

                    gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                    printf("  ");

                    enemy->enemy\_cannon[i]->condition[j] = FALSE;

                }

// 적 캐논이 오른쪽 벽에 부딪히면

                if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->enemy\_cannon[i]->pos\_x[j] == 55)

                {

// 적 캐논을 지워준다.

                    gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                    printf("  ");

                    enemy->enemy\_cannon[i]->condition[j] = FALSE;

                }

            }

        }

        break;

    // 적 캐논 플레이어 캐논 충돌

    case 3:

// 적 수량 만큼 반복

        for (int i = 0; i < ENEMY\_SIZE; i++)

        {

// 적 캐논 수량 만큼 반복

            for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

            {

// 캐논 수량 만큼 반복

                for (int k = 0; k < CANNON\_SIZE; k++)

                {

// 적 캐논과 플레이어 캐논 둘다 생성됬다면

                    if (enemy->enemy\_cannon[i]->condition[j] == TRUE && cannon->condition[k] == TRUE)

                    {

// 서로 Y값이 겹친다면

                        if (enemy->enemy\_cannon[i]->pos\_y[j] == cannon->pos\_y[k])

                        {

// X 값이 겹친다면

                            if (enemy->enemy\_cannon[i]->pos\_x[j] == cannon->pos\_x[k])

                            {

// 그 자리를 지워주고 둘다 삭제한다.

                                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                                printf("  ");

                                enemy->enemy\_cannon[i]->condition[j] = FALSE;

                                cannon->condition[k] = FALSE;

                            }

                        }

                    }

                }

            }

        }

        break;

    // 폭탄 발사시 적 캐논 폭탄 앞에서 삭제

    case 4:

// 폭탄을 발사 했을 때

        if (bomb->condition[0] == TRUE || bomb->condition[1] == TRUE)

        {

// 적 수량만큼 반복

            for (int i = 0; i < ENEMY\_SIZE; i++)

            {

// 적 캐논 수량 만큼 반복

                for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

                {

// 적 캐논을 폭탄의 앞이든 뒤든 삭제한다.

                    if (enemy->enemy\_cannon[i]->condition[j] == TRUE && (enemy->enemy\_cannon[i]->pos\_y[j] >= bomb->pos\_y[0] || enemy->enemy\_cannon[i]->pos\_y[j] >= bomb->pos\_y[1]))

                    {

                        gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                        puts("  ");

                        enemy->enemy\_cannon[i]->condition[j] = FALSE;

                    }

                }

            }

        }

        break;

    }

}

|  |  |  |
| --- | --- | --- |
| eCannon\_move.c | [돌아가기](#moc2) | 147~149 PAGE |

#include <stdio.h> // printf 사용

#include "game.h" // 구조체 사용

#include "common.h" // gotoxy 사용

// 적군 캐논 이동 함수

void eCannon\_move(ENEMY\* enemy)

{

// 적군 수량 만큼 반복

    for (int i = 0; i < ENEMY\_SIZE; i++)

    {

// 적군 캐논 수량 만큼 반복

        for (int j = 0; j < ENEMY\_CANNON\_SIZE; j++)

        {

// 적군이 생성된 상태이고 패턴이 1,0이라면

            if (enemy->enemy\_cannon[i]->condition[j] == TRUE && (enemy->pattern[i] == 1 || enemy->pattern[i] == 0))

            {

// 기존 위치를 지워준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("  ");

// 패턴 1,0은 직진 공격이기 때문에 Y값만 변경

                enemy->enemy\_cannon[i]->pos\_y[j]++;

// 새로 그려준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("\*");

            }

// 패턴이 2라면

            if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->pattern[i] == 2)

            {

// 기존 위치를 지워준다

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("  ");

// 패턴2는 대각선 공격이기 때문에 x,y값 변경

                enemy->enemy\_cannon[i]->pos\_x[j]++;

                enemy->enemy\_cannon[i]->pos\_y[j]++;

// 새로 그려준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("¤");

            }

// 패턴이 3이라면

            if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->pattern[i] == 3)

            {

// 기존 위치를 지워주고

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("  ");

// 패턴 3 또한 대각선 공격이기에 x,y 값 변경

                enemy->enemy\_cannon[i]->pos\_x[j]--;

                enemy->enemy\_cannon[i]->pos\_y[j]++;

// 다시 그려준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("¤");

            }

// 패턴 4라면

            if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->pattern[i] == 4)

            {

// 기존 위치를 지워준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("  ");

// 공격은 일직선 공격이기에 y값만 증가시킨다.

                enemy->enemy\_cannon[i]->pos\_y[j]++;

// 다시 그려준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("W");

            }

// 패턴이 5라면

            if (enemy->enemy\_cannon[i]->condition[j] == TRUE && enemy->pattern[i] == 5)

            {

// 기존 위치를 지워준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("  ");

// 패턴5는 전방, 좌우로 공격하기 때문에

// 캐논의 움직임 패턴에 따라 변경하는 값이 다르다.

                if (enemy->enemy\_cannon[i]->move\_pattern[j] == 0)

                {

                    enemy->enemy\_cannon[i]->pos\_y[j]++;

                }

                else if(enemy->enemy\_cannon[i]->move\_pattern[j] == 1)

                {

                    enemy->enemy\_cannon[i]->pos\_x[j]++;

                }

                else if (enemy->enemy\_cannon[i]->move\_pattern[j] == 2)

                {

                    enemy->enemy\_cannon[i]->pos\_x[j]--;

                }

// 다시 그려준다.

                gotoxy(enemy->enemy\_cannon[i]->pos\_x[j], enemy->enemy\_cannon[i]->pos\_y[j]);

                printf("\*");

            }

        }

    }

}