

2021-1 C++ 프로그래밍 실습과제 04

(1) 각 문제에 대한 분석과 및 해결 방법

4.1. 교재 190~196쪽의 코드를 참고하여 퍼즐 게임을 구현하라.

(1) 게임은 교재와 같이 4x4로 구현해도 되고, 3x3이나 5x5 등 다른 크기로 구현해도 됨

(2) 랭킹은 10위까지 관리되도록 구현하며, 랭킹 파일은 가급적 이진 파일로 저장할 것. (이진 파일이 어려운 경우 텍스트 파일로 저장해도 됨)

(3) 점수 계산은 이동 횟수가 아니라 시간을 기준으로 처리하도록 함. 즉, 조금이라도 빨리 해결한 사람이 더 높은 랭킹을 가짐

(보너스 최대 2점) 203쪽의 실습문제 4에서 게임 확장 기능 구현

--> (2) replay 기능

--> (3-4) 게임 저장/다시 시작 기능

--> (5) 아스키 아트를 이용한 그림 퍼즐 기능

[문제분석 및 해결방법] : 높은 품질의 콘솔 게임을 구현하기 위해서 콘솔 입출력 유틸리티들을 설계 및 제작했고 4x4 정수형 배열을 이용하여 아스키 아트 또는 숫자 블록이 화면에 그려질 수 있도록 했다. 플레이어 이름, 렌더링 타입(아스키 아트, 숫자), 시간, 움직인 횟수를 저장하는 게임 저장용 구조체를 만들고 파일 입출력을 통해 저장과 다시 시작 기능을 만들었고, 게임이 끝난 뒤에는 거품 정렬을 이용해서 상위 10등까지의 점수를 랭킹 저장용 구조체에 넣고 파일 입출력을 통해 랭킹을 저장하였다.

(2) 자신이 구현한 주요 코드

4.1. 교재 190~196쪽의 코드를 참고하여 퍼즐 게임을 구현하라.

```
<consoleutils.h>
namespace ConsoleUtils {
    void set_locale_korean();
    void set_locale_default();
    void set_encoding_cp949();
    void set_encoding_utf8();
    void set_boost_mode();
    void set_default_mode();
    short get_print_color();
    void set_print_color(int tbcolor);
    COORD get_cursor_position();
    void set_cursor_position(int x, int y);
    bool get_cursor_visibility();
    void set_cursor_visibility(bool visibility);
    void set_console_size(int cols, int lines);
    void set_cursor_size(int cursor_size);
    void set_console_title(const char *title);
    void clear_console();
    void xyprintf(int x, int y, char *format, ...);
    void xywprintf(int x, int y, wchar_t *format, ...);
    int get_key_input();
    int wait_with_handler(unsigned long ms, int (*handler)(int, void *), void *data);
}
```

```

<BufferingManager.h>
class BufferingManager {
public:
    static BufferingManager &instance();
    CHAR_INFO *char_buffer;
    void init(int width, int height);
    HANDLE get_current_screen_buffer(bool next = false);
    void draw();
    int get_width();
    int get_height();
    void BufferingManager::clear();
    void BufferingManager::xywprintf(int x, int y, wchar_t *format, ...);
    void BufferingManager::wprintf(wchar_t *format, ...);
    void BufferingManager::set_print_color(short color);
    short BufferingManager::get_print_color();
    bool BufferingManager::get_cursor_visibility();
    void BufferingManager::set_cursor_visibility(bool visibility);
    COORD get_cursor_position();
    void BufferingManager::set_cursor_position(int x, int y);
private:
    BufferingManager();
    static BufferingManager *instance_;
    int width;
    int height;
    int current_screen_buffer;
    HANDLE stdout_handle;
    HANDLE screen_buffer1;
    HANDLE screen_buffer2;
    COORD coord_buffer_size;
    COORD coord_buffer_coord;
    COORD cursor_position = {0, 0};
    CONSOLE_CURSOR_INFO cursor_info;
    short print_color = ConsoleUtils::get_print_color();
    SMALL_RECT write_rect;
};

```

```

<BufferingManager.cpp>
BufferingManager *BufferingManager::instance_ = NULL;
BufferingManager &BufferingManager::instance() {
    if (instance_ == NULL) {
        instance_ = new BufferingManager();
    }
    return *instance_;
}

```

```

/*
    https://www.ascii-art-generator.org/
    Image to Color Ascii Art - HTML with tables
    by JHLee
*/

const MATRIX_COL =256;
function to2DIndex(x, y) {
    return y * MATRIX_COL + x;
}
function to2DY(index) {
    return Math.floor(index / MATRIX_COL);
}
function to2DX(index) {
    return index % MATRIX_COL;
}
let color = {
    "#000000": 0,
    "#000aaa": 1,
    "#00aa00": 2,
    "#0aaaaa": 3,
    "#aa0000": 4,
    "#aa00aa": 5,
    "#aa5500": 6,
    "#aaaaaa": 7,
    "#555555": 8,
    "#5555ff": 9,
    "#55ff55": 10,
    "#55ffff": 11,
    "#ff5555": 12,
    "#ff55ff": 13,
    "#ffff55": 14,
    "#ffffff": 15
};
let array =new Array(24).fill(0).map(() =>new Array(40).fill(0)); ;
function rgb2hex(rgb) {
    rgb = rgb.match(/^rgbW((Wd +),Ws *(Wd +),Ws *(Wd +)W)$/);
    function hex(x) {
        return ("0"+parseInt(x).toString(16)).slice(-2);
    }
    return "#"+ hex(rgb[1]) + hex(rgb[2]) + hex(rgb[3]);
}
let x =0;
let y =0;
[...document.querySelectorAll('span, br')].forEach(e => {
    if (e.tagName === 'SPAN') {
        let attr = color[rgb2hex(e.style.color)] + color[rgb2hex(e.style.backgroundColor)] *16;
        e.textContent.split('').forEach(c => {
            array[x][y++] = to2DIndex(attr, c.charCodeAt(0));
        });
    } else {
        y =0;
        x++;
    }
});
console.log(`${array.map(e =>e.join(' ')).join(' \n')}`);

```

```

<ASCIIArt.h>
namespace ASCIIArt {
    const int KORYONGI[
        ASCIART_WIDTH * ASCIART_HEIGHT] = {12023, 15095, 15095, 15095, 15095, 15095, 12023, 15095, 15095, 15095,
        15095, 15095, 15095, 41207, 9592, 11910, 15351, 41207, 15095, 15095,
        15095, 15095, 41207, 29823, 29830, 14583, 12023, 15095, 15095, 15095,
    ...
    };
    const int LENA[
        ASCIART_WIDTH * ASCIART_HEIGHT] = {14440, 14440, 14408, 14468, 14468, 14468, 14470, 14408, 14440, 9605,
        14444, 11909, 14440, 14424, 14440,
        14456, 14456, 14456, 14454, 15239, 14440, 14408, 14440, 21381, 14436,
    ....
    };
}

```

```

<ColorTextUtils.cpp>
void ColorTextUtils::draw_color_text(int cursor_x, int cursor_y, int
start_x, int start_y, int size_x, int size_y, const int array[], int width,
int height) {
    BufferingManager &bm = BufferingManager::Instance();
    bm.set_cursor_position(cursor_x, cursor_y);
    short color = bm.get_print_color();
    for (int y = start_y; y < start_y + size_y; y++) {
        for (int x = start_x; x < start_x + size_x; x++) {
            bm.set_print_color(to_2d_x(array[x + y * width]));
            wchar_t c = to_2d_y(array[x + y * width]);
            bm.wprintf(L"%c", c);
        }
        bm.set_cursor_position(cursor_x, ++cursor_y);
    }
    bm.set_print_color(color);
}

```

```

<InterfaceUtils.h>
/**
 * @brief 메뉴 렌더링 데이터들을 포함하는 구조체
 */
typedef struct
{
    int x;
    int y;
    wchar_t *name;
    wchar_t **list;
    int length;
    int current_index;
    short outline_tbcolor;
    short name_tbcolor;
    short element_tbcolor;
    short selected_tbcolor;
    short non_selected_tbcolor;
} MenuData;
/**
 * @brief 프롬프트 렌더링 데이터들을 포함하는 구조체
 */
typedef struct
{
    int x;
    int y;
    wchar_t *message;
    int rlen;
    short outline_tbcolor;
    short message_tbcolor;
    short text_tbcolor;
} PromptData;
namespace InterfaceUtils {
    static BufferingManager & bm = BufferingManager::instance();
    void draw_menu(MenuData *data);
    int run_menu(MenuData *, bool);
    void draw_prompt(PromptData *data);
    wchar_t *run_prompt(PromptData *);
}
<Game.cpp>
MainMenu Game::run_main_menu() {
    MenuData menu;
    menu.name = L "      메뉴";
    wchar_t **list = (wchar_t **) malloc(sizeof(wchar_t *) *6);
    list[0] = L "게임 시작 (숫자)";
    list[1] = L "게임 시작 (코룡)";
    list[2] = L "게임 시작 (레나)";
    list[3] = L "      불러오기";
    list[4] = L "랭킹 (리플레이)";
    list[5] = L "      나가기";
    menu.list = list;
    menu.length =6;
    menu.current_index =0;
    menu.element_tbcolor = TO_TBCOLOR(BLUE, RED);
    menu.name_tbcolor = TO_TBCOLOR(WHITE, BLACK);
    menu.outline_tbcolor = TO_TBCOLOR(GRAY, BLACK);
    menu.selected_tbcolor = TO_TBCOLOR(LIGHT_JADE, LIGHT_PURPLE);
    menu.non_selected_tbcolor = TO_TBCOLOR(WHITE, GRAY);
    menu.x =30;
    menu.y =10;
    int index = InterfaceUtils::run_menu(&menu, true);
    free(list);
    return (MainMenu) index;
}

```

```

<beeputils.h>
#define PLAY_BEEP(o, v, t) Beep((int)((o) * pow(1.06f, v)), t)
#define PLAY_VICTORY_FANFARE() W
RA(OCTAVE_5, 200); W
RA(OCTAVE_5, 100); W
RA(OCTAVE_5, 300); W
SI(OCTAVE_5, 200); W
RA(OCTAVE_5, 200); W
SI(OCTAVE_5, 200); W
DO(OCTAVE_6, 200); W
DO(OCTAVE_6, 100); W
DO(OCTAVE_6, 700);

#define PLAY_EXIT_SOUND() W
FA_S(OCTAVE_6, 20); W
FA_S(OCTAVE_6, 20);

#define PLAY_FAIL_SOUND() W
FA_S(OCTAVE_6, 20); W

#define PLAY_SELECTION_SOUND1() W
RA(OCTAVE_4, 20);

```

```

<ConsoleUtils.cpp>
/**
 * @brief wait 중 키 핸들링을 같이 수행한다.
 * @description wait를 UNIT_WAIT로 쪼개어 수행하고 도중 키가 눌렸는지 검사하여 handler에 전달한다.
 *          handler(int c)의 반환값이 1 이상일 때 wait를 중지하고 handler의 반환 값을 반환한다.
 *          키 핸들링이 일어나지 않는 경우에는 항상 -1을 반환한다. handler가 NULL인 경우 항상 즉시 입력을 반환한
 *          다.
 * @param ms 대기할 시간 (밀리초)
 * @param handler 텍스트의 배경 색깔
 * @param data 자유형 데이터
 */
int ConsoleUtils::wait_with_handler(unsigned long ms, int (*handler)(int, void *), void *data)
{
    struct timespec begin, end;
    SystemUtils::clock_gettime(CLOCK_MONOTONIC, &begin);
    while (true)
    {
        SystemUtils::wait(UNIT_WAIT);
        if (kbhit())
        {
            int c = _getwch();
            c = (c == 0xE0 || c == 0) ? _getwch() : c;
            if (handler == NULL)
            {
                return c;
            }
            else
            {
                int result = handler(c, data);
                if (result > -1)
                {
                    return result;
                }
            }
        }
        SystemUtils::clock_gettime(CLOCK_MONOTONIC, &end);
        if (MILLISECOND_DIFF(begin, end) >= ms)
        {
            return -1;
        }
    }
}

```

```

<Game.cpp>
int Game::handle_game_key(int c, void *param) {
    void **list = (void **) param;
    Game *game = (Game *) list[0];
    GameData *data = (GameData *) list[1];
    switch (c) {
        case UP_KEY:
            if (data -> puzzle.move(UP)) {
                memcpy(data->save.current_map, data -> puzzle.map, sizeof(int) * DIM * DIM);
                data->save.move_log[data -> save.move ++] = L 'U';
                PLAY_SELECTION_SOUND1();
            } else {
                PLAY_FAIL_SOUND();
            }
            break;
        ...
        case ESCAPE_KEY:
            data->save.move_log[data -> save.move] = L 'W0';
            PLAY_EXIT_SOUND();
            return HANDLE_SAVE_AND_EXIT;
    }
    game->draw_puzzle(data);
    game->bm.draw();
    if (data -> puzzle.is_done()) {
        return HANDLE_GAME_DONE;
    }
    return HANDLE_NULL;
}

```

```

<Game.cpp>
void *param[] = {this, data};
struct timespec begin, end;
SystemUtils::clock_gettime(CLOCK_MONOTONIC, &begin);
while (true) {
    int handle_code = ConsoleUtils::wait_with_handler(10, handle_game_key, param);
    SystemUtils::clock_gettime(CLOCK_MONOTONIC, &end);
    double diff = MILLISECOND_DIFF(begin, end) / 1000.0;
    if (handle_code == HANDLE_NULL) {
        bm.xywprintf(40, 27, L "Time: %.1fs", diff + data -> save.time);
        bm.xywprintf(40, 28, L "Move: %d", data -> save.move);
        bm.draw();
    }
}

```

```

RankingData ranking_data;
FILE *fp = fopen("ranking.bin", "rb");
if (fp !=NULL) {
    fread(&ranking_data, sizeof(RankingData), 1, fp);
    fclose(fp);
} else {
    ranking_data.last_index =0;
}
if (ranking_data.last_index <10) {
    ranking_data.list[ranking_data.last_index++] = data ->save;
} else {
    if (ranking_data.list[ranking_data.last_index -1].time > data ->save.time) {
        ranking_data.list[ranking_data.last_index -1] = data ->save;
    }
}
}
GameSave temp;
for (int i =0; i < ranking_data.last_index; i++) {
    for (int j =0; j < ranking_data.last_index -1; j++) {
        if (ranking_data.list[j].time >
            ranking_data.list[j +1].time) {
            temp = ranking_data.list[j];
            ranking_data.list[j] = ranking_data.list[j +1];
            ranking_data.list[j +1] = temp;
        }
    }
}
fp = fopen("ranking.bin", "wb");
fwrite(&ranking_data, sizeof(RankingData), 1, fp);
fclose(fp);

```

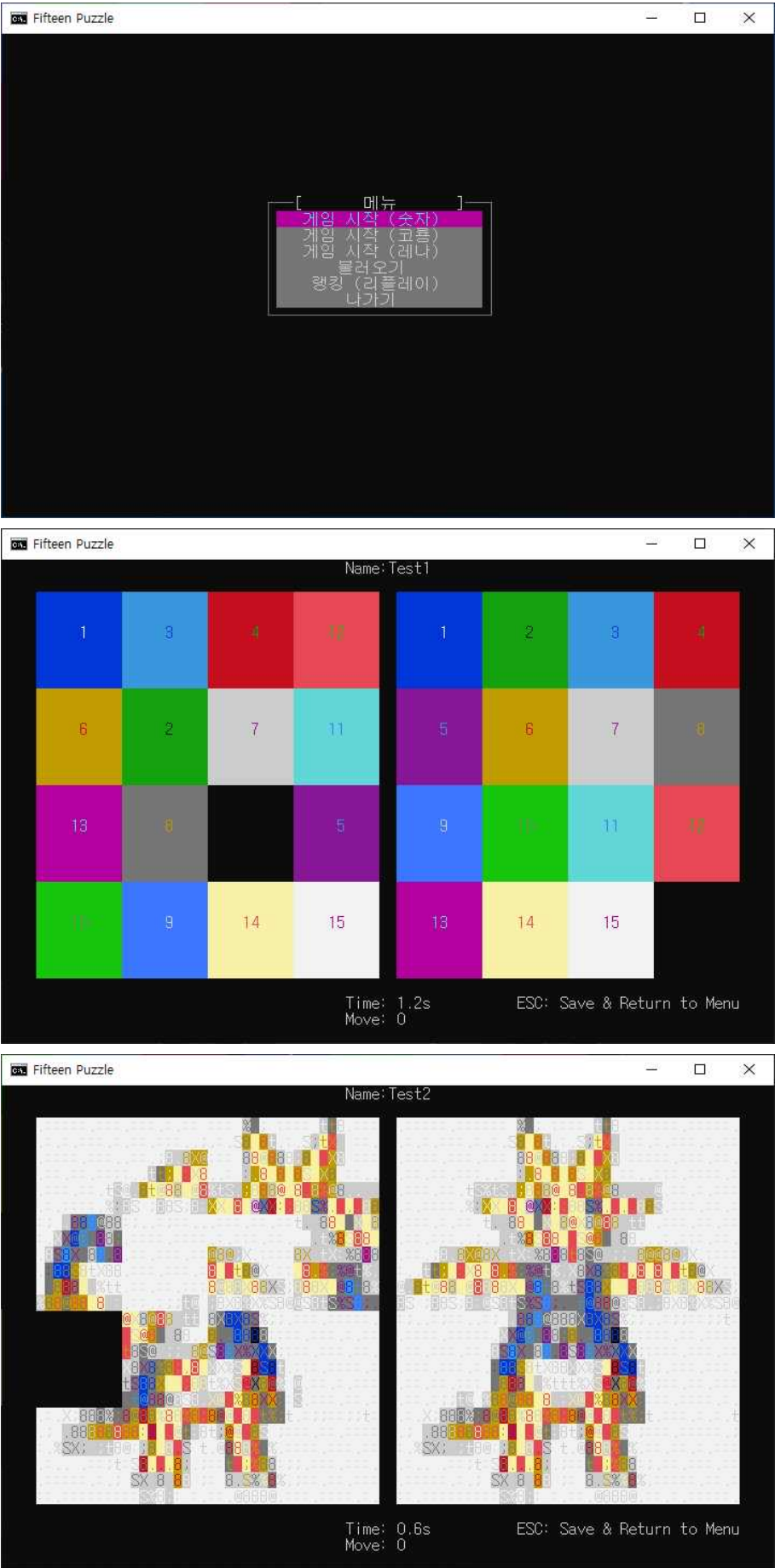
```

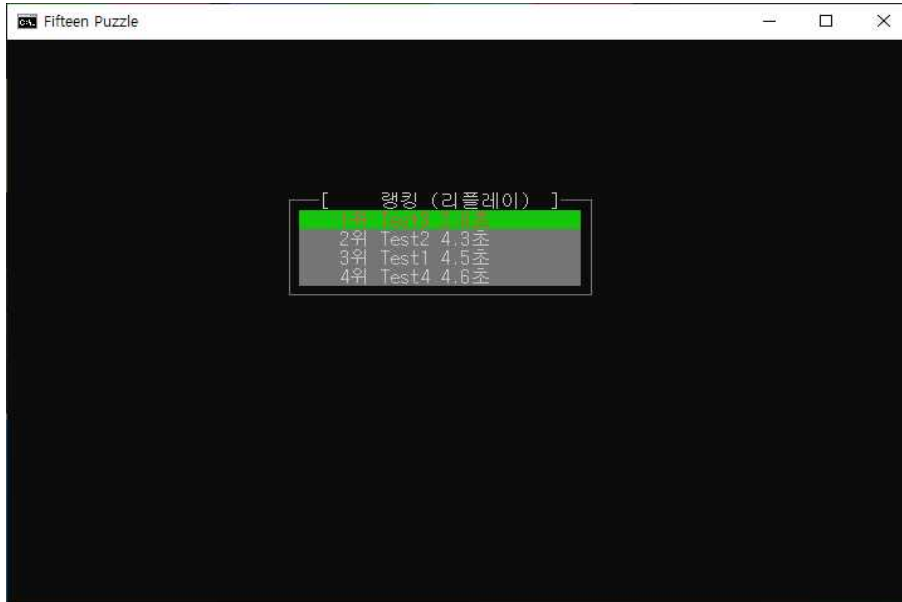
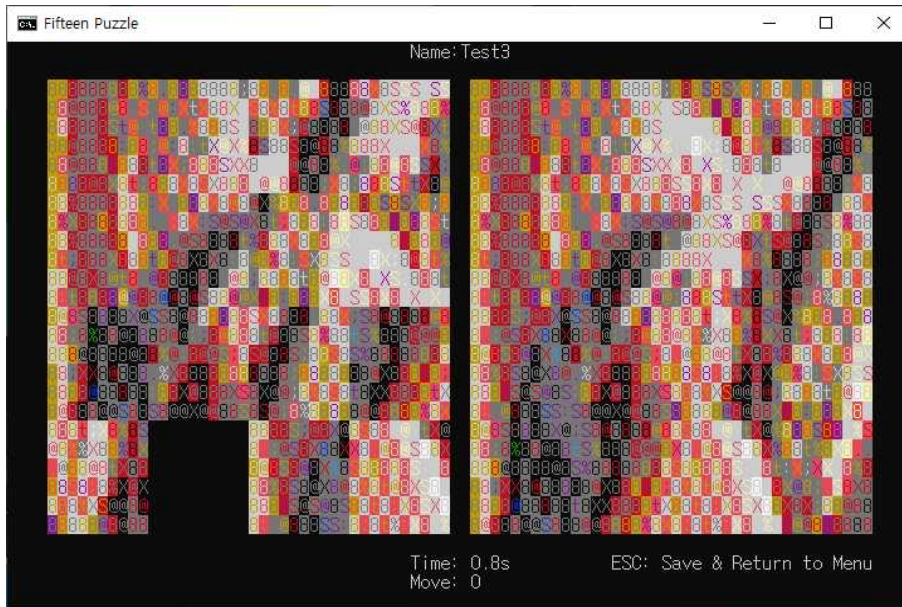
<Puzzle.cpp>
#define DIM 4
#define END_PUZZLE_INDEX (DIM *DIM -1)
#define TO_PUZZLE_DIRECTION(c) W
( (PuzzleDirection) W
( W
c == 'U' ? (UP) : W
c == 'D' ? (DOWN) : W
c == 'L' ? (LEFT) : W
c == 'R' ? (RIGHT) : W
NULL W
) W
)
#define TO_REVERSE_PUZZLE_DIRECTION(c) W
( (PuzzleDirection) W
( W
c == 'D' ? (UP) : W
c == 'U' ? (DOWN) : W
c == 'R' ? (LEFT) : W
c == 'L' ? (RIGHT) : W
NULL W
) W
)
typedef enum
{
    LEFT, RIGHT, UP, DOWN
} PuzzleDirection;
class Puzzle {
public:
    int map[DIM][DIM];
    void init_puzzle();
    bool move(PuzzleDirection dir);
    void shuffle_once();
    bool is_done();
};

```

(3) 다양한 입력에 대한 테스트 결과

4.1. 교재 190~196쪽의 코드를 참고하여 퍼즐 게임을 구현하라.





(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

4.1. 교재 190~196쪽의 코드를 참고하여 퍼즐 게임을 구현하라.

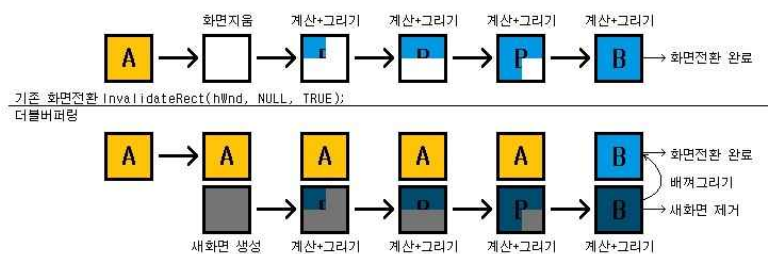
콘솔 환경에서 게임을 구현하기 위해서는 많은 입출력 함수들이 필요했고 색깔 출력, 지정 위치에 텍스트 출력, 창 크기 조절, 커서 핸들링 등의 함수를 구현하였다.

```
<consoleutils.h>
namespace ConsoleUtils {
    void set_locale_korean();
    void set_locale_default();
    void set_encoding_cp949();
    void set_encoding_utf8();
    void set_boost_mode();
    void set_default_mode();
    short get_print_color();
    void set_print_color(int tbcolor);
    COORD get_cursor_position();
    void set_cursor_position(int x, int y);
    bool get_cursor_visibility();
    void set_cursor_visibility(bool visibility);
    void set_console_size(int cols, int lines);
    void set_cursor_size(int cursor_size);
    void set_console_title(const char *title);
    void clear_console();
    void xyprintf(int x, int y, char *format, ...);
    void xywprintf(int x, int y, wchar_t *format, ...);
    int get_key_input();
    int wait_with_handler(unsigned long ms, int (*handler)(int, void *), void *data);
}
```

consoleutils에 있는 함수의 대다수는 C프로그래밍2 과목 수강간 본인이 직접 작성하여 [팀 프로젝트](#)에 사용하였던 [consoleutils.c](#)를 C++ 버전으로 포팅한 것으로, [정적 유틸 클래스](#)의 형태로 사용하기 위해 namespace를 이용하여 해당 구조로 포팅하였다.

이전에 팀 프로젝트에 사용하였던 콘솔 입출력 함수들은 모두 쓸만했지만 해결하지 못했던 고질적인 문제가 하나 있었는데, 화면을 지우고 다시 그리는 작업을 수행하면 깜빡거림(flickering)이나 찢어짐(tearing) 현상이 나타나기 때문에 화면을 지우지 않고 텍스트로 구성된 UI 컴포넌트들을 최대한 덧그리는 방법을 사용해야 해서 프로그램의 렌더링 부분에 불필요한 부분과 성능상의 오버헤드가 생기는 것이 문제였다.

```
<BufferingManager.h>
class BufferingManager {
public:
    static BufferingManager &instance();
    CHAR_INFO *char_buffer;
    void init(int width, int height);
    HANDLE get_current_screen_buffer(bool next = false);
    void draw();
    int get_width();
    int get_height();
    void BufferingManager::clear();
    void BufferingManager::xywprintf(int x, int y, wchar_t *format, ...);
    void BufferingManager::wprintf(wchar_t *format, ...);
    void BufferingManager::set_print_color(short color);
    short BufferingManager::get_print_color();
    bool BufferingManager::get_cursor_visibility();
    void BufferingManager::set_cursor_visibility(bool visibility);
    COORD get_cursor_position();
    void BufferingManager::set_cursor_position(int x, int y);
private:
    BufferingManager();
    static BufferingManager *instance_;
    int width;
    int height;
    int current_screen_buffer;
    HANDLE stdout_handle;
    HANDLE screen_buffer1;
    HANDLE screen_buffer2;
    COORD coord_buffer_size;
    COORD coord_buffer_coord;
    COORD cursor_position = {0, 0};
    CONSOLE_CURSOR_INFO cursor_info;
    short print_color = ConsoleUtils::get_print_color();
    SMALL_RECT write_rect;
};
```



그 문제를 해결하기 위해서 이번 과제 간에는 WinAPI를 이용하여 더블 버퍼링을 사용하기로 결정하였고

```
<BufferingManager.cpp>
BufferingManager *BufferingManager::instance_ =NULL;
BufferingManager &BufferingManager::instance() {
    if (instance_ ==NULL) {
        instance_ =new BufferingManager();
    }
    return *instance_;
}
```

프로그램 전반에서 잦은 전역 호출이 있을 것을 염두에 두고 Singleton 디자인 패턴을 사용하여 클래스를 설계하였다. 또한 스크린 버퍼의 원하는 x, y 위치에 포맷 텍스트를 출력해주는 함수가 있으면 좋을 것 같아, 와이드 문자 (wchar_t) 버전의 가변인자 출력 함수인 void BufferingManager::xywprintf(int x, int y, wchar_t *format, ...)도 구현하였다. 이 함수를 통해 버퍼에 쓴 문자들은 void BufferingManager::draw()로 한꺼번에 원하는 시점에 출력이 가능하게 설계하여 더블 버퍼를 사용하는 렌더링을 용이하게 할 수 있게 하였다.

콘솔 출력 부분은 이렇게 마무리 되었지만, 아스키 아트를 퍼즐 게임에 단색으로 사용하자니 심미성도 떨어지고 각 퍼즐 블록당 구별도 더 어려워 질 것 같아서, 컬러 아스키 아트를 사용하기로 결정했다.



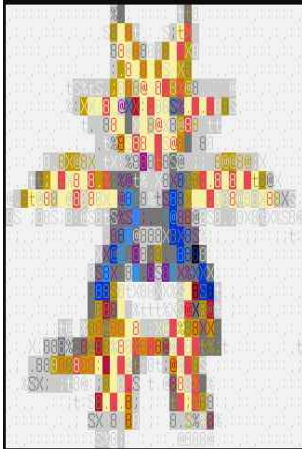
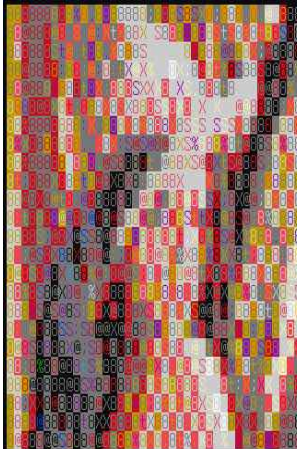
대부분 [컬러 아스키 아트 생성 사이트](#)들은 ANSI Escape Code 포맷 정도를 지원하는데, 윈도우 환경에서는 지원이 미흡하여 그대로 사용할 수 없기도 하고 직접 만든 더블 버퍼링 렌더러와 호환이 되지 않기 때문에, 사이트에서 HTML & CSS로 렌더링하여 보여주는 컬러 아스키 아트를 자바스크립트로 직접 파싱해서 배열 코드를 출력하기로 했다.

```
/*
    https://www.ascii-art-generator.org/
    Image to Color Ascii Art - HTML with tables
    by JHLee
*/

const MATRIX_COL =256;
function to2DIndex(x, y) {
    return y * MATRIX_COL + x;
}
function to2DY(index) {
    return Math.floor(index / MATRIX_COL);
}
function to2DX(index) {
    return index % MATRIX_COL;
}
let color = {
    "#000000": 0,
    "#0000aa": 1,
    "#00aa00": 2,
    "#00aaaa": 3,
    "#aa0000": 4,
    "#aa00aa": 5,
    "#aa5500": 6,
    "#aaaaaa": 7,
    "#555555": 8,
    "#5555ff": 9,
    "#55ff55": 10,
    "#55ffff": 11,
    "#ff5555": 12,
    "#ff55ff": 13,
    "#ffff55": 14,
    "#ffffff": 15
};
let array =new Array(24).fill(0).map(() =>new Array(40).fill(0));
function rgb2hex(rgb) {
    rgb = rgb.match(/^rgbW((Wd +),Ws *(Wd +),Ws *(Wd +)W)$/);
    function hex(x) {
        return ("0"+parseInt(x).toString(16)).slice(-2);
    }
    return "#"+ hex(rgb[1]) + hex(rgb[2]) + hex(rgb[3]);
}
let x =0;
let y =0;
[...document.querySelectorAll('span, br')].forEach(e => {
    if (e.tagName ==='SPAN') {
        let attr = color[rgb2hex(e.style.color)] + color[rgb2hex(e.style.backgroundColor)] *16;
        e.textContent.split("").forEach(c => {
            array[x][y++] = to2DIndex(attr, c.charCodeAt(0));
        });
    } else {
        y =0;
        x++;
    }
});
console.log(`${array.map(e =>e.join(' ')).join(' \n')}`);
```

```
<ASCIIArt.h>
namespace ASCIIArt {
    const int KORYONGI[
        ASCIIART_WIDTH * ASCIIART_HEIGHT] = {12023, 15095, 15095, 15095, 15095, 15095, 12023, 15095, 15095, 15095,
        15095, 15095, 15095, 41207, 9592, 11910, 15351, 41207, 15095, 15095,
        15095, 15095, 41207, 29823, 29830, 14583, 12023, 15095, 15095, 15095,
        ...
    }
    const int LENA[
        ASCIIART_WIDTH * ASCIIART_HEIGHT] = {14440, 14440, 14408, 14468, 14468, 14468, 14470, 14408, 14440, 9605,
        14444, 11909, 14440, 14424, 14440,
        14456, 14456, 14456, 14454, 15239, 14440, 14408, 14440, 21381, 14436,
        ....
    }
}
```

(실제로 컬러 아스키 아트가 저장된 모습)

	코롱이	레나 포르센
원본		
아스키 아트		

```
<ColorTextUtils.cpp>
void ColorTextUtils::draw_color_text(int cursor_x, int cursor_y, int
start_x, int start_y, int size_x, int size_y, const int array[], int width,
int height) {
    BufferingManager &bm = BufferingManager::Instance();
    bm.set_cursor_position(cursor_x, cursor_y);
    short color = bm.get_print_color();
    for (int y = start_y; y < start_y + size_y; y++) {
        for (int x = start_x; x < start_x + size_x; x++) {
            bm.set_print_color(to_2d_x(array[x + y * width]));
            wchar_t c = to_2d_y(array[x + y * width]);
            bm.wprintf(L"%c", c);
        }
        bm.set_cursor_position(cursor_x, ++cursor_y);
    }
    bm.set_print_color(color);
}
```

(저장된 컬러 아스키 아트를 지정한 콘솔 위치에 분할 출력하기 위해서 만든 ColorTextUtils::draw_color_text)

터미널에서 사용되는 색상이 글자색, 바탕색 16가지로, 256가지의 색깔을 가지기 때문에

[CSR\(Compressed sparse row\)](#) 방법을 이용해서 아스키 문자 코드와 문자 색깔을 한 개의 정수로 만들어 저장하는 방법을 떠올렸고 이를 40x24(아스키 아트 크기) 크기의 배열에 저장하여 사용하였다.

이미지는 [학교 마스코트 코롱이](#)와 디지털 화상 처리의 견본용 이미지로 많이 사용되는 [레나 포르센](#)을 사용하였다.

```

<InterfaceUtils.h>
/**
 * @brief 메뉴 렌더링 데이터들을 포함하는 구조체
 */
typedef struct
{
    int x;
    int y;
    wchar_t *name;
    wchar_t **list;
    int length;
    int current_index;
    short outline_tbcolor;
    short name_tbcolor;
    short element_tbcolor;
    short selected_tbcolor;
    short non_selected_tbcolor;
} MenuData;
/**
 * @brief 프롬프트 렌더링 데이터들을 포함하는 구조체
 */
typedef struct
{
    int x;
    int y;
    wchar_t *message;
    int rlen;
    short outline_tbcolor;
    short message_tbcolor;
    short text_tbcolor;
} PromptData;
namespace InterfaceUtils {
    static BufferingManager & bm = BufferingManager::instance();
    void draw_menu(MenuData *data);
    int run_menu(MenuData *, bool);
    void draw_prompt(PromptData *data);
    wchar_t *run_prompt(PromptData *);
}

<Game.cpp>
MainMenu Game::run_main_menu() {
    MenuData menu;
    menu.name = L "      메뉴";
    wchar_t **list = (wchar_t **) malloc(sizeof(wchar_t *) *6);
    list[0] = L "게임 시작 (숫자)";
    list[1] = L "게임 시작 (코룡)";
    list[2] = L "게임 시작 (레나)";
    list[3] = L "      불러오기";
    list[4] = L "랭킹 (리플레이)";
    list[5] = L "      나가기";
    menu.list = list;
    menu.length =6;
    menu.current_index =0;
    menu.element_tbcolor = TO_TBCOLOR(BLUE, RED);
    menu.name_tbcolor = TO_TBCOLOR(WHITE, BLACK);
    menu.outline_tbcolor = TO_TBCOLOR(GRAY, BLACK);
    menu.selected_tbcolor = TO_TBCOLOR(LIGHT_JADE, LIGHT_PURPLE);
    menu.non_selected_tbcolor = TO_TBCOLOR(WHITE, GRAY);
    menu.x =30;
    menu.y =10;
    int index = InterfaceUtils::run_menu(&menu, true);
    free(list);
    return (MainMenu) index;
}

```

```

<beeputils.h>
#define PLAY_BEEP(o, v, t) Beep((int)((o) * pow(1.06f, v)), t)
#define PLAY_VICTORY_FANFARE() W
RA(OCTAVE_5, 200); W
RA(OCTAVE_5, 100); W
RA(OCTAVE_5, 300); W
SI(OCTAVE_5, 200); W
RA(OCTAVE_5, 200); W
SI(OCTAVE_5, 200); W
DO(OCTAVE_6, 200); W
DO(OCTAVE_6, 100); W
DO(OCTAVE_6, 700);

#define PLAY_EXIT_SOUND() W
FA_S(OCTAVE_6, 20); W
FA_S(OCTAVE_6, 20);

#define PLAY_FAIL_SOUND() W
FA_S(OCTAVE_6, 20); W

#define PLAY_SELECTION_SOUND1() W
RA(OCTAVE_4, 20);

```

그 외에도 텍스트 기반 UI 컴포넌트를 재사용하기 interfaceutils을, 비프음을 게임에서 재생하기 위해 beeputils을 만들어 사용하여 게임의 완성도와 코드 품질을 올릴 수 있었다.

```

<ConsoleUtils.cpp>
/**
 * @brief wait 중 키 핸들링을 같이 수행한다.
 * @description wait를 UNIT_WAIT로 쪼개어 수행하고 도중 키가 눌렸는지 검사하여 handler에 전달한다.
 *          handler(int c)의 반환값이 1 이상일 때 wait를 중지하고 handler의 반환 값을 반환한다.
 *          키 핸들링이 일어나지 않는 경우에는 항상 -1을 반환한다. handler가 NULL인 경우 항상 즉시 입력을 반환한
 *          다.
 * @param ms 대기할 시간 (밀리초)
 * @param handler 텍스트의 배경 색깔
 * @param data 자유형 데이터
 */
int ConsoleUtils::wait_with_handler(unsigned long ms, int (*handler)(int, void *), void *data)
{
    struct timespec begin, end;
    SystemUtils::clock_gettime(CLOCK_MONOTONIC, &begin);
    while (true)
    {
        SystemUtils::wait(UNIT_WAIT);
        if (kbhit())
        {
            int c = _getwch();
            c = (c == 0xE0 || c == 0) ? _getwch() : c;
            if (handler == NULL)
            {
                return c;
            }
            else
            {
                int result = handler(c, data);
                if (result > -1)
                {
                    return result;
                }
            }
        }
        SystemUtils::clock_gettime(CLOCK_MONOTONIC, &end);
        if (MILLISECOND_DIFF(begin, end) >= ms)
        {
            return -1;
        }
    }
}

```

```

<Game.cpp>
int Game::handle_game_key(int c, void *param) {
    void **list = (void **) param;
    Game *game = (Game *) list[0];
    GameData *data = (GameData *) list[1];
    switch (c) {
        case UP_KEY:
            if (data -> puzzle.move(UP)) {
                memcpy(data->save.current_map, data -> puzzle.map, sizeof(int) * DIM * DIM);
                data->save.move_log[data -> save.move ++] = L 'U';
                PLAY_SELECTION_SOUND1();
            } else {
                PLAY_FAIL_SOUND();
            }
            break;
        ...
        case ESCAPE_KEY:
            data->save.move_log[data -> save.move] = L 'W0';
            PLAY_EXIT_SOUND();
            return HANDLE_SAVE_AND_EXIT;
    }
    game->draw_puzzle(data);
    game->bm.draw();
    if (data -> puzzle.is_done()) {
        return HANDLE_GAME_DONE;
    }
    return HANDLE_NULL;
}

```

```

<Game.cpp>
void *param[] = {this, data};
struct timespec begin, end;
SystemUtils::clock_gettime(CLOCK_MONOTONIC, &begin);
while (true) {
    int handle_code = ConsoleUtils::wait_with_handler(10, handle_game_key, param);
    SystemUtils::clock_gettime(CLOCK_MONOTONIC, &end);
    double diff = MILLISECOND_DIFF(begin, end) / 1000.0;
    if (handle_code == HANDLE_NULL) {
        bm.xywprintf(40, 27, L "Time: %.1fs", diff + data -> save.time);
        bm.xywprintf(40, 28, L "Move: %d", data -> save.move);
        bm.draw();
    }
}

```

또한 인게임에서 단일 스레드에서 실시간 시간 출력과 키 입력을 동시에 수행하기 위해서 wait_with_handler와 같은 함수를 설계하여 사용하였다.

```

RankingData ranking_data;
FILE *fp = fopen("ranking.bin", "rb");
if (fp !=NULL) {
    fread(&ranking_data, sizeof(RankingData), 1, fp);
    fclose(fp);
} else {
    ranking_data.last_index =0;
}
if (ranking_data.last_index <10) {
    ranking_data.list[ranking_data.last_index++] = data ->save;
} else {
    if (ranking_data.list[ranking_data.last_index -1].time > data ->save.time) {
        ranking_data.list[ranking_data.last_index -1] = data ->save;
    }
}
GameSave temp;
for (int i =0; i < ranking_data.last_index; i++) {
    for (int j =0; j < ranking_data.last_index -1; j++) {
        if (ranking_data.list[j].time >
            ranking_data.list[j +1].time) {
            temp = ranking_data.list[j];
            ranking_data.list[j] = ranking_data.list[j +1];
            ranking_data.list[j +1] = temp;
        }
    }
}
fp = fopen("ranking.bin", "wb");
fwrite(&ranking_data, sizeof(RankingData), 1, fp);
fclose(fp);

```

랭킹 처리 부분에서는 파일 입출력을 이용하여 랭킹 구조체를 저장하고, 상위 10등의 점수 기록만 기록되도록 거품 정렬을 통해 구현하였다.

```

<Puzzle.cpp>
#define DIM 4
#define END_PUZZLE_INDEX (DIM *DIM -1)
#define TO_PUZZLE_DIRECTION(c) W
( (PuzzleDirection) W
( W
c == 'U' ? (UP) : W
c == 'D' ? (DOWN) : W
c == 'L' ? (LEFT) : W
c == 'R' ? (RIGHT) : W
NULL W
) W
)
#define TO_REVERSE_PUZZLE_DIRECTION(c) W
( (PuzzleDirection) W
( W
c == 'D' ? (UP) : W
c == 'U' ? (DOWN) : W
c == 'R' ? (LEFT) : W
c == 'L' ? (RIGHT) : W
NULL W
) W
)
typedef enum
{
    LEFT, RIGHT, UP, DOWN
} PuzzleDirection;
class Puzzle {
public:
    int map[DIM][DIM];
    void init_puzzle();
    bool move(PuzzleDirection dir);
    void shuffle_once();
    bool is_done();
};

```

그리고 기존 퍼즐 코드를 확장하여, 새로 만든 렌더링 코드와 결합되어 작동할 수 있도록, 퍼즐 클래스를 설계하여 퍼즐 데이터를 내포하고, 퍼즐에 관련한 작업들을 수행할 수 있도록 하였다. 또한 리플레이 처리간 필요한 U(Up), D(Down), R(Right), L(Left)등의 문자(char)로 저장된 플레이 기록을 프로그램에서의 처리에 사용되는 열거형으로 반환하는 매크로 함수(TO_PUZZLE_DIRECTION: 정방향, TO_REVERSE_PUZZLE_DIRECTION: 역방향)를 정의하여 사용하였다.

(5) 이번 과제에 대한 느낀점

문제를 해결하면서 컬러 아스키 아트를 사용하기 위한 방법과 콘솔에서의 더블 버퍼링 기법에 대해서 공부할 수 있어서 좋았다.