뮤직 톡톡 구현

18102175 이배원

대구가톨릭대학교 컴퓨터소프트웨어학부 컴퓨터공학과

1. 서 론

이번 학년에 전과를 하여 부득이 하게 자바수업을 먼저 듣게 되어 자바수업에서 텀 프로젝트를 진행한 바 있다. 그 때의 프로젝트는 총알 피하기 게임으로 하였고, 그 때 다음 C언어 프로젝트는 리듬게임을 만들고자 생각 하게 되었고 게임 메이플스토리 게임의 구애의 춤을 만들려고 하다가 콘솔 특성상 밋밋해 보일 것 같아 문득 생각이 난 플래시 게임 뮤직톡톡을 만들게 되었다.
자바 프로젝트에서는 오디오 에셋을 가져오고, 맵과 모이미지를 직접 그려서 제작을 하였지만 C언어에서는 이미지를 가져오지 않고 일반적으로 사람들이 콘솔게임 하면 떠오르는 콘솔게임 다운 게임을 만들기 위해 기본 특수문자와 콘솔에서 제공하는 컬러와 비프음을 가지고 게임을 만들어 보았다.

우선 관련 연구에서 C언어에 대한 콘솔에서 제공되는 컬러와 비프음 등에 대해 알아본다. 보통 프로그래밍 언어를 배울 때 콘솔에서 입력 받고 출력을 하고 하지만 보통 콘솔에 대해서는 별로 배우지 않지만 콘솔에는 꽤 많은 명령어 들이 존재 한다.

다음으로 프로그램을 구성하기위한 계획과 기획, 구성을 알아보고 이 프로그램에서 이동하는 플레이어와 그 플레이어를 따라다니는 체력들을 구현한 함수들을 알아보고 프로그램의 동작 결과를 사진으로 통해 알아보고 프로그램이 어떤 기능들이 있는지 그리고 그 기능들이 잘 구현이 되었는지 알아보겠다.

2. 관련 연구

우선 이 프로그램에서 콘솔에 관해 사용하는 것은 4가지로 나뉘는데 콘솔의 사이즈, 타이틀 등 기본적인 것들을 설정하는 init과 특정 좌표로 이동 시키는 goto, 색상을 바꿔주는 setColor, 해당 주파수의 값을 소리로 나타내는 beep에 대해 알아보겠다.
이 네 개의 함수를 사용하기 위해서는 window.h 헤더파일을 추가하여 콘솔에 대한 설정을 할 수 있다.

init에서 system함수를 통해 콘솔의 명령어를 입력해줄 수 있다. 이를 이용해서 mode con col로 행을, lines를 이용하여 열의 크기를 정해줄 수 있고, title 명령어를 이용하여 해당 콘솔의 타이틀을 지정할 수 있다.

CONSOLE_CURSOR_INFO라는 커서에 대한 구조체가 있다. 이를 통해 커서의 크기(dwSize), 커서를 안 보이게(bVisible) 지정해줄 수 있다. 커서에 대한 정보를 담았으면 SetConsoleCursorInfo함수를 이용해서 그 정보를 해당 콘솔에 적용할 수 있다.

gotoxt함수에서 COORD 라는 x, y 좌표 위치를 담을 수 있는 구조체를 가져와서 매개변수로 이동시키고 싶은 x, y 좌표를 담고 SetConsoleCursorPosition함수를 통해 콘솔에 적용하면 해당 좌표로 커서가 이동하게 된다.

콘솔에서 글자색과 배경색을 지정해줄 수 있다. SetConsoleTextAttribute함수를 이용하여 색깔을 바꿀 수 있는데 배경색은 글자색과 다르게 16배로 받기 때문에 배경색깔을 background << 4(2^4 = 16)을 해주어 같은 색상으로 만들어준 후 지정해줄 수 있다.

beep는 이 프로그램에서 중요한 함수이다. 아주 간단하게 매개변수로 주파수, 실행되는 시간을 지정해주면 되는 간단한 함수이다. beep를 함수안에 구현하여 나타내고자 하는 음을 switch문으로 받아서 실행하면 된다.

3. 프로그램 설계

이 프로그램은 4개의 헤더파일과 4개의 소스파일로 이루어진다.
콘솔에 관한 각종 함수들을 모아놓은 util, 실질적인 게임관리를 담당하는 game, 게임에서 필요한 각 스테이지별로 모아놓은 map, 프로그램을 돌리기위한 main이 존재한다.

기본적인 설계는 다음과 같다.
화면에 플레이어(◎)와 방향키(→, ←, ↑, ↓, ♬)가 그려져 있고 위쪽에는 현재스테이지, 타이머, 스코어가 표시가 된다. 한 스테이지를 클리어 시 입력해야 되는 방향키의 개수는 점점 늘어나고 1부터 10개의 스테이지가 있다. 스테이지별로 해당 방향키에 맞는 키를 입력 시 해당 스테이지의 맞는 노래가 나온다.
스테이지 1부터 10까지 스테이지가 올라갈수록 알맞게 입력해야 되는 평균 시간이 점차 줄어들게 되고(0.025초씩--) 더 빠르게 눌러야 한다. 이에 따른 타이머가 필요하고 타이머가 0가 되거나 체력이 있어 알맞지 않은 방향키를 입력 시 줄어들고 추가체력이 0일 때 한 번 더 틀리면 실패가 되는 루트로 구성이 된다.
만약 스테이지 10까지 클리어 하면 엔딩 크레딧이 나오고 이때까지의 노래의 제목을 띄어준다.

우선 이 프로그램에서 가장 중요한 것은 스테이지와 노래이다. 이 두 개를 map안에서 구현을 하였고 총 10개의 스테이지와 같이 사용될 10개의 노래로 구성이 되어있다. 노래는 동요, 만화, 드라마ost, 영화ost 순으로 만들었다.
게임 안에 이스터에그를 넣고 싶어서 게임메뉴 창에 이스터에그로 오징어 게임ost를 하나 더 추가해 보았다.

만약 기존의 점수보다 높게 게임을 클리어 시 bestScore가 변경이 되며 게임정보 창에서 이를 확인할 수 있다.
왼손잡이를 위하여 방향키대신 사용할 수 있는 w, a, s, d키를 추가하였으며 게임도중과 엔딩 크레딧이 나오고 있을 때에도 esc키를 눌러 언제든지 메뉴 창으로 돌아올 수 있다.

4. 프로그램 구현

플레이어 움직임에 관련되는 것만 본다면 함수는 다음과 같다.

removePlayer(), move(), removeHealth(), drawHealth(), setLeftHealth(), drawHealth() 6개의 함수로 이루어지며, 전반적으로 플레이어가 움직이면 플레이어를 지우고 다음 위치에 플레이어를 위치시키고, 체력도 마찬가지로 체력들을 지우고 체력이 있어야 할 위치에 이동시키며 함수를 보면서 해당 기술들을 설명을 하겠다.

```
void removePlayer() {
    //for first move
    if (moveIndex == 0) {
        gotoxy(xpos, ypos);
        printf("  ");
    }
    //after that move
    else {
        gotoxy(stage[moveIndex - 1][1], stage[moveIndex - 1][0]);
        printf("  ");
    }
}
void move() {
    if (health == 0)
        removePlayer();  //움직이기전플레이어의위치에체력을그려야하기때문에플레이어를안지워도무방
    else
        removeHealth();

    setColor(Light_Yellow, Black);
    gotoxy(stage[moveIndex][1], stage[moveIndex][0]);
    printf("◎");
    setLeftHealth();
    drawHealth();
    moveIndex++;
}
void removeHealth() {
    if (getHealth) {    //체력을먹었다면플레이어가이동하면서자리가비기때문에체력을지울필요없음
        getHealth = false
        return
    }
    gotoxy(healthPos[health - 1][0], healthPos[health - 1][1]);
    printf("  ");
}
void drawHealth() {
    for (int i = 0; i < health; i++) {
        setColor(Light_Red, Black);
        gotoxy(healthPos[i][0], healthPos[i][1]);
        printf("♥");
    }
}
```

우선 방향키로 움직이면 move()가 실행된다. 플레이어에는 뒤에 체력이 따라 붙어 오는데 어차피 체력을 그려야 하기 때문에 플레이어를 지우는 것은 체력이 0일 때만 지우면 되고 체력이 있으면 플레이어를 안 지워도 체력을 그려야 하기 때문에 체력이 0이 아니면 플레이어를 다음 위치에 그리고 제일 뒤쪽에 있는 마지막 체력만 지우고 체력들을 다시 그려주면 된다.
콘솔에는 레이어 개념이 없기 때문에 해당 위치에 덮어씌울 것이 있다면 일일이 지워주지 않아도 무방하다.

```
void setLeftHealth() {
    for (int i = health - 1; i > 0; i--) {
        healthPos[i][0] = healthPos[i - 1][0];
        healthPos[i][1] = healthPos[i - 1][1];
    }

    if (moveIndex == 0) {
        healthPos[0][0] = xpos;
        healthPos[0][1] = ypos;
    }
    else {
        healthPos[0][0] = stage[moveIndex - 1][1];
        healthPos[0][1] = stage[moveIndex - 1][0];
    }
}
void drawHealth() {
    for (int i = 0; i < health; i++) {
        setColor(Light_Red, Black);
        gotoxy(healthPos[i][0], healthPos[i][1]);
        printf("♥");
    }
}
```

이 함수는 현재 가지고 있는 체력들의 위치를 설정하는 함수이다. 우선 health 는 0이 여도 끝난 것이 아니란 것을 먼저 알려두려고 한다. health가 0이란 것은 추가 체력이 없고 플레이어 자신만 남아 있다는 것이다. 따라서 health는 추가체력의 개수라고도 말할 수 있다.
입력을 잘못 해서 틀리면 체력이 플레이어와 먼 곳부터 제거가 되어야 하기 때문에 현재 체력에서 제일 마지막부터 체력의 위치를 설정한다. 그리고 스테이지가 시작되고 처음 움직일 때는 플레이어의 위치를 가리키게 한다. 그 뒤에는 스테이지에 설정된 길로 따라 갈 수 있게 stage배열에서 moveIndex로 설정하여 맨 앞 체력 위치를 설정 할 수 있게 만들었다. 그 뒤로는 for문으로 앞의 위치 -> 앞의 위치순으로 따라오게 만들었고, 실질적으로 drawHealth() 함수로 해당 추가체력 개수만큼 출력을 이루어지게 만들었다.

다음으로 게임을 마치고 나오는 엔딩 크레딧에서 사용한 기술이다.
우선 함수의 해당 코드는 다음과 같다.(소스코드 중략)

```
void ending() {
    system("cls");
  .
(생략)
  .

    while (1) {
      .
    (생략)
      .

        Sleep(150);
        sleepCount++;

        _beginthreadex(NULL, 0, (_beginthreadex_proc_type)noteSound, (int*)endingSound[soundIndex], 0,
            NULL);
        soundIndex++;

        if (soundIndex == sizeof(endingSound) / sizeof(int))
        soundIndex = 0;

        //Show Score
        if (totalScore >= bestScore) {
        gotoxy(41, 18);
        newBestScore(totalScore); //Twinkle Score
        }

        if (sleepCount % 8 == 0 && line < 40) {
            system("cls");

            if (totalScore >= bestScore) {
                gotoxy(41, 18);
            newBestScore(totalScore); //Twinkle Score
            }
            else {
            gotoxy(44, 18);
            setColor(Light_Yellow, Black); printf("Score"); setColor(White, Black);      printf(" : %d",
             totalScore); //normal Score
            }

            //Show Credit
            if (creditsYpos > 4) {
                gotoxy(creditsXpos, creditsYpos);
                setColor(Bright_White, Black);printf("[ "); setColor(Light_Yellow, Black); printf("Clear");
                  setColor(Bright_White, Black); printf(" ]");
            }
            setColor(Bright_White, Black);
            if (line >= 2 && creditsYpos > -4) {
                gotoxy(creditsXpos - 3, creditsYpos + 8);
                setColor(Light_Yellow, Black);  printf("제작자"); setColor(Bright_White, Black); printf(" : 이배
                  원");
```

```
            }
              .
            (생략)
              .

            if (line >= 34 && line < 39) {
                gotoxy(creditsXpos - 5, creditsYpos + 40);
                printf("Thank you for play");
            }
            if (line == 39) {
                gotoxy(creditsXpos - 5, creditsYpos + 41);
                printf("Thank you for play"); setColor(Light_Red, Black); printf("!");
                gotoxy(23, 14);
                setColor(Gray, Black);
                printf("Space to Menu");
                endingcreditplaying = false //Finish Endingcredit from this line //키입력가능
            }
            creditsYpos--;
            line++;
            }
        }
}

void newBestScore(int totalScore) {
    static totalscoreCount = 0;
    if (totalscoreCount == 8)
        totalscoreCount = 0;
    switch (totalscoreCount) {
        gotoxy(40, 18);
    case 0:
        setColor(Red, Black);               printf("N");
        setColor(Light_Red, Black);         printf("e");
        setColor(Light_Yellow, Black);    printf("w ");
        setColor(Light_Green, Black);    printf("S");
        setColor(Light_Cyan, Black);      printf("c");
        setColor(Light_Blue, Black);     printf("o");
        setColor(Purple, Black);          printf("r");
        setColor(Light_Purple, Black);    printf("e");

        setColor(Gray, Black);              printf(" : %d", totalScore);
        break;
          .
        (생략)
          .
    case 7:
        setColor(Light_Red, Black);             printf("N");
        setColor(Light_Yellow, Black);        printf("e");
        setColor(Light_Green, Black);        printf("w ");
        setColor(Light_Cyan, Black);          printf("S");
        setColor(Light_Blue, Black);         printf("c");
        setColor(Purple, Black);               printf("o");
```

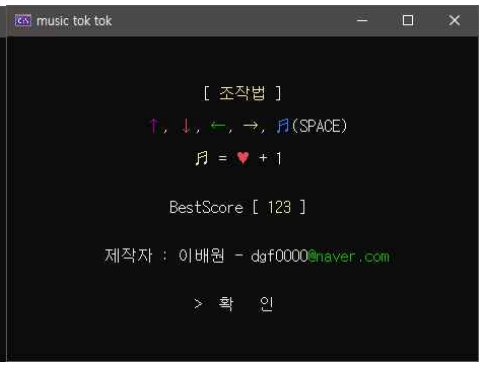
```
            setColor(Light_Purple, Black);     printf("r");
            setColor(Red, Black);              printf("e");

            setColor(Gray, Black);             printf(" : %d", totalScore);
            break;
        }
        totalscoreCount++;
    }
```

엔딩 크레딧은 보통 중간에서 글씨가 나오고 사라기지 때문에 생각이 든 것이 글씨가 나오는 위치와 사라지는 위치를 스레드 sleep함수를 이용하여 sleep이 될 때마다 sleepCount변수를 통하여 1씩 증가시켜 원하는 시간대에 출력, 사라지게 하고 엔딩 음악과 맞추어 한 칸씩 올라올 수 있도록 만들어 보았다. sleepCount가 1씩 증가 되면서 8번 증가되면 한 칸씩 올라오고 line이라는 변수를 두어 1씩 증가 하면서 글씨들이 순서대로 출력 되게 하면서 creditsXpos, creditsYpos라는 변수를 통해 모든 글씨들의 위치를 조정 할 수 있게 만들었다. endingcreditplaying(bool)값을 두어 크레딧이 모두 올라왔을 때 키 입력을 받을 수 있도록 설정 하여 크레딧이 끝나면 스페이스바를 누르면 메인으로 돌아올 수 있도록 설계하였다. 물론 편의를 위해 ESC키는 크레딧 도중에도 ESC키를 눌러 메인으로 돌아 올 수 있다.

엔딩이 나올 때 스코어를 함께 표시 하도록 하였는데 신기록일 때와 신기록이 아닌 점수로 나누었는데 기본점수 표시는 크레딧과 같이 표시 할 수 있지만 신기록은 글씨가 무지개 색으로 움직이면서 표시를 할 수 있도록 만들었다. 하지만 글씨들은 sleepCount가 8번에 한번만 움직이기 여서 같이 넣으면 끊기듯 나오기 때문에 따로 구현을 해보았다. newBestScore함수로 따로 만들었고 엔딩 크레딧에서 다른 함수를 부르면서 newscore라는 글씨가 색깔이 바뀌면서 출력이 되어야 되기 때문에 newBestScore함수를 부를 때 함수 안에 변수를 static으로 선언하여 함수 안에서 계속 색깔이 바뀌도록 만들었다.

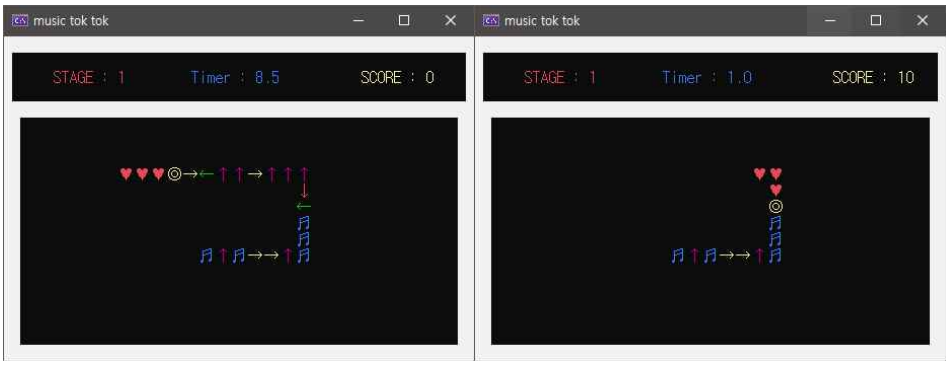
5. 프로그램의 동작과 결과



6. 결과 분석 및 토의

-메인에서 방향키를 입력 받을 시 >가 움직이면서 다른 글씨들은 어둡게 처리(음영처리)되며 화살표(>)가 있는 곳에서 스페이스 바를 누를 시 해당 루트가 실행 됨.

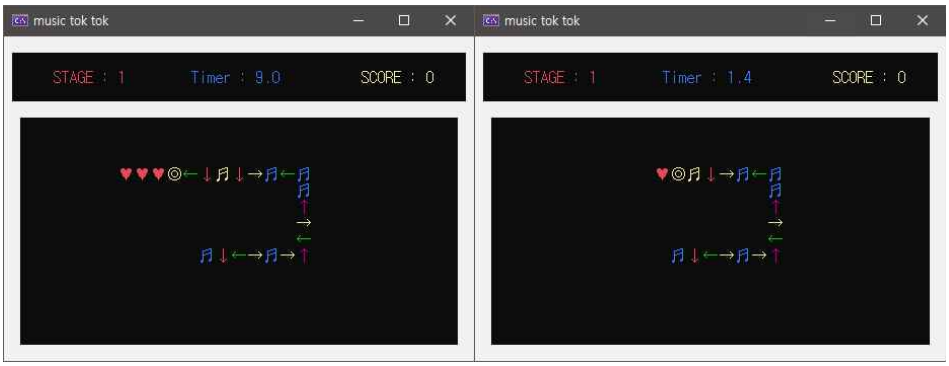
- 방향키를 계속 입력 시 오징어게임 ost가 무한루프로 돌게 된다.(이스터에그)



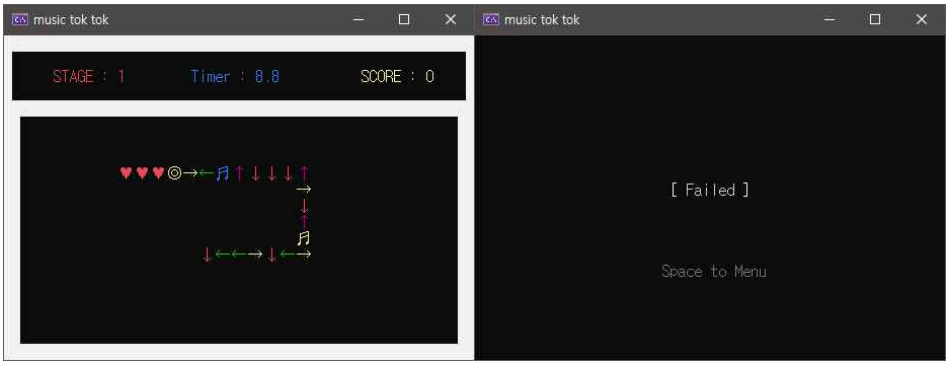
-체력(♥)이 플레이어(◎)의 뒤를 따라오며 플레이어가 움직이는 대로 따라 움직이며 플레이어 또한 스테이지를 따라 자연스럽게 움직임

- 키를 바르게 입력 시 스코어가 올라감, 해당 스테이지의 비프음이 출력됨

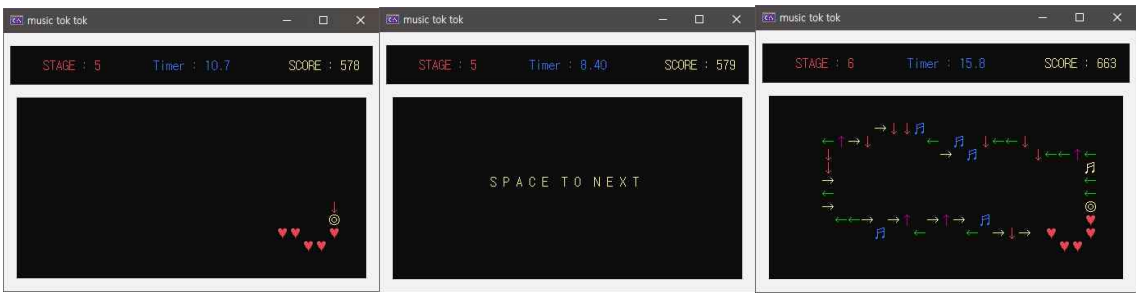
- 타이머가 작동 됨



- 키를 잘못 입력 시 체력이 1개씩 없어지며 스코어는 그대로, error 비프음이 출력

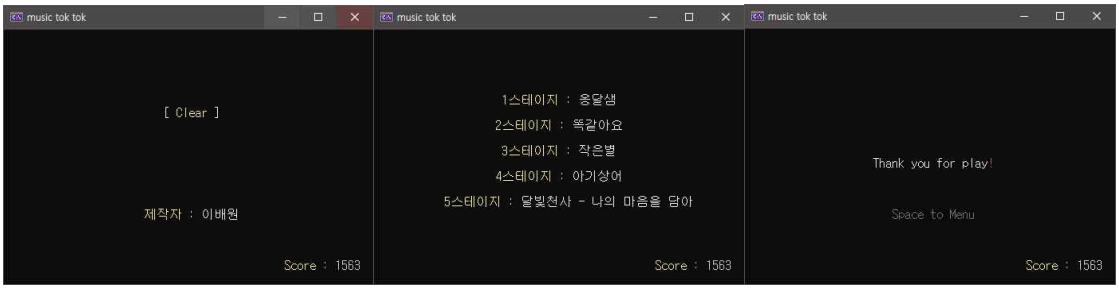


- 추가체력 없고 잘못 입력 시 혹은 타이머가 0초가 되면 failed엔딩으로 넘어감
- failed엔딩 노래가 나오고 노래가 끝나면 처음부터 다시 재생(노래 반복)



- 스테이지 클리어 시 다음 스테이지별로 설정한 타이머로 설정이 됨.
- 스테이지 클리어 시 다음 스테이지로 넘어 갈 때 플레이어(◎) 위치와 체력(♥)이 다음 스테이지로 넘어 갔을 때 그대로 잘 유지 됨.
- 스테이지 클리어 시 다음 스테이지로 넘어 갈 때 남은 시간을 스코어로 변환 되어서 넘어감(스코어 += 남은시간 * 10)

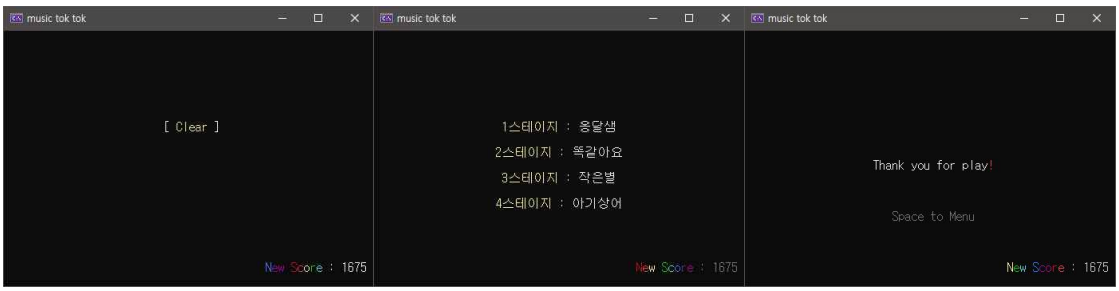
- ♬(노란색)을 space로 바르게 입력 시 체력이 추가됨(최대 추가체력 5개까지)



-모든 스테이지 클리어시 엔딩크레딧이 올라오며 끝이 날 때 까지 esc를 제외한 키 입력이 제한되며 끝이 날 시 스페이스바로 메뉴로 이동 가능

- 엔딩 크레딧의 글씨가 해당 위치에서 출력 되고 해당 위치까지 가면 사라짐

- clear엔딩 노래가 나오고 노래가 끝나면 처음부터 다시 재생



- 신기록을 세우며 클리어 시 score위치에 new Score로 바뀌며 글씨는 그대로 음악과 동시에 끊기며 올라가지만 new Score는 자연스럽게 색깔이 바뀌며 표시됨.

- 신기록을 세우고 게임정보 확인 시 신기록이 변경되어 있음

7. 결론

이 프로젝트를 통해 추억의 플래시 게임 뮤직톡톡을 구현해 보았다.
예전에 게임을 했었을 때에는 단지 재미있다 정도였는데 막상 구현을 해보고 나니 이런 식으로 만들어져있구나 라는 걸 느꼈다. 우리가 이용하는 것은 쉽지만 만드는 것은 꽤나 오랜 시간이 걸린다. 10스테이지 까지 많아 걸려야 3분밖에 소요되지 않는다. 하지만 만드는 것은 그보다 훨씬 많은 시간이 소요 되며 난이도 조절, 곡 선택 등 다양한 선택사항이 존재한다. 이렇게 힘들게 만들고 아무도 플레이를 해주지 않으면 얼마나 안타까운 일인지 생각하게 되었다. 앞으로의 게임 시장에서 이런 인디게임 또한 발전 되었으면 한다.

이 프로젝트를 하면서 구현하고자 하는 것들을 모두 구현을 하였지만 좀 더 짧게 구현이 가능 할 것 같고 포인터를 사용한다면 좀 더 최적화를 시킬 수 있을 것 같지만 포인터를 사용하지 않아도 충분하게 구현이 가능하기 때문에 코드를 이해하기 쉽게 하기 위해 포인트를 사용하지 않았다.

유니티에 사용되는 c#이나 java는 꽤나 많은 편리한 기능들이 있었는데 C에서는 제공하지 않는 것들이 꽤 있어서 편리 했었는데 이런 것들이 제공되지 않는 C에서 일일이 하나하나 코드를 작성하니 꽤나 코드가 길어지게 됨을 느꼈다.

특히 콘솔에서 영어는 1칸을 차지하고 한글이나 특수문자는 2칸을 차지하기 때문에 일일이 배열에 좌표를 찍어서 스테이지를 만들었는데 스테이지가 올라갈수록 배열의 길이가 길어져서 관리하기가 힘들어졌다. 이를 처음부터 관리할 수 있는 함수를 만들어서 쉽게 작성 할 수 있도록 만들었어야 했다는 걸 후반에 와서야 깨닫게 되었다. 코드를 작성하기 전에 계획을 충분히 하고 만들어야 되겠다는 생각이 들었다.

8. 참고문헌

커서 가리기 - https://coding-factory.tistory.com/691

좌표이동 - https://coding-factory.tistory.com/690?category=767224

컬러 - https://chanhhh.tistory.com/8

키 입력 - https://m.blog.naver.com/power2845/50143021565

구애의 춤

https://www.google.co.kr/search?q=%EB%A9%94%EC%9D%B4%ED%94%8C+%EA%B5%AC%EC%95%A0%EC%9D%98+%EC%B6%A4&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiJmaWy45z0AhVYx2EKHRA8BKsQ_AUoAXoECAEQAw&biw=1596&bih=1122&dpr=1.1

뮤직톡톡

https://www.google.co.kr/search?q=%EB%AE%A4%EC%A7%81%ED%86%A1%ED%86%A1&tbm=isch&ved=2ahUKEwifhO-y45z0AhVG6pQKHSlPCe0Q2-cCegQIABAA&oq=%EB%AE%A4%EC%A7%81%ED%86%A1%ED%86%A1&gs_lcp=CgNpbWcQAzIFCAAQgAQyBQgAEIAEMgQIABAYMgQIABAYOggIABCABBCxAzoICAAQsQMQgwFQ8QNY1RFgvBJoAXAAeAGAAa0BiAHvCpIBBDAuMTGYAQCgAQGqAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=qJWTYZ-QJMbU0wSpnqXoDg&bih=1122&biw=1596

자바 프로젝트 - https://github.com/dgf0000/Java_Dodge_Game

9. 별첨 (소스코드와 주석)

1) main.h

```
//main
#ifndef __STDIO_H__
#define __STDIO_H__
#include <stdio.h>              // 표준 입출력 함수 사용을 위한 헤더파일 추가
#endif

//util
#ifndef __WINDOWS_H__
#define __WINDOWS_H__
#include <windows.h>            //cmd 명령어를 입력하기 위한 헤더파일 추가
#endif

//game
#ifndef __STRING_H__
#define __STRING_H__
#include <string.h>             //memcpy함수를 이용 위한 헤더파일 추가
#endif

#ifndef __STDLIB_H__
#define __STDLIB_H__
#include <stdlib.h>             //random함수를 통해 난수를 받기 위한 헤더파일 추가
#endif

#ifndef __STDBOOL_H__
#define __STDBOOL_H__
#include <stdbool.h>            //bool변수를 사용하기 위한 헤더파일 추가
#endif

#ifndef __CONIO_H__
#define __CONIO_H__
#include <conio.h>              //키 입력을 위한 헤더파일 추가
#endif

#ifndef __PROCESS_H__
#define __PROCESS_H__
#include <process.h>            //비프음 쓰레드를 위한 헤더파일 추가
#endif

#ifndef __TIME_H__
#define __TIME_H__
#include <time.h>               //타이머 설정을 위한 헤더파일 추가
#endif
```

2) util.h

```
#include "main.h"

enum COLOR_LIST {          //컬러 리스트
        Black,
        Blue,
        Green,
        Cyan,
        Red,
        Purple,
        Yellow,
        White,
        Gray,
        Light_Blue,
        Light_Green,
        Light_Cyan,
        Light_Red,
        Light_Purple,
        Light_Yellow,
        Bright_White
};

void init();                                    //콘솔창 제어
void gotoxy(int, int);                  //x, y 좌표로 이동
void setColor(int, int);        //글자, 배경 색으로 설정
void noteSound(int);            //해당 비프음 출력
```

3) map.h

```
#include "main.h"

char ui[4][58];                  //UI창 크기          - 글자를 출력해야 되기대문에 1칸씩
char field[16][29];   //플레이창 크기      - 특수문자를 출력해야 되기대문에 2칸씩

int maxLevel;                    //스테이지 크기

//스테이지 배열들
int stage_1[19][3];
int stage_2[22][3];
int stage_3[29][3];
int stage_4[33][3];
int stage_5[37][3];
int stage_6[44][3];
int stage_7[56][3];
int stage_8[70][3];
int stage_9[80][3];
int stage_10[99][3];

//메뉴 이스터에그 사운드
int menuSound_1[32];

//실패엔딩 사운드
int failedSound_0[16];
int failedSound_1[16];
int failedSound_2[16];
int failedSound_3[16];
int failedSound_4[16];

//엔딩 사운드
int endingSound[288];

//사운들 배열들
int sound_1[19];
int sound_2[22];
int sound_3[29];
int sound_4[33];
int sound_5[37];
int sound_6[44];
int sound_7[56];
int sound_8[70];
int sound_9[80];
int sound_10[99];
```

4) game.h

```
#define KEY_CODE

#define UP 0
#define DOWN 1
#define LEFT 2
#define RIGHT 3
#define ENTER 4
#define ENTER_GOLD 5
#define NOTHING 6
#define ESCAPE 10
#endif

void keyControl();
void drawTitle();
int drawMenu();
void stageControl();
void drawMap();
void drawStage();
void drawInfo();
void update();
void removePlayer();
void removeHealth();
void removeAllHealth();
void setLeftHealth();
void nextStage();
void move();
void timer();
void drawUi();
void drawHealth();
void ending();
void reset();
void againTimerSet();
void failed();
void newBestScore(int);
void saveScore(const int);
void getScore();
```

5) main.c

```
#include "main.h"
#include "game.h"
#include "util.h"

int main() {
	init();	//프로그램 시작 시 콘솔창 설정
	while (1) {
		getScore();			//score파일에서 bestScore를 가져옴
		drawTitle();		//타이틀 그림 출력
		int select = drawMenu();
		switch (select) {
		case 0:				//게임 시작
			update();
			break;
		case 1:				//게임 정보
			drawInfo();
			break;
		case 2:				//게임 종료
			return 0;
		}
		system("cls");
	}
}
```

6) util.c

```
#include "util.h"

void init() {           //콘솔창 제어
        system("mode con cols=58 lines=20 | title music tok tok");
        CONSOLE_CURSOR_INFO cursorInfo = { 0, };
        cursorInfo.dwSize = 1;
        cursorInfo.bVisible = false;
        SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursorInfo);
}

void gotoxy(int x, int y) {     //x, y 좌표로 이동
        COORD pos = { x, y };
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), pos);
}

void setColor(int foreground, int background) {   //글자, 배경 색으로 설정
        foreground &= 0xf;
        background &= 0xf;
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), (background << 4) | foreground);
}

void noteSound(int note) {    //해당 비프음 출력
        switch (note) {
        case 0:             //mute
                break;
        case 1:             //Do
                Beep(262, 100);
                break;
        case 2:             //Do#
                Beep(277, 100);
                break;
        case 3:             //Re
                Beep(293, 100);
                break;
        case 4:             //Re#
                Beep(311, 100);
                break;
        case 5:             //Mi
                Beep(330, 100);
                break;
```

```
case 6:             //Fa
        Beep(349, 100);
        break;
case 7:             //Fa#
        Beep(370, 100);
        break;
case 8:             //Sol
        Beep(392, 100);
        break;
case 9:             //Sol#
        Beep(415, 100);
        break;
case 10: //Ra
        Beep(440, 100);
        break;
case 11: //Ra#
        Beep(466, 100);
        break;
case 12: //Si
        Beep(494, 100);
        break;
case 13: //Do
        Beep(523, 100);
        break;
case 14: //Do#
        Beep(554, 100);
        break;
case 15: //Re
        Beep(587, 100);
        break;
case 16: //Re#
        Beep(622, 100);
        break;
case 17: //Mi
        Beep(659, 100);
        break;
case 18: //Fa
        Beep(698, 100);
        break;
case 19: //Fa#
        Beep(740, 100);
```

```
                break;
        case 20:  //Sol
                Beep(784, 100);
                break;
        case 21:  //Sol#
                Beep(831, 100);
                break;
        case 22:  //Ra
                Beep(880, 100);
                break;
        case 23:  //Ra#
                Beep(932, 100);
                break;
        case 24:  //Si#
                Beep(988, 100);
                break;
        case 25:  //Do
                Beep(1047, 100);
                break;
        case 30:  //error
                Beep(138, 100);
        }
}
```

7) map.c

```
#include "map.h"

//UI
char ui[4][58] = {    //UI창 크기            - 글자를 출력해야 되기대문에 1칸씩
        {"11111111111111111111111111111111111111111111111111111111111111"},
        {"10000000000000000000000000000000000000000000000000000000000001"},
        {"10000000000000000000000000000000000000000000000000000000000001"},
        {"10000000000000000000000000000000000000000000000000000000000001"}
};

//Field
char field[16][29] = {            //플레이창 크기       - 특수문자를 출력해야 되기대문에 2칸씩
        {"11111111111111111111111111111"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"10000000000000000000000000001"},
        {"11111111111111111111111111111"}
};

//Stage Setting

/*
        y <=5 y >=18 +1씩
        x <=2 x >=54 +2씩
*/

int maxLevel = 10;

int stage_1[19][3] = { {8, 22}, {8, 24}, {8, 26}, {8, 28}, {8, 30}, {8, 32}, {8, 34}, {8, 36}, {9, 36}, {10, 36},
```

```
                                                    {11, 36}, {12, 36}, {13, 36}, {13, 34}, {13, 32},
{13, 30}, {13, 28}, {13, 26}, {13, 24} };

int stage_2[22][3] = { {13, 22}, {13, 20}, {13, 18}, {13, 16}, {12, 16}, {11, 16}, {10, 16}, {9, 16}, {8, 16}, {7,
16},
                                                    {6, 16}, {6, 18}, {6, 20}, {6, 22}, {6, 24}, {6, 26},
{6, 28}, {6, 30}, {7, 32}, {8, 34}, {9, 34},
                                                    {10, 34} };

int stage_3[29][3] = { {10, 32}, {10, 30}, {10, 28}, {10, 26}, {9, 24}, {9, 22}, {9, 20}, {9, 18}, {8, 16}, {8, 14},
                                                    {8, 12}, {8, 10}, {9, 8}, {10, 6}, {11, 4}, {12, 4},
{13, 4}, {14, 4}, {15, 4}, {16, 4},
                                                    {17, 6}, {17, 8}, {16, 10}, {15, 10}, {14, 12}, {14,
14}, {14, 16}, {15, 16}, {16, 16} };

int stage_4[33][3] = { {16, 18}, {16, 20}, {16, 22}, {16, 24}, {15, 26}, {15, 28}, {16, 30}, {16, 32}, {15, 34}, {15,
36},
                                                    {15, 38}, {16, 40}, {15, 42}, {14, 44}, {13, 46},
{12, 44}, {11, 44}, {10, 46}, {9, 46}, {8, 46},
                                                    {8, 44}, {8, 42}, {8, 40}, {8, 38}, {9, 36}, {9, 34},
{8, 32}, {8, 30}, {8, 28}, {9, 26},
                                                    {9, 24}, {10, 22}, {10, 20} };

int stage_5[37][3] = { {11, 18}, {11, 16}, {11, 14}, {10, 14}, {9, 12}, {8, 10}, {9, 8}, {10, 6}, {11, 6}, {12, 4},
{13, 4},
                                                    {14, 4}, {15, 6}, {16, 8}, {16, 10}, {15, 12}, {15,
14}, {15, 16}, {15, 18}, {15, 20}, {16, 22},
                                                    {16, 24}, {16, 26}, {17, 28}, {17, 30}, {17, 32},
{16, 34}, {15, 36}, {15, 38}, {15, 40}, {15, 42},
                                                    {15, 44}, {16, 46}, {16, 48}, {15, 50} , {14, 50},
{13, 50} };

int stage_6[44][3] = { {12, 50}, {11, 50}, {10, 50}, {9, 50}, {9, 48}, {9, 46}, {9, 44}, {9, 42}, {8, 40}, {8, 38}, {8,
36}, {8, 34}, {9, 32}, {8, 30}, {9, 28}, {8, 26},
                                                    {7, 24}, {7, 22}, {7, 20}, {7, 18}, {8, 16}, {8, 14},
{8, 12}, {8, 10}, {9, 10}, {10, 10}, {11, 10}, {12, 10}, {13, 10}, {14, 12}, {14, 14}, {14, 16}, {15, 18},
                                                    {14, 20}, {14, 22}, {15, 24}, {14, 26}, {14, 28},
{14, 30}, {15, 32}, {14, 34}, {15, 36}, {15, 38}, {15, 40} };

int stage_7[56][3] = { {15, 42}, {15, 44}, {16, 46}, {16, 48}, {17, 50}, {16, 52}, {15, 52}, {14, 52}, {13, 52}, {12,
52}, {11, 52}, {11, 50}, {12, 48}, {12, 46}, {11, 46},
```

{10, 46}, {9, 46}, {8, 46}, {7, 48}, {6, 46}, {6, 44}, {6, 42}, {7, 40}, {8, 40}, {8, 38}, {8, 36}, {8, 34}, {9, 34}, {10, 34}, {11, 34} , {11, 32}, {10, 30}, {9, 30}, {8, 30},

{7, 30}, {7, 28}, {7, 26}, {7, 24}, {7, 22}, {6, 20}, {6, 18}, {6, 16}, {6, 14}, {6, 12},

{7, 12}, {8, 12}, {8, 14}, {8, 16}, {9, 16}, {10, 16}, {10, 14}, {10, 12}, {11, 12}, {12, 12}, {12, 10}, {12, 8} };

int stage_8[70][3] = { {12, 6}, {11, 4}, {10, 4}, {9, 6}, {8, 4}, {7, 4}, {6, 6}, {6, 8}, {7, 10}, {6, 12}, {6, 14}, {7, 16}, {6, 18}, {7, 20}, {6, 22}, {7, 24}, {6, 26}, {7, 28}, {6, 30}, {6, 32}, {6, 34}, {6, 36},

{7, 38}, {8, 40}, {7, 42}, {8, 44}, {8, 46}, {8, 48}, {8, 50}, {8, 52}, {9, 52}, {10, 52}, {11, 52}, {12, 52}, {13, 50}, {14, 50}, {15, 50},

{15, 48}, {15, 46}, {15, 44}, {16, 42}, {16, 40}, {16, 38}, {16, 36}, {16, 34}, {16, 32}, {15, 30}, {14, 30}, {13, 30},

{13, 32}, {13, 34}, {13, 36}, {13, 38}, {13, 40}, {12, 40}, {11, 40}, {10, 40}, {10, 38}, {11, 36}, {10, 34}, {10, 32},

{11, 30}, {11, 28}, {11, 26}, {10, 24}, {10, 22}, {11, 20}, {12, 18}, {13, 16}, {14, 14} };

int stage_9[80][3] = { {15, 12}, {16, 10}, {16, 8}, {16, 6}, {15, 4}, {14, 4}, {13, 6}, {12, 4}, {11, 4}, {10, 6}, {9, 8}, {8, 8}, {7, 8}, {7, 10}, {7, 12}, {7, 14},

{7, 16}, {8, 18}, {7, 20}, {7, 22}, {8, 24}, {7, 26}, {8, 28}, {7, 30}, {6, 32}, {7, 34}, {6, 36} , {7, 38}, {6, 40}, {6, 42}, {6, 44}, {7, 46},

{7, 48}, {8, 50}, {9, 50}, {10, 50}, {11, 48}, {12, 50}, {13, 50}, {14, 48}, {15, 50}, {15, 52}, {16, 52}, {17, 52}, {17, 50}, {17, 48}, {17, 46},

{16, 44}, {15, 42}, {16, 40}, {15, 38}, {15, 36}, {16, 34}, {15, 32}, {15, 30}, {16, 28}, {15, 26}, {14, 24}, {13, 24}, {13, 26}, {13, 28}, {13, 30},

{12, 30}, {12, 32}, {12, 34}, {12, 36}, {12, 38}, {13, 40}, {12, 42}, {12, 44}, {11, 44}, {10, 44}, {10, 42}, {10, 40}, {9, 40}, {9, 38}, {9, 36}, {9, 34}, {10, 32}, {10, 30} };

int stage_10[99][3] = { {9, 30}, {9, 28}, {10, 26}, {9, 24}, {10, 22}, {9, 20}, {8, 20}, {7, 22}, {6, 20}, {6, 18}, {6, 16} , {6, 14}, {7, 14}, {7, 12}, {8, 12}, {8, 10}, {8, 8}, {8, 6},

{9, 6}, {10, 6}, {10, 8}, {11, 8}, {12, 8}, {12, 6}, {13, 6}, {14, 6}, {15, 6}, {16, 6}, {16, 8}, {17, 8}, {17, 10},

{17, 12}, {16, 12}, {15, 12}, {14, 12}, {14, 14}, {14, 16}, {13, 16}, {12, 16}, {12, 18}, {12, 20}, {13, 20}, {14, 20}, {15, 20}, {16, 20}, {17, 20}, {17, 22},

{17, 24}, {16, 24}, {15, 24}, {14, 24}, {13, 24}, {13, 26}, {12, 26}, {12, 28}, {12, 30}, {13, 32}, {13, 34}, {13, 36}, {14, 38}, {15, 38}, {16, 38}, {17, 38},

{17, 40}, {17, 42}, {17, 44}, {17, 46}, {16, 46}, {15, 46}, {15, 44}, {14, 44}, {13, 44}, {13, 42}, {12, 42}, {12, 40}, {11, 40}, {10, 40}, {10, 42}, {10, 44}, {10,

```
46},
                                                {11, 48}, {12, 48}, {12, 50}, {12, 52}, {11, 52},
{10, 52}, {9, 52}, {8, 52}, {8, 50}, {8, 48}, {7, 46}, {7, 44}, {6, 42}, {6, 40}, {7, 38}, {7, 36}, {7, 34} , {7, 32},
{7, 30}};

//Sound Setting

/*
        0 = 무음, 1 = 도, 2 = 도#, 3 = 레, 4 = 레#, 5 = 미, 6 = 파, 7 = 파#, 8 = 솔, 9 = 솔#, 10 = 라, 11
= 라#, 12 = 시,
                                13 = 도, 14 = 도#, 15 = 레, 16 = 레#, 17 = 미, 18 = 파, 19 = 파#, 20 = 솔,
21 = 솔#, 22 = 라, 23 = 라#, 24 = 시
*/
int menuSound_1[32] = { 12, 12, 12, 0, 12, 12, 12, 0, 15, 12, 12, 10, 8, 10, 12, 0, 12, 12, 12, 0, 12, 12,
12, 0, 12, 10, 8, 10, 8, 5, 5, 0 };
//오징어게임 OST <Easter Egg>

int failedSound_0[16] = { 13, 0, 0, 0, 10, 0, 0, 0, 13, 0, 0, 0, 10, 0, 0, 0 };
int failedSound_1[16] = { 13, 0, 0, 0, 12, 0, 0, 0, 13, 0, 0, 0, 12, 0, 0, 0 };
int failedSound_2[16] = { 13, 0, 0, 0, 6, 0, 0, 0, 13, 0, 0, 0, 6, 0, 0, 0 };
int failedSound_3[16] = { 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13 };
int failedSound_4[16] = { 13, 0, 0, 0, 15, 0, 0, 13, 15, 0, 0, 13, 15, 0, 0, 13 };
// 오징어게임 OST - Pink Soldiers

int endingSound[288] = { 5, 0, 10, 0, 0, 0, 0, 0, 5, 0, 10, 0, 0, 0, 0, 0, 5, 0, 10, 0, 12, 0, 14, 0, 15, 0, 17,
0, 0, 0, 10, 0, 0, 0, 0, 0, 19, 0, 17, 0, 15, 14, 0, 0, 15, 0, 12, 0, 9, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0,
12, 0, 14, 0, 15, 0, 0, 0, 14, 0, 0, 0, 12, 0, 0, 0, 0, 0,
                                                5, 0, 10, 0, 0, 0, 0, 0, 5, 0, 10, 0, 0, 0, 0, 0,
5, 0, 10, 0, 12, 0, 14, 0, 15, 0, 17, 0, 0, 0, 10, 0, 0, 0, 0, 0, 19, 0, 17, 0, 15, 14, 0, 0, 15, 0, 12, 0, 9,
10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 12, 0, 14, 0, 15, 0, 0, 0, 14, 0, 0, 0, 12, 0, 0, 0, 9, 0, 0, 0, 10, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                                17, 0, 19, 17, 0, 0, 0, 0, 17, 0, 19, 17, 0, 0,
0, 0, 17, 0, 15, 0, 14, 0, 15, 17, 0, 0, 0, 0, 0, 0, 0, 17, 0, 19, 17, 0, 0, 0, 0, 17, 0, 19, 17, 0, 0, 0, 0,
17, 0, 22, 0, 21, 0, 19, 17, 0, 0, 0, 0,
                                                17, 0, 14, 0, 10, 0, 0, 0, 12, 0, 0, 0, 14, 0, 0,
0, 19, 0, 0, 0, 17, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17, 0, 17, 0, 17, 0, 17, 0, 17, 0, 0, 0, 0, 0 };
//밋치리네코 - 밋치리네코 행진

int sound_1[19] = { 5, 8, 13, 5, 8, 0, 6, 0, 10, 0, 10, 0, 8, 12, 15, 18, 17, 15, 13 };
//옹달샘 - 깊은 산안속 옹달샘 누가와서 먹나요
```

```
int sound_2[22] = { 1, 5, 8, 1, 5, 8, 10, 10, 10, 8, 0, 0, 6, 6, 6, 5, 5, 5, 3, 3, 3, 1 };
//똑같아요 - 무엇이 무엇이 똑같은가 젓가락 두 짝이 똑같아요

int sound_3[29] = { 1, 0, 1, 0, 8, 0, 8, 0, 10, 0, 10, 0, 8, 0, 0, 0, 6, 0, 6, 0, 5, 0, 5, 0, 3, 0, 3, 0, 1 };
//작은별 - 반짝 반짝 작은별 아름답게 비치네

int sound_4[33] = { 3, 0, 5, 0, 8, 8, 8, 8, 8, 8, 8, 3, 5, 8, 0, 8, 8, 8, 8, 8, 3, 5, 8, 0, 8, 8, 8, 8, 8, 8, 8,
7, 7 };
//아기상어 - 아기상어 뚜루루뚜루 귀여운 뚜루루뚜루 바닷속 뚜루루뚜루 아기상어

int sound_5[37] = { 0, 13, 13, 13, 15, 17, 15, 13, 0, 13, 13, 13, 12, 10, 12, 13, 13, 13, 13, 13, 12, 8, 8, 6,
5, 0, 8, 13, 0, 0, 0, 0, 5, 0, 8, 17, 0 };
//달빛천사 - 나의 마음을 담아 - 슬픔에 기억들에 기쁨을 채워줄꺼야 넘치는 음악속에 리듬을 리듬을

int sound_6[44] = { 13, 13, 13, 13, 13, 13, 0, 13, 13, 1, 1, 1, 1, 0, 0, 0, 13, 13, 13, 13, 13, 13, 0, 13, 16,
16, 16, 16, 16, 0, 0, 0, 0, 16, 0, 13, 16, 0, 16, 0, 16, 13, 0, 0 };
//검정고무신 - 검정고무신 - 할아버지 할머니 어렸을 적에 신으셨던 추억의 검정고무신 검정 고무신

int sound_7[56] = { 13, 0, 11, 10, 0, 0, 0, 0, 10, 0, 8, 6, 0, 0, 0, 10, 0, 10, 13, 0, 11, 0, 0, 0, 11, 0, 3, 5,
0, 0, 0, 0, 0, 0, 13, 0, 11, 0, 11, 11, 0, 13, 0, 0, 10, 0, 10, 0, 10, 0, 10, 0, 6, 0, 0, 0 };
//카드캡터 체리 - Catch You Catch Me - 오늘도 너에게 달려가는 이 마음 난 정말정말 너를 좋아해

int sound_8[70] = { 17, 18, 20, 0, 17, 18, 20, 17, 0, 18, 20, 0, 20, 0, 22, 0, 24, 0, 24, 0, 0, 25, 0, 0, 17,
0, 0, 0, 0, 0, 0, 20, 20, 20, 20, 0, 0, 22, 0, 0, 13, 0, 0, 0, 0, 0, 0, 20, 20, 20, 20, 0, 0, 22, 0, 0, 25, 0,
24, 25, 0, 24, 0, 0, 24, 0, 25, 0, 0, 0 };
//이누야샤 - I am - 떠나자 이 길이 저 멀리 계속된다 해도 언젠간 내 안에 너만을 영원히 지키고 있어

int sound_9[80] = { 19, 17, 17, 0, 15, 15, 12, 15, 17, 15, 19, 0, 0, 0, 0, 0, 19, 17, 17, 0, 15, 15, 12, 15,
17, 15, 22, 17, 19, 0, 0, 0, 19, 17, 17, 0, 15, 15, 12, 15, 17, 15, 24, 0, 19, 17, 15, 0, 12, 14, 15, 0, 15,
15, 15, 14, 15, 14, 14, 0, 12, 0, 0, 0, 12, 14, 15, 0, 15, 15, 15, 14, 15, 14, 14, 0, 12, 0, 0, 0 };
// 꽃보다 남자 - 내 머리가 나빠서 - 니가 너무 생각나는 날엔 가슴시리고 슬픈 날에는 니가 보고싶다 입가에 맵돌아 혼자 다시 또 crying for you 혼자 다시 또 missing for you

int sound_10[99] = { 0, 15, 10, 15, 10, 15, 17, 19, 0, 0, 0, 0, 15, 0, 10, 0, 10, 15, 10, 15, 10, 15, 17, 19,
0, 0, 0, 0, 15, 0, 10, 0, 10, 22, 19, 22, 19, 17, 15, 17, 19, 0, 0, 0, 15, 0, 12, 10, 0, 0, 10, 15, 10, 15, 17,
20, 19, 15, 20, 19, 15, 20, 19, 15, 17, 0, 0, 0, 0, 0, 0, 15, 0, 14, 0, 10, 0, 15, 0, 14, 0, 10, 0, 15, 0, 14,
0, 20, 0, 19, 0, 17, 0, 15, 0, 15, 0, 0, 0 };
// 비긴어게인 - A Higher Place - You take me to another space in time You take me to a higher place So I, I'm about to get out of the race I don't mind You ought to know that everything's nothing if i don't have you
```

8) game.c

```
#define _CRT_SECURE_NO_WARNINGS //fscanf 오류 경고 메세지 제거를 위한 정의

#include "game.h"
#include "util.h"
#include "map.h"

bool isPlay = false;
bool developerMode = false;                    // <<< default = false >>>    // true : 체력이 줄지 않음,
다른 키를 눌러도 맞은 판정, 타이머 over 없음
bool developerMode_Text = false;       //현재 스테이지의 남은 개수, 현재 체력
bool isFirst = true;                           //프로그램을 처음 켜서 시작 시
bool getHealth = false;
bool isEnding = false;
bool endingcreditplaying = false;

int moveIndex = 0;                     //position value          //스테이지가 시작되고 움직인 횟수
int stageLevel = 1;                    //current level                  //현재  스테이지  레벨_스테
이지 관리
int score = 0;                                 //current score                  //나의 score
int stageLen = 0;                      //current stage length       //스테이지 끝을 알기위함
int soundIndex = 0;                    //current sound                  //현재 나와야될 소리 관리
int currentKey = 5;                    //current inputKey           //현재 눌려지고 있는 키

int bestScore = 0;                     //현재 최고기록

int health = 3;                                //기본 추가체력
int healthPos[5][2];           //max health is 5  //추가체력 최대 개수 5개

double leftTime = 9.5;
double leftTimes[10] = { 9.5, 10.5, 13, 14, 14.8, 16.5, 19.6, 22.7, 24, 27 }; //1stage => 0.5sec -- 0.025sec

//Player Position
int xpos = 0;                  //처음 위치 조정때만 사용됨
int ypos = 0;

//Current Stage, Sound
int stage[100][3];             //현재 스테이지
int sound[100];                        //현재 사운드
int menuSound[32];             //for 오징어게임 ost 이스터에그
```

```
//if added a stage, Need edit it
void stageControl() {          //현재 스테이지, 노래를 가져오는 함수, 초기위치 세팅

        switch (stageLevel) {
        case 1:
                memcpy(stage, stage_1, sizeof(stage_1));
                memcpy(sound, sound_1, sizeof(sound_1));
                stageLen = sizeof(stage_1) / sizeof(stage_1[0]);
                xpos = stage[moveIndex][1] - 2;          // - 2 = left Start, + 2 = right Start
                ypos = stage[moveIndex][0];
                //start health position setting
                for (int i = 0; i < health; i++) {                    //처음 스테이지를 시작 할 때는 체력의
위치를 정해줘야 함
                        healthPos[i][0] = xpos - (2 * (i + 1));
                        healthPos[i][1] = ypos;
                }
                break;
        case 2:
                memcpy(stage, stage_2, sizeof(stage_2));
                memcpy(sound, sound_2, sizeof(sound_2));
                stageLen = sizeof(stage_2) / sizeof(stage_2[0]);
                xpos = stage[moveIndex][1] + 2;
                ypos = stage[moveIndex][0];
                break;
        case 3:
                memcpy(stage, stage_3, sizeof(stage_3));
                memcpy(sound, sound_3, sizeof(sound_3));
                stageLen = sizeof(stage_3) / sizeof(stage_3[0]);
                xpos = stage[moveIndex][1] + 2;
                ypos = stage[moveIndex][0];
                break;
        case 4:
                memcpy(stage, stage_4, sizeof(stage_4));
                memcpy(sound, sound_4, sizeof(sound_4));
                stageLen = sizeof(stage_4) / sizeof(stage_4[0]);
                xpos = stage[moveIndex][1] - 2;
                ypos = stage[moveIndex][0];
                break;
        case 5:
                memcpy(stage, stage_5, sizeof(stage_5));
                memcpy(sound, sound_5, sizeof(sound_5));
```

```
            stageLen = sizeof(stage_5) / sizeof(stage_5[0]);
            xpos = stage[moveIndex][1] + 2;
            ypos = stage[moveIndex][0] - 1;
            break;
        case 6:
            memcpy(stage, stage_6, sizeof(stage_6));
            memcpy(sound, sound_6, sizeof(sound_6));
            stageLen = sizeof(stage_6) / sizeof(stage_6[0]);
            xpos = stage[moveIndex][1];
            ypos = stage[moveIndex][0] + 1;
            break;
        case 7:
            memcpy(stage, stage_7, sizeof(stage_7));
            memcpy(sound, sound_7, sizeof(sound_7));
            stageLen = sizeof(stage_7) / sizeof(stage_7[0]);
            xpos = stage[moveIndex][1] - 2;
            ypos = stage[moveIndex][0];
            break;
        case 8:
            memcpy(stage, stage_8, sizeof(stage_8));
            memcpy(sound, sound_8, sizeof(sound_8));
            stageLen = sizeof(stage_8) / sizeof(stage_8[0]);
            xpos = stage[moveIndex][1] + 2;
            ypos = stage[moveIndex][0];
            break;
        case 9:
            memcpy(stage, stage_9, sizeof(stage_9));
            memcpy(sound, sound_9, sizeof(sound_9));
            stageLen = sizeof(stage_9) / sizeof(stage_9[0]);
            xpos = stage[moveIndex][1] + 2;
            ypos = stage[moveIndex][0] - 1;
            break;
        case 10:
            memcpy(stage, stage_10, sizeof(stage_10));
            memcpy(sound, sound_10, sizeof(sound_10));
            stageLen = sizeof(stage_10) / sizeof(stage_10[0]);
            xpos = stage[moveIndex][1];
            ypos = stage[moveIndex][0] + 1;
            break;
        }
}
```

```
//GetKey
void keyControl() {             //키 입력 컨트롤러
        currentKey = NOTHING;
        if (GetAsyncKeyState(VK_UP) & 0x0001 || GetAsyncKeyState(0x57) & 0x0001) {
//0x57 == w     //for 왼손잡이
                currentKey = UP;
        }
        if (GetAsyncKeyState(VK_LEFT) & 0x0001 || GetAsyncKeyState(0x41) & 0x0001) { //0x57 == a
                currentKey = LEFT;
        }
        if (GetAsyncKeyState(VK_RIGHT) & 0x0001 || GetAsyncKeyState(0x44) & 0x0001) {
//0x57 == d
                currentKey = RIGHT;
        }
        if (GetAsyncKeyState(VK_DOWN) & 0x0001 || GetAsyncKeyState(0x53) & 0x0001) {
//0x57 == s
                currentKey = DOWN;
        }
        if (GetAsyncKeyState(VK_SPACE) & 0x0001 || GetAsyncKeyState(VK_RETURN) & 0x0001) {
                currentKey = ENTER;
        }
        if (GetAsyncKeyState(VK_ESCAPE) & 0x0001) {
                currentKey = ESCAPE;
        }
}

//Draw Title
void drawTitle() {    //메인 화면 타이틀 그림
        printf("\n\n\n\n");
        setColor(Light_Red, Black); printf("         ##       ##     "); setColor(Light_Yellow, Black); printf("#
  #     "); setColor(Light_Green, Black); printf(" #####    ");   setColor(Light_Blue, Black); printf("#####    ");
setColor(Light_Purple, Black); printf(" ###          \n");
        setColor(Light_Red, Black); printf("          # # # #      "); setColor(Light_Yellow, Black); printf("#
  #     "); setColor(Light_Green, Black); printf("#           ");   setColor(Light_Blue, Black); printf("   #       ");
setColor(Light_Purple, Black); printf("#    #        \n");
        setColor(Light_Red, Black); printf("          #   #   #      "); setColor(Light_Yellow, Black); printf("#
  #     "); setColor(Light_Green, Black); printf(" ####      ");   setColor(Light_Blue, Black); printf("   #       ");
setColor(Light_Purple, Black); printf("#             \n");
        setColor(Light_Red, Black); printf("          #       #      "); setColor(Light_Yellow, Black); printf("#
 #     "); setColor(Light_Green, Black); printf("       #     ");   setColor(Light_Blue, Black); printf("   #       ");
```

```
setColor(Light_Purple, Black); printf("#    #         \n");
            setColor(Light_Red, Black); printf("          ###    ###    "); setColor(Light_Yellow, Black); printf("
####       "); setColor(Light_Green, Black); printf("#####       "); setColor(Light_Blue, Black); printf("#####
"); setColor(Light_Purple, Black); printf(" ###          \n");
            setColor(Bright_White, Black);
}
//Draw Menu
int drawMenu() {    //메인 화면 키 조작

            int menuXos = 25;
            int menuYos = 12;
            int menuSoundIndex = 0;

            memcpy(menuSound, menuSound_1, sizeof(menuSound_1));

            gotoxy(menuXos - 2, menuYos);
            setColor(Bright_White, Black);
            printf("> 게임시작");
            gotoxy(menuXos, menuYos + 1);
            setColor(Gray, Black);
            printf("게임정보");
            gotoxy(menuXos, menuYos + 2);
            printf("  종료  ");

            while (1) {          //키 입력 시 '>'이동과 다른 글자들을 음영 처리를 위한 설정
                        keyControl();
                        switch (currentKey) {
                        case UP:
                                    if (menuYos > 12) {
                                                gotoxy(menuXos - 2, menuYos);
                                                printf("  ");
                                                gotoxy(menuXos - 2, --menuYos);
                                                setColor(Bright_White, Black);
                                                printf("> ");

                                                if (menuYos == 13) {
                                                            gotoxy(menuXos, ++menuYos);
                                                            setColor(Gray, Black);
                                                            printf("  종료  ");
                                                            gotoxy(menuXos, --menuYos);
                                                            setColor(Bright_White, Black);
```

```
                        printf("게임정보");
                }
                else if (menuYos == 12) {
                        gotoxy(menuXos, ++menuYos);
                        setColor(Gray, Black);
                        printf("게임정보");
                        gotoxy(menuXos, --menuYos);
                        setColor(Bright_White, Black);
                        printf("게임시작");
                }
        }
        if (menuSoundIndex == sizeof(menuSound_1) / sizeof(int))
                menuSoundIndex = 0;
        _beginthreadex(NULL,        0,        (_beginthreadex_proc_type)noteSound,
(int*)menuSound[menuSoundIndex], 0, NULL);
        menuSoundIndex++;
        currentKey = NOTHING;
        break;

    case DOWN:
        if (menuYos < 14) {
                gotoxy(menuXos - 2, menuYos);
                printf("  ");
                gotoxy(menuXos - 2, ++menuYos);
                setColor(Bright_White, Black);
                printf("> ");

                if (menuYos == 13) {
                        gotoxy(menuXos, --menuYos);
                        setColor(Gray, Black);
                        printf("게임시작");
                        gotoxy(menuXos, ++menuYos);
                        setColor(Bright_White, Black);
                        printf("게임정보");
                }
                else if (menuYos == 14) {
                        gotoxy(menuXos, --menuYos);
                        setColor(Gray, Black);
                        printf("게임정보");
                        gotoxy(menuXos, ++menuYos);
                        setColor(Bright_White, Black);
```

```
                                        printf("   종료   ");
                                }
                        }
                        if (menuSoundIndex == sizeof(menuSound_1) / sizeof(int))
                                menuSoundIndex = 0;
                        _beginthreadex(NULL,        0,        (_beginthreadex_proc_type)noteSound,
(int*)menuSound[menuSoundIndex], 0, NULL);
                        menuSoundIndex++;
                        currentKey = NOTHING;
                        break;

                case LEFT:
                case RIGHT:
                        if (menuSoundIndex == sizeof(menuSound_1) / sizeof(int))
                                menuSoundIndex = 0;
                        _beginthreadex(NULL,        0,        (_beginthreadex_proc_type)noteSound,
(int*)menuSound[menuSoundIndex], 0, NULL);
                        menuSoundIndex++;
                        currentKey = NOTHING;
                        break;

                case ENTER:
                        noteSound(12);
                        noteSound(15);
                        currentKey = NOTHING;
                        return menuYos - 12;        //초기값이 12이기 때문에 현재 선택된 값에서 12
를 빼주면 게임시작 = 0, 게임정보 = 1, 종료 = 2가 됨
                }
        }
}

//Draw Map
void drawMap() {    //문자를 그림으로 바꿔주는 함수
        system("cls");
        int h, w;
        //UI
        for (h = 0; h < sizeof(ui) / sizeof(ui[0]); h++) {
                for (w = 0; w < sizeof(ui[1]) / sizeof(char); w++) {
                        char uiCode = ui[h][w];
                        switch (uiCode) {
                        case '0': setColor(Bright_White, Black); printf(" "); break;
```

```
                        case '1': setColor(Bright_White, Bright_White); printf("#"); break;
                        }
                }
                printf("\n");
        }
        //MAP
        for (h = 0; h < sizeof(field) / sizeof(field[0]); h++) {
                for (w = 0; w < sizeof(field[1]) / sizeof(char); w++) {
                        char mapCode = field[h][w];
                        switch (mapCode) {
                        case '0': setColor(Bright_White, Black); printf("  "); break;
                        case '1': setColor(Bright_White, Bright_White); printf("■"); break;
                        }
                }
                if (h != 15) {
                        printf("\n");
                }
        }
}
//Draw Stage
void drawStage() {  //화면에 해당 스테이지를 그리는 함수
        srand(time(NULL));

        //Sgtage Setting - get current stage
        stageControl();

        //Draw Player
        gotoxy(xpos, ypos);
        setColor(Light_Yellow, Black);
        printf("◎");

        //Draw KeySet
        for (int index = 0; index < stageLen; index++) {
                int random = rand() % 5;
                switch (random) {
                case 0:
                        gotoxy(stage[index][1], stage[index][0]);
                        stage[index][2] = UP;
                        setColor(Light_Purple, Black);
                        printf("↑");
                        break;
```

```
                case 1:
                        gotoxy(stage[index][1], stage[index][0]);
                        stage[index][2] = DOWN;
                        setColor(Light_Red, Black);
                        printf("↓");
                        break;
                case 2:
                        gotoxy(stage[index][1], stage[index][0]);
                        stage[index][2] = LEFT;
                        setColor(Light_Green, Black);
                        printf("←");
                        break;
                case 3:
                        gotoxy(stage[index][1], stage[index][0]);
                        stage[index][2] = RIGHT;
                        setColor(Light_Yellow, Black);
                        printf("→");
                        break;
                case 4:
                        random = rand() % 8;                    //추가체력 확률
                        gotoxy(stage[index][1], stage[index][0]);
                        if (random == 0) {
                                setColor(Light_Yellow, Black);
                                stage[index][2] = ENTER_GOLD;
                        }
                        else {
                                setColor(Light_Blue, Black);
                                stage[index][2] = ENTER;
                        }
                        printf("♬");
                        break;
                }
        }
}

//Game Loop
void update() {      //main에서 게임 시작 시 이 함수로 넘어오며 이 루프를 반복하게 됨
        isPlay = true;
        drawMap();                              //맵을 그림
        drawStage();                    //스테이지를 그림
        drawHealth();                   //체력들을 그림
```

```
	againTimerSet();	//타이머 설정
	if (isFirst) {
		_beginthreadex(NULL, 0, (_beginthreadex_proc_type)timer, NULL, 0, NULL);
		isFirst = false;
	}
	while (isPlay) {
		keyControl();
		switch (currentKey) {
		case UP:
			if (stage[moveIndex][2] == UP || developerMode) { //알맞은 방향키를 입력 시
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)sound[soundIndex], 0, NULL);
				score++;
			}
			else {
					//잘못된 방향키를 입력 시
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)30, 0, NULL);  //note sound 30 = 틀렸을 때 소리
				removeHealth();
				health--;
			}
			move();
			soundIndex++;
			currentKey = NOTHING;
			break;
		case DOWN:
			if (stage[moveIndex][2] == DOWN || developerMode) {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)sound[soundIndex], 0, NULL);
				score++;
			}
			else {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)30, 0, NULL);
				removeHealth();
				health--;
			}
			move();
			soundIndex++;
			currentKey = NOTHING;
			break;
```

```
		case LEFT:
			if (stage[moveIndex][2] == LEFT || developerMode) {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)sound[soundIndex], 0, NULL);
				score++;
			}
			else {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)30, 0, NULL);
				removeHealth();
				health--;
			}
			move();
			soundIndex++;
			currentKey = NOTHING;
			break;
		case RIGHT:
			if (stage[moveIndex][2] == RIGHT || developerMode) {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)sound[soundIndex], 0, NULL);
				score++;
			}
			else {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)30, 0, NULL);
				removeHealth();
				health--;
			}
			move();
			soundIndex++;
			currentKey = NOTHING;
			break;
		case ENTER:
			if (stage[moveIndex][2] == ENTER || developerMode && stage[moveIndex][2]
!= ENTER_GOLD) {
				_beginthreadex(NULL,	0,	(_beginthreadex_proc_type)noteSound,
(int*)sound[soundIndex], 0, NULL);
				score++;
			}
			else if (stage[moveIndex][2] == ENTER_GOLD || developerMode) {
				if (health < 5) {
```

```
                                        getHealth = true;
                                        health++;
                                }
                                _beginthreadex(NULL,    0,    (_beginthreadex_proc_type)noteSound,
(int*)sound[soundIndex], 0, NULL);
                                score++;
                        }
                        else {
                                _beginthreadex(NULL,    0,    (_beginthreadex_proc_type)noteSound,
(int*)30, 0, NULL);
                                removeHealth();
                                health--;
                        }
                        move();
                        soundIndex++;
                        currentKey = NOTHING;
                        break;
                case ESCAPE:                    //esc를 누를 시 게임 탈출
                        isPlay = false;
                        currentKey = NOTHING;
                        break;
                case NOTHING:
                        break;
                }
                drawUi();
                if (moveIndex == stageLen) {            //현재 스테이지가 끝났을 때
                        score += (int)(leftTime * 10);  //스코어 += 남은시간 * 10
                        nextStage();                                            //다음   스테이지
로 이동
                }
                if (health < 0 && !developerMode || leftTime <= 0 && !isEnding && !developerMode) {
        //실패엔딩
                        failed();
                }
        }
        reset();
}

//Remove Player Character
void removePlayer() {           //방향키 입력시 캐릭터 있던 자리를 비우는 함수
        //for first move
```

```
	if (moveIndex == 0) {
		gotoxy(xpos, ypos);
		printf("  ");
	}
	//after that move
	else {
		gotoxy(stage[moveIndex - 1][1], stage[moveIndex - 1][0]);
		printf("  ");
	}
}
//Remove last health
void removeHealth() {		//추가체력 있던 곳의 자리를 비우는 함수
	if (getHealth) {	//체력을 먹었다면 플레이어가 이동하면서 자리가 비기 때문에 체력을 지울 필요 없
음
		getHealth = false;
		return;
	}
	gotoxy(healthPos[health - 1][0], healthPos[health - 1][1]);
	printf("  ");
}
//Remove all health
void removeAllHealth() {	//스테이지 클리어 시 SPACE  TO  NEXT을 출력해주어야 하므로 체력들을 전부
지워주는 함수
	for (int i = 0; i < health; i++) {
		gotoxy(healthPos[i][0], healthPos[i][1]);
		printf("  ");
	}
}

//Setting Health Position
void setLeftHealth() {			//체력들의 위치를 재조정 하는 함수
	for (int i = health - 1; i > 0; i--) {
		healthPos[i][0] = healthPos[i - 1][0];
		healthPos[i][1] = healthPos[i - 1][1];
	}

	if (moveIndex == 0) {
		healthPos[0][0] = xpos;
		healthPos[0][1] = ypos;
	}
	else {
```

```
                healthPos[0][0] = stage[moveIndex - 1][1];
                healthPos[0][1] = stage[moveIndex - 1][0];
        }
}

//Player Move
void move() {                   //방향키 입력을 받을 시 실행되는 함수
        if (health == 0)
                removePlayer();     //움직이기 전 플레이어의 위치에 체력을 그려야 하기 때문에 플레이어
를 안지워도 무방
        else
                removeHealth();

        setColor(Light_Yellow, Black);
        gotoxy(stage[moveIndex][1], stage[moveIndex][0]);
        printf("◎");
        setLeftHealth();
        drawHealth();
        moveIndex++;
}

//Next - stage Clear
void nextStage() {   //스테이지 클리어시 실행된는 함수
        //ending
        if (stageLevel == maxLevel)    //마지막 스테이지 클리어시 엔딩으로 넘어감
                ending();
        //nextStage
        else {
                removePlayer();
                removeAllHealth();
                setColor(Light_Yellow, Black);
                gotoxy(16, 11);
                printf(" S P A C E  T O  N E X T ");
                while (1) {
                        keyControl();
                        if (currentKey == ENTER) {
                                gotoxy(16, 11);
                                printf("                          ");
                                moveIndex = 0;
                                soundIndex = 0;
                                stageLevel++;
```

```
                                        leftTime = leftTimes[stageLevel - 1];
                                        isFirst = false;
                                        drawStage();
                                        drawHealth();
                                        drawUi();
                                        break;
                                }
                                else if (currentKey == ESCAPE) {
                                        isPlay = false;
                                        break;
                                }

                        }
                }
}

void timer() {          //쉬운 구성을 위해 타이머는 Sleep으로 구현, 쓰레드로 이 함수를 호출
                while (1) {
                        Sleep(100);
                        leftTime -= 0.1;
                }
}

//Ui Draw
void drawUi() {         //화면 위쪽에 제공되는 현재 스테이지, 남은시간, 현재 스코어가 제공됨
                setColor(Light_Red, Black);
                gotoxy(6, 2);
                printf("STAGE : %d", stageLevel);
                setColor(Light_Blue, Black);
                gotoxy(23, 2);
                printf("Timer : %0.1lf", leftTime);
                setColor(Light_Yellow, Black);
                gotoxy(44, 2);
                printf("SCORE : %d", score);

                //[Test] Show Health, Left To Text
                if (developerMode_Text) {
                        setColor(Light_Green, Black);
                        gotoxy(32, 18);
                        printf("Left : %2d", stageLen - moveIndex);
                        gotoxy(45, 18);
```

```
                printf("Health : %d", health);
        }
}

//Health Draw
void drawHealth() {             //체력들을 그려주는 함수
        for (int i = 0; i < health; i++) {
                setColor(Light_Red, Black);
                gotoxy(healthPos[i][0], healthPos[i][1]);
                printf("♥");
        }
}

//Values initialization
void reset() {          //게임이 끝날 시 변수들을 리셋 해주는 함수
        isPlay = false;
        moveIndex = 0;
        stageLevel = 1;
        score = 0;
        stageLen = 0;
        soundIndex = 0;
        health = 3;
        leftTime = leftTimes[stageLevel - 1];
        xpos = 0;
        ypos = 0;
}


void againTimerSet() {          //스테이지 별로 타이머를 조정해주는 함수
        leftTime = leftTimes[stageLevel - 1];
}

//Game Failed
void failed() {         //게임 실패시 실행되는 함수
        system("cls");

        reset();
        int count = 0;
        int failedSoundCount[16] = { 0, 1, 0, 1, 2, 1, 0, 1, 3, 4, 3, 4, 3, 4, 3, 4 };        //같은 음이 많아
배열로 관리
```

```
		gotoxy(24, 9);
		setColor(Bright_White, Black); printf("[ "); setColor(White, Black); printf("Failed"); setColor(Bright_White, Black); printf(" ]");
		gotoxy(23, 14);
		setColor(Gray, Black);
		printf("Space to Menu");

		while (1) {
			keyControl();
			if (currentKey == ENTER || currentKey == ESCAPE) {
				noteSound(15);
				noteSound(12);
				Sleep(50);
				break;
			}
			Sleep(120);		//노래박자
			if (count == sizeof(failedSoundCount) / sizeof(int))		//노래 무한반복
				count = 0;
			if (soundIndex == 0) {
				switch (failedSoundCount[count++]) {
				case 0:
					memcpy(sound, failedSound_0, sizeof(failedSound_0));
					break;
				case 1:
					memcpy(sound, failedSound_1, sizeof(failedSound_1));
					break;
				case 2:
					memcpy(sound, failedSound_2, sizeof(failedSound_2));
					break;
				case 3:
					memcpy(sound, failedSound_3, sizeof(failedSound_3));
					break;
				case 4:
					memcpy(sound, failedSound_4, sizeof(failedSound_4));
					break;
				}
			}
			_beginthreadex(NULL, 0, (_beginthreadex_proc_type)noteSound, (int*)sound[soundIndex], 0, NULL);
			soundIndex++;
			if (soundIndex == sizeof(failedSound_0) / sizeof(int))
```

```
                soundIndex = 0;
        }
}

//Game Ending
void ending() {       //모든 스테이지 클리어시 실행되는 함수
        system("cls");

        isEnding = true;
        endingcreditplaying = true;
        int sleepCount = 6;                     //sleep횟수 카운트
        int creditsXpos = 25;                   //크레딧 Y위치
        int creditsYpos = 8;            //크레딧 X위치
        int line = 0;                                   //크레딧 N번째 줄
        int totalScore = score;                 //스코어 temp

        int endingCreditSound[288];             //엔딩크레딧은 사운드가 길어 따로 사운드배열 선언
        int endingSoundCount = 0;               //사운드 반복 카운트

        memcpy(endingCreditSound, endingSound, sizeof(endingSound));

        if (totalScore >= bestScore)    //스코어 저장
                saveScore(totalScore);

        gotoxy(creditsXpos, creditsYpos);
        setColor(Bright_White, Black); printf("[     ");     setColor(Light_Yellow,     Black);     printf("Clear");
setColor(Bright_White, Black); printf(" ]");

        reset();
        while (1) {
                keyControl();
                if (!endingcreditplaying && currentKey == ENTER || currentKey == ESCAPE) {
                        leftTime = 10;                  //엔딩이 끝나고 스페이스바 누르자마자 실패되는
것 방지
                        isEnding = false;
                        Sleep(50);
                        break;
                }

                Sleep(140);             //노래박자
                sleepCount++;
```

```
                _beginthreadex(NULL, 0, (_beginthreadex_proc_type)noteSound, (int*)endingSound[soundIndex], 0, NULL);
                soundIndex++;

                if (soundIndex == sizeof(endingSound) / sizeof(int))
                        soundIndex = 0;

                //Show Score
                if (totalScore >= bestScore) {
                        gotoxy(41, 18);
                        newBestScore(totalScore); //Twinkle Score
                }

                if (sleepCount % 8 == 0 && line < 43) {//노래의 음과 크레딧이 같이 올라가기 위해 구현한 장치
                        system("cls");

                        if (totalScore >= bestScore) {
                                gotoxy(41, 18);
                                newBestScore(totalScore); //Twinkle Score
                        }
                        else {
                                gotoxy(44, 18);
                                setColor(Light_Yellow, Black); printf("Score"); setColor(White, Black); printf(" : %d", totalScore); //normal Score
                        }

                        //Show Credit
                        if (creditsYpos > 4) {
                                gotoxy(creditsXpos, creditsYpos);
                                setColor(Bright_White, Black); printf("[ "); setColor(Light_Yellow, Black); printf("Clear"); setColor(Bright_White, Black); printf(" ]");
                        }
                        setColor(Bright_White, Black);
                        if (line >= 2 && creditsYpos > -4) {
                                gotoxy(creditsXpos - 3, creditsYpos + 8);
                                setColor(Light_Yellow, Black); printf("제작자"); setColor(Bright_White, Black); printf(" : 이배원");
                        }
                        if (line >= 4 && creditsYpos > -6) {
```

```
                gotoxy(creditsXpos - 2, creditsYpos + 10);
                setColor(Light_Yellow, Black); printf("기획"); setColor(Bright_White,
Black); printf(" : 이배원");
        }
        if (line >= 6 && creditsYpos > -8) {
                gotoxy(creditsXpos - 6, creditsYpos + 12);
                setColor(Light_Yellow, Black); printf("원작 게임");
setColor(Bright_White, Black); printf("  : 뮤직톡톡");
        }
        if (line >= 9 && creditsYpos > -11) {
                gotoxy(creditsXpos - 5, creditsYpos + 15);
                setColor(Light_Yellow, Black); printf("1스테이지");
setColor(Bright_White, Black); printf(" : 옹달샘");
        }
        if (line >= 11 && creditsYpos > -13) {
                gotoxy(creditsXpos - 6, creditsYpos + 17);
                setColor(Light_Yellow, Black); printf("2스테이지");
setColor(Bright_White, Black); printf(" : 똑같아요");
        }
        if (line >= 13 && creditsYpos > -15) {
                gotoxy(creditsXpos - 5, creditsYpos + 19);
                setColor(Light_Yellow, Black); printf("3스테이지");
setColor(Bright_White, Black); printf(" : 작은별");
        }
        if (line >= 15 && creditsYpos > -17) {
                gotoxy(creditsXpos - 6, creditsYpos + 21);
                setColor(Light_Yellow, Black); printf("4스테이지");
setColor(Bright_White, Black); printf(" : 아기상어");
        }
        if (line >= 17 && creditsYpos > -19) {
                gotoxy(creditsXpos - 14, creditsYpos + 23);
                setColor(Light_Yellow, Black); printf("5스테이지");
setColor(Bright_White, Black); printf(" : 달빛천사 - 나의 마음을 담아");
        }
        if (line >= 19 && creditsYpos > -21) {
                gotoxy(creditsXpos - 11, creditsYpos + 25);
                setColor(Light_Yellow, Black); printf("6스테이지");
setColor(Bright_White, Black); printf(" : 검정고무신 - 검정고무신");
        }
        if (line >= 21 && creditsYpos > -23) {
                gotoxy(creditsXpos - 18, creditsYpos + 27);
```

```
                    setColor(Light_Yellow, Black); printf("7스테이지");
setColor(Bright_White, Black); printf(" : 카트캡터 체리 - Catch You Catch Me");
                    }
                    if (line >= 23 && creditsYpos > -25) {
                              gotoxy(creditsXpos - 10, creditsYpos + 29);
                              setColor(Light_Yellow, Black); printf("8스테이지");
setColor(Bright_White, Black); printf(" : 이누야샤 - I am");
                    }
                    if (line >= 25 && creditsYpos > -27) {
                              gotoxy(creditsXpos - 17, creditsYpos + 31);
                              setColor(Light_Yellow, Black); printf("9스테이지");
setColor(Bright_White, Black); printf(" : 꽃보다 남자 - 내 머리가 나빠서");
                    }
                    if (line >= 27 && creditsYpos > -29) {
                              gotoxy(creditsXpos - 16, creditsYpos + 33);
                              setColor(Light_Yellow, Black); printf("10스테이지");
setColor(Bright_White, Black); printf(" : 비긴어게인 - A Higher Place");
                    }
                    if (line >= 29 && creditsYpos > -31) {
                              gotoxy(creditsXpos - 13, creditsYpos + 35);
                              setColor(Light_Yellow, Black); printf("실패곡");
setColor(Bright_White, Black); printf(" : 오징어게임 - Pink Soldiers");
                    }
                    if (line >= 31 && creditsYpos > -33) {
                              gotoxy(creditsXpos - 8, creditsYpos + 37);
                              setColor(Light_Yellow, Black); printf("엔딩곡");
setColor(Bright_White, Black); printf(" : 밋치리네코 행진");
                    }
                    if (line >= 37 && line < 42) {
                              gotoxy(creditsXpos - 5, creditsYpos + 43);
                              printf("Thank you for play");
                    }
                    if (line == 42) {
                              gotoxy(creditsXpos - 5, creditsYpos + 44);
                              printf("Thank you for play"); setColor(Light_Red, Black); printf("!");
                              gotoxy(23, 14);
                              setColor(Gray, Black);
                              printf("Space to Menu");
                              endingcreditplaying = false; //Finish Endingcredit from this line
          //스페이스, 엔터키 입력 가능
                    }
```

```
                creditsYpos--;
                line++;
            }
        }
}

void newBestScore(int totalScore) {     //신기록을 세울 때 엔딩함수에서 불리는 함수
        static totalscoreCount = 0;    //엔딩 함수에서 계속 부르면 static 변수를 선언하였기 때문에 newscore의 색깔이 계속 바뀜
        if (totalscoreCount == 8)
            totalscoreCount = 0;
        switch (totalscoreCount) {     //색깔이 바뀌어야 되기 때문에 글자의 색깔이 swicth문으로 계속 바뀜
            gotoxy(40, 18);
        case 0:
            setColor(Red, Black);                           printf("N");
            setColor(Light_Red, Black);             printf("e");
            setColor(Light_Yellow, Black); printf("w ");
            setColor(Light_Green, Black);  printf("S");
            setColor(Light_Cyan, Black);   printf("c");
            setColor(Light_Blue, Black);   printf("o");
            setColor(Purple, Black);                printf("r");
            setColor(Light_Purple, Black); printf("e");

            setColor(Gray, Black);                          printf(" : %d", totalScore);
            break;
        case 1:
            setColor(Light_Purple, Black); printf("N");
            setColor(Red, Black);                           printf("e");
            setColor(Light_Red, Black);             printf("w ");
            setColor(Light_Yellow, Black); printf("S");
            setColor(Light_Green, Black);  printf("c");
            setColor(Light_Cyan, Black);   printf("o");
            setColor(Light_Blue, Black);   printf("r");
            setColor(Purple, Black);                printf("e");

            setColor(White, Black);                         printf(" : %d", totalScore);
            break;
        case 2:
            setColor(Purple, Black);                printf("N");
            setColor(Light_Purple, Black); printf("e");
            setColor(Red, Black);                           printf("w ");
```

```
            setColor(Light_Red, Black);                 printf("S");
            setColor(Light_Yellow, Black); printf("c");
            setColor(Light_Green, Black);  printf("o");
            setColor(Light_Cyan, Black);   printf("r");
            setColor(Light_Blue, Black);   printf("e");

            setColor(Bright_White, Black);  printf(" : %d", totalScore);
            break;
case 3:
            setColor(Light_Blue, Black);   printf("N");
            setColor(Purple, Black);                    printf("e");
            setColor(Light_Purple, Black); printf("w ");
            setColor(Red, Black);                                   printf("S");
            setColor(Light_Red, Black);                 printf("c");
            setColor(Light_Yellow, Black); printf("o");
            setColor(Light_Green, Black);  printf("r");
            setColor(Light_Cyan, Black);   printf("e");

            setColor(Bright_White, Black);  printf(" : %d", totalScore);
            break;
case 4:
            setColor(Light_Cyan, Black);   printf("N");
            setColor(Light_Blue, Black);   printf("e");
            setColor(Purple, Black);                    printf("w ");
            setColor(Light_Purple, Black); printf("S");
            setColor(Red, Black);                                   printf("c");
            setColor(Light_Red, Black);                 printf("o");
            setColor(Light_Yellow, Black); printf("r");
            setColor(Light_Green, Black);  printf("e");

            setColor(Bright_White, Black);  printf(" : %d", totalScore);
            break;
case 5:
            setColor(Light_Green, Black);  printf("N");
            setColor(Light_Cyan, Black);   printf("e");
            setColor(Light_Blue, Black);   printf("w ");
            setColor(Purple, Black);                    printf("S");
            setColor(Light_Purple, Black); printf("c");
            setColor(Red, Black);                                   printf("o");
            setColor(Light_Red, Black);                 printf("r");
            setColor(Light_Yellow, Black); printf("e");
```

```
                setColor(Bright_White, Black);   printf(" : %d", totalScore);
                break;
        case 6:
                setColor(Light_Yellow, Black); printf("N");
                setColor(Light_Green, Black);  printf("e");
                setColor(Light_Cyan, Black);   printf("w ");
                setColor(Light_Blue, Black);   printf("S");
                setColor(Purple, Black);                printf("c");
                setColor(Light_Purple, Black); printf("o");
                setColor(Red, Black);                           printf("r");
                setColor(Light_Red, Black);             printf("e");

                setColor(White, Black);                         printf(" : %d", totalScore);
                break;
        case 7:
                setColor(Light_Red, Black);             printf("N");
                setColor(Light_Yellow, Black); printf("e");
                setColor(Light_Green, Black);  printf("w ");
                setColor(Light_Cyan, Black);   printf("S");
                setColor(Light_Blue, Black);   printf("c");
                setColor(Purple, Black);                printf("o");
                setColor(Light_Purple, Black); printf("r");
                setColor(Red, Black);                           printf("e");

                setColor(Gray, Black);                          printf(" : %d", totalScore);
                break;
        }
        totalscoreCount++;
}

void saveScore(const int _totalScore) {  //엔딩시 스코어 저장
        FILE* file;
        file = fopen("TokTokScore.txt", "w");
        if (file == NULL) {
                bestScore = 0;
        }
        else {
                fprintf(file, "%d", _totalScore);
                fclose(file);
        }
```

```
}

void getScore() {     //bextScore 가져오기
        FILE* file;
        file = fopen("TokTokScore.txt", "r");
        if (file == NULL) {
                bestScore = 0;
        }
        else {
                fscanf(file, "%d", &bestScore);
                fclose(file);
        }
}

//Infomation
void drawInfo() {     //게임 정보에를 누를 시 제공되는 함수
        system("cls");
        printf("\n\n\n");
        setColor(Bright_White, Black); printf("                                    ");
        printf("[ "); setColor(Light_Yellow, Black); printf("조작법"); setColor(Bright_White, Black); printf("
]\n\n");
        printf("                          ");
        setColor(Light_Purple, Black); printf("↑");
        setColor(Bright_White, Black); printf(", ");
        setColor(Light_Red, Black);                 printf("↓");
        setColor(Bright_White, Black); printf(", ");
        setColor(Light_Green, Black);  printf("←");
        setColor(Bright_White, Black); printf(", ");
        setColor(Light_Yellow, Black); printf("→");
        setColor(Bright_White, Black); printf(", ");
        setColor(Light_Blue, Black);    printf("♬");
        setColor(Bright_White, Black); printf("(SPACE)");
        printf("\n\n");
        printf("                                  ");
        setColor(Light_Yellow,  Black);      printf("♬");  setColor(Bright_White,  Black);  printf("  =  ");
setColor(Light_Red, Black); printf("♥"); setColor(Bright_White, Black); printf(" + 1");
        printf("\n\n\n");
        setColor(Bright_White, Black); printf("                                BestScore [ "); setColor(Light_Yellow,
Black); printf("%d", bestScore); setColor(Bright_White, Black); printf(" ] \n\n\n");
        setColor(Bright_White, Black);    printf("                          제작자 : 이배원 - "); printf("dgf0000");
setColor(Light_Green, Black); printf("@naver.com\n\n\n");
```

```
        setColor(Bright_White, Black);   printf("                    > 확   인");

        while (1) {
                keyControl();
                if (currentKey == ENTER || currentKey == ESCAPE) {
                        noteSound(15);
                        noteSound(12);
                        Sleep(50);
                        break;
                }
        }
}
```