

C++ 프로그래밍 및 실습

팩맨 게임 개발

진척 보고서 #2

제출일자: 2024.12.01

제출자명: 이상현

제출자학번: 214784

1. 프로젝트 목표

1) 배경 및 필요성

최근 수업 시간에서 MUD GAME에 대해 접하다 보니, 이러한 게임 개발에 흥미를 가지게 되었음. 그래서 이번 프로젝트의 주제로 가장 널리 알려져 있고 죽기 전에 꼭 해야 할 비디오 게임 명단에도 존재하는 팩맨(Pac-Man)을 직접 구현해보고자 하였음.

2) 프로젝트 목표

주어진 미로 안에서 팩맨이 유령들을 피해다니며 미로에 놓인 코인들을 모두 줍게 되면 승리하도록 게임 구현하기.

3) 차별점

기존 팩맨 게임의 유령들의 AI는 각 색깔의 유령마다 고유의 AI가 존재했다. 빨간 유령(Blinky)는 현재 팩맨의 위치로 이동하고, 분홍 유령(Pinky)는 팩맨의 4칸 앞 지점으로 이동하고, 초록 유령(Inky)는 팩맨의 2칸 앞 지점을 중심으로 빨간 유령(Blinky)의 대칭된 좌표로 이동하고, 노란 유령(Clyde)는 팩맨으로부터 도망가도록 설계되어 있었다. 그러나 이번 프로젝트에서는 게임 난이도 설정을 도입하여 사용자가 정한 난이도에 따라 유령들의 이동 AI와 유령의 수를 변경하는걸 차별점으로 두었다.

2. 기능 계획

1) 메인 화면 구현

- 본격적인 팩맨 게임 시작 전 화면을 만들고 사용자의 입력에 따라 게임 시작, 점수판, 게임 종료를 다룰 메인 화면을 구현

세부기능

(1) 게임 시작을 뜻하는 입력을 받았다면 팩맨 게임을 실행하도록 구현

- 진행할 게임의 난이도를 입력으로 받아 그에 맞게끔 유령의 움직임과 수가 조절되도록 구현

- 팩맨 게임을 하다가 게임 오버가 된다면 다시 메인 화면으로 오도록 구현

(2) 게임이 끝날 때 마다 현재 점수를 자동으로 기록하고, 메인 화면에서 점수판을 뜻하는 입력을 받았다면 기록된 점수들을 볼 수 있는 점수판 구현.

(3) 게임 종료를 뜻하는 입력을 받았다면 프로그램을 종료하도록 구현

2) 팩맨 게임 맵 구현

- 팩맨과 유령이 움직이는 미로 형식의 맵 구현

세부기능

(1) 벽, 코인, 아이템을 2차원 배열로 구현(각각을 뜻하는 문자열을 설정하여 구분)

- 2차원 배열로 전체 맵을 구현하고, 팩맨과 유령이 지나갈 수 없는 벽을 지정함.

- 팩맨이 맵 통로에 있는 코인을 먹게되면 (지나가면) 점수가 증가하도록 함.

- 팩맨이 맵 통로에 있는 아이템을 먹게되면 (지나가면) PowerUp 모드가 되도록 함.

- 유령의 스폰 장소를 지정해두고, 일정 시간이 지날 때 마다 한 마리씩 추가로 움직이게 함.

3) 사용자의 입력에 따른 팩맨의 움직임 구현

- 사용자로부터 상, 하, 좌, 우를 입력 받아 팩맨의 이동 방향을 결정

세부기능

(1) 사용자의 input에 따라 방향을 전환하고, 설정한 프레임에 따라 움직이기

- input으로 상, 하, 좌, 우를 사용자에게 받아오고 그에 따라 앞으로 팩맨이 움직일 방향을 결정함.

- 프레임 속도를 설정하고, 그 시간 간격에 따라 콘솔창을 갱신하면서 팩맨과 유령이 움직이는 것을 구현함.

- 평상시에는 팩맨과 유령의 이동 속도는 동일하나, 팩맨이 PowerUp 모드가 된다면 유령보다 조금 더 빠르게 이동하도록 프레임 속도를 조절함.

4) 유령들의 움직임 알고리즘 구현

- BFS를 활용하여 유령과 팩맨 사이의 최단 거리를 계산하여 이동하도록 구현

세부기능

(1) 유령들의 움직임 알고리즘 구분

- (A모드) BFS를 통해 현재 팩맨의 위치까지 최적의 경로를 찾아내서 이동하는 모드

- (B모드) 팩맨의 2칸 앞을 중심으로 하고, A모드 유령 위치와 대칭되는 좌표로 이동하는 모드

- (C모드) 일정 범위까지는 팩맨을 향해서 이동하다가, 팩맨과 너무 가까워지면 팩맨과 멀어지도록 이동하는 모드

- (D모드) 팩맨의 주변 4칸으로 이동하는 모드

- (E모드) 팩맨이 PowerUp 모드가 되었을 때, 유령이 팩맨으로부터 도망치는 모드

(2) 사용자가 설정한 난이도에 따라 유령들의 이동 알고리즘을 바꿔줄 예정

- 쉬움, 보통, 어려움 난이도로 구별함.

- 쉬움 난이도에서는 A, C모드를 사용하는 2마리의 유령을 소환

- 보통 난이도에서는 A, B, C, D모드를 각각 사용하는 4마리의 유령을 소환

- 어려움 난이도에서는 A모드 2마리, B모드 2마리, C모드 1마리, D모드 1마리 총 6마리의 유령을 소환

5) 팩맨이 유령을 잡아먹을 수 있는 PowerUp 모드 구현

- 팩맨이 맵에 놓여있는 아이템을 먹게 되면 일정 시간 동안 유령을 잡아먹을 수 있는 PowerUp 모드 구현

세부기능

- (1) 팩맨이 PowerUp 모드일 때 유령과 같은 좌표가 되면 유령을 없애고 초기 장소로 돌려보냄.
- (2) 팩맨이 PowerUp 모드가 된다면 유령들의 움직임 패턴을 E모드로 설정하여 팩맨으로부터 멀어지도록 구현
- (3) 팩맨이 PowerUp 모드일 때는 유령보다 이동 속도가 더 빨라지도록 구현

6) 잡아먹혔던 유령 부활 기능 구현

- PowerUp 모드로 인해 죽은 유령은 일정 시간 후 다시 부활하여 맵을 돌아다니도록 구현

세부기능

- (1) PowerUp 모드의 팩맨에 의해 죽게 된 유령들을 리스폰 장소에서 5초 후에 리스폰 하도록 구현

7) 게임 종료 후 점수 기록 기능 구현

세부기능

- (1) 게임이 종료되면 점수판에 점수를 자동으로 저장함.
 - 각 게임마다 팩맨(유저)에게 주어진 목숨은 한 게임.
 - 팩맨이 유령과 같은 좌표에 있게 되면 목숨이 닳아서 게임이 종료됨.
 - 게임이 종료되면 게임을 진행하면서 먹은 코인들로 인해 얻은 점수를 1차원 배열로 저장함.

- 이를 메인 화면의 점수판 항목에서 볼 수 있도록 구현.

3. 진척 사항

1) 기능 구현

(1) 메인 화면 구현

- 본격적인 팩맨 게임 시작 전 화면을 만들고 사용자의 입력에 따라 게임 시작, 점수판, 게임 종료를 다룰 메인 화면을 구현
- 입력으로 1을 받을 시 게임 맵을 출력하고 게임 시작함
- 입력으로 2를 받을 시 점수판을 출력함
- 입력으로 3을 받을 시 게임을 종료함
- 1~3을 제외한 입력을 받을 시 재입력을 요청함.

1. 코드 블록 스크린샷

```
int main() {  
    DisableCursorBlinking(); // 커서 깜빡임 제거  
    MainMenu();  
}
```

```

// 메인메뉴를 구현하는 함수
void MainMenu() {
    bool game_flag = true;
    while(game_flag){
        cout << "===== MAIN MENU =====" << endl;
        cout << "1. 게임 시작" << endl;
        cout << "2. 점수판 보기" << endl;
        cout << "3. 게임 종료" << endl;
        cout << "선택: ";
        int menu_number;
        cin >> menu_number;
        switch (menu_number) {
            case 1: {
                Initialize();
                MapSetting();
                DisplayMap();
                char input;
                int move_counter = 0; // 팩맨의 이동 횟수를 세는 카운터
                const int ghost_speed = 2; // 팩맨이 2번 움직일 때 고스트 1번 움직임
                bool game_running = true;

                while(game_running) {
                    if(_kbhit()) {
                        input = _getch();
                    }
                    MovePacman(input);
                    // 고스트의 이동 횟수를 조절하기 위함 (팩맨 2번 이동 마다 고스트 한번 이동)
                    if (move_counter % ghost_speed == 0) {
                        for (int i = 0; i < 4; i++){
                            ghosts[i].MoveGhost(pac_x, pac_y, map);
                        }
                        move_counter = 0;
                    }
                    move_counter++;
                    ClearScreen();
                    DisplayMap();
                    this_thread::sleep_for(chrono::milliseconds(100));

                    // 게임 종료 조건 (팩맨과 유령의 좌표가 같으면 종료)
                    for (int i = 0; i < 4; i++){
                        if (pac_x == ghosts[i].ghost_x && pac_y == ghosts[i].ghost_y) {
                            score_board.push_back(score); // 점수판에 현재 점수 저장
                            cout << "게임 종료!! 점수 = " << score << endl;
                            game_running = false;
                            break;
                        }
                    }
                }
            }
            break;
        }
    }
}

```

```

        case 2: {
            Initialize();
            ShowScoreboard();
            break;
        }
        case 3: {
            game_flag = false;
            break;
        }
        default: {
            cout << "잘못된 입력입니다. (1~3 입력)" << endl;
            break;
        }
    }
}
}

```

```

// 맵을 출력하는 함수
void DisplayMap() {
    for (int i = 0; i < HEIGHT; i++) {
        cout << " ";
        for (int j = 0; j < WIDTH+1; j++) {
            if (i == pac_y && j == pac_x)
                cout << PACMAN;
            else {
                bool is_ghost = false;
                for (int g = 0; g < 4; g++) {
                    if (i == ghosts[g].ghost_y && j == ghosts[g].ghost_x)
                    {
                        if (i == pac_y && j == pac_x) {
                            break;
                        }

                        cout << GHOST; // 유령 출력
                        is_ghost = true;
                        break;
                    }
                }

                if (!is_ghost)
                    cout << map[i][j];
            }
        }
        cout << endl;
    }
    cout << "현재 점수 : " << score << endl;
}
}

```

```

// 커서 깜빡임 제거 함수
void DisableCursorBlinking() {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); // 콘솔 핸들 가져오기
    CONSOLE_CURSOR_INFO cursorInfo;

    GetConsoleCursorInfo(hConsole, &cursorInfo); // 현재 커서 정보 가져오기
    cursorInfo.bVisible = FALSE; // 커서를 보이지 않게 설정
    SetConsoleCursorInfo(hConsole, &cursorInfo); // 커서 정보 갱신
}

// 점수판 출력 함수 (MAIN 메뉴에서 2 입력 시 실행 가능, 지금까지 실행했던 게임의 점수 저장)
void ShowScoreboard() {
    cout << "===== 점수판 =====< endl;
    if (score_board.size() <= 0) {
        cout << "No Game Data. Play Game Please" << endl;
    }
    else {
        for(int i = 0; i < score_board.size(); i++) {
            cout << i+1 << ". 점수: " << score_board[i] << endl;
        }
    }
}

// 콘솔 커서 위치를 맨 위로 이동시키는 함수
void ClearScreen() {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD coord = {0, 0}; // 커서를 (0, 0)으로 이동
    SetConsoleCursorPosition(hConsole, coord);
}

void Initialize()
{
    pac_x = 12;
    pac_y = 14;

    ghosts.clear();
    ghosts.push_back(Ghost(1, 1)); // 유령 1
    ghosts.push_back(Ghost(25, 1)); // 유령 2
    ghosts.push_back(Ghost(1, 26)); // 유령 3
    ghosts.push_back(Ghost(25, 26)); // 유령 4

    score = 0;
    for (int i = 0; i < 40; i++)
    {
        cout << " " << endl;
    }
    ClearScreen();
}
}

```


2. 입력

- MainMenu() 함수의 입력

- menu_number = 유저의 입력값 저장

3. 반환값

- 입력이 1이면 Initialize(), MapSetting(), DisplayMap() 함수를 실행하고 게임을 시작함
- 입력이 2면 Initialize(), ShowScoreboard() 함수 실행
- 입력이 3이면 game_flag를 false로 바꿈.
- 그 외의 값이라면 다시 입력 받음.

4. 결과

- 입력이 1일때 Initialize(), MapSetting(), DisplayMap() 함수를 실행하여 초기 설정과 맵 초기화를 하고 맵을 출력한다. 그 후 while문을 통해 유저의 키보드 입력에 따라 팩맨을 움직이며 게임을 진행함
- 입력이 2일때 Initialize(), ShowScoreboard() 함수를 실행하여 콘솔창을 지우고 지금까지 저장된 게임 점수를 출력함
- 입력이 3일때 game_flag를 false로 바꿔 while문이 반복되지 않게 하여 프로그램을 종료함

5. 설명

- pacman.cpp 파일의 main() 함수에서 DisableCursorBlinking(), MainMenu() 함수를 실행함. 이를 통해 콘솔창에서 커서의 깜빡임을 없애고, 메인 메뉴 창을 출력함.
- MainMenu() 함수에서 사용자의 입력을 받아 1이면 현재의 게임 맵을 출력하고 게임을 시작함, 2이면 게임이 종료될 때 현재 점수를 저장해 둔 점수판을 출력, 3이면 프로그램을 종료, 그 외의 값이면 다시 입력 받도록 하

였음.

- DisplayMap() 함수는 2중 for문을 통해 2차원 배열 map을 하나씩 순회하면서 해당 값과 팩맨의 좌표나 유령의 좌표가 동일하다면 팩맨과 유령에 해당하는 아이콘을 출력하고, 그렇지 않다면 map배열에 저장된 값을 출력함 (# = 벽, . = 코인(점수), * = 아이템, P = 팩맨, G = 유령)
- ShowScoreboard() 함수는 게임이 종료될 때 마다 현재의 score의 값을 저장해 둔 score_board 벡터의 값을 불러오며 몇 번째 게임에서 몇 점을 얻었는지 출력해줌. 만약 게임을 하지 않아서 점수가 없다면 게임을 먼저 수행하라는 안내 문구 출력.
- Initialize() 함수는 게임 맵을 출력하기 전에 팩맨과 고스트의 좌표, 점수를 초기화 시키면서 게임의 기초 세팅을 해주고, 콘솔창을 깔끔하게 해주기 위한 역할로 cout << " " endl을 여러 번 반복하여 맵 출력을 깔끔하게 하기 위한 기초 작업 함수이다.
- ClearScreen() 함수는 콘솔창에서 커서의 위치를 맨 위로 올려주는 역할로 맵의 출력이나 점수판의 출력이 콘솔창의 맨 위에서부터 이뤄지도록 하기 위한 함수이다
- DisableCursorBlinking() 함수는 콘솔창에서 커서를 보이지 않게 하여 게임을 진행하는 동안 맵에 깜빡임이 사라지도록 하는 역할이다.

6. 적용된 배운 내용: 함수(선언, 호출) (6~7주차), switch문(4주차), while문(4주차), for문(4주차), 벡터(10주차)

(2) 팩맨 게임 맵 구현

- 팩맨과 유령이 움직이는 미로 형식의 맵 구현

- 벽, 코인, 아이템을 2차원 배열로 구현(각각을 뜻하는 문자열을 설정하여 구분)

1. 코드 블록 스크린샷

```
// 맵 구현 (# = 벽, . = 코인(점수), * = 아이템)
char map[HEIGHT][WIDTH + 1];
char original_map[HEIGHT][WIDTH + 1] = {
    "#####",
    "#.....#",
    "#.####.#.####.###.",
    "#*####.####.#.####.*#",
    "#.####.#.####.###.",
    "#.....#",
    "#.####.##.#####.##.####.",
    "#.####.##.#####.##.####.",
    "#.....#.....#.....#",
    "#####.#####.#####.",
    "   #.####.#####.   ",
    "   #.##      ##.   ",
    "   #.## ##   ##.   ",
    "#####.##.##.#####.",
    "#   .   .   #",
    "#####.##.##.#####.",
    "   #.## ##   ##.   ",
    "   #.##      ##.   ",
    "   #.##.#####.##.   ",
    "#####.##....#....#####.",
    "#.....#####.#####.",
    "#.####.####.#.####.###.",
    "#.####.####.#.####.###.",
    "#*...#####.##.*#",
    "###.##.##.#####.##.###.",
    "###.##.##....#....###.###.",
    "#.....#####.#####.",
    "#.#####.#####.",
    "#.....#",
    "#####",
};
```

```
void MapSetting() {
    for (int i = 0; i < HEIGHT; i++) {
        for (int j = 0; j < WIDTH + 1; j++) {
            map[i][j] = original_map[i][j];
        }
    }
}
```

2. 입력 - X

3. 반환값 - X

4. 결과 - X

5. 설명

- original_map이라는 2차원 배열을 다음과 같이 선언함.

- # = 벽, . = 코인(점수), * = 아이템, P = 팩맨, G = 유령을 뜻함.

- 게임을 할 때마다 MapSetting() 함수를 실행하여 원본 역할인

original_map을 복사본인 map 배열로 복사해옴. 게임을 진행하는 동안에는 원본을 쓰지 않고 복사본인 map 배열의 값을 수정하면서 진행함

- 팩맨을 움직이면서 팩맨이 코인 (.)을 지나가면 현재 점수가 10점이 증가하도록 구현함. (이는 기능3에서 더 자세히 설명함)

6. 적용된 배운 내용: 2차원 배열(5주차), for문(4주차)

(3) 사용자의 입력에 따른 팩맨의 움직임 구현

- 사용자로부터 상, 하, 좌, 우를 입력 받아 팩맨의 이동 방향을 결정
- 사용자의 입력에 따라 방향을 전환하고, 설정한 프레임에 따라 움직이기
- 프레임 속도를 설정하고, 그 시간 간격에 따라 콘솔창을 갱신하면서 팩맨과 유령이 움직이는 것을 구현함.

1. 코드 블록 스크린샷

```
// 팩맨 이동 함수
void MovePacman(char input) {
    int dx = pac_x, dy = pac_y;
    if (input == 'w') dy--; // 위
    if (input == 's') dy++; // 아래
    if (input == 'a') dx--; // 왼쪽
    if (input == 'd') dx++; // 오른쪽

    if (map[dy][dx] == ' ' || map[dy][dx] == '.' || map[dy][dx] == '*' || map[dy][dx] == 'G') {
        if (map[dy][dx] == '.') {
            score += 10;
        }

        map[pac_y][pac_x] = ' ';

        pac_x = dx;
        pac_y = dy;
    }
}
```

2. 입력 - input

3. 반환값 - X

4. 결과 - 사용자의 입력인 input을 MainMenu()함수의 case1의 while문에서 _kbhit()이 true일 때, _getch()를 통해 저장하고, 이를 매개변수로 받아와 해당 방향에 따른 팩맨의 x, y좌표를 변경함. 그리고 옮겼을 때 팩맨의 좌표가 유효하다면 (빈 공간, 코인, 아이템, 유령의 위치) 해당 위치에 맞는

상호작용을 하고 기존 팩맨의 위치를 빈칸으로 바꿔주고, 팩맨의 좌표를 최신화 한다.

5. 설명

- input에 따라 현재 팩맨의 위치에 변화를 주고 이를 dx, dy로 저장함. 해당 dx와 dy가 빈칸이나, 코인(.), 아이템(*), 유령(G)의 위치에 있다면 이동 가능하단 뜻이므로 코인이 위에 있다면 점수를 10점 추가하고, 기존 팩맨의 위치인 map[pac_y][pac_x]를 빈칸으로 바꾸고 pac_x와 pac_y를 최신화한 dx, dy 값으로 바꿔준다.

6. 적용된 배운 내용: 함수(6~7주차), 관계연산자(4주차)

(4) 유령들의 움직임 알고리즘 구현 (구현 중)

- BFS를 활용하여 유령과 팩맨 사이의 최단 거리를 계산하여 이동하도록 구현
- BFS를 포함한 이동뿐만 아니라 여러가지 움직임 알고리즘을 만들어서 유령을 이동함
- 사용자가 난이도를 입력하면 그에 따라 유령의 수나 이동 알고리즘을 수정함

1. 코드 블록 스크린샷

ghost.h 헤더파일의 코드

```
#include <queue>
#include <utility>

class Ghost {
public:
    int ghost_x, ghost_y; // 유령의 위치 (x, y 좌표)

    Ghost(int x, int y) : ghost_x(x), ghost_y(y) {}

    void MoveGhost(int pac_x, int pac_y, char map[30][28]); // 유령 이동 함수
private:
    void BFS(int pac_x, int pac_y, char map[30][28]); // BFS 알고리즘으로 최단 경로 찾기
};
```

ghost.cpp 파일의 코드

```
#include "ghost.h"
#include <queue>
#include <iostream>
#include <utility>

using namespace std;
// 유령이 이동하는 함수 (BFS 알고리즘을 사용)
void Ghost::MoveGhost(int pac_x, int pac_y, char map[30][28]) {
    BFS(pac_x, pac_y, map); // Pac-Man의 위치를 기준으로 BFS 수행
}

// BFS 알고리즘을 사용하여 Pac-Man의 위치로 최단 경로를 찾아 이동하는 함수
void Ghost::BFS(int pac_x, int pac_y, char map[30][28]) {

    // 방향 벡터 (상, 하, 좌, 우)
    int dx[] = {0, 0, -1, 1};
    int dy[] = {-1, 1, 0, 0};

    // BFS를 위한 큐와 방문 체크 배열
    queue<pair<int, int>> q;
    bool visited[30][28] = {false};

    // 시작 위치를 큐에 삽입하고 방문 처리
    q.push({ghost_y, ghost_x});
    visited[ghost_y][ghost_x] = true;

    // 최단 경로를 저장하는 배열
    pair<int, int> prev[30][28]; // 배열 선언

    // 배열 초기화
    for (int i = 0; i < 30; ++i) {
        for (int j = 0; j < 28; ++j) {
            prev[i][j] = {-1, -1}; // 초기값으로 -1, -1을 설정
        }
    }
}
```

```
while (!q.empty()) {
    // 큐에서 front 요소를 추출
    pair<int, int> current = q.front(); // 맨 앞 요소 가져오기
    int cy = current.first;           // y 좌표
    int cx = current.second;          // x 좌표

    q.pop(); // 맨 앞 요소 제거

    // 팩맨의 위치에 도달하면 BFS 종료
    if (cy == pac_y && cx == pac_x) break;

    // 상, 하, 좌, 우로 이동
    for (int i = 0; i < 4; i++) {
        int ny = cy + dy[i];
        int nx = cx + dx[i];

        // 유효한 위치인지 확인
        if (ny >= 0 && ny < 30 && nx >= 0 && nx < 28 &&
            !visited[ny][nx] && map[ny][nx] != '#') {
            q.push({ny, nx});
            visited[ny][nx] = true;
            prev[ny][nx] = {cy, cx};
        }
    }
}

if (prev[pac_y][pac_x] == make_pair(-1, -1)) {
    for (int i = 0; i < 4; i++) {
        int ny = ghost_y + dy[i];
        int nx = ghost_x + dx[i];
        if (ny >= 0 && ny < 30 && nx >= 0 && nx < 28 && map[ny][nx] != '#') {
            ghost_y = ny;
            ghost_x = nx;
            return;
        }
    }
    return;
}

// 경로 역추적하여 유령이 이동할 위치 찾기
pair<int, int> nextMove = {pac_y, pac_x};
while (prev[nextMove.first][nextMove.second] != make_pair(ghost_y, ghost_x)) {
    nextMove = prev[nextMove.first][nextMove.second];
}

ghost_y = nextMove.first; // 유령의 y 좌표 업데이트
ghost_x = nextMove.second; // 유령의 x 좌표 업데이트
}
```

pacman.cpp 파일에서의 관련 코드

```
// 유령 객체 4개를 관리할 벡터
vector<Ghost> ghosts;

// 팩맨과 고스트의 위치를 시각적으로 표현
char PACMAN = 'P';
char GHOST = 'G';
```

```
void Initialize()
{
    pac_x = 12;
    pac_y = 14;

    ghosts.clear();
    ghosts.push_back(Ghost(1, 1));    // 유령 1
    ghosts.push_back(Ghost(25, 1));   // 유령 2
    ghosts.push_back(Ghost(1, 26));   // 유령 3
    ghosts.push_back(Ghost(25, 26));  // 유령 4

    score = 0;
    for (int i = 0; i < 40; i++)
    {
        cout << " " << endl;
    }
    ClearScreen();
}
```

```
switch (menu_number) {
    case 1: {
        Initialize();
        MapSetting();
        DisplayMap();
        char input;
        int move_counter = 0; // 팩맨의 이동 횟수를 세는 카운터
        const int ghost_speed = 2; // 팩맨이 2번 움직일 때 고스트 1번 움직임
        bool game_running = true;

        while(game_running) {
            if(_kbhit()) {
                input = _getch();
            }
            MovePacman(input);
            // 고스트의 이동 횟수를 조절하기 위함 (팩맨 2번 이동 마다 고스트 한번 이동)
            if (move_counter % ghost_speed == 0) {
                for (int i = 0; i < 4; i++){
                    ghosts[i].MoveGhost(pac_x, pac_y, map);
                }
                move_counter = 0;
            }
            move_counter++;
            ClearScreen();
            DisplayMap();
            this_thread::sleep_for(chrono::milliseconds(100));

            // 게임 종료 조건 (팩맨과 유령의 좌표가 같으면 종료)
            for (int i = 0; i < 4; i++){
                if (pac_x == ghosts[i].ghost_x && pac_y == ghosts[i].ghost_y) {
                    score_board.push_back(score); // 점수판에 현재 점수 저장
                    cout << "게임 종료!! 점수 = " << score << endl;
                    game_running = false;
                    break;
                }
            }
        }
        break;
    }
}
```

2. 입력

- MoveGhost(), BFS() 함수: 팩맨의 x, y좌표인 pac_x, pac_y, 게임을 진행하고 있는 맵인 map을 입력으로 받는다.

3. 반환값 - X

4. 결과

- pacman.cpp에서는 전역변수로 ghosts라는 고스트 객체를 관리할 벡터를 선언해두고, Initialize() 함수에서 벡터를 초기화한 후, 맵의 가장자리 부분에 한 마리씩 고스트를 배치하도록 ghosts벡터에 고스트 객체를 선언하여 넣어준다. MainMenu에서 MoveGhost()를 통해 고스트가 BFS를 통해 팩맨의 위치로 움직이도록 한다. 이를 DisplayMap을 통해 고스트 4마리와 팩맨을 포함한 맵을 출력하여 가시화한다.
- ghost.h는 고스트 클래스의 원형을 선언한 헤더 파일로 ghost의 x, y 좌표를 저장할 ghost_x, ghost_y를 생성자를 통해 초기화 한다. 또한 MoveGhost(), BFS() 함수를 선언하여 ghost.cpp 파일에서 구현하도록 하였다.
- ghost.cpp는 MoveGhost(), BFS()함수를 구체화하여 고스트의 움직임에 따른 x, y 좌표 변화를 구현하였다.

5. 설명

- BFS() 함수는 queue를 활용하여 현재 팩맨의 x, y좌표에 따른 고스트의 최단거리 이동을 구현하였으며, BFS 알고리즘을 활용하였다.
- MoveGhost() 함수는 BFS()함수를 호출하는 역할로써 추후에 다른 고스트의 이동 알고리즘을 구현했을 때, 난이도에 따라, 유형의 종류에 따라 이동 방식을 바꾸도록 구현할 예정이다. (아직 미완성)
- pacman.cpp 파일에서 MainMenu() 함수에서 MoveGhost()를 호출할 때는 팩맨이 2번 움직인 후에 고스트가 1번 움직이도록 하는 조건이 존재한다

다. 이는 둘이 동시에 움직이면 게임 진행 속도가 너무 빨라 난이도를 조절하기 위함으로써, 수치적으로 보면 팩맨이 고스트보다 2배 정도 빠르다. 이를 `move_counter`와 `ghost_speed` 변수로 구현했으며 팩맨이 움직일 때마다 `move_counter`를 1 증가시키고, `ghost_speed`를 2로 설정하여, `move_counter % ghost_speed`가 0일 때, `MoveGhost()`를 호출하도록 했다.

- 고스트의 움직임에 따라 팩맨과 `x`, `y` 좌표가 같아지면 고스트가 팩맨을 잡아먹은 것으로 간주해 게임을 종료하도록 `pacman.cpp`의 `MainMenu()`에서 구현하였다. `for`문을 사용해 `ghosts[i].ghost_x`와 `pac_x`가 같고, `ghosts[i].ghost_y`와 `pac_y`가 같다면 현재의 점수를 점수판을 뜻하는 `score_board` 벡터에 저장하고 게임을 종료한다. (기능 7의 내용)

6. 적용된 배운 내용: 헤더파일(9주차), 클래스 (생성자, 함수 원형, 함수 구현)(9주차), 벡터(10주차)

2) 테스트 결과

프로그램 실행 방법 (맵의 길이가 길어서 맵을 한번에 봐야하므로 콘솔창을 최대한 늘리고 pacman.cpp 파일 실행해야함, 그러지 않으면 콘솔창에서 무한 루프하는거 처럼 보임)

(1) 메인 화면 구현

```
===== MAIN MENU =====
1. 게임 시작
2. 점수판 보기
3. 게임 종료
선택: 
```

- 입력이 1일 때, 게임 맵 출력과 게임 시작

[illegible]

- 입력이 2일 때, 점수판 출력

===== 점수판 =====
 1. 점수: 40 (게임을 한 판 진행하고 점수가 40점이었을 때)

```

===== 점수판 =====
No Game Data. Play Game Please
===== MAIN MENU =====
1. 게임 시작
2. 점수판 보기
3. 게임 종료
선택:

```

(게임을 진행하지 않고 점수판을 출력했을 때)

- 입력이 3일 때, 게임 종료

```

===== MAIN MENU =====
1. 게임 시작
2. 점수판 보기
3. 게임 종료
선택: 3
PS C:\CPP2409-P> 

```

- 그 외의 값 입력 시 에러 문구 출력

```

===== MAIN MENU =====
1. 게임 시작
2. 점수판 보기
3. 게임 종료
선택: 6
잘못된 입력입니다. (1~3 입력)
===== MAIN MENU =====
1. 게임 시작
2. 점수판 보기
3. 게임 종료
선택: 

```

(2) 게임 맵 구현, 팩맨과 고스트의 움직임, 게임 실행 화면 구현

```
#####
#.....#.....#
#.#.#.#.#.#.#.#.#.#.#
#*#####.#.#.#.#.#*#
#.#.#.#.#.#.#.#.#.#.#
#.....#
#.#.#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#.#.#
#####
##### P#####
#####.#####
#.#.#.#.#.#.#.#
#.#G#.#.#
#.#G##.#.#.#
#####.#.#.#.#.#
#.#.#.#.#
#####.#.#.#.#
#.#.#.#.#.#.#
#.#.#.#.#.#
#.#.#.#.#.#
#.#.#.#.#.#
#####
#####.#.#.#.#.#
#.#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#.#
#*#####*#
#####
#####.#.#.#.#.#
#####
#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#
#.#.#.#.#.#.#.#.#
#####
```

4. 계획 대비 변경 사항

(11. 17일자 내용)

1) 팩맨 게임 맵 구현

- 원래 계획상 11월 17일까지 팩맨 게임 맵 구현 및 팩맨의 위치에 따른 아이템과의 상호 작용 기능을 완성하려 하였음.

그러나 2차원 배열로 맵을 구현하여 벽, 코인, 아이템, 팩맨, 유령을 표기하는 것은 계획대로 되었으나 팩맨과 유령의 움직임 기능 구현을 다음 진척 보고서 발표인 12월 1일까지로 계획해서 이번 진척 보고서에서는 아이템과의 상호 작용 기능을 구현할 수 없었음.

그에 따라 12월 1일까지 제출인 진척 보고서에서 이번에 구현하지 못했던 기능2) 팩맨 게임 맵 구현의 세부 기능 구현을 마저 진행하려 함.

구현해야 할 세부 기능은 다음과 같음.

-> 팩맨의 유효한 움직임 판단, 팩맨이 코인을 먹을 시 점수 증가, 아이템을 먹을 시 PowerUp 모드 변신, 일정 시간마다 맵에 유령 추가하기, 팩맨과 유령이 동일 좌표에 있을 시 팩맨 사망 처리.

(12.01일자 내용)

1) 메인 화면 구현

- 고스트 클래스 구현이 아직 미흡하여 난이도에 따른 유령의 움직임과 수가 조절되도록 하는 기능 1의 난이도 입력 받는 기능 구현을 아직 못했음. (12/15 중간 보고서 제출때는 가능할 예정)

2) 팩맨 게임 맵 구현

- 아직 PowerUp 모드 구현을 하지 않아 팩맨이 움직이면서 아이템을 먹었을 때 PowerUp 모드가 되도록하는 기능 2의 내용을 완전히 구현하지 못함. (12/15 중간 보고서 제출때는 가능할 예정)

- 기능 2의 유령의 스폰 장소 설정과, 시간이 지날 때 마다 한마리 씩 추가하도록 하는 것은 기능 6 유령의 부활 기능을 구현하면서 같이 할 예정

3) 사용자의 입력에 따른 팩맨의 움직임 구현

- 초기 제안서에서 기능 3 에 팩맨과 고스트의 속도가 같도록 하려했으나, 직접 게임을 진행해보니 속도가 같으면 난이도가 너무 높아져서 팩맨이 2 배 빠르도록 하였음. 대신 PowerUp 모드일 때 속도 증가를 없앨 예정.

4) 유령들의 움직임 알고리즘 구현

- 기능 4 의 고스트 움직임 알고리즘 중 BFS 를 사용하는 A 모드만 구현했고, 나머지는 12/15 중간 보고서 제출 때 완성할 예정

- 게임 종료 기능을 구현하다보니 점수 저장 기능인 기능 7 을 앞당겨서 구현하게 되었음.

5. 프로젝트 일정

(11/17 보고서에서 수정된 진척도가 존재함)

업무		11/3	11/17	12/1	12/15	12/22
제안서 작성		완료				
기능1	세부기능1		-----95% 구현----->			
	세부기능2		완료			
	세부기능3		완료			
기능2	세부기능1		-----90% 구현----->			
기능3	세부기능1			-----90% 구현----->		
기능4	세부기능1			-----30% 구현----->		
	세부기능2			----->		
기능5	세부기능1				----->	
	세부기능2				----->	
	세부기능3				----->	
기능6	세부기능1					----->
기능7	세부기능1			완료		