v

C++프로그래밍및실습

프로젝트 제목

진척 보고서 #01

제출일자:2024-11-17
제출자명:최민규
제출자학번:233935

1. 프로젝트 목표 (16 pt)

1) 배경

프로젝트의 주제로 콘솔 그래픽을 기반으로 한 게임을 만드는 것이 좋겠다고 생각했습니다. 이 주제로 적합한 장르의 게임을 생각하던 중 예전에 플레이했던 게임인 CAPCOM사의 록맨 시리즈가 생각이 나서, 이 게임의 스테이지 중 하나인 에어맨 스테이지를 구현하려 합니다.

2) 프로젝트 목표

액션 플랫포머 게임 록맨 2의 에어맨 스테이지 구현

3) 차별점

콘솔 그래픽으로 액션 플랫포머 게임을 구현

2. 기능 계획

1) 주인공

- 좌, 우, 점프, 공격의 유저 조작

- 점프가 입력되는 시간에 따라 점프 높이 조절 가능

- 체력이 존재하고 체력이 0이하가 되면 게임오버

- 목숨이 2개 존재하며 2개의 목숨이 모두 사라지면 세이브 포인트를 무시하고 처음부터 다시 시작

2) 적

- 스테이지 진행 시 나오는 6개의 적과 스테이지 보스

- 보스를 제외한 모든 적은 처치 시 일정 확률로 소형 및 대형 체력 회복 아이템을 드랍

(1) 플랫폼 형 적

- 근접하면 페이드 인 되며 나타남

- 등장하고 나서 몇 초 후 적의 좌 우에서 번갈아가며 소형 적이 등장

- 이 적의 머리 위를 플랫폼처럼 이용 가능

(2) 소형 적

- 플랫폼 형 적에게서 등장

- 유저를 향해 수직, 수평 방향으로 돌진(일직선으로)

(3) 구름 발판을 타고 있는 적

- 구름 발판 위의 적을 공격해야 타격 가능

- 적을 처치하고 남은 구름 발판을 유저가 이용할 수 있음

- 이 적은 투사체를 포물선으로 던지는 공격을 할 수 있음

(4) 새

- 빠른 속도로 수평으로 이동하며 유저가 아래에 있으면 알을 떨궈서 공격

(5) 회오리

- 유저가 근접하면 유저 쪽으로 세 번 회오리를 던짐

- 회오리는 포물선 형태로 나타나며 유저가 공격해 부술 수 있음

(6) 강풍기

- 등장하면 유저를 강풍기가 바라보는 방향으로 밈

(7) 보스

- 3번 유저의 투사체를 막는 탄환을 발사하고 2번 점프하며 반대편으로 이동

- 탄환을 발사할 때 유저를 보스가 바라보는 방향으로 밈

3) 스테이지와 스테이지 스크롤

- 스테이지에는 정해진 적과 플랫폼이 있으며 유저가 이동하면 그에 맞춰서 스테이지가 스크롤 되고, 그에 맞춰서 적이 등장함.

- 일정 진행도를 만족하면 사망해도 이어서 진행할 수 있는 지점(세이브포인트) 존재

- 스테이지의 끝에는 보스와 맞붙을 수 있는 보스 스테이지로 입장하는 문이 있음

- 주인공이 플랫폼이 없는 구멍으로 빠지게 되면 체력이 얼마나 남아있든 게임 오버

3. 진척사항

1) 기능 구현

클래스 Engine

```h
#include <time.h>
#include <iostream>
#include <Windows.h>
#include "rockman.h"

class Engine
{
    private:
        Controls* buttons;
        Rockman* rockman;
        Map* map;

        HANDLE CIN = 0;
        HANDLE COUT = 0;

        bool Key_input();
    public:
        double FPS;// = 30.0;

        bool Render();
        void Shutdown();
        void E_Init();
        void gotoxy(int x, int y);

        //void renderRoom1();
        //void renderRoom2();
        //void renderRoom3();
        //void renderBossRoom();

        void renderTestRoom(Map* map);
        void updateScreenOffset(Map* map, int xPos);

        void renderRockman(Rockman* rockman, Map* map);
};
```

(1) 일정한 주사율로 출력되는 화면

- 입력: 없음, 출력: true

- 설명: 코드의 시작 시간과 끝나는 시간의 차를 계산해 일정한 딜레이를 주어 원하는 속도의 주사율로 출력할 수 있게 한다.

- 적용된 배운 내용 (반복문, 조건문, 클래스, 함수)

- 코드 스크린샷

```cpp
bool Engine::Render()
{
    static int Frames = 0;
    bool Return  = true;
    while(Return){
        time_t start, end;
        start = clock();
        end = clock();
        while((double)(end - start) < (1000.0 / FPS)){
            if(!Key_input()){
                Return = false;
            }
            end = clock();
        }

        rockman->R_Update(buttons, map);

        updateScreenOffset(map, rockman->getXPos());
        renderTestRoom(map);

        renderRockman(rockman, map);

        gotoxy(15, 18);
        //printf("xPos: %d, yPos: %d, xVel: %f, yVel: %f, prevXpos: %d, sc

        end = clock();
        std::cout << "FPS: " << Frames << " : " << end - start << "ms" ;
        Frames++;
    }

    return true;
}
```

(2) 키 입력

- 입력: 없음, 출력: true

- 설명: 키들의 입력을 받아 구조체 Controls(함수 내 변수명으로는 buttons)의 flag를 변경한다.

- 적용된 배운 내용 (함수, 포인터)

- 코드 스크린샷

```
bool Engine::Key_input()
{
    switchControls('W', buttons->upDown);
    switchControls('S', buttons->downDown);
    switchControls('A', buttons->leftDown);
    switchControls('D', buttons->rightDown);

    switchControls('J', buttons->actionPressed);
    switchControls('K', buttons->jumpPressed);

    return true;
}
```

```
void switchControls(int VirtualKey, bool &key){
    if (GetAsyncKeyState(VirtualKey) < 0 && key == false)
    {
        key = true;
    }
    if (GetAsyncKeyState(VirtualKey) == 0 && key == true)
    {
        key = false;
    }
}
```

(3) 스테이지 출력 및 화면 스크롤링

- 입력: 클래스 Map 포인터, xPos, 출력: 없음

- 설명: 클래스 Map과 록맨의 x좌표 xPos를 받아 현재 록맨의 좌표에 맞는 화면을 출력한다. (현재는 테스트를 하기 위한 테스트 맵으로 출력하도록 함) 맵의 끝에서는 화면 스크롤링이 되지 않는다.

- 적용된 배운 내용 (함수, 포인터, 클래스, 조건문, 반복문)

- 코드 스크린샷

```
void Engine::renderTestRoom(Map* map) {
    for (int y = 0; y < map->ScreenHeight; ++y) {
        for (int x = 0; x < map->ScreenWidth; ++x) {
            int mapX = x + map->screenOffsetX;
            if (mapX < map->ScreenWidth * map->test_room_numberofScreen) {
                gotoxy(x, y);  // 커서 위치 설정
                switch (map->test_room[y][mapX]) {
                    case 0: printf(" "); break;
                    case 1: printf("="); break;
                }
            }
        }
    }
}

void Engine::updateScreenOffset(Map* map, int xPos) {
    int dx = xPos - rockman->previousXpos;
    int mapWidth = map->ScreenWidth * map->test_room_numberofScreen;

    if (xPos < map->ScreenWidth / 2) {
        map->screenOffsetX = 0;
    } else if (xPos > mapWidth - map->ScreenWidth / 2) {
        map->screenOffsetX = mapWidth - map->ScreenWidth;
    } else {
        map->screenOffsetX += dx;
    }

    rockman->previousXpos = xPos;
}
```

(4) 록맨 출력

- 입력: 클래스 Rockman 포인터, 클래스 Map포인터 출력: 없음

- 설명: 콘솔 화면에 록맨을 그려준다. 맵의 끝에서는 화면 스크롤링이 되지 않으므로, 현재 좌표에 맞는 위치에 록맨을 출력한다.

- 적용된 배운 내용 (함수, 포인터, 클래스, 조건문)

- 코드 스크린샷

```
void Engine::renderRockman(Rockman* rockman, Map* map) {
    int mapWidth = map->ScreenWidth * map->test_room_numberofScreen;
    int renderX = rockman->getXPos() - map->screenOffsetX;

    if (rockman->getXPos() <= map->ScreenWidth / 2) {
        renderX = rockman->getXPos();
    } else if(rockman->getXPos() >= mapWidth - map->ScreenWidth / 2){
        renderX = rockman->getXPos() - (mapWidth - map->ScreenWidth);
    }

    gotoxy(renderX, rockman->getYPos());
    if (rockman->getfacingRight()) {
        printf("[\\");
    } else {
        printf("/]");
    }
}
```

클래스 Rockman

```h
#include <iostream>
#include <math.h>
#include <algorithm>
#include "map.h"
#include "controls.h"

class Rockman {
private:
    int xPos;
    int yPos;

    float xVel, yVel;
    float const acc = 0.5f;
    float const xMaxVel = 1.0f, yMaxVel = 1.0f;
    float const gravity = 0.5f;
    float const jumpforce = 1.0f;

    bool onGround;
    bool facingRight;
    bool isInvincible;
public:
    int previousXpos;

    void R_Init();
    void R_Update(Controls* control, Map* map);

    bool XCollision(Map* map, Controls* control);
    bool YCollision(Map* map, Controls* control);

    void setXPos(int x);
    void setYPos(int y);

    int getXPos();
    int getYPos();

    float getXVel();
    float getYVel();

    bool getfacingRight();
};
```

(5) 록맨 상태 업데이트

- 입력: 구조체 Controls 포인터, 클래스 Map 포인터 출력: 없음

- 설명: 키 입력 flag를 모아둔 구조체 Controls를 이용해 키 입력에 따른 록맨의 움직임

과 x, y 충돌을 처리한다.

- 적용된 배운 내용 (함수, 포인터, 클래스, 조건문)

- 코드 스크린샷

```cpp
void Rockman::R_Update(Controls* control, Map* map){

    if(xVel != 0 && !control->leftDown && !control->rightDown){
        if (xVel > 0){
            xVel = std::max<float>(xVel - acc, 0);
        } else {
            xVel = std::min<float>(xVel + acc, 0);
        }
    } else {
        if(control->leftDown && xVel > -xMaxVel){
            xVel = std::max<float>(xVel - acc, -xMaxVel);
            facingRight = false;
        }

        if(control->rightDown && xVel < xMaxVel) {
            xVel = std::min<float>(xVel + acc, xMaxVel);
            facingRight = true;
        }
    }

    xPos += xVel;

    if(xVel != 0 && XCollision(map, control)){ // 충돌 시 xVel = 0 설정
        xVel = 0;
    }

    if(control->jumpPressed && onGround){
        yVel -= jumpforce;
    }else if (yVel < yMaxVel){
        yVel = std::min<float>(yVel + gravity, yMaxVel);
    }

    if(!control->jumpPressed && yVel < 0){
        yVel = 0;
    }

    yPos += yVel;

    if(yVel > 0) {
        onGround = YCollision(map, control);
        if(onGround) {
            yVel = 0;
        }
    }else if (yVel < 0) {
        if(YCollision(map, control)){
            yVel = 0;
```

(6) x, y 충돌 확인

- 입력: 구조체 Controls 포인터, 클래스 Map 포인터 출력: true or false

- 설명: 각 맵에서의 x, y 충돌을 키 입력 flag와 록맨의 x, y 좌표를 이용해 처리한다. 충동하면 true, 그렇지 않으면 false를 반환한다.

- 적용된 배운 내용 (함수, 포인터, 클래스, 조건문)

- 코드 스크린샷

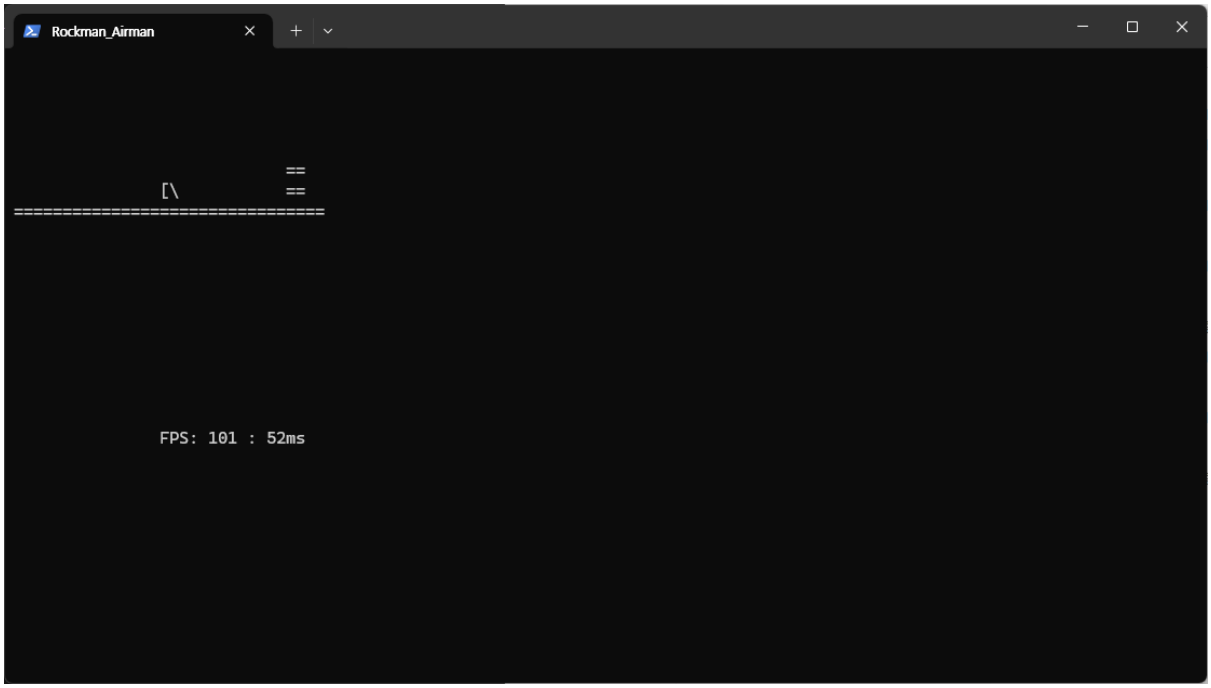
```
bool Rockman::XCollision(Map* map, Controls* control){
    if (!map || !control) { return false;} // 유효하지 않으면 충돌로 간주하지 않음

    if(map->In_testroom){
        if(  control->leftDown
        && ((map->test_room[yPos][xPos - 1] == 1)
        ||   xPos - 1 <= 0)){
            std::cout << "Left Wall Collision Detected\n";
            return true;
        }
        else if(control->rightDown
            &&((map->test_room[yPos][xPos + 2] == 1)
            ||  (xPos + 2) >= (map->ScreenWidth * map->test_room_numberofScreen))){
            std::cout << "Right Wall Collision Detected\n";
            return true;
        }
    }

    return false;
}

bool Rockman::YCollision(Map* map, Controls* control){
    if(map->In_testroom){
        if((map->test_room[yPos + 1][xPos] == 1) || map->test_room[yPos + 1][xPos + 1] == 1){
            return true;
        }else if((map->test_room[yPos - 2][xPos] == 1) || map->test_room[yPos - 2][xPos + 1] == 1){
            return true;
        }
    }

    return false;
}
```

2) 테스트 결과

(1) 화면 출력 및 일정한 주사율

- 일정한 주사율로 화면 출력

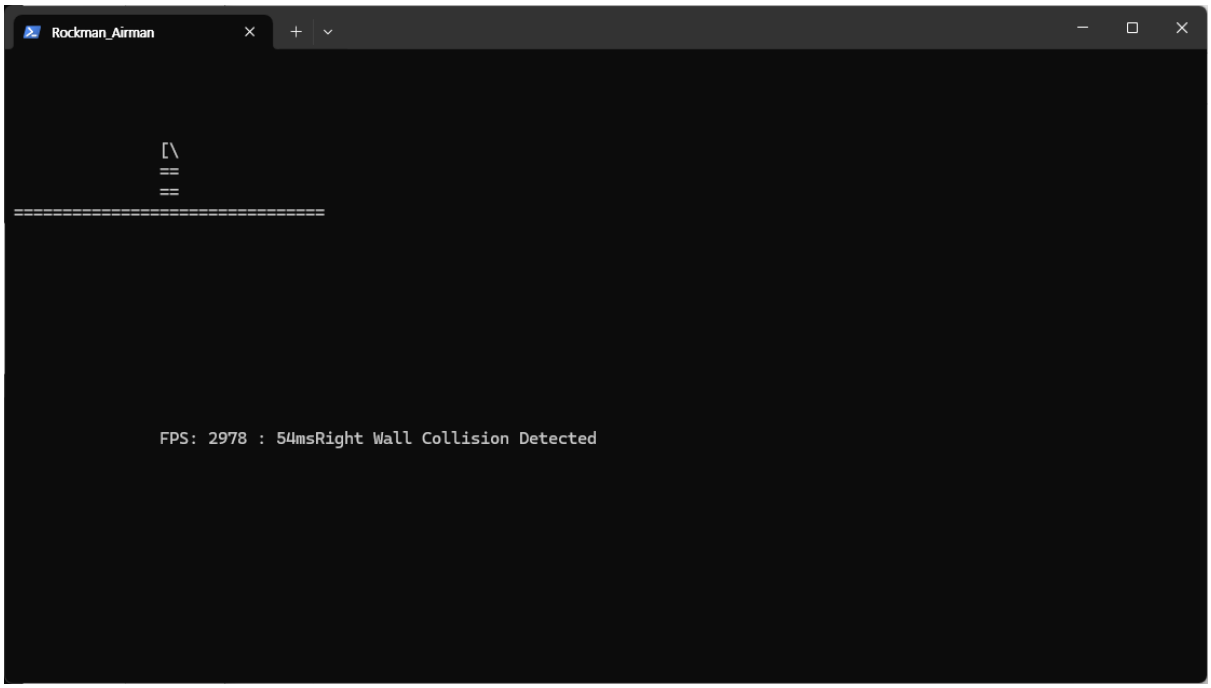
- 테스트 결과 스크린샷



(2) 키 입력 및 록맨의 이동

- 키 입력을 받으면 록맨이 이동한다.

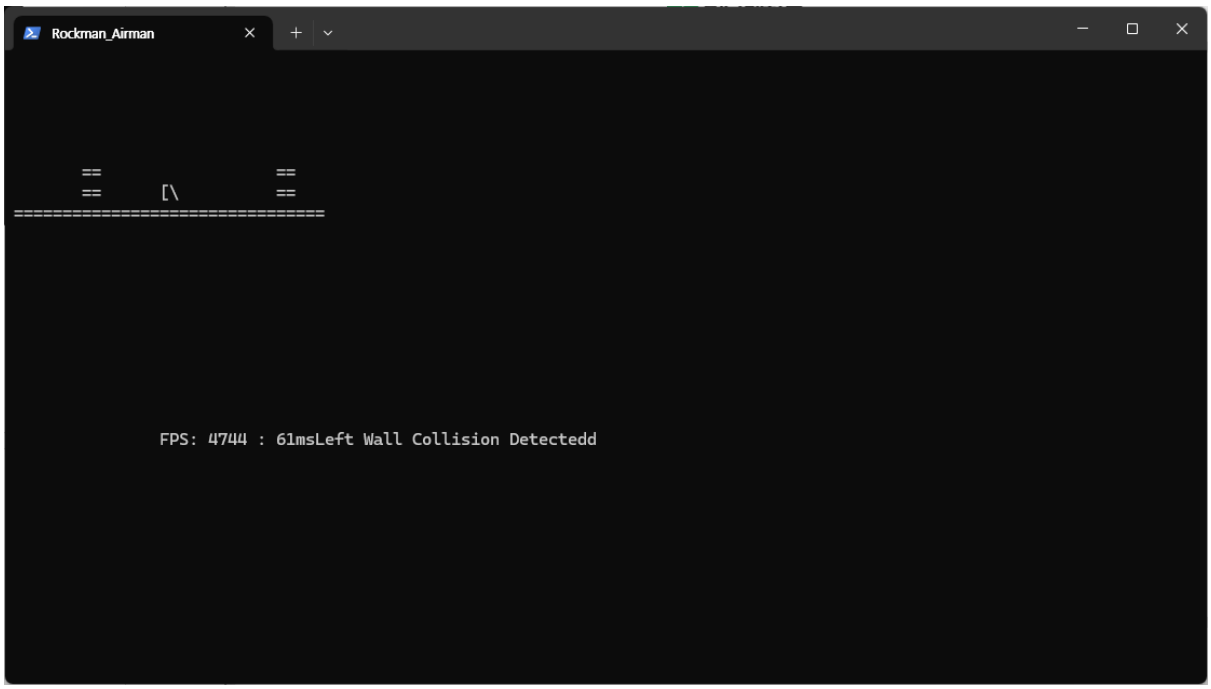
- 테스트 결과 스크린샷

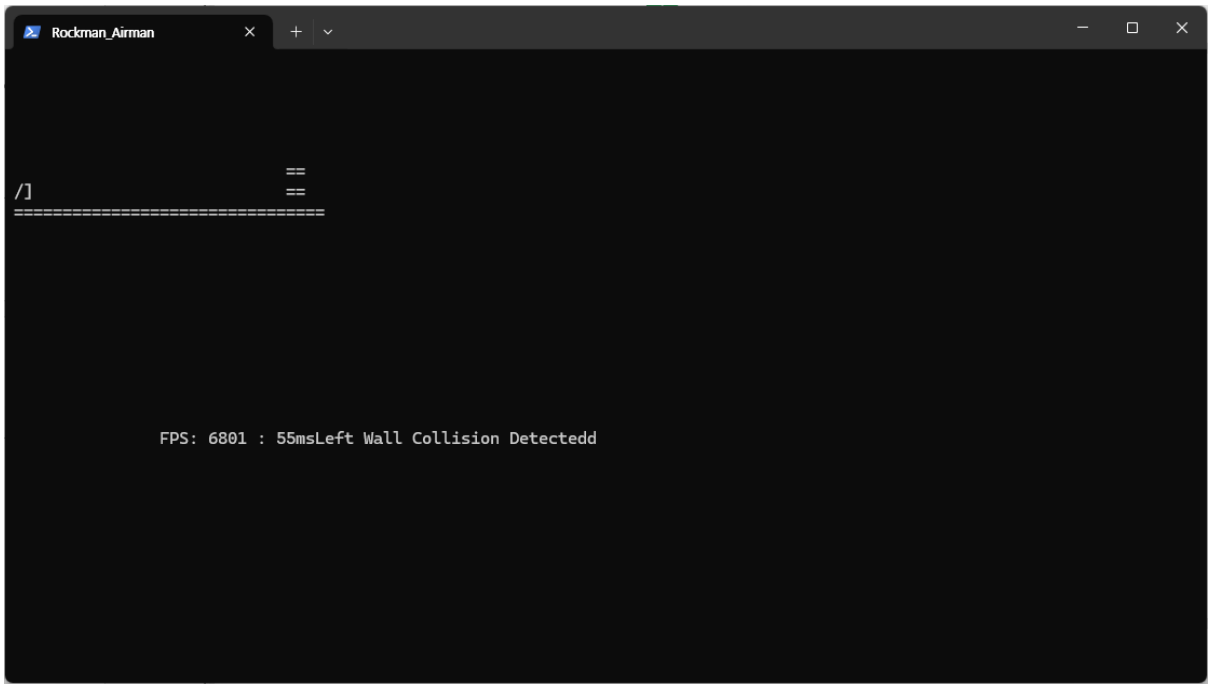


(3) 스크린 스크롤링

- 록맨의 위치에 따라서 화면이 움직인다.

- 테스트 결과 스크린샷

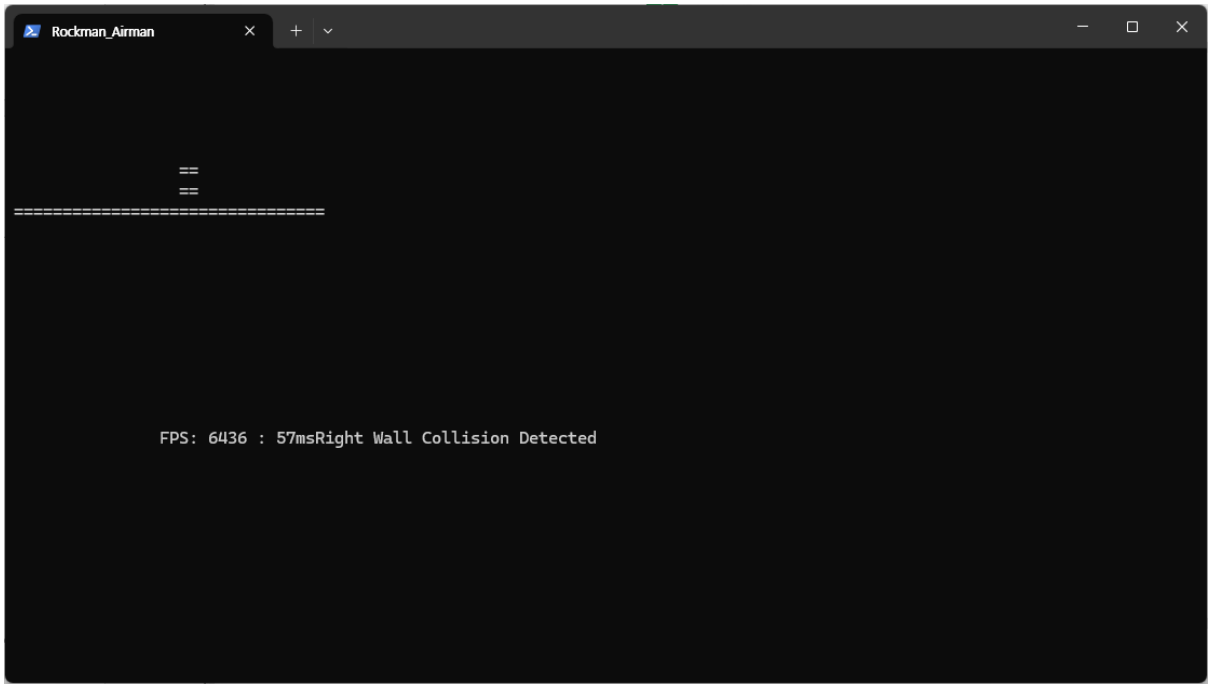




(4) 충돌 확인

- 충돌 확인

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

1) 일정 조정

- 스테이지의 완벽한 구현은 프로젝트 후반부로 옮겨야 할 것 같음

- 사유: 해당 스테이지가 적을 먼저 구현하지 않으면 진행할 수 없어서 적들을 먼저 구현한 후 스테이지를 구현해야 할 것 같음

- 록맨의 조작과 물리 조정을 조금 더 해야 할 것 같음

- 사유: 화면 주사율과 내가 원하는 키 입력 정도를 구현하는 것에 시간을 너무 써서 액션 기능과 물리 구현을 끝내지 못했다.

5. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/10	11/17	11/24
제안서 작성		완료			
주인공			진행 중		
스테이지				진행중	
중간 보고서 1				완료	