**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | C++ Snakegame with ncurses |
| 팀 명 | 5조5억만 |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.2 |
| **Date** | 2020-06-15 |

|  |  |
| --- | --- |
| **팀원** | 윤서영 (팀장) |
| 이한정 |
| 조규현 |
| 황영서 |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Snake Project”를 수행하는 팀 “5조5억만”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “5조5억만”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서 – C++ Snakegame with ncurses.doc |
| **원안작성자** | 윤서영, 이한정, 조규현, 황영서 |
| **수정작업자** | 윤서영, 이한정, 조규현, 황영서 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2020-06-15 | 황영서 | 1.0 | 최초 작성 |  |
| 2020-06-16 | 윤서영 | 1.1 | 수정 |  |
| 2020-06-16 | 이한정 | 1.2 | 수정 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_Toc43103653)

[2 개발 내용 및 결과물 5](#_Toc43103654)

[2.1 목표 5](#_Toc43103655)

[2.2 개발 내용 및 결과물 6](#_Toc43103656)

[2.2.1 개발 내용 6](#_Toc43103657)

[2.2.2 시스템 구조 및 설계도 6](#_Toc43103658)

[2.2.3 활용/개발된 기술 6](#_Toc43103659)

[2.2.4 현실적 제한 요소 및 그 해결 방안 6](#_Toc43103660)

[2.2.5 결과물 목록 7](#_Toc43103661)

[3 자기평가 8](#_Toc43103662)

[4 참고 문헌 8](#_Toc43103663)

[5 부록 8](#_Toc43103664)

[5.1 사용자 매뉴얼 8](#_Toc43103665)

[5.2 설치 방법 8](#_Toc43103666)

# 개요

|  |
| --- |
| **평가기준 (10점)**  **프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.**  **또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.** |

## 프로젝트 소개

**국민대학교 C++프로그래밍 교과목에서 C++언어를 사용하고, ncurses 라이브러리를 활용하여 제작한 Snake Game Project입니다.**

## 개발 방법

* 개발OS : Linux Ubuntu 16.04 LTS 기준
* 언어: C++ 14 Version

## 사용 라이브러리

* ncursesw 라이브러리(Unicode지원)

## 설치 방법(Linux기준)

* sudo apt-get update
* sudo apt-get install libcurses5-dev libcursesw5-dev

# 개발 내용 및 결과물

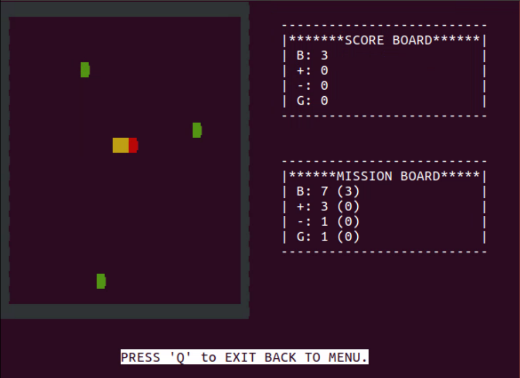
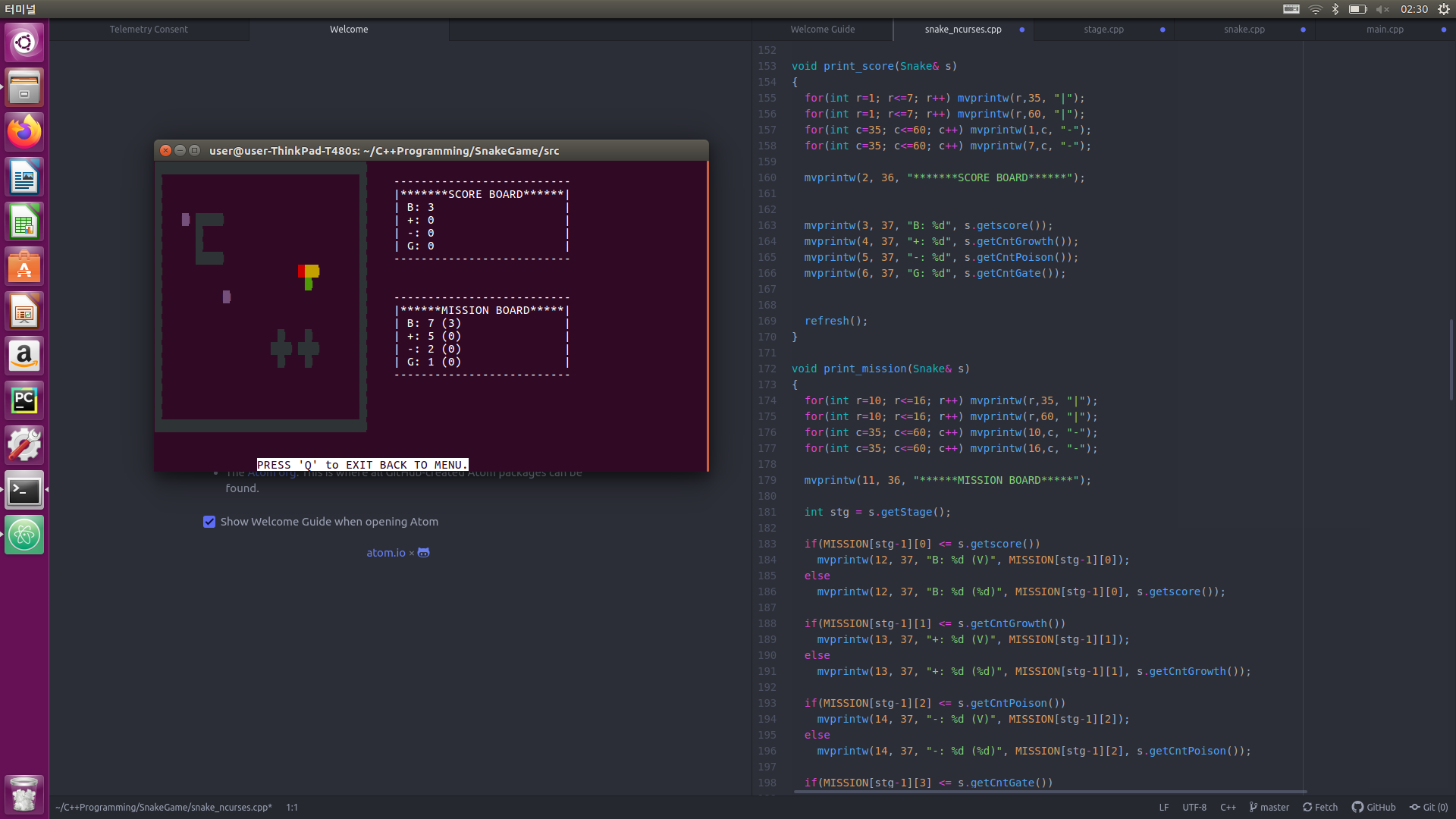
## 목표

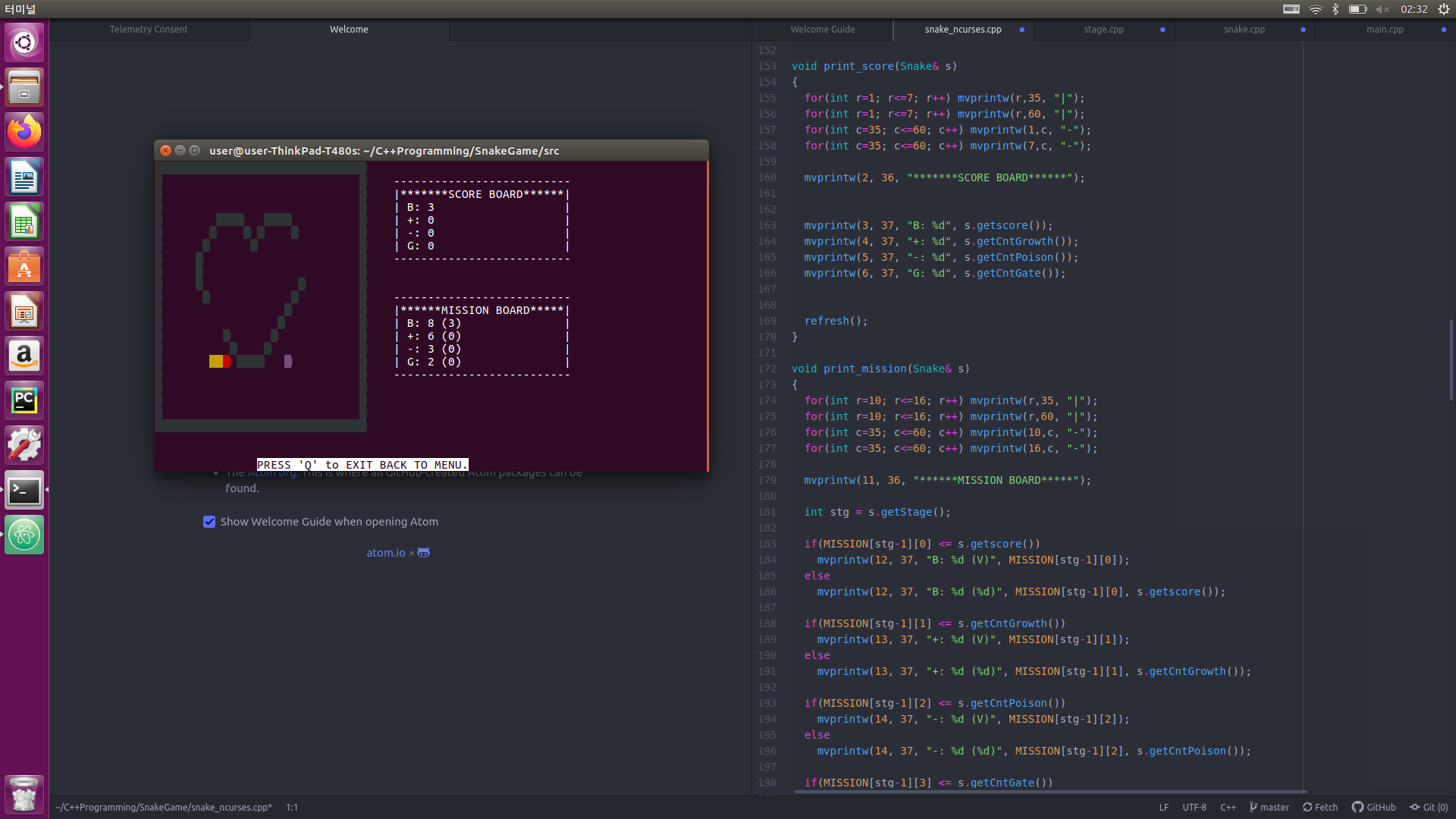
|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.** |

|  |  |  |
| --- | --- | --- |
| 적용단계 | 내용 | 적용 여부 |
| 1단계 | Map의 구현 | 적용 |
| 2단계 | Snake 표현 및 조작 | 적용 |
| 3단계 | Item 요소의 구현 | 적용 |
| 4단계 | Gate 요소의 구현 | 적용 |
| 5단계 | 점수 요소의 구현 | 적용 |

**1. Map**

각 스테이지에 따른 Map을 구현한다.

 컴퓨터이(가) 표시된 사진

자동 생성된 설명

**2. Snake**

**·** 초기 길이: 3

**·** 초기 방향: 오른쪽

**·** 머리의 색은 빨강, 몸통의 색은 노랑색이다.

**·** 사용자가 방향키를 입력하게 되면, Snake의 좌표는 1초마다 진행방향 쪽으로

변하게 된다.

**·** 몸의 길이가 3미만으로 내려가게 되면 게임이 종료된다.

**·** 머리가 벽에 닿거나, 몸통에 닿게 되면 게임이 종료된다.

**·** 머리가 아이템의 좌표와 같아질 때 아이템을 얻는다.

**3. Item**

아이템은 5초마다 랜덤하게 나타난다.

**·** 아이템은 Snake와 Wall과 겹치지 않도록 생성한다.

**·** 최소 1개부터 최대 3개까지의 아이템이 랜덤하게 생성된다.

**·** 아이템은 Growth와 Poison중 랜덤하게 생성된다.

- Growth: 몸 길이 1증가

- Poison: 몸 길이 1 감소

**·** 아이템은 획득 시 Map에서 사라진다.

**4. Gate**

**·** Gate는 기본적으로 Snake 길이가 6이상일 때, 출현하게 된다.

**·** 10초마다 Gate를 새로 출현하게 되는데, Snake가 통과 중일 때는 생성하지

않는다.

**·** Gate는 IMMUNE WALL에 출현하지 않는다.

**5. Score**

Score Board에서는 현재까지 Snake가 획득한 점수들을 출력한다.

**·** B: 현재 Snake의 길이

**·** +: 현재까지 획득한 Growth 아이템 수

**·** -: 현재까지 획득한 Poison 아이템 수

**·** G: 현재까지 통과한 Gate 수

## 개발 내용 및 결과물

### 개발 내용

|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.** |

**1. Map**

**·** Map 크기: 21 X 31 (ROW, COL)

**·** Map은 Stage마다 다르게 구성되어있다.

**·** Map은 3차원의 구조로 되어있다. [STAGE][ROW][COL]

**·** 각 Cell은 WALL/IMMUNEWALL 에 해당하는 값이다.

- MAP[STAGE][ROW][COL] == 0: WALL

- MAP[STAGE][ROW][COL] == 2: IMMUNEWALL

**2. Snake**

**2.1 Snake Class**

**·** Snake Class에서는 Snake 생성, 아이템 생성, 초기 Map구성, Gate출현의 기능을

수행하는 메소드가 존재한다.

**2.2 Snake 생성**

**·** Snake클래스의 생성자를 통해 Snake를 생성한다. 인자 값으로는 stage의 값을 넘겨

받게 된다.

**·** Snake는 deque 구조로 생성되며 배열의 원소는 Map의 각 좌표들이다.

**2.3 방향키 입력**

**·** 사용자가 방향키를 입력하게 되면 Snake가 방향키에 따라 움직인다.

**·** 다음과 같은 상황에는 방향키 입력이 무시된다.

**-** 현재 진행방향과 같은 방향

**-** 현재 진행방향과 반대 방향

**2.4 Snake의 움직임 표현**

**·** 방향키 입력에 따른 좌표 변화

| **방향키** | **좌표변화** |
| --- | --- |
| 상 | 행이 1만큼 감소 |
| 하 | 행이 1만큼 증가 |
| 좌 | 열이 1만큼 감소 |
| 우 | 열이 1만큼 증가 |

**·** 구현방법

**-** 변화된 좌표를 snake(deque) front에 push한다.

**-** pop하여 맨 뒤 원소를 삭제한다.

**2.5 화면 출력**

**·** Snake(deque)의 위치를 표현하기 위해 화면을 매초 refresh하여 출력해준다.

**3. Item**

**3.1 Item 출현**

**·** Item 출현 시 종류를 랜덤하게 한다.

**·** function rand\_score(): 1 또는 -1을 리턴한다.

**-** 1: Growth 아이템

**-** -1: Poison 아이템

**3.2 Item 획득**

**·** Snake Head와 Item이 부딪히게 됐을 경우 아이템을 획득한다.

**·** Snake Head: cells(snake의 deque).front()

**·** Head와 Item의 좌표가 같아질 경우, 획득한 아이템에 대한 정보들을 저장한다.

**-** 현재 아이템의 Iterator(삭제를 위함)

**-** 현재 아이템의 좌표

**-** 아이템 flag(Growth or Poison)

**·** Item 획득 시, 해당 Item은 화면에서 사라진다.

**-** items(MAP에 나타나는 아이템 배열).erase(curitem.t);

**·** Item 획득 시, 현재까지 획득한 Item 개수를 저장하는 변수를 1 증가시킨다.

**-** Score Board에서 Growth, Poison을 표시하기 위함.

**4. Gate**

**4.1 Gate 출현**

**·** Snake의 길이가 6 이상일 때 출현한다.

**·** 10초마다 새로운 Gate를 생성한다.

**·** Snake가 Gate를 통과 중인지 체크하고 통과 중이면 생성하지 않는다.

**·** Wall중에(Not immune) Gate 한 쌍을 랜덤하게 출현시킨다.

**-** Snake/Item의 좌표와 겹치지 않게 생성한다.

**-** Gate는 한 쌍이므로, 총 두개의 Gate를 생성하여 deque에 저장한다.

**4.2 Gate 통과**

**·** Gate를 통과 중인지 체크하고 통과 중이라면 flag값을 true로 설정한다.

**·** 매번 출구 좌표에 Gate의 양쪽 값을 대입하다가 Gate 통과 flag가 true일 시

해당 게이트 좌표의 대입을 건너뛴다

=> 출구 좌표는 통과 중이지 않은 좌표로 설정된다.

**·** Gate flag가 true일 경우

**-** 출구 Gate의 좌표에 따라 이동방향을 설정해준다.

| **Gate(out)** | **Direction** |
| --- | --- |
| Row==0 | ↓ |
| Row==end | ↑ |
| Col==0 | → |
| Col==end | ← |

- Gate통과 횟수를 저장하는 변수를 1 증가시킨다.(Score Board)

**5. Score & Mission**

**5.1 Score Board**

Score Board에서는 현재까지 Snake가 획득한 점수들을 출력한다.

- Snake.getScore(): Snake의 현재 길이

- Snake.getCntGrowth(): 현재까지 획득한 Growth Item 수

- Snake.getCntPoision(): 현재까지 획득한 Poison Item 수

- Snake.getCntGate(): 현재까지 통과한 Gate 수

**5.2 Mission Board & Stage**

**·** Mission Board에서는 각 Stage마다 수행해야 할 Mission들이 출력된다.

Ex) 3단계

앉아있는, 컴퓨터, 노트북이(가) 표시된 사진

자동 생성된 설명

**·** 각 Mission을 완수할 경우 괄호 안에 ‘V’자를 표시한다.

컴퓨터이(가) 표시된 사진

자동 생성된 설명

### 시스템 구조 및 설계도

|  |
| --- |
| **작성요령 (30점)**  **프로젝트의 각 세부 목표의 주요 기능(알고리즘 등)에 대해서 기술한다. 세부 목표별로 수정한 프로그램 소스 파일을 나열하고, 해당 파일에서 세부 목표를 달성하기 위해 작성한 클래스/함수에 대해 나열하고, 각 요소에 대해 간략한 설명을 작성한다. 또한 각 요소의 개발자를 명시한다.** |

|  |  |  |
| --- | --- | --- |
| 적용단계 | 내용 | 소스 파일 |
| 1단계 | Map의 구현 | stage.cpp / snake.cpp |
| 2단계 | Snake 표현 및 조작 | snake.cpp / snake\_ncurses.cpp |
| 3단계 | Item 요소의 구현 | snake.cpp / snake\_ncurses.cpp |
| 4단계 | Gate 요소의 구현 | stage.cpp / snake.cpp |
| 5단계 | 점수 요소의 구현 | snake.cpp / snake\_ncurses.cpp |

**1. Map** (개발자: 이한정)

**1.1 setStageWall()**

: 초기 Map을 구성하는 함수

// stage.cpp

std::deque<Cell> setStageWall(int stage)

{

std::deque<Cell> tmpWall;

for(int i=0; i<MAX\_ROW; i++){

Cell temp;

temp.p.row = i;

for(int j=0; j<MAX\_COL; j++){

temp.p.col = j;

if(MAP[stage-1][i][j] == 2){

temp.type = IMMUNEWALL;

tmpWall.push\_back(temp);

}else if(MAP[stage-1][i][j] == 0){

temp.type = WALL;

tmpWall.push\_back(temp);

}

}

}

return tmpWall;

}

* 임시변수 tmpWall을 deque 자료형으로 선언함.
* 미리 저장되어있는 MAP 배열의 값을 확인하여 ImmuneWall/Wall인지를 판단한다.
  + MAP[STAGE][ROW][COL]==0 : WALL
  + MAP[STAGE][ROW][COL]==2 : IMMUNEWALL

**1.2 Snake::setWall(int stage)**

// snake.cpp

void Snake::setWall(int stage)

{

walls = setStageWall(stage);

}

: Snake 클래스의 메소드이며, Snake가 생성됨과 동시에 Map을 생성하도록 해준다. 그리고 setStageWall 메소드를 호출할 때, stage 인자값을 넘겨준다.

**2. Snake** (개발자: 윤서영)

**2.1 Snake Class**

: Snake 클래스에서는 Snake생성, 아이템 생성, 초기 Map구성, Gate출현의 기능을 수행하는 메소드가 존재한다.

// snake.hpp

class Snake{

protected:

int len;

int stage;

time\_t item\_starttime; // 아이템의 생성시간을 담는 변수

time\_t gate\_starttime;

std::deque<Cell> cells; // Snake head,body 좌표를 저장하는 queue

std::deque<Item> items; // Item의 좌표를 저장하는 queue

std::deque<Cell> walls; // WALL,IMMUNEWALL,GATE 정보들

std::deque<Cell> gates;

CurrentItem curitem; // 먹은 아이템에 대한 정보(좌표, 종류) 저장

int fdir; // Snake의 현재진행방향

int score, coll; // score=Snake의 길이, coll=충돌했는지에 대한 flag

int poison; // 현재까지 획득한 Poison아이템수

int growth; // 현재까지 획득한 Growth아이템수

int gate; // 현재까지 통과한 Gate의 수

public:

Snake(int s=1);

void createNode(int d);

void makeItem(); // 아아템 생성(랜덤, 최대3개)

void eatItem(); // 아이템을 먹었을 때의 변화

void movesnake(); // snake가 움직였을때 row,col를 변화시키는 함수

void render(); // 변화된 rol,col를 화면에 출력

int collide(); // Snake의 머리가 몸통에 부딪히면 충돌판단

void setWall(int stage); // 초기 벽 설정(WALL,IMMUNEWALL)

int wallcollid(); // 벽 충돌여부 확인 함수

void makeGate(); // Snake길이가 일정 이상일때 Gate 출현

int isWall(Point p); // 해당 좌표가 Wall인지 Check하는 함수

int getscore(); // Score는 Snake의 길이

int getcoll();

int getfdir();

void setfdir(int);

int getCntGrowth();

int getCntPoison();

int getCntGate();

int getStage();

};

**2.2 Snake 생성(Snake 생성자)**

: Snake클래스의 생성자를 통해 Snake를 생성한다. 인자값으로는 stage의 값을 넘겨받게 된다

// snake.cpp

Snake::Snake(int s) : len(DEF\_LEN\_CLASSIC), stage(s)

{

int i;

Cell temp;

// Score Board를 위한 초기화

growth = 0;

poison = 0;

gate = 0;

// Wall 설정

setWall(stage);

// time초기화

item\_starttime = 0;

gate\_starttime = 0;

fdir=RIGHT;

temp.p=rand\_point(cells, items, walls);

// Snake길이때문에 가끔 화면을 넘어가는 경우가 발생해서 예외조건설정.

Point tmpP = temp.p;

tmpP.col += DEF\_LEN\_CLASSIC;

if((temp.p.col+DEF\_LEN\_CLASSIC) >= MAX\_ROW-1){

temp.p.col -= (MAX\_ROW-DEF\_LEN\_CLASSIC);

}else if(isWall(tmpP)){

temp.p.col -= MAX\_ROW-DEF\_LEN\_CLASSIC;

}

for(i=0; i<len-1; i++) {

temp.p.col+=i;

if(temp.p.col+i >= MAX\_COL-1){

temp.p.col=MAX\_COL-i-1;

}

cells.push\_front(temp);

}

temp.p.col+=i;

cells.push\_front(temp);

// 아이템생성(랜덤)

makeItem();

coll=0;

}

* 초기 길이가 3인 Snake를 생성한다.
* Score Board에 표시할 점수 변수들을 초기화한다. (growth/poison/gate)
* item 획득한 시간을 초기화한다.
* 기본 이동방향을 오른쪽(fdir=RIGHT)로 설정한다.
* 초기 Map과 Item을 할당해준다. (좌표는 random하되, 겹치지 않도록 한다.)

**2.3 방향키 입력(게임 진행시)**

: 사용자가 방향키를 입려하게 되면, Snake가 움직이게 된다.

// snake\_ncurses.cpp

while(1) {

ch=getch();

if(ch=='q') {

return S.getscore();

}

else if(ch==KEY\_UP || ch==KEY\_DOWN || ch==KEY\_RIGHT || ch==KEY\_LEFT) {

switch(ch) {

case KEY\_UP:

d=UP;

break;

case KEY\_DOWN:

d=DOWN;

break;

case KEY\_RIGHT:

d=RIGHT;

break;

case KEY\_LEFT:

d=LEFT;

break;

}

if(diff(S.getfdir(), d)!=2 && d!=S.getfdir()) {

S.createNode(d); // 진행방향 설정

}

}

S.movesnake();

S.render();

print\_score(S);

print\_mission(S);

// 벽에 부딪히거나 자기몸에 부딪히면 GameOver

if(S.getcoll()) {

timeout(TIMEOUT\_LONG);

getch();

show\_gameover(S.getscore());

return S.getscore();

}

// 길이가 3미만이면 GameOver

if(S.getscore()<DEF\_LEN\_CLASSIC) {

timeout(TIMEOUT\_LONG);

getch();

show\_gameover(S.getscore());

return S.getscore();

}

if(missionClear(stage, S.getscore(), S.getCntGrowth(), S.getCntPoison(), S.getCntGate())){

if(stage==4) break;

print\_nextStage(stage);

stage++;

S = Snake(stage);

continue;

}

}

* 다음과 같은 상황에는 방향키 입력이 무시된다.
  + 현재 진행방향과 같은 방향
  + 현재 진행방향과 반대 방향
* 게임 진행 중, 'q' 입력 시, 메인 화면으로 돌아간다.
* 벽에 부딪히거나 자기몸에 부딪히면 게임 종료
* Snake의 몸의 길이가 3미만이면 게임 종료
* 해당 Stage에 대한 Mission을 Clear하면, 다음 Stage로 넘어가게 된다.

**2.4 Snake::moveSnake()**

: 사용자가 방향키를 입력하게 되면, Snake의 좌표는 1초마다 진행방향쪽으로 변하게 된다.

// snake.cpp

// Snake::movesnake()

Point f, b;

Cell temp;

f=cells.front().p;

b=cells.back().p;

switch(fdir) {

case UP:

f.row-=1;

break;

case DOWN:

f.row+=1;

break;

case RIGHT:

f.col+=1;

break;

case LEFT:

f.col-=1;

break;

}

temp.p=f;

// 충돌하면 게임 종료

if(wallcollid() || collide()){

coll=1;

return;

}

cells.push\_front(temp);

cells.pop\_back();

* Snake는 기본적으로 “deque” 라는 STL 라이브러리를 사용하여 구현하였다.
  + 변화된 좌표를 push\_front 메소드를 통해 삽입한다.
  + 특별한 이벤트가 발생하지 않았을 경우, pop\_back 메소드를 호출하여 맨 뒤 원소를 삭제한다.
* Snake가 Wall에 부딪히거나, 자기 몸에 부딪혔으면 return하여 게임을 종료시킨다.

**2.5 Snake::render()**

: Snake가 움직임을 표현하기위해 매번 refresh하여 출력해준다.

// snake.cpp

void Snake::render()

{

erase();

for(std::deque<Cell>::iterator it=cells.begin(); it!=cells.end(); ++it){

if(it==cells.begin()){

attron(COLOR\_PAIR(COLOR\_SNAKE\_HEAD));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_SNAKE\_HEAD));

}else{

attron(COLOR\_PAIR(COLOR\_SNAKE\_BODY));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_SNAKE\_BODY));

}

}

// item

for(std::deque<Item>::iterator it=items.begin(); it!=items.end(); ++it){

if(it->points == 1){

attron(COLOR\_PAIR(COLOR\_ITEM\_GROWTH));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_ITEM\_GROWTH));

}else if(it->points == -1){

attron(COLOR\_PAIR(COLOR\_ITEM\_POISON));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_ITEM\_POISON));

}

}

// Wall

for(std::deque<Cell>::iterator it=walls.begin(); it!=walls.end(); ++it){

if(it->type == IMMUNEWALL){

attron(COLOR\_PAIR(COLOR\_IMMUNEWALL));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_IMMUNEWALL));

}else if(it->type == WALL){

attron(COLOR\_PAIR(COLOR\_WALL));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_WALL));

}

}

// Gate

for(std::deque<Cell>::iterator it=gates.begin(); it!=gates.end(); ++it){

attron(COLOR\_PAIR(COLOR\_GATE));

mvprintw(it->p.row, it->p.col, "\u2B1B");

attroff(COLOR\_PAIR(COLOR\_GATE));

}

refresh();

}

**3. Item** (개발자: 황영서)

**3.1 Snake::makeItem()**

: 아이템은 5초가 지나면 랜덤하게 나타나게 된다.

// snake.cpp

void Snake::makeItem()

{

item\_starttime = time(NULL);

srand(time(NULL));

items.clear();

Item tempItem;

int cnt = (rand()%MAX\_ITEM)+1;

for(int i=0; i<cnt; i++){

tempItem.p = rand\_point(cells, items, walls);

tempItem.points = rand\_score();

items.push\_back(tempItem);

}

curitem.itemFlag = 0;

}

* 아이템의 좌표는 Snake와 Wall과 겹치지 않도록 생성한다.
* 최소 1개부터 최대 3개까지의 아이템이 랜덤하게 생성된다.
* 아이템의 종류 또한 랜덤하게 출현한다.
  + rand\_score()함수 : 1 또는 -1을 리턴한다.
    - 1 : Growth아이템
    - -1 : Poison아이템

**3.2 아이템 획득**

: Snake Head 부분이 Item과 부딪히게 됐을 경우를 획득했다라고 판단하며, 획득 시 몸의 길이가 증가하거나 감소한다.

// snake.cpp

void Snake::eatItem()

{

items.erase(curitem.t);

}

// snake.cpp

// Snake::movesnake()

bool itemFlag = false;

for(std::deque<Item>::iterator it=items.begin(); it!=items.end(); ++it){

if(cells.front().p.row==it->p.row && cells.front().p.col==it->p.col) {

itemFlag = true;

curitem.t = it;

curitem.p.row = it->p.row;

curitem.p.col = it->p.col;

curitem.itemFlag = it->points;

break;

}

}

if(itemFlag){

if(curitem.itemFlag == -1){ //Poison Item

cells.pop\_back();

cells.pop\_back();

poison++;

}else if(curitem.itemFlag == 1){ //Growth Item

growth++;

}

eatItem();

}

// 아이템은 5초가 지나면 자동으로 나타난다.

time\_t cur\_time;

cur\_time = time(NULL);

if(cur\_time-item\_starttime >= ITEM\_TIME){

makeItem();

}

* Snake Head 부분 = cells.front()
* Head와 아이템 좌표가 같아질 경우, 획득한 아이템에 대한 정보들을 저장한다.
  + 현재 아이템의 iterator(삭제를 위함)
  + 현재 아이템의 좌표
  + 아이템 flag (Poison인지/Growth인지 구별하기 위함)
* 아이템을 획득하면, 해당 아이템은 화면에서 사라진다.
  + eatItem()
* 아이템 획득 시, 현재까지 획득한 아이템 갯수를 저장하는 변수를 1 증가시킨다.

**4. Gate** (개발자: 조규현)

**4.1 Snake::makeGate()**

: Gate는 기본적으로 Snake 길이가 6이상일 때, 출현하게 된다. 10초마다 Gate를 새로 출현하게 되는데, Snake가 통과 중일 때는 생성하지 않는다.

// snake.cpp

void Snake::makeGate()

{

//snake가 gate 통과중인지 check;

for(std::deque<Cell>::iterator snake=cells.begin(); snake!=cells.end(); ++snake){

for(std::deque<Cell>::iterator gate=gates.begin(); gate!=gates.end(); ++gate){

if(snake->p.row==gate->p.row && snake->p.col==gate->p.col) {

return;

}

}

}

gates.clear();

gate\_starttime = time(NULL);

// WALL 중에 GATE를 두개 뽑아낸다.

for(int i=0; i<2; i++){

Cell temp;

temp.p = rand\_point(cells, items, walls, gates);

gates.push\_back(temp);

}

}

// snake.cpp

// Snake::movesnake()

if(cells.size() >= APPEAR\_GATE\_LEN && (cur\_time-gate\_starttime >= GATE\_TIME || gate\_starttime==0)){

makeGate();

}

* Snake(cells) 길이가 6 이상이고, gate가 출현한지 10초이상이면 새로운 Gate를 출현시킨다.
* Snake가 gate를 통과 중인지 Check하고, 통과 중이면 생성하지 않는다.
* Wall 중에 Gate 한 쌍을 랜덤하게 출현 시킨다.
  + Snake/Item과 좌표가 겹치지 않게 좌표를 가져온다.
  + Gate는 한 쌍이므로, 총 두개의 Gate를 생성하여 gates deque에 저장한다.

**4.2 Gate 통과**

: Snake가 하나의 Gate에 통과 중일 경우, 다른 Gate로 나와야한다.

// snake.cpp

// Snake::movesnake()

// Gate통과중인지 Check

bool gateFlag = false;

Point gateP; // 통과중인 Gate말고 반대 Gate

for(std::deque<Cell>::iterator gate=gates.begin(); gate!=gates.end(); ++gate){

if(cells.front().p.row==gate->p.row && cells.front().p.col==gate->p.col) {

gateFlag = true;

}else{

gateP = gate->p;

}

}

if(gateFlag){

Cell temp;

temp.p = gateP;

// 진입 gate가 가장자리면

if(gateP.row == 0){

fdir = DOWN;

}else if(gateP.row == MAX\_ROW-1){

fdir = UP;

}else if(gateP.col == 0){

fdir = RIGHT;

}else if(gateP.col == MAX\_COL-1){

fdir = LEFT;

}else{

// 진입할 방향이 Wall 인지 확인

for(int i=0; i<4; i++){

Point tmp = gateP;

switch(fdir) {

case UP:

tmp.row-=1;

break;

case DOWN:

tmp.row+=1;

break;

case RIGHT:

tmp.col+=1;

break;

case LEFT:

tmp.col-=1;

break;

}

// 벽이 아니면 진행방향으로.

if(!isWall(tmp)) break;

// 1) 진행방향과 반대

// 2) 그다음부턴 시계방향

if(i==0) fdir = (fdir+2)%4;

else fdir = (fdir+1)%4;

}

}

cells.push\_front(temp);

cells.pop\_back();

gate++;

}

// 벽이 아니면 진행방향으로.

if(!isWall(tmp)) break;

// 1) 진행방향과 반대

// 2) 그다음부턴 시계방향

if(i==0) fdir = (fdir+2)%4;

else fdir = (fdir+1)%4;

}

}

cells.push\_front(temp);

cells.pop\_back();

gate++;

}

* 우선 Gate가 통과 중인 지 Check하고, gateFlag 값을 true로 설정한다.
* 그렇지 않은경우, 반대편 Gate의 좌표를 저장한다.
* gateFlag==true일 경우
  + 앞에서 설정한 기준값에 따라 Gate의 이동방향을 설정해준다.
  + gate통과 횟수를 저장하는 변수를 1 증가시킨다.

**5. ScoreBoard 표시** (개발자: 윤서영)

**5.1 print\_score()**

: 획득한 점수들을 화면에 표시해준다.

// snake.cpp

int Snake::getscore()

{

return cells.size();

}

int Snake::getCntGrowth()

{

return growth;

}

int Snake::getCntPoison()

{

return poison;

}

int Snake::getCntGate()

{

return gate;

}

// snake\_ncurses.cpp

void print\_score(Snake& s)

{

for(int r=1; r<=7; r++) mvprintw(r,35, "|");

for(int r=1; r<=7; r++) mvprintw(r,60, "|");

for(int c=35; c<=60; c++) mvprintw(1,c, "-");

for(int c=35; c<=60; c++) mvprintw(7,c, "-");

mvprintw(2, 36, "\*\*\*\*\*\*\*SCORE BOARD\*\*\*\*\*\*");

mvprintw(3, 37, "B: %d", s.getscore());

mvprintw(4, 37, "+: %d", s.getCntGrowth());

mvprintw(5, 37, "-: %d", s.getCntPoison());

mvprintw(6, 37, "G: %d", s.getCntGate());

refresh();

}

* Snake.getscore() : Snake의 현재 길이
* Snake.getCntGrowth() : 현재까지 획득한 Growth 아이템 수
* Snake.getCntPoison(): 현재까지 획득한 Poison 아이템 수
* Snake.getCntGate() : 현재까지 통과한 Gate 수

**5.2 print\_mission()**

: 각 미션과 미션의 수행 여부를 출력해준다.

// snake\_ncurses.cpp

void print\_mission(Snake& s)

{

for(int r=10; r<=16; r++) mvprintw(r,35, "|");

for(int r=10; r<=16; r++) mvprintw(r,60, "|");

for(int c=35; c<=60; c++) mvprintw(10,c, "-");

for(int c=35; c<=60; c++) mvprintw(16,c, "-");

mvprintw(11, 36, "\*\*\*\*\*\*MISSION BOARD\*\*\*\*\*");

int stg = s.getStage();

if(MISSION[stg-1][0] <= s.getscore())

mvprintw(12, 37, "B: %d (V)", MISSION[stg-1][0]);

else

mvprintw(12, 37, "B: %d (%d)", MISSION[stg-1][0], s.getscore());

if(MISSION[stg-1][1] <= s.getCntGrowth())

mvprintw(13, 37, "+: %d (V)", MISSION[stg-1][1]);

else

mvprintw(13, 37, "+: %d (%d)", MISSION[stg-1][1], s.getCntGrowth());

if(MISSION[stg-1][2] <= s.getCntPoison())

mvprintw(14, 37, "-: %d (V)", MISSION[stg-1][2]);

else

mvprintw(14, 37, "-: %d (%d)", MISSION[stg-1][2], s.getCntPoison());

if(MISSION[stg-1][3] <= s.getCntGate())

mvprintw(15, 37, "G: %d (V)", MISSION[stg-1][3]);

else

mvprintw(15, 37, "G: %d (%d)", MISSION[stg-1][3], s.getCntGate());

refresh();

}

### 활용/개발된 기술

|  |
| --- |
| **작성요령 (10점)**  **프로젝트 수행에 사용한 외부 기술/라이브러리를 나열하여 작성한다. 각각 기술을 이 프로젝트에 적용할 때, 도움 받거나 해결하고자 하는 기능에 대해 상세히 설명한다.**  **NCURSES / STL 라이브러리 등을 포함하여 설명한다.**  **또한, 이 프로젝트를 수행하면서, 새롭게 고안한 알고리즘 등이 있다면 설명한다.** |

### 2.2.3.1 ncurses/ncursesw

* GUI기반 Game을 만들기위해 사용하였다.
* Snake/Item/Gate를 Unicode 문자열로 표시할 때 사용한 라이브러리다.

### 2.2.3.2 deque(STL라이브러리)

* Snake/Item/Gate 좌표와 같은 정보를 담기 위해, 2차원 배열이 아닌 deque라는 STL라이브러리를 사용하였다.
* 특히, Snake 움직임을 구현하는 과정에서, push\_front같은 함수를 제공해주는 것 때문에 vector가 아닌 deque 컨테이너를 사용하게 되었다.

### 2.2.3.3 ctime

* random하게 좌표를 생성하기위해 ctime라이브러리를 활용하였다.
* Item의 경우, 초단위로 생성하게 될 경우, 동시 출현 아이템들의 생성시간이 거의 같아서 종류가 모두 같아지는 버그가 발생하였다. 따라서 초단위가 아닌 millisecond 기준으로 seed를 생성하도록 clock\_gettime 함수를 사용하였다.

### 2.2.3.4 cstdlib

* rand() 함수를 호출하여 랜덤한 값을 추출하기위해 사용하였다.

### 현실적 제한 요소 및 그 해결 방안

|  |
| --- |
| **작성요령 (5점)**  **제안된 프로젝트의 단계 별 수행에 있어, 제한 요소를 찾아 작성한다. 해당 제한 요소를 해결하기 위해서 어떤 방법으로 해결하였는지 작성한다.** |

* + 각 Stage마다 Map을 구성할 때, Immunewall/wall에 대한 정보를 2차원 배열로 선언했었는데, 매번 2차원배열의 값을 수정하는 것 때문에 시간이 많이 소요되는 문제점이 있었다.
  + 따라서, 2차원 배열이 아닌 deque에 해당 좌표들을 저장하여 속도를 향상시킬 수 있도록 수정하였다.

### 결과물 목록

|  |
| --- |
| **작성요령 (5점)**  **결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.** |

|  |  |
| --- | --- |
| 파일명 | 역할 |
| Makefile | 컴파일을 하기위한 makefile |
| main.cpp | main함수 호출 |
| snake.hpp | Snake Class가 선언되어있음 |
| snake.cpp | Snake Class 메소드 구현 |
| snake\_ncurses.hpp | Game에 필요한 요소들을 선언 |
| snake\_ncurses.cpp | Game에 필요한 핵심 로직들이 구현 |
| stage.hpp | stage.cpp 에서 사용되는 메소드 선언 |
| stage.cpp | 각 stage마다 map을 구성하는 로직 구현 |

# 자기평가

|  |
| --- |
| **작성요령 (5점)**  **프로젝트를 수행한 자기 평가를 서술한다. 팀원 개개인의 자기 평가가 포함되어야 하며, 본인의 역할, 프로젝트 수행 시 어려운 점, 도움이 되었던 점, 이 프로젝트 운영에 개선이 필요하다고 생각하는 점을 충분히 서술한다.** |

|  |  |  |
| --- | --- | --- |
| 이름 | 느낀점 | 평가 |
| 윤서영 | 프로젝트 리더를 맞게 되어 프로젝트를 기한 내로 끝낼 수 있을 지와 요구사항에 맞게 완벽하게 구현할 수 있을 지에 대한 부담감이 있었다. 프로젝트를 진행하면서 오픈소스 라이브러리에 대한 중요성을 알게 되었고, 하다가 막히는 부분이 있으면 오픈소스를 활용하여 극복해낼 수 있었다. | **9점** |
| 이한정 |  |  |
| 조규현 |  |  |
| 황영서 |  |  |

# 참고 문헌

**.**

# 부록

|  |
| --- |
| **작성요령 (15점)**  **프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.** |

## 사용자 매뉴얼

**프로젝트 실행 후 사용안내, 따라하기 등 포함**

## 설치 방법

### 컴파일

* 개발OS: Linux Ubuntu 16.04 LTS
* Version: C++ 14
* Compile Command: **make**

### 실행

* Command: **./snake\_ncurses**