



국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	<i>Snake game</i>
팀 명	<i>안현섭, 표우진</i>
문서 제목	결과보고서

Version	1.2
Date	2021-06-19

팀원	안현섭
	표우진

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 **C++**프로그래밍 수강 학생 중 프로젝트 "**Snake game**"을 수행하는 팀 "안현섭, 표우진"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "안현섭, 표우진"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	최종보고서-Snakegame.doc
원안작성자	안현섭, 표우진
수정작업자	안현섭, 표우진

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2021-06-19	안현섭	1.0	최초 작성	원안 작성, snake.cpp, snake.h, makefile 구현 내용 설명
2021-06-19	표우진	1.1	내용 수정	CreateTerm.cpp, CreateTerm.h 설명

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

목 차

1	개요	4
2	개발 내용 및 결과물	5
2.1	목표	5
2.2	개발 내용 및 결과물	7
2.2.1	개발 내용	7
2.2.2	시스템 구조 및 설계도	18
2.2.3	활용/개발된 기술	19
2.2.4	현실적 제한 요소 및 그 해결 방안	19
2.2.5	결과물 목록	20
3	자기평가	20
4	참고 문헌	22
5	부록	22
5.1	사용자 매뉴얼	22
5.2	설치 방법	23

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

1 개요

평가기준 (10점)

프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.

또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.

2021년 1학기 C++ 프로그래밍 프로젝트로 진행된 Snakegame Project
C++ 언어로 구현하였고 ncurses 라이브러리를 이용하여 Snakegame을 구현한다.

Ncurses 획득/설치 방법


```
sudo apt-get update
sudo apt-get install libncurses5-dev libncursesw5-dev
```

● 컴파일 방법

\$ g++ -o snake 소스-화일-이름들 -lncurses

● 프로그램 소스가 snake.cpp, rule.cpp 라고 할 때

\$ g++ -o snake snake.cpp rule.cpp -lncurses

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

2 개발 내용 및 결과물

2.1 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	미적용

1단계 - Map

Snake 게임의 원활한 실행을 위한 Map을 생성한다.

NCurses Library 함수들을 사용하여 2차원 배열로 된 Snake Map을 화면으로 표시한다. 이 때 배열의 크기는 60 by 30 으로 정한다.

2단계 – Snake

- Snake 의 초기 길이는 3이고 초기 방향은 왼쪽이다.
- 사용자가 방향키를 입력하면, Snake 의 좌표는 0.11초마다 진행방향으로 변한다.
- 사용자가 Snake 의 방향과 반대 방향 키를 누르면 게임이 종료된다.
- 머리가 벽에 닿거나 몸통에 닿으면 게임이 종료된다.
- Snake 는 속이 꽉찬 원으로 표현한다.
- Snake 의 길이가 3이하로 줄어든 경우 게임이 종료된다.
- 머리가 아이템을 먹으면 아이템에 따라 몸이 늘어나거나 줄어든다.
- 머리가 게이트를 통과하면 다른 게이트에 나와야 한다.

3단계 – Item

- 아이템은 Snake 와 Wall 에 겹치지 않도록 생성된다.
- 3개까지 랜덤으로 생성된다.
- Growth 는 ★이고, Poison 은 ♠이다.

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

4단계 – Gate

- Snake 가 게이트를 통과하는 동안 게이트는 없어져선 안된다.
- 게이트의 입구가 공유되면 안된다.
- 스네이크의 진행방향에 따라 게이트에서 나오는 스네이크의 방향이 바뀌어야한다.
- 게이트는 한번에 한쌍만 갖는다.
- 게이트는 Immune Wall 에 나타나지 않는다.

5단계 – Score

다음 단계로 넘어가기 위한 목표 점수 및 현재 점수 시스템
 모든 미션을 완료해야 다음 라운드로 넘어갈 수 있다.

획득한 Grow, Poison Item 개수와 게이트 이용횟수, 목표 뱀 길이

 <div> 국민대학교 소프트웨어학부 C++ </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

2.2 개발 내용 및 결과물

2.2.1 개발 내용

작성요령 (10점)

프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.

1. Map (1단계)

개별 라운드에 사용될 맵의 정보를 담은 텍스트 파일과 맵으로 사용할 60by30 크기의 Stage 텍스트 파일을 미리 작성하여 C++에서 제공하는 파일 입출력 기능을 이용해 프로그램 내의 2차원 배열에 저장하여 맵을 출력한다.

```

void SetStage() {
    std::fstream setting("mapfile.txt");
    std::fstream map;
    for (int i = 0; getline(setting, mapstr) && i < RoundCount - 1; i++) {
    }
    map.open(mapstr);
    for (int i = 0; i < 30; i++) {
        for (int j = 0; j < 60; j++) {
            map >> buffer[i][j];
        }
    }
}

```

mapfile 텍스트 파일에 미리 Stage 정보를 입력하여 라운드별로 해당 스테이지를 불러올 수 있도록 구현하였다.

 <div> 국민대학교 소프트웨어학부 C++ </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

```

void Terminals::GameBoard() {
    Board_GAME = newwin(30, 60, 1, 1); // column, row, point y, point x;
    nodelay(stdscr, true);
    wbkgd(Board_GAME, COLOR_PAIR(8));
    wattron(Board_GAME, COLOR_PAIR(8));
    for (int i = 0; i < 30; i++) {
        wattron(Board_GAME, COLOR_PAIR(4));
        move(i, 2);
        for (int j = 0; j < 60; j++) {
            if (buffer[i][j] == '0')
                wprintw(Board_GAME, " ");
            else if (buffer[i][j] == '1' || buffer[i][j] == 'W')
                wprintw(Board_GAME, "■");
            else
                wprintw(Board_GAME, " ");
        }
        wattroff(Board_GAME, COLOR_PAIR(4));
    }
    wrefresh(Board_GAME);
    Snake snake;
    snake.play();
}

```

Map 파일 내의 숫자에 따라 벽을 표현하였다.

3. Snake, Item, Gate 로직 (2,3,4단계)

1) 좌표 생성 / Snake, Item, Gate

```

Pos::Pos(int col,int row)
{
    x=col;
    y=row;
}

Pos::Pos()
{
    x=0;
    y=0;
}

```

Snake, Item, Gate의 좌표를 불러와 vector에 push_back하기 위한 함수

2) Snake 생성 및 초기화, 소멸자

```

Snake::Snake()
{
    // nodelay(stdscr,true);
    keypad(stdscr,true); // 키패드 생성
    noecho();
    curs_set(0); // 커서 제거
}

```




```
fail=false;
for(int i = 0; i<maxheight; i++){
    for(int j = 0; j<maxwidth; j++){
        if(buffer[i][j] == '3')
            snake.push_back(Pos(j,i));
    }
}
point=0;
len=3;
growCnt=0;
poisonCnt=0;
gateCnt=0;
delay=110000;
getGrow=false;
getPoison=false;
gateNum = -1;
dir='l';
srand(time(NULL));
putGate();
for(int i = 0; i<3; i++){
    putGrow(); // growItem 생성
    putPoison(); // poisonItem 생성
}
for(int i=0;i<snake.size();i++) // snake 맵에 그리기
{
    mvwprintw(Board_GAME,snake[i].y,snake[i].x,"●");
}
}
```

Snake의 생성과 함께 필요한 조건들을 초기화 해주고 맵에 Snake를 그려준다. 생성과 동시에 Gate, Grow, Poison을 맵에 삽입할 수 있는 함수도 같이 넣어주었다.

```
Snake::~Snake() // Snake Destructor
{
    nodelay(stdscr,false);
    getch();
    endwin();
}
```

프로그램 종료 후 destructor를 통해 메모리를 비워준다.

3) Snake 동작

```
void Snake::moveSnake(){
    kbhit();
    if(getGrow){ // grow item 을 먹을 때
        if(dir=='l')
```



```
        snake.insert(snake.begin(),Pos(snake[0].x-1,snake[0].y));
    else if(dir=='r')
        snake.insert(snake.begin(),Pos(snake[0].x+1,snake[0].y));
    else if(dir=='u')
        snake.insert(snake.begin(),Pos(snake[0].x,snake[0].y-1));
    else if(dir=='d')
        snake.insert(snake.begin(),Pos(snake[0].x,snake[0].y+1));
    }
    else if(getPoison){ // poison item 을 먹을 때
        wmove(Board_GAME,snake[snake.size()-1].y,snake[snake.size()-
1].x);
        wprintw(Board_GAME," ");
        wmove(Board_GAME,snake[snake.size()-2].y,snake[snake.size()-
2].x);
        wprintw(Board_GAME," ");
        wrefresh(Board_GAME);
        snake.pop_back();
        snake.pop_back();
        if(dir=='l')
            snake.insert(snake.begin(),Pos(snake[0].x-1,snake[0].y));
        else if(dir=='r')
            snake.insert(snake.begin(),Pos(snake[0].x+1,snake[0].y));
        else if(dir=='u')
            snake.insert(snake.begin(),Pos(snake[0].x,snake[0].y-1));
        else if(dir=='d')
            snake.insert(snake.begin(),Pos(snake[0].x,snake[0].y+1));
    }

    else{ // 아무것도 먹지 않았을 때
        wmove(Board_GAME,snake[snake.size()-1].y,snake[snake.size()-1].x);
        wprintw(Board_GAME," ");
        wrefresh(Board_GAME);
        snake.pop_back();
        if(dir=='l')
            snake.insert(snake.begin(),Pos(snake[0].x-1,snake[0].y));
        else if(dir=='r')
            snake.insert(snake.begin(),Pos(snake[0].x+1,snake[0].y));
        else if(dir=='u')
            snake.insert(snake.begin(),Pos(snake[0].x,snake[0].y-1));
        else if(dir=='d')
            snake.insert(snake.begin(),Pos(snake[0].x,snake[0].y+1));
    }
    mvwprintw(Board_GAME,snake[0].y,snake[0].x,"●");
    wrefresh(Board_GAME);
}

void Snake::moveGate(){ // 게이트안에서 움직일 때

    if(gateNum == 0){ // 진입 = 0, 진출 = 1
        if(gate[1].x == 0)
            dir = 'r';
        else if(gate[1].x == 59)
            dir = 'l';
```



```
else if(gate[1].y == 0)
    dir = 'd';
else if(gate[1].y == 29)
    dir = 'u';
else{
    if(buffer[gate[1].y-1][gate[1].x] != '1' ||
buffer[gate[1].y+1][gate[1].x] != '1'){
        if(dir == 'u' || dir == 'l')
            dir = 'u';
        else
            dir = 'd';
    }
    else{
        if(dir == 'u' || dir == 'l')
            dir = 'l';
        else
            dir = 'r';
    }
}
for(int i = 0; i<snake.size(); i++){
    kbhit();
    wmove(Board_GAME,snake[snake.size()-1].y,snake[snake.size()-
1].x);
    wprintw(Board_GAME," ");
    wrefresh(Board_GAME);
    snake.pop_back();
    if(dir=='l'){
        snake.insert(snake.begin(),Pos(gate[1].x-1,gate[1].y));
        gate[1].x = gate[1].x-1;
    }
    else if(dir=='r'){
        snake.insert(snake.begin(),Pos(gate[1].x+1,gate[1].y));
        gate[1].x = gate[1].x+1;
    }
    else if(dir=='u'){
        snake.insert(snake.begin(),Pos(gate[1].x,gate[1].y-1));
        gate[1].y = gate[1].y-1;
    }
    else if(dir=='d'){
        snake.insert(snake.begin(),Pos(gate[1].x,gate[1].y+1));
        gate[1].y = gate[1].y+1;
    }
    mvwprintw(Board_GAME,snake[0].y,snake[0].x,"●");
    wrefresh(Board_GAME);
    usleep(delay);
}
}

else if(gateNum == 1){ // 진입 = 1, 진출 = 0
    if(gate[0].x == 0)
        dir = 'r';
    else if(gate[0].x == 59)
```



```
        dir = 'l';
    else if(gate[0].y == 0)
        dir = 'd';
    else if(gate[0].y == 29)
        dir = 'u';
    else{
        if(buffer[gate[0].y-1][gate[0].x] != '1' ||
buffer[gate[0].y+1][gate[0].x] != '1'){
            if(dir == 'u' || dir == 'l')
                dir = 'u';
            else
                dir = 'd';
        }
        else{
            if(dir == 'u' || dir == 'l')
                dir = 'l';
            else
                dir = 'r';
        }
    }
    for(int i = 0; i<snake.size(); i++){
        kbhit();
        wmove(Board_GAME,snake[snake.size()-1].y,snake[snake.size()-
1].x);
        wprintw(Board_GAME," ");
        wrefresh(Board_GAME);
        snake.pop_back();
        if(dir=='l'){
            snake.insert(snake.begin(),Pos(gate[0].x-1,gate[0].y));
            gate[0].x = gate[0].x-1;
        }
        else if(dir=='r'){
            snake.insert(snake.begin(),Pos(gate[0].x+1,gate[0].y));
            gate[0].x = gate[0].x+1;
        }
        else if(dir=='u'){
            snake.insert(snake.begin(),Pos(gate[0].x,gate[0].y-1));
            gate[0].y = gate[0].y-1;
        }
        else if(dir=='d'){
            snake.insert(snake.begin(),Pos(gate[0].x,gate[0].y+1));
            gate[0].y = gate[0].y+1;
        }
        mvwprintw(Board_GAME,snake[0].y,snake[0].x,"●");
        wrefresh(Board_GAME);
        usleep(delay);
    }
}
```

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

Snake의 움직임을 vector를 통해 좌표를 받아와 움직임을 구현하였다.

Kbhit()함수는 ncurses 라이브러리의 getch() 함수를 통해 키를 받아오기 위한 함수를 따로 구현 하였다. Grow, Poison 아이템을 만났는지 확인하여 grow를 만나면 길이를 +1 해주고, Poison을 만났을 경우는 -1를 해준다. 아무것도 만나지 않았을 경우는 vector를 통해 좌표만 바꿔준다.

Gate를 진입했을 경우 vector좌표를 gate 좌표로 바꿔줘야 하기 때문에 따로 구현하였다. 구현 방식은 위의 방식과 동일하다.

4) Snake가 gate, poison, grow, wall을 만났을 경우를 구현

```
bool Snake::collision()
{
    // 벽과 충돌했을 때
    for(int i = 0; i<maxheight; i++){
        for(int j = 0; j<maxwidth; j++){
            if(buffer[i][j] == '1')
                if(snake[0].x == j && snake[0].y == i)
                    return true;
        }
    }
    for(int i=2;i<snake.size();i++)
    {
        if(snake[0].x==snake[i].x && snake[0].y==snake[i].y) //
        몸통과 충돌할 때 GameOver
            return true;
    }
    for(int i = 0; i<2; i++){
        if(snake[0].x == gate[i].x && snake[0].y == gate[i].y){ //
        gate 와 마주쳤을 때
            gateNum = i;
            int gate1tmpX = gate[0].x;
            int gate1tmpY = gate[0].y;
            int gate2tmpX = gate[1].x;
            int gate2tmpY = gate[1].y;
            moveGate(); // 게이트안에서 이동
            wattron(Board_GAME,COLOR_PAIR(4));
            mvwprintw(Board_GAME,gate1tmpY,gate1tmpX,"■");
            mvwprintw(Board_GAME,gate2tmpY,gate2tmpX,"■");
            buffer[gate1tmpY][gate1tmpX] = '1';
            buffer[gate2tmpY][gate2tmpX] = '1';
            wrefresh(Board_GAME);
            wattroff(Board_GAME,COLOR_PAIR(4));
            gate.clear();
            putGate();
            gateCnt+=1;
            break;
        }
    }
}
```



```
}
for(int i = 0; i<3; i++){
    if(snake[0].x==growItem[i].x && snake[0].y==growItem[i].y)
// growItem 과 마주쳤을 때
    {
        growItem.erase(growItem.begin()+i);
        getGrow=true;
        growCnt+=1;
        putGrow();
        point+=1;
        len+=1;
        break;
    }
    else if(snake[0].x==poisonItem[i].x &&
snake[0].y==poisonItem[i].y) // poisonItem 과 마주쳤을 때
    {
        poisonItem.erase(poisonItem.begin()+i);
        getPoison=true;
        poisonCnt+=1;
        putPoison();
        point-=1;
        len-=1;
        break;
    }
    else{
        getGrow=false;
        getPoison=false;
    }
}

return false;

}
bool Snake::gameOver(){
    fail=true;
    return fail;
}
```

경우를 4가지로 나눠서 구현하였다.

1번째 snake 몸통과 충돌했을 경우

2번째 gate와 충돌했을 경우

3번째 growitem과 충돌했을 경우

4번째 poisonitem과 충돌했을 경우

5) 방향전환을 위한 kbhit()



```
void Snake::kbhit(){ // 방향전환
    int tmp = getch();
    switch(tmp)
    {
        case KEY_LEFT:
            if(dir!='r')
                dir='l';
            else
                gameOver();
            break;
        case KEY_UP:
            if(dir!='d')
                dir='u';
            else
                gameOver();
            break;
        case KEY_DOWN:
            if(dir!='u')
                dir='d';
            else
                gameOver();
            break;
        case KEY_RIGHT:
            if(dir!='l')
                dir='r';
            else
                gameOver();
            break;
        case KEY_BACKSPACE:
            dir='q';
            break;
    }
}
```

Ncurses 라이브러리의 getch()를 이용하여 왼쪽 방향키를 누르면 dir를 'l'로 오른쪽 방향키는 'r' 위쪽 방향키는 'u' 아래방향키는 'd'으로 방향을 변경 시켜준다.

6) Item 생성을 위한 putGate(), putGrow(), putPoison()

```
void Snake::putGate(){
    while(gate.size()<2)
    {
        int tmpX=rand()%maxwidth;
        int tmpY=rand()%maxheight;
        if(buffer[tmpY][tmpX] != '1') // 벽에 게이트 생성
            continue;
        for(int i = 0; i<gate.size(); i++){
            if(gate[i].x == tmpX && gate[i].y == tmpY)
                continue;
        }
    }
}
```



```
    }
    buffer[tmpY][tmpX] = ' ';
    gate.push_back(Pos(tmpX,tmpY));
}
for(int i = 0; i<gate.size(); i++){
    mvwprintw(Board_GAME, gate[i].y, gate[i].x, "□");
}
wrefresh(Board_GAME);
}
void Snake::putGrow()
{
    int tmpX, tmpY;
    while(1)
    {
        tmpX=rand()%maxwidth+1;
        tmpY=rand()%maxheight+1;
        for(int i=0; i<snake.size(); i++){
            if(snake[i].x==tmpX && snake[i].y==tmpY){ // 스네이크
몸통에 Grow 생성불가
                continue;
            }
            if(buffer[tmpY][tmpX] == '1') // 벽에 Grow 생성불가
                continue;
            growItem.push_back(Pos(tmpX,tmpY));
            break;
        }
        mvwprintw(Board_GAME, tmpY, tmpX, "★");
        wrefresh(Board_GAME);
    }
}
void Snake::putPoison()
{
    int tmpX, tmpY;
    while(1)
    {
        tmpX=rand()%maxwidth+1;
        tmpY=rand()%maxheight+1;

        for(int i=0; i<snake.size(); i++){
            if(snake[i].x==tmpX && snake[i].y==tmpY) // 스네이크
몸통에 Poison 생성불가
                continue;
        }
        if(buffer[tmpY][tmpX] == '1') // 벽에 Poison 생성불가
            continue;
        poisonItem.push_back(Pos(tmpX,tmpY));
        break;
    }
    mvwprintw(Board_GAME, tmpY, tmpX, "♠");
    wrefresh(Board_GAME);
}
}
```


 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

putGate()는 Map에 '1'인 벽에 랜덤으로 gate를 생성한다. 이 때 두 게이트가 겹치지 않게 하기 위해 겹칠 경우 다시 반복문이 실행 되게 작성했다.

putGrow()는 Map에 벽과 스네이크를 제외한 곳에 Grow를 3개를 생성한다. 이 때 위와 동일하게 3개의 grow가 겹치지 않게 한다.

putPoison()은 putGrow와 동일하게 구현하였다.

7) 게임 종료를 위한 gameOver()


```
bool Snake::gameOver(){
    fail=true;
    return fail;
}
```

게임종료를 위한 조건을 달성시 fail을 ture로 변경시킨다.

8) 게임 종료를 위한 gameOver()

```
void Snake::play()
{
    while(1)
    {
        if(collision() || fail || len < 3) // 충돌 or 정반대 방향을
        입력하면 return true;
        {
            break;
        }
        moveSnake();
        if(dir=='q') // 백스페이스 입력시 게임 stop
            break;
        usleep(delay); // delay
    }
}
```

Snake의 전체로직을 수행하기 위한 Play()함수를 구현하였다.

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

2.2.2 시스템 구조 및 설계도

작성요령 (30 점)


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.

Snake – 전체적인 로직 구현

함수명	인자	리턴타입	설명	제작자
Kbhit()		void	키보드 입력	안현섭
moveGate()		Void	Gate 안에 진입했을 때 snake 움직임	안현섭
putGate()		Void	Gate 생성	안현섭
putGrow()		Void	Growth 생성	안현섭
putPoison()		void	Poison 생성	안현섭
Collision()		Bool	Snake 가 벽과 충돌했는지 판단	안현섭
gameOver()		bool	게임 종료	안현섭
moveSnake ()		void	Snake 움직임 구현	안현섭
Play()		void	전체 게임을 판단 및 실행	안현섭

CreateTerm – 함수 구현

함수명	인자	리턴타입	설명	제작자
GameBoard()		void	게임 윈도우 생성	표우진
ScoreBoard()		Void	스코어 윈도우 생성	표우진
MissionBoard()		Void	임무 목표 윈도우 생성	표우진
EndStage()		Void	게임이 종료되었을시에 다음라운드 or 종료를 묻는 함수	표우진
StartScreen()		void	시작화면 출력	표우진
SetStage()		void	2 차원 배열에 맵 타일 저장	표우진

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

2.2.3 활용/개발된 기술

작성요령 (10 점)

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.

<vector>를 이용하여 Snake, Gate, Item 의 좌표를 저장하는 용도로 사용하였다.

<vector>를 사용하여 Snake 의 몸이 길어졌다가 줄어듦과 벡터의 값을 바꾸는 것으로 snake 가 움직이는 걸 구현했다.

<fstream>를 이용하여 map 과 mission data 를 파일 입출력으로 불러와 저장한다.

<string>를 이용하여 스크린에 점수 값 등을 표현하였다.

<locale>를 이용하여 유니코드 작성이 가능하도록 하였다.

<wchar.h>를 이용하여 wchar_t 자료형을 사용하도록 하였다.

<ncurses>

ncurses 에 게임 로직을 이용하여 얻은 데이터를 바탕으로 snake, map, item, gate, mission 등의 정보를 출력하고 User Interface 를 구현하였다.

2.2.4 현실적 제한 요소 및 그 해결 방안

작성요령 (5 점)

제안된 프로젝트의 단계 별 수행에 있어, 제한 요소를 찾아 작성한다. 해당 제한 요소를 해결하기 위해서 어떤 방법으로 해결하였는지 작성한다.

1. 호환성 문제

저희 팀은 단계별로 역할 분담을 맡아 구현하였는데, 이 때 만들어진 Map과 Snake를 호환 시키는데 상당한 어려움을 겪었다. 서로의 멤버함수와 멤버변수를 고려하지않은 패착이었다. 이 경우 회의를 통해 서로의 코드를 맞춰 해결 할 수 있었다.

2. 터미널 크기 문제

터미널에서 ncurses를 실행 시 화면을 전체화면으로 실행시키지 않을 경우 window가 깨지는 현상이 발생하였다. 이 같은 경우 문제해결을 위해 시도를 해봤으나 결국 해결 하

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

지 못하였다. 그래서 사용자 매뉴얼에 꼭 전체화면으로 실행하라는 언급을 하였다.

3. Gate 인아웃 문제

게이트를 진입하고 빠져나올 때 요구사항의 방향대로 나오는게 아닌 이상한 방향으로 이동되어 나오는 문제를 겪었다. 이 경우 snake.cpp내에서 movegate()함수를 따로 구현하여 작동시키니 해당 문제를 해결 할 수 있었다.

4. 다른 소스 파일의 다른 클래스간 변수 or 함수 호환 문제

다른 파일간의 멤버 변수와 함수를 불러오는데에 어려움을 겪었다.
객체 선언과 포인터 선언을 이용하여 해결하였다.

2.2.5 결과물 목록

작성요령 (5 점)

결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.

CreateTerm.cpp

- 전체적인 UI 와 맵 생성을 담당하는 소스파일

CreateTerm.h

- CreateTerm 의 헤더파일, Snake 헤더파일과의 호환

Snake.cpp

- Snake 움직임과 게이트 및 아이템 구현을 위한 소스파일


Snake.h

- Snake 의 헤더파일, 뱀의 좌표와 각종 변수들 선언

Main.cpp

- Snake Game 구동을 위한 main 함수 소스파일

3 자기평가

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

작성요령 (5 점)

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.

20175166 안현섭

이번 과제에서 전체적인 Snake 의 움직임을 표현하는 메인 로직 구성을 맡게 되었다. ncurses 를 사용해보는게 처음이다 보니 터미널에 text UI 로 나타내 구현하는데 많은 어려움을 겪었다. 그래서 처음엔 알고리즘을 짜는데 시간을 투자하기 보다는 ncurses 라는 라이브러리를 공부하는데 시간을 많이 투자 했던 것 같다. 차근차근 하나씩 배워가는 느낌으로 프로젝트를 수행 하였고, 처음에 스크린에 snake 를 움직이게 구현했을 때의 쾌감은 아직도 잊혀지지 않는다. 이번 강의에서 C++를 처음 접해 문법에 익숙하지 않았는데 프로젝트를 수행하면서 강의에서 배운 내용들이 생각나며 복습효과도 있었다. 하지만 강의에서 배운 거에 비해 몰라서 찾아 사용해야 하는 것들이 많아 상대적으로 프로젝트가 처음 시작해야할 때 어떻게 해야 할지 막막한 느낌이 없지 않아 있었다. 그래서 코딩을 시작한지 별로 안되었거나 경험이 적은 학생들을 위해 예시 코드를 넣어서 방향성을 제시해주면 처음에 시작할 때 더 수월하게 진행 할 수 있을 거란 생각이 들었다. 마지막으로 이번에 게임을 만드는 것이 처음인데 이 경험을 발판 삼아 나중에 현업에 나가서도 어떠한 개발상황이 생기더라도 할 수 있다는 자신감을 불어넣어준 좋은 경험이었던 것 같다.

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

20185291 표우진

다양한 맵의 UI 와 각 클래스와 멤버 함수간의 원활한 구동을 계획하였지만 생각만큼 잘 실행하지는 못하였다. ncurses 라는 라이브러리는 거의 새로운 문법을 배우는 것 같은 느낌이 들었으며 또한 팀 프로젝트에서는 서로 간의 소통과 초기 단계에서의 계획이 중요하다는 것을 뼈저리게 느낄 수 있는 시간이었다.

C++ 수업에서 과제로만 잠깐잠깐 하던 문법들을 총 망라하여 사용하는 것이다보니 내가 모르는 게 아직 참 많구나 라는 생각을 하게 되었다.

이번 프로젝트를 치러비행삼아 다음번에는 더욱더 완성도 있고 직관적인 프로그램을 작성할 수 있을 것이라고 확신한다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	web	Cpp Reference	https://www.cplusplus.com/reference/			
2	web	stackoverflow	https://stackoverflow.com/			
3	web	ncurses 프로그래밍	https://psman2.tistory.com/entry/ncurses-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D	2003-03-10	윤상배	
4	web	Ncurses Guide	http://www.cs.ukzn.ac.za/~hughm/os/notes/ncurses.html			

5 부록

작성요령 (15 점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

5.1 사용자 매뉴얼

※주의사항

ncurses 라이브러리를 설치한 후 진행한다.

실행 시 터미널의 크기가 전체크기가 아니면 화면이 깨질 수 있으니 꼭 전체크기로 실행

 국민대학교 소프트웨어학부 C++	결과보고서		
	프로젝트 명	Snake game	
	팀 명		
	Confidential Restricted	Version 1.2	2021-06-19

시켜야 한다.

-게임 설명- (수정 요망)

방향키 : ↑ ↓ ← →

■ : 벽

● : Snake 머리

○ : Snake 몸통

♣ : Grow 아이템 (획득시 길이+1)

✕ : Poison 아이템 (획득시 길이-1)

Ⓜ : Gate (진입시 반대 Gate로 진출)

게임은 총 3개의 스테이지로 구성되어 있습니다.

스테이지의 미션을 모두 클리어하시면 다음 스테이지로 넘어갑니다.

마지막 스테이지를 완료하시면 게임이 종료됩니다.

gameover

■ 에 충돌 시

반대 방향 방향키 입력 시

○(몸통)에 ●(머리) 충돌 시

Snake의 길이가 3이하로 줄어들 시

5.2 설치 방법

1. 파일 다운로드
2. 터미널 실행
3. 다운받은 폴더로 이동 후 Command : make Main
4. ./Main

5.3 데모 동영상 링크

<https://youtu.be/ZABU5NwBwaY>