




국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	Snake Game
팀 명	6조
문서 제목	SNAKE GAME 결과 보고서

Version	1.4
Date	2022-JUN-13

팀원	박창용 (20213003)
	안시현 (20180502)
	이정훈 (20171678)

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

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 "**6조**"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "**6조**"의 팀원들의 서면 허락 없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	최종보고서_SnakeGame_6조.docx
원안작성자	박창용, 안시현, 이정훈
수정작성자	박창용, 안시현, 이정훈

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2022-06-13	이정훈	1.0	최초 작성	결과 보고서 초안 작성
2022-06-15	박창용	1.1	내용 추가	자기평가 및 macOS 설치 방법 작성
2022-06-15	안시현	1.2	내용 추가	자기평가 작성
2022-06-16	안시현	1.3	내용 추가	개발내용 작성
2022-06-16	이정훈	1.4	내용 추가	내용 통합

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

목 차

1	개요	4
1.1	프로젝트 소개	4
1.2	개발환경	4
1.3	사용 라이브러리 소개	4
1.3.1	ncurses	4
1.3.2	unistd	4
1.4	게임 규칙	5
1.4.1	아이템	5
1.4.2	게이트	5
1.4.3	점수 및 미션	5
1.4.4	클리어 조건	5
1.4.5	패배 조건	5
2	개발 내용 및 결과물	6
2.1	목표	6
2.2	개발 내용 및 결과물	6
2.2.1	개발 내용	6
2.2.2	시스템 구조 및 설계도	8
2.2.3	활용/개발된 기술	24
2.2.4	현실적 제한 요소 및 그 해결 방안	25
2.2.5	결과물 목록	26
3	시연영상	26
4	자기평가	27
5	참고 문헌	28
6	부록	28
6.1	Github Repository	28
6.2	설치 방법	28
6.2.1	설치 방법 1 - git clone	28
6.2.2	설치 방법 2 - Download ZIP	28
6.3	사용자 매뉴얼	29

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

1 개요

1.1 프로젝트 소개

본 프로젝트는 국민대학교 소프트웨어학부 C++ 프로그래밍 과목 팀 프로젝트의 일환으로 C++ 프로그래밍 언어로 ncurses 라이브러리를 사용하여 Snake Game을 구현하고자 한다. Snake Game은 화면에 있는 뱀을 상하좌우로 조작하여 뱀이 화면 끝이나 장애물에 부딪히지 않고 머리가 몸통에 닿지 않으면서 화면에 무작위로 생성되는 여러 아이템을 획득하며 진행하는 게임이다.

1.2 개발환경

본 프로젝트는 macOS와 국민대학교 소프트웨어학부 C++ 프로그래밍 과목에서 요구하는 표준 환경인 Linux Ubuntu 20.04(이하 Ubuntu)를 개발 환경으로 하고 있으며 C++ 소스 코드 컴파일을 위한 컴파일러는 GCC/G++ 컴파일러를 사용한다.

1.3 사용 라이브러리 소개

1.3.1 ncurses

ncurses 라이브러리는 UNIX System에서 터미널 기반 프로그래밍을 위해 사용하던 curses 라이브러리를 재작성한 것으로, 프로그래머가 텍스트 사용자 인터페이스를 터미널 독립 방식으로 기록할 수 있도록 API를 제공하는 라이브러리이다. 터미널에서 화면 제어, 키보드 제어를 돕거나 윈도우를 출력하는 등의 작업을 가능하게 한다. 본 프로젝트에서는 Snake Game을 진행하기 위한 윈도우를 사용자에게 출력하며 사용자의 키보드 입력을 바탕으로 게임 진행, Snake의 움직임을 제어하기 위해 ncurses 라이브러리를 사용한다.

Ubuntu 설치 방법

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

macOS 설치 방법

```
brew install ncurses
```

1.3.2 unistd

unistd 라이브러리는 UNIX OS에서 쓰이는 표준 심볼과 상수를 정의해 놓은 표준 라이브러리로, UNIX 시스템 프로그래밍을 위해 주로 쓰인다. UNIX 기반 컴파일러를 사용하면 별다른 설치 없이 사용할 수 있다. 본 프로젝트에서는 unistd 라이브러리의 마이크로초 간격으로 호출 프로세스의 실행을 일시 중지하는 usleep 함수를 사용하여 Snake의 움직임을 표현한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

1.4 게임 규칙

키보드의 WSAD 키 또는 방향키를 이용하여 **Snake** 를 상하좌우로 조작한다.

1.4.1 아이템

Snake 는 다음과 같은 아이템을 획득할 수 있다.

- 🍷 Grow Item - 몸의 길이가 1 만큼 증가한다.
- ☠️ Poison Item - 몸의 길이가 1 만큼 감소한다.

- ◆ **Item** 은 **Snake** 가 있지 않은 임의의 위치에 출현한다.
- ◆ **Item** 은 출현 후 일정시간이 지나면 사라지고 다른 위치가 나타난다.
- ◆ **Item** 은 동시에 3 개 이하만 출현할 수 있다.

1.4.2 게이트

- ◆ **Gate** 는 두 개가 한쌍으로 한번에 한 쌍만 나타나며 생성 위치는 서로 겹치지 않는다.
- ◆ **Gate** 는 임의의 위치의 벽에서 생성된다.
- ◆ **Gate** 에 **Snake** 가 진입하면 연결된 다른 **Gate** 로 진출한다.
- ◆ **Gate** 에 **Snake** 가 진입하면 **Gate** 는 사라지지 않으며 다른 위치에서 **Gate** 가 나타나지 않는다.

1.4.3 점수 및 미션

- ◆ 몸의 최대길이, 획득한 아이템의 수, 게이트 사용 횟수, 게임 시간을 각각 계산한다.
- ◆ 계산한 점수들을 토대로 미션 달성 여부를 확인한다.

1.4.4 클리어 조건

- ◆ 미션을 모두 달성한다.

1.4.5 패배 조건

- ◆ **Snake** 가 진행방향의 반대방향으로 이동한다.
- ◆ **Snake** 가 자신의 몸통에 부딪힌다.
- ◆ **Snake** 가 벽에 부딪힌다.
- ◆ **Snake** 의 몸의 길이가 3 보다 작아진다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

2 개발 내용 및 결과물

2.1 목표

적용단계	내용	적용 여부
1단계	Map의 구현	적용
	Stage의 구현 - 다섯가지 Map의 구현	적용
2단계	Snake 표현 및 조작	적용
	Snake Rule	적용
	키보드 입력 구현	적용
	Head와 Body의 구분	미적용
3단계	Item 요소의 구현	적용
	Grow Item의 구현	적용
	Poison Item의 구현	적용
4단계	Gate 요소의 구현	적용
5단계	점수 요소의 구현	적용
	스코어보드의 구현	적용
	미션의 구현	적용


2.2 개발 내용 및 결과물

2.2.1 개발 내용

1단계. Map의 구현

Stage를 5단계로 설정하여 Stage 별로 각기 다른 형태의 Map을 구현하였다. Map의 사이즈는 40x60이며 각 Map은 정수 배열로 이루어져 있다. 배열의 원소와 대응하는 Map의 요소는 다음과 같다.

배열의 원소	Map의 요소
0	Snake가 움직일 수 있음
1	충돌하면 Snake가 죽는 벽, Gate가 생성될 수 있는 벽
2	Gate가 생성될 수 없는 벽
3	Grow Item
4	Map 테두리

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

6	Poison Item
8	Gate 1, Gate 2와 짝을 이룸
9	Gate 2, Gate 1과 짝을 이룸
h	Snake의 Head
b	Snake의 Body

Snake의 Head와 Body는 초기 Map 배열에는 표현되어 있지 않으나, 정수 배열을 문자 배열로 변환하는 과정에서 추가된다.

2단계. Snake 표현 및 조작

Snake의 좌표를 Vector 클래스로 설정해 vector에 저장한다. Map을 나타낸 문자 배열에서 Snake의 Head는 h, Body는 b로 표현되며 사용자에게는 Head와 Body 모두 #로 출력된다. Snake는 키보드의 WSAD 키와 방향키를 사용해 조종할 수 있는데, ncurses 라이브러리의 wgetch 함수를 사용해 사용자의 키보드 입력을 받는다. keypad 함수를 활성화함으로 키보드의 방향키 입력을 받고 각 문자 입력을 통해 WSAD 키 입력을 받는다. Snake의 진행 방향으로 향하는 키보드 입력은 무시한다. Snake는 진행 방향의 역방향으로 움직일 수 없으며 진행 방향의 역방향 키를 입력하면 Game Over가 된다. 또한 벽에 부딪히면 Game Over가 되며, Snake의 길이를 1만큼 감소시키는 Item을 획득하여 Snake의 길이가 3보다 작을 시에도 Game Over가 된다.

unistd 라이브러리의 usleep 함수를 이용해 Snake의 움직임을 표현한다. usleep의 인자로 Snake의 getSpeed 함수를 포함하는데, Snake 생성자를 통해 초기 speed를 설정하며 getSpeed 함수는 이를 반환한다. 초기 speed는 100000으로 설정하였다.

Head와 Body의 구현은 코드 상에서 이루어지고 있으나 사용자에게 출력될 때에는 모두 #로 표현된다.

3단계. Item 요소의 구현

Map 상에서 Random하게 Grow Item과 Poison Item이 나타난다. Grow Item 획득 시에는 Snake의 길이가 1만큼 증가하고 Poison Item 획득 시에는 Snake의 길이가 1만큼 감소한다. Map 상에서 Grow Item은 3으로, Poison Item은 6으로 표현되고 사용자에게 출력되는 모습은 Grow Item은 0, Poison Item은 x이다. Item은 position 클래스를 통해 Random하게 좌표가 설정되며 vector에 push하고 pop하는 과정을 통해 생성과 소멸을 구현했다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

4단계. Gate 요소의 구현

Random하게 Gate가 생성될 수 있는 벽에 Gate가 나타난다. Gate가 생성될 수 있는 벽은 Map 상에서 1로 표현되며 사용자에게 출력되는 모습은 *이다. Gate가 생성되면 생성된 벽은 각각 8과 9로 바뀌고 사용자에게는 1과 2로 바뀌어 출력된다. 1과 2의 Gate는 서로 연결되어 한 곳으로 진입하면 다른 한 곳으로 진출한다. Snake가 Gate에 진입하면 이어진 Gate의 진출 방향에 따라 Snake의 진행 방향을 바꾼다. 왼쪽 벽에 Gate 1이 생기고 아래쪽 벽에 Gate 2가 생겼고 Snake가 Gate 1에 진입하였다고 가정하면 Gate 2를 통해 진출 시 Snake의 방향은 왼쪽 방향에서 위쪽 방향으로 바뀐다.

5단계. 점수 요소의 구현

터미널 우측 상단에는 스코어보드를, 우측 하단에는 미션 창을 출력한다. 스코어보드는 현재 Snake의 길이와 최대 Snake의 길이, Grow Item과 Poison Item의 획득 횟수, Gate 진입 횟수를 기록한다. 미션 창에는 스코어보드에서 기록 내용에 대한 미션을 출력한다. 사용자가 완수해야 하는 미션은 다음과 같다.

현재 Snake의 길이를 6까지 증가시킨다.
Grow Item을 2회 획득한다.
Poison Item을 2회 획득한다.
Gate에 1회 진입한다.

미션을 모두 달성하면 다음 Stage로 이동할 수 있으며 총 Stage는 5개가 존재한다.

2.2.2 시스템 구조 및 설계도

item.cpp - Snake가 획득할 수 있는 아이템인 Grow item과 Poison item의 정의와 생성과 소멸을 구현한다. Grow item을 만나면 Snake의 길이가 1만큼 증가하고 Poison item을 만나면 Snake의 길이가 1만큼 감소한다.

```
void updategrowth(int stage_num){
    growth_position = vgrow_item.back();
    map[stage_num][growth_position.y][growth_position.x] = 3;
}

void updateposion(int stage_num){
    poison_position = vpoison_item.back();
    map[stage_num][poison_position.y][poison_position.x] = 6;
}
```

각 Item vector에 저장된 Item을 Map 상에 표현하는 기능을 구현하였다. Map 배열 상의 원소만 변경하는 함수이다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

void appearposion(int stage_num, WINDOW *win1){
    nodelay(win1, true);
    while(1){
        if(map[stage_num][poison_position.randomPosition().y][poison_position.randomPosition().x] != 0){
            poison_position.randomPosition();
        }
        else break;
    }
    vpoison_item.push_back(poison_position);
    updateposion(stage_num);
}

void appeargrowth(int stage_num, WINDOW *win1){
    nodelay(win1, true);
    while(1){
        if(map[stage_num][growth_position.randomPosition().y][growth_position.randomPosition().x] != 0){
            growth_position.randomPosition();
        }
        else break;
    }
    vgrow_item.push_back(growth_position);
    updategrowth(stage_num);
}

```

Map 배열에서 Item 이 생성될 수 있는 좌표인지 판단하여 random 한 position 에 Item 이 생성되는 기능을 구현하였다.

```

void disappearPoison(int stage_num, WINDOW *win1){
    nodelay(win1, true);
    if(vpoison_item.empty() == 0 ){
        poison_position = vpoison_item.back();
        map[stage_num][poison_position.y][poison_position.x] = 0;
        vpoison_item.pop_back();
    }
}

void disappeargrowth(int stage_num, WINDOW *win1){
    nodelay(win1, true);
    if(vgrow_item.empty() == 0 ){
        growth_position = vgrow_item.back();
        map[stage_num][growth_position.y][growth_position.x] = 0;
        vgrow_item.pop_back();
    }
}

```

Item 이 소멸되는 기능을 구현하였다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

map.cpp - 40x60 사이즈의 맵 다섯 개를 정수형 삼차원배열에 정의한다.

Position.h - 맵 상의 Random 한 좌표를 표현하기 위해 필요한 것을 정의한다.

```
class position{
public:
    int x;
    int y;
    position();
    position(int tempx, int tempy);
    position& randomPosition();

    position& operator=(const position& v);
    bool operator==(const position v);
};
```

position.cpp - position.h 에서 정의한 것을 구현한다.

```
position::position(){
    x = 0;
    y = 0;
}


position::position(int tempx, int tempy){
    x = tempx;
    y = tempy;
}

position& position::randomPosition(){
    srand(time(NULL));
    this ->x = rand()%60;
    this ->y = rand()%40;
    return *this;
}

position& position::operator=(const position& v){
    this -> x = v.x;
    this -> y = v.y;
    return *this;
}

bool position::operator==(const position v){
    if((x == v.x) && ( y == v.y))
        return true;
    else
        return false;
}
```

생성자를 통해 초기 좌표를 설정하며, srand 를 이용해 난수를 생성해 random 한 좌표를 반환한다.
또한 = 연산과 == 연산을 오버로딩하여 구현함으로 좌표 간 연산을 구현하였다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

snakeGame.h - Snake 의 움직임과 아이템, 게이트, 벽과의 충돌 시 이벤트 발생 시에 필요한 것을 정의한다.

```
const int snakeMaxLen=6;

class Snake
{
private:
    vector<Vector> snake_vec;
    Vector direction;
    const int row, col;
    vector<Vector> wall;
    Vector gate[2];
    bool end;
    int speed;
    char* map_list;
    int gateCnt;
    int level;
    int snakeLen;

public:
    Snake(int r, int c);
    ~Snake();
    int growthItem=0;
    int poisonItem=0;
}
```

Snake Game 을 진행하기 위한 변수 및 생성자, 소멸자를 선언한다. Snake 의 Max Length 를 6 으로 설정하였고 Snake 의 좌표를 저장하기 위한 Vector 형 vector 와 Snake 의 진행 방향을 나타내기 위한 direction 을 선언하였다. Wall 의 좌표를 저장하기 위한 Vector 형 vector 과 Gate 는 한 번에 한쌍, 즉 두개 밖에 출현할 수 없으므로 2 개의 좌표만 저장할 수 있는 배열을 선언한다. 이 외 Snake Game 을 진행하기 위해 필요한 변수를 선언하고 생성자와 소멸자, 초기 Item 개수를 선언한다.

```
void setDirection(int d);
void moveSnakeHead(int map[40][60]);
void moveSnakeBody();
char* setMaptoList(int map[40][60]);
int gateDirection(Vector gate, int map[40][60]);
char getDirection();
void setEnd(bool e);
bool getEnd();
int getSpeed();
int getRow();
int getCol();

void setGate(int map[40][60]);
void removeGate(int map[40][60]);
void setGateCnt(int i);
int getGateCnt();
int getLevel();
int getSnakeLen();

int getSize();
void setLevel(int new_level);
void minusSnake(WINDOW *win1);
void crushItem(WINDOW *win1);
position getHead();
void resize(int new_size);
void changeSnakeLen();
```

Snake Game 을 진행하기 위한 각종 함수를 선언하였다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

SnakeGame.cpp – Snake 가 가는 방향으로 Head 가 증가하게 만들고 Snake 가 Gate 에 진입하면 이어진 Gate 로 진출할 수 있게 Snake 의 방향을 바꾼다. Gate 가 아닌 벽을 만났을 경우는 exit 을 true 로 변경하고 벽에 부딪혔다는 판정 후 Game Over 문구를 출력한다. Head 에 따라 body 도 함께 변경한다. 2 차원 배열을 리스트로 변경하고 int 를 char 값으로 변경한 다음 1 차원 배열로 변경한다. Gate 가 생길 수 있는 벽일 경우 wall 에 벡터에 좌표 값을 추가한다.

```
Snake::Snake(int r, int c) : row(r), col(c)
{
    for(int i=0; i<3; i++)
        snake_vec.push_back(Vector(col/2, row/2+i));
    setDirection(0);
    end = false;
    speed = 100000;
    map_list = new char[row*col];
    level=1;
    snakeLen=3;
    setGateCnt(0);
}

Snake::~Snake(){ delete [] map_list; }
```

Snake 생성자와 소멸자이다. 초기 Snake 의 길이를 3 으로 하여 Snake vector 에 좌표를 저장한다. setDirection 함수를 통해 초기 방향을 윗방향으로 설정하고 게임의 종료 여부를 설정하는 end 를 false 로 설정하며 초기 speed 를 100000 으로 설정한다. map_list 는 각 stage 별 map 을 game 에 맞게 변환하여 저장하기 위한 배열이다. 초기 레벨과 Gate 의 수를 설정한다.

```
int Snake::gateDirection(Vector gate, int map[40][60]){
    Vector A = gate + Vector(0,-1) ;
    if(map[A.getY()][A.getX()]==0) return 0;
    Vector B = gate + Vector(1,0) ;
    if(map[B.getY()][B.getX()]==0) return 1;
    Vector C = gate + Vector(0,1) ;
    if(map[C.getY()][C.getX()]==0) return 2;
    Vector D = gate + Vector(-1,0) ;
    if(map[D.getY()][D.getX()]==0) return 3;

    return -1;
}
```

Snake 가 Gate 를 통과하고 난 후의 방향 전환을 구현한다. 하 방향 Gate 로 진출하면 상 방향으로, 우 방향 Gate 로 진출하면 좌 방향으로, 상 방향 Gate 로 진출하면 하 방향으로, 좌 방향 Gate 로 진출하면 우 방향으로 Snake 의 진행 방향을 반환한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

void Snake::moveSnakeHead(int map[40][60]){
    snake_vec[0] += direction;
    for(int i=0; i<wall.size(); i++) {
        if(snake_vec[0] == wall[i]) {
            if(snake_vec[0] == gate[1]) {
                snake_vec[0].setX(gate[0].getX());
                snake_vec[0].setY(gate[0].getY());
                setDirection(gateDirection(gate[0], map));
                setGateCnt(1);
                break;
            }
            else if(snake_vec[0] == gate[0]) {
                snake_vec[0].setX(gate[1].getX());
                snake_vec[0].setY(gate[1].getY());
                setDirection(gateDirection(gate[1], map));
                setGateCnt(1);
                break;
            }
            else {
                setEnd(true);
                removeGate(map);
                whyDead = "You can't go through the wall!";
            }
        }
    }
}

void Snake::moveSnakeBody(){
    for(unsigned int i=snake_vec.size()-1; i>0; --i) snake_vec[i] = snake_vec[i - 1];
}

```

Snake 의 Head 와 Body 의 움직임을 구현한다. Head 가 Wall 과 부딪혔을 때 Wall 이 Gate 이면 Gate 에 진입하고 진출하는 것이라 판정하고, Gate 가 아니라면 Game Over 이라 판정한다.



```
char* Snake::setMaptoList(int map[40][60]){
    memset(map_list, ' ', row*col);
    for(unsigned int i=0; i<40; i++) {
        for(int j=0; j<60; j++){
            switch(map[i][j]) {
                case 0 : map_list[i*col+j] = '0'; break;
                case 1 :
                    map_list[i*col+j] = '1';
                    wall.push_back(Vector(j,i));
                    break;
                case 2 : map_list[i*col+j] = '2'; break;
                case 3 : map_list[i*col+j] = '3'; break;
                case 4 : map_list[i*col+j] = '4'; break;
                case 6 : map_list[i*col+j] = '6'; break;
                case 98 : map_list[i*col+j] = '8'; break;
                case 99 : map_list[i*col+j] = '9'; break;
            }
        }
    }

    map_list[snake_vec[0].getY()*col+snake_vec[0].getX()] = 'h';
    for(unsigned int i=1; i<snake_vec.size(); ++i)
        map_list[snake_vec[i].getY()*col+snake_vec[i].getX()] = 'b';
    return map_list;
}
```

정수형 배열 Map 을 Game 에서 이용할 수 있게 문자형 배열로 변환하는 기능을 구현하였다. map을 인자로 받아 배열을 탐색하여 원소에 따라 각 Map 의 요소로 변환한다. 배열의 각 원소에 대응하는 Map 의 요소는 본 문서 2.2.1 항의 1 단계. Map 의 구현에서 확인할 수 있다.

```
void Snake::setEnd(bool e) {end = e;}
bool Snake::getEnd() {return end;}
int Snake::getSpeed() {return speed;}
int Snake::getRow() {return row;}
int Snake::getCol() {return col;}
int Snake::getLevel() {return level;}
int Snake::getSnakeLen(){return snakeLen;}

char Snake::getDirection() {
    if(direction.getX()==1) return 'r';
    else if(direction.getX()==-1) return 'l';
    else if(direction.getY()==-1) return 'u';
    else return 'd'; //아랫쪽
}
```

Snake 의 진행 방향을 설정하는 함수와 Game 의 종료 여부, Snake 의 Speed, Map 의 행과 열의 수, Game 의 레벨, Snake 의 길이를 반환하는 함수를 구현하였다.



```
void Snake::setGate(int map[40][60]) {
    int randWall = rand() % wall.size();
    int randWall2 = rand() % wall.size();
    if(randWall == randWall2) setGate(map);
    gate[0] = wall[randWall];
    gate[1] = wall[randWall2];
    map[gate[0].getY()][gate[0].getX()] = 98;
    map[gate[1].getY()][gate[1].getX()] = 99;
}

void Snake::removeGate(int map[40][60]) {
    map[gate[0].getY()][gate[0].getX()] = 1;
    map[gate[1].getY()][gate[1].getX()] = 1;
    gate[0].setX(0);
    gate[0].setY(0);
    gate[1].setX(0);
    gate[1].setY(0);
}
```

Gate 를 Map 상에 생성하고 소멸하는 기능을 구현하였다. Wall 중 Random 하게 두 개의 Wall 을 골라 Gate 로 설정하여 생성하고, 다시 Gate 를 Wall 로 설정함으로 Gate 를 소멸한다.

```
void Snake::setGateCnt(int i) {
    if (i==0){
        gateCnt = 0;
    }
    else{
        gateCnt += 1;
    }
}

int Snake::getGateCnt() {return gateCnt;}
```

Gate 의 수를 설정하고 반환하는 기능을 구현하였다.

```
int Snake::getSize(){return snake_vec.size();}

void Snake::setLevel(int new_level){
    level = new_level;
}
```

Snake 의 현재 사이즈를 반환하는 함수와 현재 Level 을 설정하는 함수를 구현하였다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

void Snake::minusSnake(WINDOW *win1){
    disappearPoison(level-1,win1);
    snake_vec.pop_back();
    appearposion(level-1,win1);
}

void Snake::crushItem(WINDOW *win1){
    disappeargrowth(level -1,win1);
    snake_vec.push_back(snake_vec.back());
    appeargrowth(level-1,win1);
}

```

Snake 가 획득할 수 있는 Item 인 Grow Item 과 Poison Item 이 사라짐에 따라 다른 위치에 Item 을 생성하는 기능을 구현하였다.

```


position Snake::getHead(){
    position head(snake_vec[0].getX(),snake_vec[0].getY());
    return head;
}

void Snake::resize(int new_size){
    snake_vec.resize(new_size);
}

void Snake::changeSnakeLen(){snakeLen = snake_vec.size();}

```

Snake 의 Head 좌표를 반환하는 함수와 Snake 의 사이즈를 재설정하는 함수, Snake 의 길이를 현재 Snake vector 의 사이즈로 설정하는 함수를 구현하였다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

Vector.h - 맵 상의 좌표를 표현하기 위해 필요한 것을 정의한다.

```

class Vector{
protected:
    int x,y;
public:
    Vector();
    Vector(int x, int y);

    void setX(int x);
    void setY(int y);
    int getX() const;
    int getY() const;

    Vector& operator=(const Vector& v);
    Vector& operator+=(const Vector& v);
    Vector operator+(const Vector& v);
    Vector& operator-=(const Vector& v);
    Vector operator-(const Vector& v);
    bool operator==(const Vector& v);
    bool operator!=(const Vector& v);
};

```

맵 상의 좌표를 표현하기 위하여 X 좌표와 Y 좌표를 설정하고 각종 연산자를 오버로딩하기 위해 선언한다.

Vector.cpp - vector.h 에서 정의한 것을 구현한다.

```

Vector::Vector() : x(0), y(0){}
Vector::Vector(int x, int y) {setX(x); setY(y);}

void Vector::setX(int x){this->x = x;}
void Vector::setY(int y){this->y = y;}
int Vector::getX() const {return this->x;}
int Vector::getY() const {return this->y;}

```

초기 좌표를 설정하는 생성자와 각 X 좌표, Y 좌표를 설정 및 반환하는 함수를 구현하였다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

Vector& Vector::operator=(const Vector& v){
    setX(v.getX());
    setY(v.getY());
    return *this;
}
Vector& Vector::operator+=(const Vector& v){
    setX(getX()+v.getX());
    setY(getY()+v.getY());
    return *this;
}
Vector Vector::operator+(const Vector& v){
    Vector v2(getX(),getY());
    return v2+=v;
}

Vector& Vector::operator-=(const Vector& v){
    setX(getX()-v.getX());
    setY(getY()-v.getY());
    return *this;
}
Vector Vector::operator-(const Vector& v){
    Vector v2(getX(),getY());
    return v2-=v;
}
bool Vector::operator==(const Vector& v){
    return getX()==v.getX() && getY()==v.getY();
}
bool Vector::operator!=(const Vector& v){
    return !operator==(v);
}

```

좌표 연산에 필요한 각종 연산자를 오버로딩하여 구현하였다.

Stage.cpp - 게임을 진행하기 위한 윈도우 관련 메소드, 게임 진행 로직을 구현한다. 미션이 성공했는지 여부를 표시해주는 캐릭터를 표시한다. 왜 Game Over 가 되었는지 알려주는 문자열도 추가한다. 미션 기준도 추가한다. 새 윈도우를 만들고 인풋을 받고 게임을 시작하고 게임을 재시작하는 메소드도 추가한다. Map 상에서 Snake 가 움직일 수 있는 공간을 0 으로 설정하고 Gate 가 생길 수 있는 벽을 1 로 설정하며 Gate 가 생길 수 없는 벽을 2 로 설정한다. 그리고 맵 테두리는 @으로 설정한다. Snake 의 길이가 3 보다 작으면 Game Over 로 판정한다.

```

char missionB = 'X';
char missionGrowth = 'X';
char missionPoison = 'X';
char missionGate = 'X';
string whyDead = "";

int num_missionB = 6;
int num_missionGrowth = 2;
int num_missionPoison = 2;
int num_missionGate = 1;

```

미션 달성 여부와 달성해야 하는 수치를 설정하였다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

void newWindow(float y, float x){
    clear();
    initscr();
    noecho();
    getmaxyx(stdscr, y, x);
}

int UserInput(){
    int UserInput = getch();
    refresh();
    endwin();
    clear();
    return UserInput;
}

```

새로운 윈도우를 생성하고 사용자의 키보드 입력을 받기 위한 함수이다. 새로운 윈도우를 생성할 때에는 기존 윈도우를 모두 clear 하고 사용자가 입력한 것을 출력하지 않는다.

```

int startGame(float y, float x) {
    newWindow(y,x);
    printw("Do you want to start snake game? (y/n)");
    return UserInput();
}

int finishWindow(float y, float x){
    newWindow(y,x);
    printw(whyDead.data());
    printw("\n \nYou Dead! Press any button to finish~");
    return UserInput();
}

```

게임을 시작하고 끝낼 때에 출력되는 윈도우이다. 처음 게임을 시작하면 "Do you want to start snake game? (y/n)"이라는 문구가 출력되고 UserInput 함수를 실행해 사용자의 입력을 받는다. Game Over 되면 왜 Game Over 되었는지에 대한 문구와 "You Dead! Press any button to finish~"이라는 문구가 출력되고 UserInput 함수를 실행해 사용자의 입력을 받는다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

void printScoreBoard(WINDOW* w, int snakeLen, int level, int growthItem, int poisonItem, int Gate){
    werase(w);
    wbkgd(w, COLOR_PAIR(level));
    wborder(w, '|', '|', '~', '~', '.', '.', '.', '.');
    mvwprintw(w, 1, 1, "Score Board");
    mvwprintw(w, 2, 1, "B:(CurrentLength)/(Max Length) %d/%d", snakeLen, snakeMaxLen);
    mvwprintw(w, 3, 1, "+:(GrowthItems Num): %d ", growthItem);
    mvwprintw(w, 4, 1, "-:(PoisonItems Num): %d ", poisonItem);
    mvwprintw(w, 5, 1, "G(Number of Entered Gate) : %d ", Gate);
    mvwprintw(w, 6, 1, "Level : %d ", level);
    wrefresh(w);
}

```

스코어보드를 출력하는 기능을 구현하였다. 스코어보드에 포함되는 내용은 본 문서 2.2.1 항의 5 단계. 점수 요소 구현에서 확인할 수 있다.

```

void printMission(WINDOW* w, int level){
    werase(w);
    wbkgd(w, COLOR_PAIR(level));
    wborder(w, '|', '|', '~', '~', '.', '.', '.', '.');
    mvwprintw(w, 1, 1, "Mission");
    mvwprintw(w, 2, 1, "B: 6 ( %c )", missionB);
    mvwprintw(w, 3, 1, "+: 2 ( %c )", missionGrowth);
    mvwprintw(w, 4, 1, "-: 2 ( %c )", missionPoison);
    mvwprintw(w, 5, 1, "G: 1 ( %c )", missionGate);
    wrefresh(w);
}

```

미션을 출력하는 기능을 구현하였다. 미션에 포함되는 내용은 본 문서 2.2.1 항의 5 단계. 점수 요소 구현에서 확인할 수 있다.



```
int noticeChangeLevel(float y, float x, int level){
    clear();
    initscr();
    noecho();

    getmaxyx(stdscr, y, x);
    newWindow(y,x);
    if (level==1){
        printw("Let's start Game! \n Level 1");
    }
    else{
        string lev = to_string(level);
        printw("Let's go to next level! \n");
        printw(lev.data());
        printw("\nPress Enter button~!");
    }
    return UserInput();
}
```

현재 레벨을 사용자에게 알려주고 Stage 클리어 시 다음 레벨로 넘어가 진행할 수 있는 기능을 구현하였다. 현재 레벨이 1 인지 아닌지를 판단하여 1 이 아니면 다음 레벨로 넘어가기 위하여 사용자의 입력을 받는다.

```
void nextLevel(Snake& snake, WINDOW *win1){
    if((missionB == 'O')&&(missionGate=='O')&&(missionGrowth=='O')&&(missionPoison=='O')){
        snake.resize(3);
        snake.growthItem =0;
        snake.poisonItem =0;
        snake.setGateCnt(0);
        missionB = 'X';
        missionGrowth = 'X';
        missionPoison = 'X';
        missionGate = 'X';
        disappeargrowth(snake.getLevel()-1,win1);
        disappearPoison(snake.getLevel()-1,win1);
        snake.setLevel(snake.getLevel()+1);
        if (noticeChangeLevel(0,0, snake.getLevel()) == 13) {};
    }
}
```

다음 레벨로 넘어간 후 윈도우를 다시 초기 상태로 설정하는 기능을 구현하였다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

void setMission(Snake& snake, WINDOW *win1){
    if(vgrow_item.empty() ==0){
        position head = snake.getHead();
        if(head == vgrow_item.back()){
            snake.crushItem(win1);
            snake.changeSnakeLen();
            snake.growthItem++;
        }
    }
    if(vpoison_item.empty() ==0){
        position head = snake.getHead();
        if(head == vpoison_item.back()){
            snake.minusSnake(win1);
            snake.changeSnakeLen();
            snake.poisonItem++;
        }
    }
    if(snake.getSize() == num_missionB) {missionB = '0';}
    if(snake.growthItem == num_missionGrowth) {missionGrowth = '0';}
    if(snake.poisonItem == num_missionPoison) {missionPoison = '0';}
    if(snake.getGateCnt() == num_missionGate) {missionGate = '0';}
}

```

미션을 설정하는 함수를 구현하였다. Snake 의 Item 획득 여부를 판단하며, 미션 달성 여부를 판단하여 미션이 달성되었으면 미션 달성을 사용자에게 알린다.

```

start_color();
init_pair(1, COLOR_WHITE, COLOR_CYAN);
init_pair(2, COLOR_WHITE, COLOR_GREEN);
init_pair(3, COLOR_BLACK, COLOR_MAGENTA);
init_pair(4, COLOR_BLACK, COLOR_YELLOW);
init_pair(5, COLOR_BLACK, COLOR_BLUE);

getmaxyx(stdscr, y, x);
WINDOW *win1 = newwin(40, 60, 0, 0);
WINDOW *scoreBoard = newwin(10, 40, 0,60);
wrefresh(scoreBoard);

WINDOW *mission = newwin(10, 40, 12,60);
wrefresh(mission);

```

각 Stage 별 Map 의 Color 를 설정한다. 게임이 진행될 윈도우와 스코어보드, 미션이 출력될 윈도우를 생성한다.



```

switch(input)
{
    case 'w':
    case KEY_UP:
        if(d!='u' && d!='d') snake.setDirection(0);
        else if (d=='d') {
            snake.setEnd(true);
            whyDead = "You've moved in the tail direction! You must move towards head.";
        }

        break;
    case 's':
    case KEY_DOWN:
        if(d!='d' && d!='u') snake.setDirection(2);
        else if (d=='u') {
            snake.setEnd(true);
            whyDead = "You've moved in the tail direction! You must move towards head.";
        }

        break;
    case 'a':
    case KEY_LEFT:
        if(d!='l' && d!='r') snake.setDirection(3);
        else if (d=='r') {
            snake.setEnd(true);
            whyDead = "You've moved in the tail direction! You must move towards head.";
        }

        break;
    case 'd':
    case KEY_RIGHT:
        if(d!='r' && d!='l') snake.setDirection(1);
        else if (d=='l') {
            snake.setEnd(true);
            whyDead = "You've moved in the tail direction! You must move towards head.";
        }

        break;
    case 'r' :
    case 'R' :
        snake.setEnd(true);
        snake.removeGate(map[snake.getLevel()]);
        disappeargrowth(snake.getLevel()-1,win1);
        disappearPoison(snake.getLevel()-1,win1);
        game();
}

```

사용자의 키보드 입력을 받아 Snake 를 조종하는 기능을 구현하였다. Snake 를 조종하는 키 대응은 본 문서 2.2.3 항 ncurses 의 세번째 문단에서 확인할 수 있다. Snake 의 진행 방향을 설정하며 진행 방향의 역방향키를 입력하면 Game Over 이라 판정한다. 추가로 r 또는 R 키를 입력하면 윈도우가 초기화되며 게임을 재시작할 수 있다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

```

        if(snake.getSize() <3) {
            snake.setEnd(TRUE);
            whyDead = "Snake length is less than 3!";
        }

        snake.moveSnakeBody();
        snake.moveSnakeHead(map[snake.getLevel()-1]);
        nextLevel(snake, win1);
        usleep(snake.getSpeed());
    }

```

Snake 의 길이가 3 보다 작으면 Game Over 라 판정하며 Snake 의 움직임을 구현하였다. usleep 함수의 인자가 되는 getSpeed 함수는 Snake 의 Speed 를 반환하며 반환된 Speed 만큼 호출 프로세스의 실행을 중지한다. 따라서 Snake 의 Speed 를 작게 설정하면 사용자가 느끼는 Speed 의 속도는 현저히 증가한다.

Makefile - 프로그램을 실행시키기 위해 필요한 코드를 편리성을 위해 한 번에 컴파일 및 실행한다.


2.2.3 활용/개발된 기술

ncurses

본 프로젝트에서는 Snake Game을 진행하기 위한 윈도우를 사용자에게 출력하며 사용자의 키보드 입력을 바탕으로 게임 진행, Snake의 움직임을 제어하기 위해 ncurses 라이브러리를 사용한다. 게임을 처음 실행하면 게임을 실행할 것인지 사용자에게 묻는 문구 출력, 스테이지를 클리어할 때마다 다음 스테이지로 넘어가기 위한 문구 출력, 스테이지를 클리어하지 못했을 시의 문구 출력을 위해 사용한다.

map을 나타내기 위한 윈도우, 스코어보드를 나타내기 위한 윈도우, 미션 창을 나타내기 위한 윈도우를 생성한다. mvwprintw, mvwaddch 와 같은 함수를 사용하여 배열로 이루어진 map을 각 원소의 요소 별로 Snake, Wall, Item, Gate 여부를 판단하여 사용자가 보는 윈도우 상에 나타내며, Snake의 길이, Item 획득 횟수, Gate 진입 횟수, 레벨을 나타내는 스코어보드와 미션을 윈도우 상에 나타낸다.

wgetch 함수를 사용해 사용자의 키보드 입력을 받아 이를 바탕으로 Snake를 조종한다. 키보드의 WSAD 키 또는 방향키로 Snake를 조종할 수 있으며 W키와 상방향키, S키와 하방향키, A키와 좌방향키, D키와 우방향키가 서로 대응된다. 방향키 입력은 keypad 함수를 활성화하여 ncurses 라이브러리에서 미리 정의해놓은 키보드 입력 상수를 이용해 처리하며 WSAD 키 입력은 각 문자 입력으로 처리한다. 이 외에도 ncurses 모드를 시작하기 위한 initscr 함수, ncurses 모드를 종료하기 위한 endwin 함수, 사용자로부터 입력 받은 문자를 출력하지 않기 위한 noecho 함수, ncurses 모드의 출력을 갱신하기 위한 refresh 함수 등을 사용하여 Snake Game을 구현한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

vector

본 프로젝트에서는 가변적인 Snake의 길이, map에 표현되는 Item을 표현하기 위해 vector 라이브러리를 사용한다. Snake 생성자를 통해 초기 길이를 3으로 설정하고 Item을 획득함에 따라 달라지는 Snake의 길이와 시간이 지남에 따라 나타나고 사라지는 Item을 표현하기 위해 동적으로 확장/축소가 가능하고 요소에 대한 접근이 쉬운 vector를 사용하여 Snake Game을 구현한다.

unistd

본 프로젝트에서는 unistd 라이브러리의 마이크로 간격으로 호출 프로세스의 실행을 일시 중지하는 usleep 함수를 사용하여 Snake의 움직임을 표현한다. usleep의 인자로 Snake의 getSpeed 함수를 포함하는데, Snake 생성자를 통해 초기 speed를 설정하며 getSpeed 함수는 이를 반환한다. 초기 speed는 100000으로 설정하였다.

Vector


본 프로젝트에서는 Vector 클래스를 선언하여 map의 좌표를 다루고 있다. map 상의 x좌표, y좌표를 설정하고 =, +=, -=, +, -, ==, != 연산자를 오버로딩하여 Snake의 방향 설정, Gate의 진입 및 진출 방향 설정 등을 표현한다.

position

본 프로젝트에서는 position 클래스를 선언하여 map의 random한 좌표를 다루고 있다. Item은 map 상의 random한 좌표에서 등장하게 되는데 =, == 연산자를 오버로딩하여 이를 표현한다.

2.2.4 현실적 제한 요소 및 그 해결 방안

개발을 완료하고 Game 을 진행해보니 생각보다 미션을 달성하기가 쉽지 않았다. 특히 Snake 의 속도를 조절하기가 어려워서 1 레벨을 달성하기가 어려울 수 있을 것 같았다. 따라서 속도를 조절하는 대신 레벨 별 수행 미션을 낮게 잡아서 최대한 이용자가 간편하게 이용할 수 있게 하였다.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

2.2.5 결과물 목록

파일명	역할
item.cpp	Snake 가 획득할 수 있는 아이템인 grow item 과 poison item 의 정의와 생성과 소멸을 구현한다.
map.cpp	40x60 사이즈의 맵 다섯 개를 정의한다.
position.h	맵 상의 Random 한 좌표를 표현하기 위해 필요한 것을 정의한다.
position.cpp	position.h 에서 정의한 것을 구현한다.
snakeGame.h	Snake 의 움직임과 아이템, 게이트, 벽과의 충돌 시 이벤트 발생 시에 필요한 것을 정의한다.
snakeGame.cpp	snakeGame.h 에서 정의한 것을 구현한다.
stage.cpp	게임을 진행하기 위한 윈도우 관련 메소드, 게임 진행 로직을 구현한다.
vector.h	맵 상의 좌표를 표현하기 위해 필요한 것을 정의한다.
vector.cpp	vector.h 에서 정의한 것을 구현한다.
makefile	프로그램을 실행시키기 위해 필요한 코드를 편리성을 위해 한번에 컴파일 및 실행한다.

3 시연영상

https://youtu.be/i1wlwtEv4rE https://youtu.be/ySrJOY4cv5s
--

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

4 자기평가

팀원	자기평가
박창용	본 프로젝트에서 ncurses 라이브러리를 기반으로 한 프레임워크 개발 및 뱀 게임 일부 구현을 수행하였다. 프레임워크 개발하는 것에 대부분의 시간을 사용하였고, 정작 뱀 게임 구현은 많이 진행하지 못했기 때문에 아쉬웠다. ncurses 라이브러리를 프레임워크로 래핑하면서 비동기에 관한 이슈 등이 있었는데, 이러한 이슈들을 해결하는 것이 가장 어려웠다. 그러나 좋은 실력의 팀원을 만난 덕분에, 작업에 집중할 수 있었고 개발을 잘 완료할 수 있었다.
안시현	본 프로젝트에서 전반적인 코드 구현을 하였다. 코드 구현이 생각보다 쉽지 않았지만 코드를 잘 구현하는 선배에게 조언을 구해서 잘 마칠 수 있었다. 우분투를 이용해서 프로젝트를 구현한 것은 처음이었는데 서툴렀던 점을 기억하며 방학 때 개인프로젝트를 해봐야겠다고 생각했다.
이정훈	본 프로젝트에서 팀원 간 연락 수단 마련, 일정 관리를 수행하였다. 게임 진행을 위한 코드 대부분을 안시현 학우가 작성하였고 이를 베이스로 하여 개발을 진행하였기 때문에 본인의 개발 기여도는 낮다고 평가한다. 프로젝트를 진행하면서 ncurses 라이브러리 사용법을 습득하여 코드로 표현하는 것이 가장 어려웠으며, 이번 학기에 캡스톤 디자인과 병행하며 프로젝트를 수행해야 했기에 부담감이 컸지만 좋은 팀원을 만나 프로젝트를 잘 끝마칠 수 있었다. 기회가 된다면 게임 서버를 구현해 랭킹 시스템이나 시중의 타 게임(slither.io 등)과 같이 온라인에서 즐길 수 있도록 개발해보고 싶다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

5 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
01	WEB	ncurses	Wikipedia			
02	WEB	NCURSES Programming HOWTO	KLDP Wiki		gsong	원문 번역
03	WEB	[Linux] NCURSES 프로그래밍	TISTORY	2013	투명잉크	블로그 발췌
04	DOC	Ncurses 기반 텍스트 사용자 인터페이스	국민대학교 소프트웨어 학부		김준호 교수	

6 부록

6.1 Github Repository

<https://github.com/yellowoov/SnakeGame>

6.2 설치 방법

6.2.1 설치 방법 1 - git clone

1. 게임을 설치하고자 하는 디렉토리로 이동하기 위해 터미널에 다음과 같이 입력한다.

```
cd 게임을 설치하고자 하는 디렉토리
ex) cd Desktop
```

2. 다음과 같은 명령어를 입력한다.

```
git clone https://github.com/yellowoov/SnakeGame
```

6.2.2 설치 방법 2 – Download ZIP

1. 다음과 같은 주소를 인터넷 브라우저 주소창에 입력한다.

<https://github.com/yellowoov/SnakeGame/archive/refs/heads/main.zip>

2. 다운로드가 완료되면 파일을 게임을 설치하고자 하는 디렉토리에 압축 해제한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	6조	
	Confidential Restricted	Version 1.4	2022-JUN-16

6.3 사용자 매뉴얼

1. 설치방법을 따라 설치를 완료한 후 터미널에 다음과 같이 입력한다.

```
cd 게임을 설치한 디렉토리
ex) cd Desktop/SnakeGame
```

2. make 명령어로 컴파일한다.

```
make run
```

3. 실행 파일을 실행한다.

```
./game
```