



국민대학교  
소프트웨어융합대학  
소프트웨어학부


# C++프로그래밍 프로젝트

프로젝트 명	<i>Snake Game</i>
팀 명	18조
문서 제목	결과보고서

Version	2.1
Date	2023-06-19

팀원	김유진 (팀장)
	정성엽
	정성윤

CONFIDENTIALITY/SECURITY WARNING


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Snake Game”를 수행하는 팀 “18조”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “18조”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

## 문서 정보 / 수정 내역


Filename	최종보고서-snakeGame.doc
원안작성자	김유진, 정성엽, 정성윤
수정작업자	김유진, 정성엽, 정성윤

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2023-06-14	정성엽	1.0	최초 작성	개발 개요와 목표 작성
2023-06-15	정성윤	1.1	내용 추가	개발 내용, 활용 개발된 기술 작성
2023-06-16	김유진	1.2	내용 추가	시스템 구조 및 설계도 작성 1, 결과물 목록 작성
2023-06-17	정성엽	1.3	내용 추가	시스템 구조 및 설계도 작성 2, 매뉴얼 및 설치 방법 작성
2023-06-18	정성윤	2.0	내용 추가 및 수정	부족한 내용 추가 및 수정
2023-06-19	김유진	2.1	내용 추가 및 수정	부족한 내용 추가 및 수정

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 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	10
2.2.3	활용/개발된 기술	36
2.2.4	현실적 제한 요소 및 그 해결 방안	38
2.2.5	결과물 목록	39
3	자기평가	40
4	참고 문헌	42
5	부록	43
5.1	사용자 매뉴얼	43
5.2	설치 방법	44

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

# 1 개요

먼저 프로젝트를 개발하기 위해 사용한 언어는 C++와 Ncurses입니다. 총 3개의 파일로 구성하였고 각 파일에 대한 설명은 다음과 같습니다.

## 1. snakeGame.h

snakeGame에 필요한 변수들과 메소드들을 선언했습니다. snakeGame에 사용된 라이브러리는 다음과 같습니다.

```
#include <iostream>
#include <vector>
#include <ncurses.h>
#include <cstdlib>
#include <ctime>
```

이 때 ncurses.h는 외부 라이브러리가기 때문에 따로 설치를 해야 합니다. 먼저 우분투 터미널에 접속합니다. 그 후

Sudo apt-get update

를 통해 업데이트를 진행한 후

Sudo apt-get install libncurses5-dev libncursesw5-dev


를 통해 두 개의 패키지를 설치합니다. 그러면 이제 nurses.h 라이브러리를 사용할 수 있습니다.

## 2. snakeGame.cpp

snakeGame.cpp에서는 snakeGame.h 헤더파일을 참조하여 헤더파일에 선언된 변수들과 메소드들을 정의했습니다. 그리고 컴파일에 필요한 라이브러리인 <unistd.h>도 사용하였습니다.

## 3. Main.cpp

Main.cpp에서는 snakeGame.h 헤더파일을 참조해서 snakeGame이 실행될 수 있도록 구성했습니다. snakeGame.cpp와 마찬가지로 컴파일을 위해 unistd.h 라이브러리도 사용했습니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 2 개발 내용 및 결과물

### 2.1 목표

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

#### 1단계 : Map의 구현


- 사용자의 터미널의 크기에 맞게 화면을 조정하도록 한다.
- 외벽은 하얀색으로 설정한다.
- 각 level마다 Wall의 위치가 바뀌도록 구성한다.

#### 2단계 : Snake 표현 및 조작

- 방향키는 키보드의 어떤 키로 입력할 지 직접 정한다.
- Head 방향 이동은 무시한다.
- Tail 방향으로 이동시 실패한다.
- Snake는 자신의 Body를 통과할 수 없다.
- Snake는 벽을 통과할 수 없다.
- Head 방향 이동은 일정시간에 의해 이동한다.
- 몸의 길이가 3보다 작아지면 실패다.

#### 3단계 : Item 요소의 구현

- Snake의 Head 위치와 Item의 위치가 같은 경우 Item을 획득한다.
- Growth Item의 경우 몸의 길이가 1 증가한다.
- Poison Item의 경우 몸의 길이가 1 감소한다.
- speed Item의 경우 Snake의 이동 속도가 증가한다.
- Item의 경우 Snake Body가 있지 않은 임의의 위치에 출현하도록 한다.
- 출현 후 일정시간이 지나면 사라지고 다른 위치에 나타나야 한다.
- 동시에 출현할 수 있는 Item의 개수는 3개로 제한한다.
- Growth Item을 획득하면 몸의 길이는 진행방향으로 증가한다.
- Poison Item을 획득하면 꼬리 부분이 1 감소한다.


 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

#### 4단계 : Gate 요소의 구현

- Gate는 두 개가 한 쌍이다.
- Gate는 겹치지 않는다.
- Gate는 임의의 위치에 있는 벽에서 나타난다.
- Gate에 Snake가 진입하면, 다른 Gate로 진출한다.
- Gate에 Snake가 진입중인 경우 Gate는 사라지지 않고, 다른 위치에서 Gate가 나타나지 않는다.
- Gate는 한 번에 한쌍만 나타난다.
- Gate가 나타나는 벽이 가장자리에 있을 때 항상 Map의 안쪽 방향으로 진출한다.
- Gate가 나타나는 벽이 Map의 가운데 있을 때 다음의 순서로 진출한다.
  - (1) 진입 방향과 일치하는 방향이 우선
  - (2) 진입 방향의 시계방향으로 회전하는 방향
  - (3) 진입 방향의 역시계방향으로 회전하는 방향
  - (4) 진입 방향과 반대방향

#### 5단계 : 점수 요소의 구현

- 먼저 Score Board에는 다음의 내용을 포함하도록 한다.
  - (1) Snake 현재 길이
  - (2) 먹은 Growth Item 개수
  - (3) 먹은 Poison Item 개수
  - (4) 먹은 Speed Item 개수
  - (5) 통과한 Gate 개수
- 다음으로 Mission에는 다음 스테이지로 넘어가기 위한 목표가 있고 그 목표를 달성하면 화면에 'v'로 체크 표시가 된다.
- Mission을 모두 완료하면 다음 스테이지로 넘어간다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 2.2 개발 내용 및 결과물


### 2.2.1 개발 내용

#### 1단계 : Map의 구현

- `setWindow()` 함수는 터미널 창 설정을 하는 함수입니다. `ncurses`의 내장함수인 `initscr()`과 `noehco()`를 사용하여 터미널을 초기화하고, 사용자 입력을 터미널 화면에 나타나지 않게 합니다.
- `renderWindow()` 함수를 이용해 게임 레벨에 따른 **stage** 화면을 디자인합니다. 총 4개의 **level**이 있고 각 **stage** 화면은 다르게 디자인 됩니다. 외벽은 하얀색으로 구성하였습니다.
- 터미널의 크기를 받아 화면을 구성하기 때문에 사용자의 터미널 크기에 따라 가변적으로 디자인이 변합니다.
- 커서를 숨기기 위해서 `curs_set = 0`으로 설정합니다.

#### 2단계 : Snake 표현 및 조작

- `renderSnake()` 함수를 이용해 초기의 **snake**의 모양을 구성합니다. 총 길이 3의 **snake**를 만듭니다. **head**와 **tail** 부분이 존재하고 이에 대한 정보는 `charPosition()` 구조에 저장됩니다. `move()`를 이용해서 커서를 해당 **snake**의 위치를 변경하고, `addch`를 사용하여 **snakeShape**를 문자를 해당 위치에 출력합니다.. 이때 **snakeShape**는 'X'를 사용합니다.
- 생성된 **snake**는 **head**의 방향으로 계속하여 이동합니다.
- 게임이 시작하기 전에 사용자로 부터 **snake**를 움직일 방향을 지정하는 키를 입력받습니다. `getKey()` 함수를 만들어 이를 적용하면 방향키 뿐만아니라 다른 키보드 입력도 **snake**의 방향 전환에 사용할 수 있습니다.
- 'GameOver'가 되는 경우에 대한 정의를 `checkCollision` 함수를 이용하였습니다.
  - 1) 게임 진행 중 `getch()` 함수를 이용해 만약 **snake**의 진행 방향과 반대되는 방향이 입력되었을 때에는 "Game Over"가 됩니다. 예를 들어 **snake**의 이동방향이 아래인 상황에서 사용자가 위로 방향전환하는 키를 입력하였을 때에는 게임을 중단합니다.
  - 2) `BackSpace`가 입력된다면 "Game Over"가 되도록 설계하였습니다.
  - 3) **snake**의 크기가 아이템으로 인해 커지는 경우 자신의 몸에 **head**가 부딪히는 경우에도 게임이 중단됩니다.
  - 4) **snake**의 크기가 아이템으로 인해 작아져 크기가 3미만이 되는 경우 게임이 종료됩니다.
  - 5) 1단계에서 구현한 **map**에서 **snake**의 **head** 부분이 벽의 좌표와 같게 된다면 충돌이 일어나 게임이 중단됩니다. 이런 상황에서는 `checkCollision` 함수를 이용해서 두 좌표값을 비교합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

### 3단계 : Item 요소의 구현

- 3가지의 item을 구현하였습니다.

1) growthItem: snake의 head 부분이 아이템의 좌표와 일치하면 snake의 크기가 1 증가합니다.

2) poisonItem: snake의 head 부분이 아이템의 좌표와 일치하면 snake의 크기가 1 감소합니다.

3) speedItem: snake의 head 부분이 아이템의 좌표와 일치하면 snake의 속도가 증가합니다.

- 각 item은 "+", "-", "\*"의 문자, 초록, 빨강, 노랑색으로 표현됩니다.

- placeGrowthItem(), placePoisonItem(), placeSpeedItem() 함수를 이용해 화면의 랜덤한 좌표값에 각 아이템을 배치합니다. 또한 일정 시간이 지나면 아이템은 다시 새로운 좌표값으로 갱신합니다. 이 경우에는 growthItemTimer, poisonItemTimer, speedItemTimer 변수를 사용합니다.

- 이때 아이템의 좌표는 벽을 제외한 좌표값입니다. 만약 아이템이 배치된 위치가 snake의 몸체 일부분이거나 벽일 경우에는 while 문으로 다시 생성하게 합니다.

### 4단계 : Gate 요소의 구현

- gate는 snake가 다른 좌표로 이동할수 있게 하는 요소로 gate\_idx1, gate\_idx2 를 이용하여 한 쌍의 랜덤한 벽 좌표를 받아 생성합니다.

- 이때 바깥 벽 뿐 만 아니라 화면 안에 있는 벽의 랜덤한 좌표도 포함됩니다. 또한 두 벽 좌표가 중복되지 않도록 하고 모서리 부분, immune wall에 gate가 생성되지 않도록 합니다.

- gate는 한 위치에 고정되지 않고 item 요소들과 같이 gateTimer 변수를 이용해 특정 시간이 지나면 랜덤하게 재배치 됩니다.


- nucurses 함수를 이용해 파란색으로 표시하여 다른 벽들과 다름을 구별합니다.

- gate 좌표가 snake의 head 부분과 만나면 snake는 다른 Gate로 이동(워프)합니다.

- getWarpDirection으로 snake가 들어오는 방향과 두 gate의 좌표를 받아 snake의 집입방향을 정의합니다.

- 이동(워프) 후 뱀의 방향은 우선적으로 벽에서 화면 안쪽으로 설계하고, 시계방향으로 방향의 우선순위를 지정합니다.



 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 5단계 : 점수 요소의 구현

- 각 level에 따라 사용자가 수행해야하는 미션이 존재합니다. 미션은 다음과 같습니다.


- 1) B: currentLength, snake의 몸 길이
- 2) + : growthItem, snake가 획득한 growthItem의 개수
- 3) - : poisionItem, snake가 획득한 poisionItem의 개수
- 4) \* : speedItem, snake가 획득한 speedItem의 개수
- 5) G: gate, snake가 통과한 gate 횟수

- 각 level에 따라 요구되는 미션의 개수, 횟수가 다릅니다.. 이를 requiredLength, requiredGrowthItem, requiredPoisionItem, requireSpeedItem, requiredGate 변수로 정의하고 각 스테이지가 끝나면 nextStage()함수로 기준을 확인합니다.

- scoreGrowthItem, scorePoisionItem, scoreSpeedItem, scoreGate 변수로 사용자가 수행한 미션의 수를 정의합니다.

- displayScore함수를 이용하여 화면에 ScoreBoard와 Mission을 출력하여 현재 사용자가 수행한 미션의 횟수(score 변수))와 주어진 미션(required 변수)을 보여줍니다.

- 사용자가 주어진 미션을 수행하면 요구되는 미션의 수를 -1로 갱신해주고, 해당 아이템 또는 gate통과 횟수를 모두 충족하면 'V'으로 갱신하여 모두 수행하였음을 사용자에게 알려줍니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 2.2.2 시스템 구조 및 설계도

### 1단계 : Map의 구현(정성엽 담당)

#### - set window

```
// window 초기화
void snakeGame::setWindow(int level)
{
    initscr(); // initialise the screen
    nodelay(stdscr, TRUE);
    keypad(stdscr, true);
    noecho(); // user input is not displayed on the screen
    curs_set(0); // cursor symbol is not displayed on the screen (Linux)
    getmaxyx(stdscr, maxHeight, maxWidth); // define dimensions of game window
    maxHeight -= (level - 1) * 3;
    maxWidth -= (level - 1) * 6;
    return;
}
```

- 기본적인 윈도우 세팅을 설정해주었습니다. 단말기와 키패드를 허용하고, 사용자의 input을 따로 보여주지 않기 위해 `noecho()`를 지정합니다. 또한 따로 커서를 보이지 않게 하기 위해 `curs_set(0)`을 선언해주었습니다.
- 또한 스테이지가 올라갈수록 난이도를 높이기 위해 레벨에 따라 윈도우가 작아지도록 설정했습니다



## - renderwindow

```
int k, q;
for (int i = 1; i < maxWidth - 12; i++) // 윗벽
{
    wall.push_back(CharPosition(i, 0));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(0, i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}

for (int i = 1; i < maxWidth - 12; i++) // 아래 벽
{
    wall.push_back(CharPosition(i, maxHeight - 1));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(maxHeight - 1, i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}

for (int i = 1; i < maxHeight - 1; i++) // 왼쪽 벽
{
    wall.push_back(CharPosition(0, i));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(i, 0);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}

for (int i = 1; i < maxHeight - 1; i++) // 오른쪽 벽
{
    wall.push_back(CharPosition(maxWidth - 12, i));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(i, maxWidth - 12);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}
```

- maxWidth, maxHeight에 따라 가장자리 외벽을 생성해줍니다.
- score board 자리를 위해 오른쪽은 12만큼 빼주었습니다.
- 하나씩 벽을 생성하며 미리 선언해둔 vector에 wall의 좌표를 담은 객체를 저장합니다. 이를 이용하여 gate와 같은 wall을 이용한 각종 기능과 동작을 구현합니다.
- 벽은 하얀색으로 나타내도록 하였습니다.



```
// 레벨에 따라 맵 구성
switch (level)
{
case 1: // KMU 모양의 장애물
    for (int i = 0; i < (maxWidth - 12) / 10; i++) {
        start_color();
        init_pair(3, COLOR_WHITE, COLOR_WHITE);
        attron(COLOR_PAIR(3));

        wall.push_back(CharPosition((maxWidth - 12) / 10 * 1, H));
        move(H, (maxWidth - 12) / 10 * 1);
        addch(edgechar);

        wall.push_back(CharPosition((maxWidth - 12) / 10 * 2 - i, H));
        move(H, (maxWidth - 12) / 10 * 2 - i);
        addch(edgechar);
        attroff(COLOR_PAIR(3));
        refresh();
        H++;
    }
    for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...

    // M
    H = (maxHeight - 1) / 5;
    for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
    for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
    H = (maxHeight - 1) / 5;

    //U
    for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
    for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
    break;
}


case 2: { // 3분할 장애물 ...
case 3: { // X모양의 장애물 ...
case 4: { // 9분할 장애물
    { ...
break;
}
}
```

- switch 문을 이용하여 레벨(스테이지)별로 다른 맵을 형성하도록 구현하였습니다.
- 이전의 외부벽을 세운것과 같은 맥락으로 wall을 추가하고, 색을 바꾸어주었습니다.
- stage1은 팀의 정체성을 담아 kmu모양의 장애물로 구성하였습니다 (정성엽 담당)
- stage2는 화면을 3분할 하는 장애물로 구성하였습니다. (정성윤 담당)
- stage3은 X모양으로 화면을 분할하는 장애물로 구성하였습니다.(김유진 담당)
- stage4는 화면을 9분할 하는 장애물로 구성하였습니다. (정성윤 담당)



```
// 모서리 부분을 다른 색으로 표시
start_color();
init_pair(4, COLOR_BLACK, COLOR_BLACK);
attron(COLOR_PAIR(4));
move(0, 0);
addch(edgechar);
move(0, maxWidth - 12);
addch(edgechar);
move(maxHeight - 1, 0);
addch(edgechar);
move(maxHeight - 1, maxWidth - 12);
addch(edgechar);
attroff(COLOR_PAIR(4));
refresh();
```

- 벽의 가장자리에 위치한 immune wall은 색을 검은색으로 설정하여 일반 벽과 차이를 두었습니다.


 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 2단계 Snake 표현 및 조작 (김유진 담당)

### - moveSnake

```
// Snake 움직임 정의
void snakeGame::moveSnake()
{
    KeyPressed = getch();
    if (KeyPressed == leftKey) {
        if (direction != 'r')
        {
            direction = 'l';
        }
        else
            checkCollision();
    }
    else if (KeyPressed == rightKey) {
        if (direction != 'l')
        {
            direction = 'r';
        }
        else
            checkCollision();
    }
    else if (KeyPressed == upKey) {
        if (direction != 'd')
        {
            direction = 'u';
        }
        else
            checkCollision();
    }
    else if (KeyPressed == downKey) {
        if (direction != 'u')
        {
            direction = 'd';
        }
        else
            checkCollision();
    }
    else if (KeyPressed == KEY_BACKSPACE)
        direction = 'q';
}
```

- 방향키 입력에 대하여 snake의 방향을 움직이도록 구현하였습니다. 백스페이스를 누르면 게임은 종료됩니다.

 <div> 국민대학교  소프트웨어부  C++프로그래밍 </div>	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## - checkCollision

```
// 게임 종료 조건
bool snakeGame::checkCollision()
{
    for (int i = 0; i < wall.size(); i++)
    {
        if (snake[0].x == wall[i].x && snake[0].y == wall[i].y)
        {
            if (!((snake[0].x == gate_1.x && snake[0].y == gate_1.y) || (snake[0].x == gate_2.x && snake[0].y == gate_2.y)))
            {
                return true;
                break;
            }
        }
    }
}
```

- Snake는 벽(Wall)을 통과할 수 없고, 자신의 몸을 통과할 수 없습니다. 따라서 if 문을 이용하여 스네이크의 헤드부분이 벽의 좌표와 일치한다면, 게이트인지 확인 후 게이트가 아니라면 게임이 종료되도록 구현하였습니다.

```
for (int i = 2; i < snake.size(); i++)
{
    if (snake[0].x == snake[i].x && snake[0].y == snake[i].y)
    {
        return true;
    }
}
```

- 또한 Snake의 Head가 자신의 Body 닿게 되면 게임을 중단하도록 합니다.




```
if (snake.size() < 3)
{
    return true;
}

if (direction == 'r' && KeyPressed == leftKey)
{
    return true;
}
if (direction == 'l' && KeyPressed == rightKey)
{
    return true;
}
if (direction == 'u' && KeyPressed == downKey)
{
    return true;
}
if (direction == 'd' && KeyPressed == upKey)
{
    return true;
}

return false;
```

- snake의 몸 길이가 3보다 작아진 경우에도 게임은 종료됩니다.
- 또한 if문을 이용하여 snake의 진행방향(direction)과 반대방향으로 가는 키가 눌린다면 게임은 종료되도록 구현하였습니다.



 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

### 3단계 : Item 요소의 구현 (정성윤 담당)


#### (1) GrowthItem 구현

```
// 맵에 GrowthItem 표현
void snakeGame::placeGrowthItem()
{
    int tmpx, tmpy;
    bool clear = false;
    while (!clear)
    {
        tmpx = rand() % (maxWidth - 13) + 1;
        tmpy = rand() % (maxHeight - 2) + 1;
        clear = true;

        for (int i = 0; i < snake.size(); i++)
        {
            if (snake[i].x == tmpx && snake[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }

        for (int i = 0; i < wall.size(); i++)
        {
            if (wall[i].x == tmpx && wall[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }
    }
}
```

- 난수를 이용해서 임의의 위치를 받고 그 위치가 Snake와 Wall에 있지 않도록 합니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

```

growthItem.x = tmpx;
growthItem.y = tmpy;
start_color();
init_pair(1, COLOR_WHITE, COLOR_GREEN);
attron(COLOR_PAIR(1));
move(growthItem.y, growthItem.x);
addch(growthItemShape);
attroff(COLOR_PAIR(1));
refresh();
}

```

- 그 위치에 색상은 초록색으로 표현한 후 맵에 표시합니다.

```

// GrowthItem 위치 변경
void snakeGame::growthItemTime()
{
    growthItemTimer++;
    if (growthItemTimer % maxItemTime == 0)
    {
        move(growthItem.y, growthItem.x);
        printw(" ");
        placeGrowthItem();
        growthItemTimer = 0;
    }
}


```

- 일정시간이 지나면 GrowthItem의 위치를 변경합니다.



```
// GrowthItem 먹었는지 확인
bool snakeGame::checkGrowth()
{
    if (snake[0].x == growthItem.x && snake[0].y == growthItem.y)
    {
        growthItemTimer = 0;
        placeGrowthItem();
        currentLength++;
        scoreGrowthItem++;
        displayScore();
        return eatGrowthItem = true;
    }
    else
    {
        return eatGrowthItem = false;
    }
}
```

- Snake가 GrowthItem을 먹었는지 확인합니다.
- Snake의 Head 위치가 GrowthItem의 위치와 일치해야 합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18


## (2) PoisonItem 구현

```
// 맵에 PoisonItem 표현
void snakeGame::placePoisonItem()
{
    int tmpx, tmpy;
    bool clear = false;
    while (!clear)
    {
        tmpx = rand() % (maxWidth - 13) + 1;
        tmpy = rand() % (maxHeight - 2) + 1;
        clear = true;

        for (int i = 0; i < snake.size(); i++)
        {
            if (snake[i].x == tmpx && snake[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }

        for (int i = 0; i < wall.size(); i++)
        {
            if (wall[i].x == tmpx && wall[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }
    }
}
```

- 난수를 이용해서 Snake의 위치와 Wall의 위치와 겹치지 않는 곳을 찾습니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

```

poisonItem.x = tmpx;
poisonItem.y = tmpy;
start_color();
init_pair(2, COLOR_WHITE, COLOR_RED);
attron(COLOR_PAIR(2));
move(poisonItem.y, poisonItem.x);
addch(poisonItemShape);
attroff(COLOR_PAIR(2));
refresh();
}

```

- 해당 위치를 찾으면 색상은 빨간색으로 한 후 맵에 표시합니다.

```

// PoisonItem 위치 변경
void snakeGame::poisonItemTime()
{
    poisonItemTimer++;
    if (poisonItemTimer % maxItemTime == 0)
    {
        move(poisonItem.y, poisonItem.x);
        printw(" ");
        placePoisonItem();
        poisonItemTimer = 0;
    }
}


```

- 일정시간이 지나면 PoisonItem의 위치가 바뀌도록 합니다.



```
// PoisonItem 먹었는지 확인
bool snakeGame::checkPoison()
{
    if (snake[0].x == poisonItem.x && snake[0].y == poisonItem.y)
    {
        poisonItemTimer = 0;
        placePoisonItem();
        currentLength--;
        scorePoisonItem++;
        displayScore();
        return eatPoisionItem = true;
    }
    else
    {
        return eatPoisionItem = false;
    }
}
```

- Snake가 PoisonItem을 먹었는지 확인합니다.
- 이 때 Snake의 Head 부분이 PoisonItem의 위치와 일치해야 합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

### (3) SpeedItem 구현

```
// 맵에 SpeedItem 표현
void snakeGame::placeSpeedItem()
{
    int tmpx, tmpy;
    bool clear = false;
    while (!clear)
    {
        tmpx = rand() % (maxWidth - 13) + 1;
        tmpy = rand() % (maxHeight - 2) + 1;
        clear = true;

        for (int i = 0; i < snake.size(); i++)
        {
            if (snake[i].x == tmpx && snake[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }

        for (int i = 0; i < wall.size(); i++)
        {
            if (wall[i].x == tmpx && wall[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }
    }
}
```

- 난수를 이용해서 임의의 위치를 찾습니다.
- 이 때 Snake의 위치와 Wall의 위치가 아닌 곳을 찾아야 합니다.



```
speedItem.x = tmpx;
speedItem.y = tmpy;
start_color();
init_pair(6, COLOR_WHITE, COLOR_YELLOW);
attron(COLOR_PAIR(6));
move(speedItem.y, speedItem.x);
addch(speedItemShape);
attroff(COLOR_PAIR(6));
refresh();
}
```

- 위치를 찾으면 그 위치를 노란색으로 표시한 후 맵에 표시합니다.


```
// SpeedItem 위치 변경
void snakeGame::speedItemTime()
{
    speedItemTimer++;
    if (speedItemTimer % maxItemTime == 0)
    {
        move(speedItem.y, speedItem.x);
       printw(" ");
        placeSpeedItem();
        speedItemTimer = 0;
    }
}
```

- 일정 시간이 지나면 SpeedItem의 위치가 바뀌도록 합니다.

```
// SpeedItem 먹었는지 확인
void snakeGame::checkSpeed()
{
    if (snake[0].x == speedItem.x && snake[0].y == speedItem.y)
    {
        speedItemTimer = 0;
        placeSpeedItem();
        snakeSpeed -= 10000;
        scoreSpeedItem++;
        displayScore();
    }
}
```

- Snake가 SpeedItem을 먹었는지 확인합니다.
- 이 때 Snake의 Head 부분이 SpeedItem의 위치와 동일해야 합니다.



 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

#### 4단계 : Gate 요소의 구현 (정성엽 담당)

```
//gate
void snakeGame::placeGate() //게이트 생성
{
    // 난수를 통해 idx를 받음
    int gate_idx1 = rand() % wall.size();
    int gate_idx2 = rand() % wall.size();
    while (gate_idx2 == gate_idx1) //중복되지않도록
    {
        gate_idx2 = rand() % wall.size();
    }
    gate_1 = wall[gate_idx1];
    gate_2 = wall[gate_idx2];
    start_color();
    init_pair(5, COLOR_BLUE, COLOR_BLUE);
    attron(COLOR_PAIR(5));
    move(gate_1.y, gate_1.x);
    addch(edgechar);
    move(gate_2.y, gate_2.x);
    addch(edgechar);
    attroff(COLOR_PAIR(5));
    refresh();
}
```

- Gate를 표시할 때는 임의의 위치에 놓일 수 있도록 난수를 받았습니다.
- 위치를 받으면 그 위치에 게이트를 놓고 색상은 파란색으로 표시했습니다.



```
// Gate 위치 변경
void snakeGame::gateTime()
{
    gateTimer++;
    if (gateTimer % maxGateTime == 0)
    {
        attron(COLOR_PAIR(3));
        move(gate_1.y, gate_1.x);
        addch(edgechar);
        move(gate_2.y, gate_2.x);
        addch(edgechar);
        attroff(COLOR_PAIR(3));
        refresh();
        placeGate();
        gateTimer = 0;
    }
}
```

- 일정 시간이 지나면 Gate의 위치를 바꾸도록 구현했습니다.
- 기존의 Gate 위치는 다시 하얀색 Wall로 설정했습니다.

```
// Gate에 들어갔는지 확인
bool snakeGame::checkGate()
{
    if (snake[0].x == gate_1.x && snake[0].y == gate_1.y)
    {
        gateTimer = maxGateTime - snake.size() - 1;
        scoreGate++;
        displayScore();
        return atGate1 = true;
    }
    else if (snake[0].x == gate_2.x && snake[0].y == gate_2.y)
    {
        gateTimer = maxGateTime - snake.size() - 1;
        scoreGate++;
        displayScore();
        return atGate2 = true;
    }
}
```

- Snake가 Gate에 들어갔는지 확인합니다.
- 이 때 Snake의 Head 부분이 Gate의 위치와 일치해야 합니다.



```
// 게이트를 통과할 때
char snakeGame::getWarpDirection(char d, CharPosition gate)
{
    char result;
    CharPosition leftBlock(gate.x - 1, gate.y);
    CharPosition rightBlock(gate.x + 1, gate.y);
    CharPosition upBlock(gate.x, gate.y - 1);
    CharPosition downBlock(gate.x, gate.y + 1);

    bool isLeftWall = false;
    bool isRightWall = false;
    bool isUpWall = false;
    bool isDownWall = false;
```

- 게이트를 통과했을 때 다른 게이트의 어떤 곳으로 나오게 될지 정하는 함수입니다.
- 먼저 나오게 될 게이트 기준으로 상하좌우가 Wall인지 확인하기 위해서 다음과 같은 구조체와 변수를 정의합니다.



```
for (int i = 0; i < wall.size(); i++)
{
    if (wall[i].x == leftBlock.x && wall[i].y == leftBlock.y)
    {
        isLeftWall = true;
    }
    if (wall[i].x == rightBlock.x && wall[i].y == rightBlock.y)
    {
        isRightWall = true;
    }
    if (wall[i].x == upBlock.x && wall[i].y == upBlock.y)
    {
        isUpWall = true;
    }
    if (wall[i].x == downBlock.x && wall[i].y == downBlock.y)
    {
        isDownWall = true;
    }
}
```

- 게이트를 기준으로 왼쪽 블록이 벽인지 확인합니다. 만약 벽이라면 true로 놓습니다.
- 오른쪽, 위와 아래도 마찬가지로 벽이라면 true로 놓습니다.

```
if ((leftBlock.x == 0 && leftBlock.y == 0) || (leftBlock.x == 0 && leftBlock.y == maxHeight - 1))
{
    isLeftWall = true;
}
if ((rightBlock.x == maxWidth - 12 && rightBlock.y == 0) || (rightBlock.x == maxWidth - 12 && rightBlock.y == maxHeight - 1))
{
    isRightWall = true;
}
if ((upBlock.x == 0 && upBlock.y == 0) || (upBlock.x == maxWidth - 12 && upBlock.y == 0))
{
    isUpWall = true;
}
if ((downBlock.x == 0 && downBlock.y == maxHeight - 1) || (downBlock.x == maxWidth - 12 && downBlock.y == maxHeight - 1))
{
    isDownWall = true;
}
```

- 게이트를 기준으로 상하좌우 블록이 Immune Wall이라도 true로 놓습니다.



```
if (gate.x == 0)
{
    isLeftWall = true;
    isUpWall = true;
    isDownWall = true;
}
if (gate.x == maxWidth - 12)
{
    isRightWall = true;
    isUpWall = true;
    isDownWall = true;
}
if (gate.y == 0)
{
    isUpWall = true;
    isRightWall = true;
    isLeftWall = true;
}
if (gate.y == maxHeight - 1)
{
    isDownWall = true;
    isRightWall = true;
    isLeftWall = true;
}
```

- 게이트의 x좌표가 0이라면 왼쪽 벽입니다.
- 따라서 왼쪽, 위와 아래는 true로 합니다.
- 마찬가지로 게이트가 오른쪽 벽, 위쪽 벽, 아래쪽 벽이면 해당하는 벽을 각각 true로 합니다.



```
if (d == 'l')
{
    if (!isLeftWall)
    {
        result = 'l';
    }
    else if (!isUpWall)
    {
        result = 'u';
    }
    else if (!isRightWall)
    {
        result = 'r';
    }
    else
    {
        result = 'd';
    }
}


if (d == 'u')
{
    if (!isUpWall)
    {
        result = 'u';
    }
    else if (!isRightWall)
    {
        result = 'r';
    }
    else if (!isDownWall)
    {
        result = 'd';
    }
    else
    {
        result = 'l';
    }
}
```

```
if (d == 'r')
{
    if (!isRightWall)
    {
        result = 'r';
    }
    else if (!isDownWall)
    {
        result = 'd';
    }
    else if (!isLeftWall)
    {
        result = 'l';
    }
    else
    {
        result = 'u';
    }
}

if (d == 'd')
{
    if (!isDownWall)
    {
        result = 'd';
    }
    else if (!isLeftWall)
    {
        result = 'l';
    }
    else if (!isUpWall)
    {
        result = 'u';
    }
    else
    {
        result = 'r';
    }
}

return result;
}
```

- 들어온 방향에 따라 순서대로 처리를 합니다.
- 만약 기존 Gate에서 왼쪽으로 들어왔다면 다른 Gate에서 왼쪽이 벽인지 확인합니다.
- 벽이 아니라면 왼쪽 방향을 리턴하고 왼쪽이 벽이라면 위, 오른쪽 그리고 아래가 벽인지 차례대로 확인합니다.
- 방향이 위쪽, 오른쪽, 아래쪽일 때도 마찬가지로 Gate 나가는 우선순위에 따라 확인하고 결과를 return합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 5단계 : 점수요소의 구현(정성엽 담당)

```
// 점수 출력
void snakeGame::displayScore()
{
    //Score Board
    move(0, maxWidth - 11);
    printf("Score Board");

    move(1, maxWidth - 9);
    printf("B : (%d)", currentLength);

    move(2, maxWidth - 9);
    printf("+ : (%d)", scoreGrowthItem);

    move(3, maxWidth - 9);
    printf("- : (%d)", scorePoisonItem);

    move(4, maxWidth - 9);
    printf("* : (%d)", scoreSpeedItem);

    move(5, maxWidth - 9);
    printf("G : (%d)", scoreGate);
}
```

- Score Board에서는 총 5개를 표현했습니다.
- Snake 현재 길이, 각각 먹은 아이템 개수 그리고 통과한 게이트 수 입니다.
- 각 점수는 해당하는 변수를 이용해서 구현했습니다.



```
//Mission
move(7, maxWidth - 11);
printw("Mission");

move(8, maxWidth - 9);
if (requiredLength - currentLength <= 0)
{
    printw("B : (%s)", "v");
}
else
{
    printw("B : (%d)", requiredLength - currentLength);
}

move(9, maxWidth - 9);
if (requiredGrowthItem - scoreGrowthItem <= 0)
{
    printw("+ : (%s)", "v");
}
else
{
    printw("+ : (%d)", requiredGrowthItem - scoreGrowthItem);
}

move(10, maxWidth - 9);
if (requiredPoisonItem - scorePoisonItem <= 0)
{
    printw("- : (%s)", "v");
}
else
{
    printw("- : (%d)", requiredPoisonItem - scorePoisonItem);
}
```

- 다음 단계로 가기 위한 미션 성공 여부는 체크 표시 'v'로 표현했습니다.
- 각 조건 별로 요구되는 조건을 만족했다면 'v' 표시를 하고 아직 만족을 못했다면 필요한 만큼 표시합니다.
- 이 코드에는 Snake Length, GrowthItem, PoisonItem의 조건이 있습니다.





```
move(11, maxWidth - 9);
if (requiredSpeedItem - scoreSpeedItem <= 0)
{
    printf("* : (%s)", "v");
}
else
{
    printf("* : (%d)", requiredSpeedItem - scoreSpeedItem);
}

move(12, maxWidth - 9);
if (requiredGate - scoreGate <= 0)
{
    printf("G : (%s)", "v");
}
else
{
    printf("G : (%d)", requiredGate - scoreGate);
}
```

- 마찬가지로 SpeedItem과 Gate에도 조건 충족 여부를 표시했습니다.



```
// 뱀 그래픽
for (int i = 1; i < 8; i++) {
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(14, maxWidth - 9 + i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

for (int i = 1; i < 8; i++) {
    attron(COLOR_PAIR(3));
    move(18, maxWidth - 9 + i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

for (int i = 1; i < 5; i++) {
    attron(COLOR_PAIR(3));
    move(14 + i, maxWidth - 9);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

for (int i = 1; i < 5; i++) {
    attron(COLOR_PAIR(3));
    move(14 + i, maxWidth - 1);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}
```


```
//뱀 혀
attron(COLOR_PAIR(3));
move(19, maxWidth - 6);
addch(edgechar);
attroff(COLOR_PAIR(3));

attron(COLOR_PAIR(3));
move(20, maxWidth - 7);
addch(edgechar);
attroff(COLOR_PAIR(3));

//뱀 눈
attron(COLOR_PAIR(3));
move(16, maxWidth - 7);
addch(edgechar);
attroff(COLOR_PAIR(3));


attron(COLOR_PAIR(3));
move(16, maxWidth - 5);
addch(edgechar);
attroff(COLOR_PAIR(3));
```

- 기존의 스코어 보드 외에 새로운 이미지를 추가했습니다.
- 모양은 뱀이고 색상은 White를 이용했습니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

```
//다음 스테이지로 넘어가는 기준 확인
bool snakeGame::nextStage()
{
    if (currentLength >= requiredLength &&
        scoreGrowthItem >= requiredGrowthItem &&
        scorePoisonItem >= requiredPoisonItem &&
        scoreGate >= requiredGate &&
        scoreSpeedItem >= requiredSpeedItem)
    {
        return true;
    }
    return false;
}
```

- 조건을 충족했다면 다음 스테이지로 넘어가도록 구현했습니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

### 2.2.3 활용/개발된 기술

#### 1) 활용 라이브러리

##### 1-1) Ncurses

**ncurses** 라이브러리를 활용하여 **stage** 디자인, 사용자 입력을 받은 제어키 설정, 색상 설정 등에 사용하여 화면을 구성 및 **snake** 제어를 할 수 있었습니다.

#### 2) 추가된 기능

##### 2-1) snake 조작 키 설정

기존에는 오직 키보드의 방향키를 이용해서 **snake**을 이동시킬 수 있었습니다. 저희는 추가적으로 게임이 시작되기 전 화면에 사용자로 하여금 오른쪽, 왼쪽, 위, 아래의 키보드 입력을 받아 사용자가 직접 키를 설정 할 수 있게 하였습니다.




##### 2-2) speedItem(\*)

크기를 늘려주는 **growthItem(+)** 과 크기를 줄여주는 **poisonItem(-)**이 기존에 있었다면 **snak**의 이동 속도를 증가하게 해주는 **speedItem(\*)** 추가적으로 구현하였습니다. 해당 아이템은 '\*'로 표현하였습니다. **level**이 증가할 때 마다 더 많은 아이템을 먹어야 클리어가 됩니다. 따라서 미션을 클리어하기 위해서는 더 많은 **speedItem**이 필요하고, 스네이크의 이동속도가 빨라지면 세밀한 컨트롤을 하기 쉽지 않기 때문에 **level** 이 증가할 때 마다 난이도가 올라갑니다.

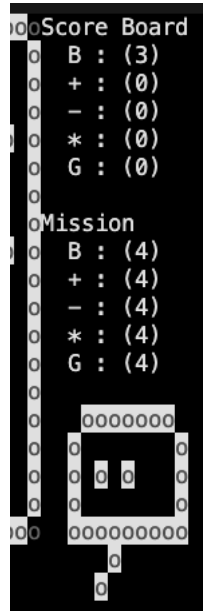
##### 2-3) 스테이지 디자인


각 스테이지는 **Map()**을 사용해서 그리지 않고 사용자 화면의 **maxWidth**, **maxHeight**를 받아 제작하였습니다. 따라서 스테이지의 디자인은 사용자의 터미널 크기에 따라 변할 수 있게 구성하였습니다.

##### 2-4) 스코어 보드 디자인

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

기존에 착착한 스코어 보드에 **snake** 모양의 그림을 추가하였습니다.



 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 2.2.4 현실적 제한 요소 및 그 해결 방안

### 1. 1단계

- 맵 중간에 **Wall**을 만들 때 원하는 모양으로 나오지 않아서 시간이 오래 걸렸습니다. **Wall** 모양이 대칭이라면 반복문을 활용해서 쉽게 구현할 수 있을 것입니다.

### 2. 2단계

- 사용자로부터 키를 입력 받고 그 키를 **Snake** 조작에 쓰이게 하는 것이 어려웠습니다. 그래서 조작키를 클래스 변수로 선언한 다음 사용자 입력으로부터 조작키를 받았습니다. 그 조작키로 **Snake** 움직임을 구현했습니다.

### 3. 3단계


- **GrowthItem**이나 **PoisonItem**을 먹었을 때 **Snake** 크기를 늘리거나 줄이는 것을 반영하는 것이 어려웠습니다. 그래서 따로 **vector**를 이용해서 **Snake** 길이를 저장한 후 아이템을 먹었을 때 길이 조절이 되도록 구현했습니다.

### 4. 4단계

- **Snake**가 **Gate**를 통과할 때 다른 **Gate**에서 우선순위에 따라 나오게 하는 것이 어려웠습니다. **Snake**가 들어가는 방향하고 **Gate**에서 나올 수 있는 방향이 매번 다르기 때문에 모든 경우를 하나씩 따져가며 구현했습니다.


### 5. 5단계

- 없음

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 2.2.5 결과물 목록

파일명	역할
snakeGame.h	snakeGame 동작을 위해 필요한 것들을 정의합니다.
snakeGame.cpp	snakeGame을 위한 구체적인 코드를 정의합니다.
main.cpp	snakeGmae.cpp에 정의된 기능들을 실행하기 위한 파일 입니다.
makefile	프로그램 실행을 위한 컴파일 과정을 간단하게 하기위한 파일입니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

### 3 자기평가

팀원명	평가
김유진	<p>저는 이번 프로젝트에서 <b>snake</b>의 움직임과 게임 종료 조건을 구현하는 역할을 담당했습니다. 아 이걸 어떻게 구현해야하나 처음 구현을 시작했을 당시에는 막막했지만, 팀원들과 함께 전체적인 그림을 그려나가는 과정에서 혼자 복잡하게 생각했다는것을 깨달았습니다. <b>if</b>문이나 <b>for</b>문, <b>list</b>같이 지금까지 배운, 정말 단순한 것 만으로도 충분히 가능한 이야기였습니다. 그 후에는 순조롭게 구현을 진행하였습니다. 하나씩 구현하고 완성해 나가는것이 퍼즐을 맞춰나가는것 같아 즐거웠습니다.</p> <p>게임이 원하는 대로 작동하지 않아 머리를 부여잡고 오류를 찾아내려했지만, 도저히 모르겠어서 팀원들에게 도움을 요청했었는데 원인이 정말 단순한 오타하나, 코드 한줄이었다는것에 놀랐습니다.</p> <p><b>c++</b>를 처음 접해봐서 모르는것도 많고 미숙한점도 많았는데 팀원들에게 배우고, 공부하고, 직접 <b>snakegame</b>을 구현하며 여러모로 많은것을 배웠습니다. 팀원 모두 바쁜시간을 쪼개가며 열심히 참여해주어 너무 감사합니다.</p>
정성엽	<p>저는 <b>Snake Game Project</b>에서 <b>Map</b>과 <b>Score Board</b> 구현을 담당했습니다. 맵을 만들 때 벽에 색상을 추가하는 법을 몰라서 계속 찾아봤었습니다. 그리고 맵 중간에 <b>Wall</b>을 구현할 때에도 원하는 모양이 나오지 않아 시간이 오래 걸렸습니다. 그래도 나중에는 대칭을 이용해서 <b>Wall</b>을 쉽게 구현할 수 있었습니다. 이 프로젝트를 하면서 처음부터 잘하지 못하고 시간이 오래 걸렸습니다. 그래도 하면 할수록 알아가는 것이 많았고 나중에는 코드를 더 효율적으로 구현할 수 있었습니다. 이번 프로젝트는 혼자 하는 것이 아니라 같이 했기 때문에 <b>Snake Game</b>을 만들 수 있었다고 생각합니다.</p>





국민대학교  
소프트웨어학부  
C++프로그래밍

결과보고서

프로젝트 명

snake game

팀 명

18조


Confidential Restricted

Version 2.1

2023-06-18


정성윤

이번 snake Game 구현 프로젝트에서 아이템 요소와 gate 요소를 맡았습니다. growth, poison, speed 아이템이 올바른 위치에 랜덤하게 배치해야 했습니다. 그리고 해당 아이템을 snake가 먹게 된다면 변경된 snake의 크기를 보여주는 부분에서 많은 시간을 들였습니다. 다양한 참고자료를 통해 구현할 수 있었습니다. 또한 아이템으로 인한 snake의 속도를 ncurses로 표현하는 부분에서 어려움을 겪었으나 팀원들의 피드백과 아이디어로 해당 부분을 완성할 수 있었습니다. gate의 위치가 올바르게 생성될 수 있도록 하였습니다. gate의 위치가 중복되거나 Immune wall에 생성되지 않도록 하는 코드가 필요하였습니다. 특히 gate를 이용해 이동한 snake의 진행 방향의 경우의 수를 처리하는 부분이 매우 어려웠으나 케이스 분류를 차근차근 정리하여 이를 코드로 천천히 구현할 수 있었습니다. 이번 프로젝트를 통해 c++과 ncurses 라이브러리에 대해서 배울 수 있었습니다. 팀원들과의 지속적인 만남과 업무 분할 및 협업을 통해 성공적으로 snake Game 프로그램을 완성할 수 있어 보람찬 프로젝트였습니다. 프로젝트에서 팀원들간의 소통과 서로의 피드백이 중요함을 깨닫게 되었습니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

## 4 참고 문헌

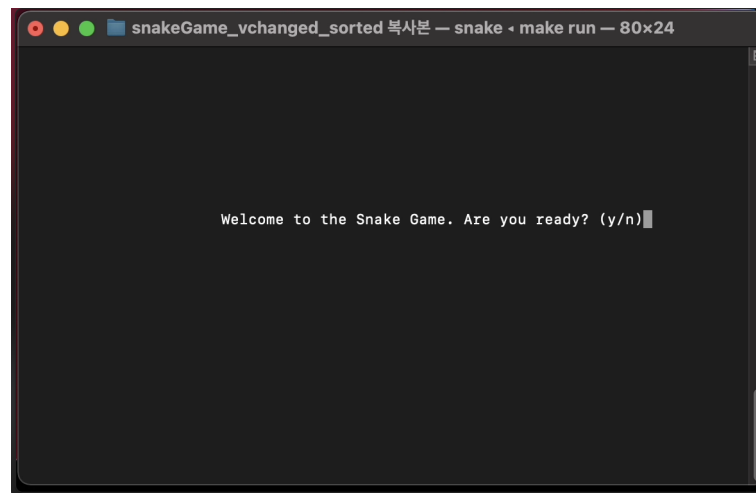
번호	종류	제목	출처	발행년도	저자	기타
1	웹 페이지	Ncurses 라이브러리	IBM docs: <a href="https://www.ibm.com/docs/ko/aix/7.3?topic=concepts-curses-library">https://www.ibm.com/docs/ko/aix/7.3?topic=concepts-curses-library</a>	2023-03-24	IBM	
2	웹 페이지	Ncurses Programming HowTo	<a href="http://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO">http://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO</a>	2005-06	Praeep Padala	

 <div> 국민대학교  소프트웨어부  C++프로그래밍 </div>	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

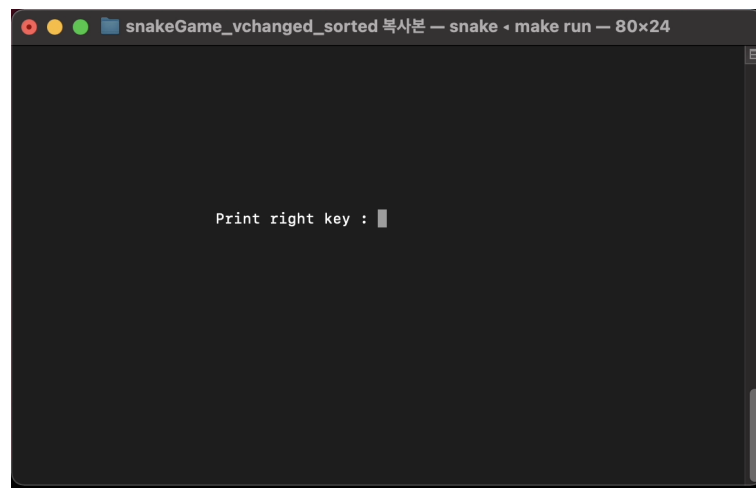
## 5 부록


### 5.1 사용자 메뉴얼

1. 터미널에 **make run**을 입력하면 다음과 같이 게임이 실행됩니다. **y(yes)**를 누르면 게임을 시작 할 수 있습니다.



2. 화면의 요구에 따라 상하좌우로 이용할 키를 지정합니다.



 <div> 국민대학교  소프트웨어부  C++프로그래밍 </div>	결과보고서		
	프로젝트 명	snake game	
	팀 명	18조	
	Confidential Restricted	Version 2.1	2023-06-18

- 지정한 키를 이용해 게임을 진행합니다.



- 아이템을 먹고 일정점수 이상을 획득하게 되면 다음 스테이지로 넘어가게 됩니다. 총 4개의 스테이지가 있으며, 벽에 충돌하지 않고 모든 스테이지를 클리어하면 성공입니다.
- 아래의 링크를 들어가면 영상을 통해 사용자 메뉴얼을 보실 수 있습니다.

<https://youtu.be/OsmhDhg0cAs>

## 5.2 설치 방법

- 다음의 주소를 인터넷 브라우저 주소창에 입력후 다운로드 합니다.

<https://github.com/Seongyoong-Jung/snakeGame>

- 게임을 설치하고자 하는 디렉토리에 압축 해제합니다.
- 터미널에서 압축해제한 디렉토리에 접근 후 다음과 같이 입력합니다.

make run

- 게임이 실행됩니다.