




국민대학교  
소프트웨어융합대학  
소프트웨어학부

# C++프로그래밍 프로젝트

프로젝트 명	Snake Game
팀 명	소행소식
문서 제목	결과보고서

Version	1.3
Date	2023-06-18

팀원	김채현 (팀장)
	김민승
	이다현
	한윤구

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

#### CONFIDENTIALITY/SECURITY WARNING

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

## 문서 정보 / 수정 내역

<b>Filename</b>	최종보고서- <b>SnakeGame</b> .doc
<b>원안작성자</b>	김채현, 김민승, 이다현, 한윤구
<b>수정작성자</b>	김채현, 김민승, 이다현, 한윤구

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2023-06-17	한윤구	1.0	최초 작성	
2023-06-17	김채현	1.1	최초 작성	
2023-06-17	이다현	1.2	최초 작성	
2023-06-17	김민승	1.3	최초 작성	

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	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 .....	8
2.2.3	활용/개발된 기술 .....	47
2.2.4	현실적 제한 요소 및 그 해결 방안 .....	48
2.2.5	결과물 목록 .....	49
3	자기평가 .....	51
4	참고 문헌 .....	52
5	부록 .....	52
5.1	사용자 매뉴얼 .....	52
5.2	설치 방법 .....	53

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

# 1 개요

## 평가기준 (10점)


프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.

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

SnakeGame 프로젝트를 완성하기 위해 사용한 개발 방법은 카카오톡을 이용한 의사소통과 google docs 를 활용한 소스 코드 공유이다. google docs 의 경우 전체적인 Snake Game 의 흐름과 기본 구조를 공유하였고, google docs 의 목차 생성 기능으로 클래스를 구분하여 본인이 맡은 파트의 코드를 공유하고 다른 팀원이 코드를 같이 확인하고 수정하는 방식을 통하여 개발을 진행하였다. 기본적인 의사소통을 비롯하여 코드의 오류 발견이나 의견 공유 외에도 참고할만한 자료 공유 등은 카카오톡 단체 채팅방을 활용하여 개발을 진행하였다.

Snake Game 프로젝트에서 사용한 외부 라이브러리는 ncurses 이다. ncurses 라이브러리를 사용하기 위한 개발환경인 리눅스 우분투를 통하여 개발을 진행하였다. 리눅스 우분투를 사용하기 위해 윈도우 사용자의 경우 wsl 을 통해 리눅스를 사용하였고 리눅스 환경에서 ncurses 라이브러리를 다운로드하기 위해 “sudo apt-get install libncurses5-dev libncursesw5-dev” 명령어를 사용하였다.

파일의 구성은 main.cpp, Board.hpp, SnakeGame.hpp, Snake.hpp, Drawable.hpp, Apple.hpp, Grape.hpp, Map.hpp, Empty.hpp, Scoreboard.hpp, Gate.hpp 로 구성하였으며 이에 대한 자세한 상속 구조와 클래스 설명은 후술하고자 한다.

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

## 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	미적용
5단계	점수 요소의 구현	적용

#### 1 단계 : Map 구현목표


- 맵은 기본 사각형 형태로 구성하고 각 스테이지로 넘어갈 때 **obstacle** 메서드를 통해 스테이지에 맞는 장애물을 추가하는 방식으로 맵을구성하며 기본 사각형 안쪽은 **Empty.hpp** 를 통해 빈공간을 공백으로 채운다. Map 의 크기는 **main.cpp** 의 상수로 선언하며 **12\*36** 으로 설정한다.

#### 2 단계 : Snake 표현 및 조작 목표

- Snake** 는 '#'문자로 표시한다. 길이는 **4** 로 게임을 시작하며 **queue** 헤더를 사용하여 **Snake** 의 길이를 **push** 함수와 **pop** 함수로 늘렸다 줄였다 조절한다. **Snake** 의 조작은 사용자의 방향키 입력에 따라 진행방향을 정할 정할 수 있다. 예를 들어 기본 진행방향에서 사용자의 방향키 입력대로 방향을 꺾고 아무 입력을 하지 않으면 기존 진행 방향을 유지하며 이동한다. 방향키를 빠르게 연타하거나 누른 상태를 유지하면 속도가 더욱 빨라진다. 주의할 점은 진행방향의 반대 키를 입력하면 스테이지는 종료된다.

#### 3 단계 : Item 요소의 구현 목표

- Apple** 은 'A'로 게임에서 표기하고 **Snake** 의 머리가 'A'에 닿으면 점수가 **100** 점 증가하고 꼬리에 '#'을 추가하여 꼬리가 길어지며 몸이 **1** 칸 늘어난다. 'A'의 발생 위치는 **Board** 클래스 내의 **getEmptyCoordinate** 메서드를 통해 맵 내의 랜덤한 공백의 위치를 받아 해당 위치에 생성된다.

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18


- Grape 는 'B'로 게임에서 표기한다. Snake 의 머리가 'B'에 닿으면 점수가 100 점 감소하고 고리의 '#'을 제거하여 몸이 1 칸 감소한다. 'B'의 생성위치는 Apple 과 동일하게 Board 클래스 내의 getEmptyCoordinate 메서드를 통해 맵 내의 랜덤한 공백의 위치를 받아 해당 위치에 생성된다.

#### 4 단계 : Gate 요소의 구현 목표

- Gate 는 'T'로 게임에서 표기한다. 'B'의 생성위치는 Board 클래스 내의 getCkCoordinate 메서드를 통해 맵 장애물내의 랜덤한 위치를 받아 해당 위치에 생성된다.

#### 5 단계 : 점수 요소의 구현 목표

- 점수는 Board 하단에 Apple 을 획득하여 득점한 점수와 Grape 를 획득하여 감점된 점수를 계산하여 Score 로 출력한다. +는 Apple 을 획득한 횟수, -는 Grape 를 획득한 횟수, B 는 현재의 길이를 나타낸다. Apple 과 Grape 를 획득한 횟수에 따라 값이 계속 업데이트 된다. 각 스테이지마다 목표를 정해두었는데 B 는 7, +는 3, -는 1 을 달성하게 되면 다음 스테이지로 넘어간다. 목표를 달성할 경우 괄호 안에 v 자로 체크가 되며 B 의 경우 목표를 달성한 상태였다가도 길이가 줄어들어 목표보다 작아질 경우 v 가 사라진다. 모든 항목의 목표를 달성하고, 최종 점수가 300 점 이상이면 스테이지가 클리어어 되도록 구현하고자 한다.

 <div> <p>국민대학교</p> <p>소프트웨어학부</p> <p>C++프로그래밍</p> </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

## 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 단계 : map 의 구현
  - Map 은 Drawable 클래스를 상속받아 작성된다. Map 은 int 타입의 y 와 x 를 매개변수로 갖는 생성자를 작성하고 해당 y 와 x 를 Drawable 클래스의 생성자를 호출하여 y 와 x 좌표값과 ACS\_CKBOARD 값을 전달하여 ncurses 라이브러리에서 체커보드(점각기호)로 벽이 표현할 수 있게 했다. 또한 각 스테이지 별로 달라지는 맵의 구성은 SnakeGame.hpp 파일에서 obstaclePosition 2 차원 배열을 통해 내부의 장애물을 생성하여 맵을 재구성하는 식으로 구현하였다. 벽과 장애물이 없는 맵의 내부는 Empty 클래스를 통해 공백으로 채울 수 있게 하였다.
- 2 단계 : Snake 표현 및 조작
  - 먼저 Snake 의 표현이다. SnakePiece 클래스를 통해 나타나는데 x 좌표와 y 좌표를 0 으로 초기화 하고 아이콘은 '#'을 통해 Snake 를 나타낸다.
  - 다음으로는 Snake 의 조작이다. Snake 의 조작은 Snake 클래스를 통해 표현한다. 해당 클래스는 SnakePiece 큐를 사용한다. Snake 의 맨 끝 꼬리와 맨 끝에서 두번째 꼬리의 방향을 고려하여 진행방향을 파악하고 addPiece 메서드와 removePiece 메서드를 통해 Snake 의 길이를 늘리거나 줄이거나의 동작을 수행한다. SnakeGame 클래스 내에서 processInput 메서드를 통해 키보드에서 키를 입력 받아 위치를 이동시킨다. 진행방향 반대쪽 키를 입력하거나 벽에 Snake 가 닿거나 길이가 3 이하가 되면 게임은 종료된다.
- 3 단계 : Item 요소의 구현
  - Item 요소를 구현하기 위해 먼저 점수와 Snake 의 길이를 늘려주는 Apple 이 있다. Apple 클래스는 Drawable 클래스를 상속받아 사용되고 x 와 y 좌표에 생성되고 'A'문자를 아이콘으로 생성된다. A 에 Snake 가 닿으면 점수가 100 점 올라가며 꼬리에 '#'가 추가되며 길이가 1 증가한다.
  - 다음으로 Grape 를 구현했는데 Grape 는 Apple 과 정반대의 역할을 한다고 보면 된다. 생성되는 형식은 같으며 아이콘만 'B'로 표기한다. Snake 가 'B'에 닿으면 꼬리쪽 '#'문자가 하나 삭제 되며 길이가 1 감소하고, 점수가 100 감소한다.

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

○ **4 단계 : Gate 요소의 구현**

- Gate 요소는 Drawable 클래스를 상속받는 Gate 클래스로 구성된다. x, y 로 초기화 되는 위치에 생성되고 아이콘은 T 로 생성된다. 그 후 SnakeGame 클래스의 createGate1, createGate2 메서드를 각각 실행하면서 Gate 가 맵의 벽에 생성된다

○ **5 단계 : 점수 요소의 구현**

- 점수 계산 결과와 길이 점수 요소에 대한 것을 맵 밑에 위치하도록 한다. Scoreboard 클래스의 initialize 메서드와 clear 메서드, refresh 메서드를 통해 스테이지 이동이나 게임시작시 점수판을 초기화하도록 구성하였다. 또한 Snake 가 먹은 사과와 포도로 인해 계산된 점수를 나타내는 'score', 길이의 증감 횟수를 각각 나타내는 '+', '-', Snake 의 길이를 나타내는 'B', Grape 를 먹은 횟수를 나타내는 'G'의 점수가 각각 6, 2, 2, 2 가 되고 해당 목표치에 달성한 항목에는 'v'표시가 되며 목표치를 달성했음을 보여준다. 또한 모든 항목의 목표를 달성하고 'score'의 점수가 300 점 이상되면 스테이지를 클리어하게 된다.


## 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.

- Drawable.hpp
- #pragma once
- class Drawable {
- public:
- Drawable() {
- y = x = 0;
- icon = ' ';
- }
- Drawable(int y, int x, ctype ch) {
- this->y = y;



 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

- this->x = x;
- this->icon = ch;
- }
- int getX() {
- return x;
- }
- int getY() {
- return y;
- }
- chtype getIcon() {
- return icon;
- }
- protected:
- int y, x;
- chtype icon;
- };
- 

-       상속이 많이 되고 많이 등장할 **Drawable.hpp** 파일의 **Drawable** 클래스 먼저 설명하고자 한다. 해당 클래스는 객체의 초기 위치를 0,0 으로 설정하고 아이콘을 공백 문자로 설정한다. 매개변수가 있는 생성자는 주어진 y,x 좌표와 아이콘을 받아 값을 저장한다. include 하는 경우가 많으므로 “#pragma once”를 통해 중복선언을 방지해주고자 한다.

- 1 단계 : Map 의 구현

<map.hpp 파일>


```
#ifndef Map_hpp
```

```
#define Map_hpp
```

```
#pragma once
```

```
#include <ncurses.h>
```

```
#include "Drawable.hpp"
```

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```
class Map : public Drawable {
```

```
public:
```

```
    Map(int y, int x) : Drawable(y, x, ACS_CKBOARD) {}
```

```
};
```

```
#endif
```

- Map 클래스는 Drawable 클래스를 상속받아 작성된다. y 와 x 좌표로 Drawable 생성자를 호출하여 맵의 위치를 설정하고 벽을 ncurses 의 ACS\_CKBOARD 라는 체커보드 기호로 구성한다.

<Empty.hpp>

```
#pragma once
```

```
#include <curses.h>
```

```
#include "Drawable.hpp"
```

```
class Empty : public Drawable {
```

```
public:
```

```
    Empty(int y, int x) {
```

```
        this->y = y;
```

```
        this->x = x;
```

```
        this->icon = ' ';
```

```
    }
```

```
};
```

- Empty 클래스는 Drawable 클래스를 상속하며 아이콘을 공백으로 설정하여 맵 내부에 공백이 생길 수 있도록 작성하였다.

<SnakeGame 클래스 내 addObstacles 메서드>


```
void addObstacles(int stage) {
```

```
    switch (stage) {
```

```
    case 1: {
```

```
        int obstaclePositions[][2] = {
```

```
            {5, 10},{5, 11}, {5, 12}, {5, 13},{5, 14}, {5, 15},{5, 16},{5, 17}, {5, 18},{5, 19},{5, 20}, {5, 21},
```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

{5, 22},{5, 23}, {5, 24}

};

for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {

    int y = obstaclePositions[i][0];

    int x = obstaclePositions[i][1];

    Map* obstacle = new Map(y, x);

    board.add(*obstacle);

}

break;

}

case 2: {

    int obstaclePositions[][2] = {

        {8, 17},

        {7, 17},

        {6, 17},

{5, 13},{5, 14}, {5, 15},{5, 16},{5, 17}, {5, 18},{5, 19},{5, 20}, {5, 21},

        {4, 17},

        {3, 17},

        {2, 17}

    };

    for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {

        int y = obstaclePositions[i][0];

        int x = obstaclePositions[i][1];

        Map* obstacle = new Map(y, x);

        board.add(*obstacle);

    }

    break;

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

}

case 3: {

    int obstaclePositions[][2] = {

        {4, 1},{4, 2}, {4, 3},{4, 4}, {4, 5},{4, 6},{4, 7}, {4, 8},{4, 9},{4, 10}, {4, 11},{4,12},{4, 13},{4, 14},
        {4, 15}, {4, 16},

        {4, 17}, {4, 18},{4, 19},{4, 20}, {4, 21},{4,22},{4, 23},{4, 24}, {4, 25},

        {7, 11},{7,12},{7, 13},{7, 14}, {7, 15},{7, 16},{7, 17}, {7, 18},{7, 19},{7, 20}, {7, 21},{7,22},{7,
        23},{7, 24}, {7, 25}, {7, 22}, {7, 23}, {7, 24},

        {7, 25}, {7, 26}, {7, 27}, {7, 28}, {7, 29}, {7, 30}, {7, 31}, {7, 32},

        {7, 33}, {7, 34},

        {9, 1}, {9, 2}, {9, 3}, {9, 4},

        {9, 5}, {9, 6}, {9, 7}, {9, 8},

        {9, 9}, {9, 10}, {9, 11}, {9, 12},

        {9, 13}, {9, 14}, {9, 15}, {9, 16},

        {9, 17}, {9, 18}, {9, 19}, {9, 20},

        {9, 21}, {9, 22}, {9, 23}, {9, 24},

        {9, 25}

    };

    for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {

        int y = obstaclePositions[i][0];

        int x = obstaclePositions[i][1];

        Map* obstacle = new Map(y, x);

        board.add(*obstacle);

    }


    break;

}

case 4: {

    int obstaclePositions[][2] = {

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

{3, 9},{3, 10}, {3, 11},{3, 12},{3, 13}, {3, 14},

{4, 9},

{5, 9},                                {5, 17},{4, 17},                                {5, 22},{4, 22},

{6, 9},                                {6, 15},{6, 16},{6, 17},{6, 18},{6, 19}, {6, 22},{6, 21},{6, 22},{6,
23},{6, 24},

{7, 9},                                {7, 17},{8, 17},                                {7, 22},{8, 22},

{8, 9},{8, 10}, {8, 11},{8, 12},{8, 13}, {8, 14}

};

for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {

    int y = obstaclePositions[i][0];

    int x = obstaclePositions[i][1];

    Map* obstacle = new Map(y, x);

    board.add(*obstacle);

}

break;

}

default:

    break;

}

}

```

- 스테이지 다양화를 위해 SnakeGame.hpp 파일내 SnakeGame 클래스의 addObstacle 메서드를 활용하고, obstaclePosition 배열을 활용하여 맵 내에 벽을 만들어 새로운 맵을 만들어 내는 방식을 채택하였다.

○ 2 단계 : Snake 표현 및 조작

<Snake.hpp>

#pragma once

#include <ncurses.h>

#include "Drawable.hpp"

#include "SnakeGame.hpp"

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```
#include "Scoreboard.hpp"

#include <queue>

enum Direction {

    up = -1,

    down = 1,

    left = -2,

    right =2

};

class SnakePiece : public Drawable {

public:

    SnakePiece() {

        this->x = this->y = 0;

        this->icon = '#';

    }

    SnakePiece(int y, int x) {

        this->x = x;

        this->y = y;

        this->icon = '#';

    }

};

class Snake {

    std::queue<SnakePiece> prev_pieces;

    Direction cur_direction;


public:

    Snake() {

        cur_direction = down;

    }

};
```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

void addPiece(SnakePiece piece) {
    prev_pieces.push(piece);
}

void removePiece() {
    prev_pieces.pop();
}

SnakePiece tail() {
    return prev_pieces.front();
}


SnakePiece head() {
    return prev_pieces.back();
}

Direction getDirection() {
    return cur_direction;
}

void setDirection(Direction d) {
    if (cur_direction + d != 0)
        cur_direction = d;
}

Direction getTailDirection() {
    if (prev_pieces.size() < 2) {
        // 꼬리가 한 조각만 있는 경우, 현재 방향을 반환합니다.
        return cur_direction;
    } else {
        // 꼬리의 두 번째 조각과 첫 번째 조각의 위치를 비교하여 방향을 결정합니다.
        SnakePiece tail = prev_pieces.front();
        SnakePiece secondTail = prev_pieces.back();
    }
}

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

int tailRow = tail.getY();

int tailCol = tail.getX();

int secondTailRow = secondTail.getY();

int secondTailCol = secondTail.getX();

if (tailRow < secondTailRow) {

    return up;

} else if (tailRow > secondTailRow) {

    return down;

} else if (tailCol < secondTailCol) {

    return left;

} else {

    return right;

}

}

```

```

SnakePiece nextHead() {

    int row = head().getY();

    int col = head().getX();

    switch (cur_direction) {

    case down:

        row++;

        break;

    case up:

        row--;

        break;

    case left:

        col--;

```



 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        break;

    case right:

        col++;

        break;

    }

    return SnakePiece(row, col);

}

};

```

- Snake 를 만드는 파일은 Drawable.hpp, snakeGame.hpp, Scoreboard.hpp, ncurses, queue 헤더를 include 하여 작성되었다. Snake.hpp 파일에는 클래스가 두개 정의 되어있는데 그중 먼저 SnakePiece 클래스이다. 해당 클래스는 Drawable 클래스를 상속받았고, 0,0 좌표에서 '#'을 아이콘으로 Snake 가 생성된다.

- Snake 클래스는 SnakePiece 큐를 활용하여 Snake 의 꼬리를 생성하고 없애는 역할을하며, 꼬리 끝과 그 앞의 움직임을 통해 Snake 의 움직이는 방향을 확인하고 머리가 나아갈 방향을 설정해준다.

.

<SnakeGame 클래스에서의 Snake 조작>

```

void processInput() {

    chtype input = board.getInput();

    switch (input) {

    case KEY_UP:

    case 'w':

        if (snake.getDirection() != down) {

            snake.setDirection(up);

        }

        else {

            game_over = true;

        }

        break;

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

case KEY\_DOWN:

case 's':

```
if (snake.getDirection() != up) {
    snake.setDirection(down);
}
```

else {

```
    game_over = true;
```

}

break;

case KEY\_RIGHT:

case 'd':

```
if (snake.getDirection() != left) {
    snake.setDirection(right);
}
```

else {

```
    game_over = true;
```

}

break;

case KEY\_LEFT:

case 'a':

```
if (snake.getDirection() != right) {
    snake.setDirection(left);
}
```

else {

```
    game_over = true;
```

}

break;

 <div> <b>국민대학교</b>  <b>소프트웨어학부</b>  <b>C++프로그래밍</b> </div>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

case 'p':

board.setTimeout(-1);

while (board.getInput() != 'p')

board.setTimeout(1000);

break;

default:

break;

}

}

- 해당 메서드를 통해 생성된 **Snake** 를 방향키로 조작할 수 있다.

- 3 단계 : Item 요소의 구현

< Apple.hpp 와 Grape.hpp>

- #pragma once
- #include <curses.h>
- #include "Drawable.hpp"
- class Apple : public Drawable {
- public:
- Apple(int y, int x) {
- this->y = y;
- this->x = x;
- this->icon = 'A';
- }
- };
- #pragma once
- #include <curses.h>
- #include "Drawable.hpp"

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

○ class Grape : public Drawable {
○ public:
○     Grape(int y, int x) {
○         this->y = y;
○         this->x = x;
○         this->icon = 'B';
○     }
○ };
○

```

- 각 클래스는 동일한 구조로 되어있다. **Drawable** 클래스를 상속받고 있으며, 랜덤한 좌표에서 생성되도록 객체를 생성하였다. **Icon** 은 각각 'A'와 'B'이다


<Snakegame 내의 Item 요소 생성 및 효과>

```

void createApple() {
    int y, x;
    board.getEmptyCoordinates(y, x);
    apple = new Apple(y, x);
    board.add(*apple);
}

void eatApple() {
    delete apple;
    apple = NULL;
    score += 100;
    length++;
    if (apple_cnt >= 0) apple_cnt++;
    scoreboard.updateScore(score,length,apple_cnt, grape_cnt);
}

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

void createGrape() {

    int y, x;

    board.getEmptyCoordinates(y, x);

    grape = new Grape(y, x);

    board.add(*grape);

}

void eatGrape() {

    delete grape;

    grape = NULL;

    length--;

    if (length <= 3) {

        game_over = true;

        return;

    }

    if (grape_cnt >= 0) grape_cnt++;

    score -= 100;

    scoreboard.updateScore(score, length, apple_cnt, grape_cnt);

    int emptyRow = snake.tail().getY();

    int emptyCol = snake.tail().getX();

    Direction tailDirection = snake.getTailDirection();

    // 이전 꼬리 위치의 빈 공간 추가

    board.add(Empty(emptyRow, emptyCol));


    // 이전 꼬리 위치에서의 이동 방향에 따라 추가적인 빈 공간 추가

    switch (tailDirection) {

        case up:

            board.add(Empty(emptyRow + 1, emptyCol));

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        break;

    case down:

        board.add(Empty(emptyRow - 1, emptyCol));

        break;

    case left:

        board.add(Empty(emptyRow, emptyCol + 1));

        break;

    case right:

        board.add(Empty(emptyRow, emptyCol - 1));

        break;

    }

    snake.removePiece();

}

```

- createApple 과 createGrape 의 경우 Board 클래스의 메서드들을 통해 맵내의 특정 좌표에서 생성 되도록 구현되었다. 또한 eatApple 의 경우 apple 을 Snake 가 먹었을 때 길이가 하나 늘어나도록 구현하였으며 eatGrape 의 경우 먹었을 때 이전 꼬리 위치를 '#' 대신 빈공간을 추가하여 꼬리가 지워지도록 구현하였다.

<SnakeGame 추가>

```

void handleNextPiece(SnakePiece next) {

    if (apple != NULL) {

        switch (board.getCharAt(next.getY(), next.getX())) {

            case 'A':

                eatApple();

                break;

            case 'B':{

                eatGrape();

                board.add(Empty(next.getY(), next.getX()));

            }

        }

    }

}

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        snake.removePiece();

        break;

    }

    case 'T':{

        contactGate();

        break;

    }

    case ' ': {

        int emptyRow = snake.tail().getY();

        int emptyCol = snake.tail().getX();

        board.add(Empty(emptyRow, emptyCol));

        snake.removePiece();

        break;

    }

    default:

        game_over = true;

        break;

    }

}

board.add(next);

snake.addPiece(next);

}

```

- 각 요소를 먹었을 때 어떤 함수가 호출 되는지를 구현하였다.

○ 4 단계 : Gate 요소의 구현

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

- 5 단계 : 점수 요소의 구현

<Scoreboard.hpp>

#pragma once

```
class Scoreboard {
```

```
    WINDOW *score_win;
```

```
    int goal[4] = {5,1,1,1};
```

```
public:
```

```
    Scoreboard() {
```

```
    }
```

```
    Scoreboard(int width, int y, int x) {
```

```
        score_win = newwin(5, width, y, x);
```

```
    }
```

```
void initialize(int initial_score) {
```

```
    clear();
```

```
    mvwprintw(score_win, 0, 0, "Score");
```

```
    mvwprintw(score_win, 1, 0, "B : ");
```

```
    mvwprintw(score_win, 2, 0, "+ : ");
```

```
    mvwprintw(score_win, 3, 0, "- : ");
```

```
    mvwprintw(score_win, 4, 0, "G : ");
```

```
    updateScore(initial_score, 4, 0, 0);
```

```
    refresh();
```



 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

}

```
void updateScore(int score, int length, int apple_cnt, int grape_cnt) {
    mvwprintw(score_win, 0, score_win->_maxx - 20, "%i", score);

    if (length > goal[0]) mvwprintw(score_win, 1, score_win->_maxx - 20, "%i / 5 ( V )", length);
    else mvwprintw(score_win, 1, score_win->_maxx - 20, "%i / 10 (   )", length);

    if (apple_cnt > goal[1]) mvwprintw(score_win, 2, score_win->_maxx - 20, "%i / 5 ( V )", apple_cnt);
    else mvwprintw(score_win, 2, score_win->_maxx - 20, "%i / 5 (   )", apple_cnt);

    if (grape_cnt > goal[2]) mvwprintw(score_win, 3, score_win->_maxx - 20, "%i / 5 ( V )", grape_cnt);
    else mvwprintw(score_win, 3, score_win->_maxx - 20, "%i / 5 (   )", grape_cnt);

    if (grape_cnt > goal[3]) mvwprintw(score_win, 4, score_win->_maxx - 20, "%i / 5 ( V )", grape_cnt);
    else mvwprintw(score_win, 4, score_win->_maxx - 20, "%i / 5 (   )", grape_cnt);

    refresh();
}

void clear() {
    wclear(score_win);
}

void refresh() {
```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```
wrefresh(score_win);


}

};
```

- Score 보드 판을 만들기 위해 배열을 사용하였다. goal 배열을 통해 전체 스코어의 목표치를 설정해놓았으며 initialize 메서드를 통해 Snake의 길이, 사과·포도·점수 합계, 점수판을 초기화하고 처음 상태를 출력한다. 게임을 진행하면서 Snake가 아이템 요소를 먹으면서 점수가 변화할 때 상태를 표시해주고 목표를 달성했다는 'v'표시를 해주도록 구현하였다.

<SnakeGame 내의 아이템 요소>

```
void initialize() {
    apple = NULL;
    grape = NULL;
    gate1 = NULL;
    gate2 = NULL;
    board.initialize();
    score = 0;
    scoreboard.initialize(score);
    game_over = false;
    srand(time(NULL));
    snake.setDirection(down);
    addObstacles(round);
    handleNextPiece(SnakePiece(1, 1));
    handleNextPiece(snake.nextHead());
    handleNextPiece(snake.nextHead());
    snake.setDirection(right);
    handleNextPiece(snake.nextHead());
    if (apple == NULL) {
        createApple();
```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

    }

    if (grape == NULL) {
        createGrape();
    }

    if (gate1 == NULL) {
        createGate1();
    }

    if (gate2 == NULL) {
        createGate2();
    }

}

```

- 해당 메서드에서는 게임 시작시 아이템과 다른 기본 요소들이 나타날 수 있게 구현되었다.

#### < SnakeGame 내의 아이템 요소 2>

```

void updateState()
{
    handleNextPiece(snake.nextHead());

    if (apple == NULL) {
        createApple();
    }

    if (grape == NULL) {
        createGrape();
    }

}

```

- Snake 가 먹어서 아이템 요소가 사라졌을 때 다시 생성될 수 있도록 구현하였다.

- 전체 등장하지 않은 코드의 전체 코드

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

<SnakeGame.hpp>

#ifndef SnakeGame_hpp
#define SnakeGame_hpp

#pragma once

#include <ncurses.h>

#include "Board.hpp"
#include "Drawable.hpp"
#include <time.h>
#include <vector>
#include <stdlib.h>
#include "Apple.hpp"
#include "Map.hpp"
#include "Empty.hpp"
#include "Snake.hpp"
#include "Grape.hpp"
#include "Scoreboard.hpp"
#include "Gate.hpp"

class SnakeGame {

    Board board;

    bool game_over;

    Grape* grape;

    Apple* apple;

    Map* map;


    Gate* gate1;

    Gate* gate2;

    Snake snake;

    Scoreboard scoreboard;

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

int score = 0;

int round;

int apple_cnt = 0;

int grape_cnt = 0;

int length = 4;

int previousRound;

int goal[4] = {7,3,1,1};

void handleNextPiece(SnakePiece next) {

    if (apple != NULL) {

        switch (board.getCharAt(next.getY(), next.getX())) {

            case 'A':

                eatApple();

                break;

            case 'B':{

                eatGrape();

                board.add(Empty(next.getY(), next.getX()));

                snake.removePiece();

                break;

            }

            case 'T':{

                contactGate();


                break;

            }

            case ' ': {

                int emptyRow = snake.tail().getY();

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        int emptyCol = snake.tail().getX();

        board.add(Empty(emptyRow, emptyCol));

        snake.removePiece();

        break;
    }

    default:

        game_over = true;

        break;

    }

}

board.add(next);

snake.addPiece(next);

}

void createApple() {

    int y, x;

    board.getEmptyCoordinates(y, x);

    apple = new Apple(y, x);

    board.add(*apple);

}

void eatApple() {

    delete apple;

    apple = NULL;

    score += 100;


    length++;

    if (apple_cnt >= 0) apple_cnt++;

    scoreboard.updateScore(score,length,apple_cnt, grape_cnt);

}

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

void createGrape() {

    int y, x;

    board.getEmptyCoordinates(y, x);

    grape = new Grape(y, x);

    board.add(*grape);

}

void eatGrape() {

    delete grape;

    grape = NULL;

    length--;

    if (length <= 3) {

        game_over = true;

        return;

    }

    if (grape_cnt >= 0) grape_cnt++;

    score -= 100;

    scoreboard.updateScore(score, length, apple_cnt, grape_cnt);

    int emptyRow = snake.tail().getY();

    int emptyCol = snake.tail().getX();

    Direction tailDirection = snake.getTailDirection();

    // 이전 꼬리 위치의 빈 공간 추가

    board.add(Empty(emptyRow, emptyCol));


    // 이전 꼬리 위치에서의 이동 방향에 따라 추가적인 빈 공간 추가

    switch (tailDirection) {

        case up:

            board.add(Empty(emptyRow + 1, emptyCol));

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        break;

    case down:

        board.add(Empty(emptyRow - 1, emptyCol));

        break;

    case left:

        board.add(Empty(emptyRow, emptyCol + 1));

        break;

    case right:

        board.add(Empty(emptyRow, emptyCol - 1));

        break;

    }

    snake.removePiece();
}

void createGate1() {

    int y, x;

    board.getChCoordinates(y, x);

    gate1 = new Gate(y, x);

    board.add(*gate1);

}

void createGate2() {

    int y, x;

    board.getChCoordinates(y, x);

    gate2 = new Gate(y, x);

    board.add(*gate2);

}

```



 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```
void contactGate() {
```

```
}
```

```
std::vector<Drawable*> drawables;
```

```
public:
```

```
SnakeGame(int height, int width, int speed = 300)
```

```
{
```

```
board = Board(height, width, speed);
```

```
int sb_row = board.getStartRow() + height;
```

```
int sb_col = board.getStartCol();
```

```
scoreboard = Scoreboard(width, sb_row, sb_col);
```

```
initialize();
```

```
}
```

```
~SnakeGame() {
```

```
delete apple;
```

```
delete grape;
```

```
delete gate1;
```

```
delete gate2;
```

```
}
```

```
std::vector<Drawable*> & getDrawables() {
```

```
return drawables;
```

```
}
```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18


```

void addObstacles(int stage) {
    switch (stage) {
    case 1: {
        int obstaclePositions[][2] = {
            {5, 10},{5, 11}, {5, 12}, {5, 13},{5, 14}, {5, 15},{5, 16},{5, 17}, {5, 18},{5, 19},{5, 20}, {5, 21},
            {5, 22},{5, 23}, {5, 24}
        };

        for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {
            int y = obstaclePositions[i][0];
            int x = obstaclePositions[i][1];
            Map* obstacle = new Map(y, x);
            board.add(*obstacle);
        }
        break;
    }
    case 2: {
        int obstaclePositions[][2] = {
            {8, 17},
            {7, 17},
            {6, 17},
            {5, 13},{5, 14}, {5, 15},{5, 16},{5, 17}, {5, 18},{5, 19},{5, 20}, {5, 21},
            {4, 17},
            {3, 17},
            {2, 17}
        };

        for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {
            int y = obstaclePositions[i][0];

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

int x = obstaclePositions[i][1];

Map* obstacle = new Map(y, x);

board.add(*obstacle);

}

break;

}

case 3: {

int obstaclePositions[][2] = {

    {4, 1},{4, 2}, {4, 3},{4, 4}, {4, 5},{4, 6},{4, 7}, {4, 8},{4, 9},{4, 10}, {4, 11},{4,12},{4, 13},{4, 14},
    {4, 15}, {4, 16},

    {4, 17}, {4, 18},{4, 19},{4, 20}, {4, 21},{4,22},{4, 23},{4, 24}, {4, 25},

    {7, 11},{7,12},{7, 13},{7, 14}, {7, 15},{7, 16},{7, 17}, {7, 18},{7, 19},{7, 20}, {7, 21},{7,22},{7,
    23},{7, 24}, {7, 25}, {7, 22}, {7, 23}, {7, 24},

    {7, 25}, {7, 26}, {7, 27}, {7, 28}, {7, 29}, {7, 30}, {7, 31}, {7, 32},

    {7, 33}, {7, 34},

    {9, 1}, {9, 2}, {9, 3}, {9, 4},

    {9, 5}, {9, 6}, {9, 7}, {9, 8},

    {9, 9}, {9, 10}, {9, 11}, {9, 12},

    {9, 13}, {9, 14}, {9, 15}, {9, 16},

    {9, 17}, {9, 18}, {9, 19}, {9, 20},

    {9, 21}, {9, 22}, {9, 23}, {9, 24},

    {9, 25}

};

for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {

int y = obstaclePositions[i][0];

int x = obstaclePositions[i][1];

Map* obstacle = new Map(y, x);

board.add(*obstacle);

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

    }

    break;

}

case 4: {

    int obstaclePositions[][2] = {

        {3, 9},{3, 10}, {3, 11},{3, 12},{3, 13}, {3, 14},

        {4, 9},

        {5, 9},                                {5, 17},{4, 17},                                {5, 22},{4, 22},

        {6, 9},                                {6, 15},{6, 16},{6, 17},{6, 18},{6, 19}, {6, 22},{6, 21},{6, 22},{6,
23},{6, 24},

        {7, 9},                                {7, 17},{8, 17},                                {7, 22},{8, 22},

        {8, 9},{8, 10}, {8, 11},{8, 12},{8, 13}, {8, 14}

    };

    for (int i = 0; i < sizeof(obstaclePositions) / sizeof(obstaclePositions[0]); i++) {

        int y = obstaclePositions[i][0];

        int x = obstaclePositions[i][1];

        Map* obstacle = new Map(y, x);

        board.add(*obstacle);

    }

    break;

}

default:

    break;

}


}

void initialize() {

    apple = NULL;

    grape = NULL;

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

gate1 = NULL;

gate2 = NULL;

board.initialize();

score = 0;

scoreboard.initialize(score);

game_over = false;

srand(time(NULL));

snake.setDirection(down);

addObstacles(round);

handleNextPiece(SnakePiece(1, 1));

handleNextPiece(snake.nextHead());

handleNextPiece(snake.nextHead());

snake.setDirection(right);

handleNextPiece(snake.nextHead());

if (apple == NULL) {
    createApple();
}

if (grape == NULL) {
    createGrape();
}

if (gate1 == NULL) {
    createGate1();
}

if (gate2 == NULL) {
    createGate2();
}
}


```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

void processInput() {
    ctype input = board.getInput();
    switch (input) {
    case KEY_UP:
    case 'w':
        if (snake.getDirection() != down) {
            snake.setDirection(up);
        }
        else {
            game_over = true;
        }
        break;
    case KEY_DOWN:
    case 's':
        if (snake.getDirection() != up) {
            snake.setDirection(down);
        }
        else {
            game_over = true;
        }
        break;
    case KEY_RIGHT:
    case 'd':
        if (snake.getDirection() != left) {
            snake.setDirection(right);
        }
        else {

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        game_over = true;
    }

    break;

case KEY_LEFT:
case 'a':

    if (snake.getDirection() != right) {
        snake.setDirection(left);
    }

    else {
        game_over = true;
    }

    break;

case 'p':

    board.setTimeout(-1);

    while (board.getInput() != 'p')

        board.setTimeout(1000);

    break;

default:

    break;


}

}

void updateState()
{
    handleNextPiece(snake.nextHead());

    if (apple == NULL) {
        createApple();
    }
}

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

    if (grape == NULL) {
        createGrape();
    }

    if (gate1 == NULL) {
        createGate1();
    }

    if (gate2 == NULL) {
        createGate2();
    }
}

void goalAchieved() {
    if (length >= goal[0] && apple_cnt >= goal[1] && grape_cnt >= goal[2]) {
        game_over = true;
    }
}

void redraw() {
    board.refresh();
    scoreboard.refresh();
}

bool isOver() {
    return game_over;
}

int getScore() {
    return score;
}

};

#endif /* SnakeGame_h */

```



 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

<Board.hpp>

#pragma once

#include "Drawable.hpp"

#include <cstdlib>

#include <vector>

class Board

{

public:

Board() {

construct(0, 0, 300);

}

Board(int height, int width, int speed) {

construct(height, width, speed);

}

void initialize() {

clear();

refresh();

}

void addBorder() {

wborder(board\_win, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD, ACS\_CKBOARD);


}

void add(Drawable drawable) {

addAt(drawable.getY(), drawable.getX(), drawable.getIcon());

}

void addAt(int y, int x, chtype ch) {

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

        mvwaddch(board_win, y, x, ch);
    }

    chtype getInput() {
        return wgetch(board_win);
    }

    void getEmptyCoordinates(int& y, int& x) {
        while ((mvwinch(board_win, y = std::rand() % height, x = std::rand() % width)) != ' ');
    }

    void getCkCoordinates(int& y, int& x) {
        int randomY, randomX;

        bool foundCk = false;

        while (!foundCk) {
            randomY = std::rand() % height;
            randomX = std::rand() % width;

            if (mvwinch(board_win, randomY, randomX) == ACS_CKBOARD) {
                foundCk = true;
            }
        }

        y = randomY;
        x = randomX;
    }

    void remove(Drawable& object) {
        // Remove the object from the board grid

        int y = object.getY();

        int x = object.getX();

        mvwaddch(board_win, y, x, ACS_CKBOARD) ;
    }

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

    }

    chtype getCharAt(int y, int x) {
        return mvwinch(board_win, y, x);
    }

    void clear() {
        wclear(board_win);
        addBorder();
    }

    void refresh() {
        wrefresh(board_win);
    }

    void setTimeout(int timeout) {
        wtimeout(board_win, timeout);
    }

    int getStartRow() {
        return start_row;
    }

    int getStartCol() {
        return start_col;
    }

    int getHeight() {
        return height;
    }

    int getWidth() {
        return width;
    }

private:

```

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

WINDOW* board_win;

int height, width, start_row, start_col;

std::vector<std::vector<ctype>>> grid;

void construct(int height, int width, int speed) {

    int xMax, yMax;

    getmaxyx(stdscr, yMax, xMax);

    this->height = height;

    this->width = width;

    start_row = (yMax / 2) - (height / 2);

    start_col = (xMax / 2) - (width / 2);

    board_win = newwin(height, width, start_row, start_col);

    setTimeout(speed);

    keypad(board_win, true);

}

};

```

```

<Main.cpp>

#include <ncurses.h>

#include <iostream>

#include "Board.hpp"

#include "SnakeGame.hpp"

#include "Drawable.hpp"

#define BOARD_DIM 12


#define BOARD_ROWS BOARD_DIM

#define BOARD_COLS BOARD_DIM * 3

int main(int argc, char** argv)

{

```

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

```

int round =1;

for (; round <= 4; round++) {

    initscr();

    refresh();


    noecho();

    curs_set(0);

    SnakeGame game(BOARD_ROWS, BOARD_COLS, 1000);

    game.addObstacles(round); //장애물 추가


    while (!game.isOver()) {

        game.processInput();

        game.updateState();

        game.redraw();

        game.goalAchieved();

    }

    getch();

    endwin();

    std::cout << "Round: " << round << std::endl;

    std::cout << "Score: " << game.getScore() << " points" << std::endl;

    std::cout << std::endl;

}


return 0;

}

```

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

-위에 코드들에서 볼 수 있듯이 게임이 종료되는 것은 **SnakeGame** 클래스의 **eatGrape** 메서드의 if 문에서 볼 수 있듯이 **Snake**의 길이가 3 이하가 됐을 때, 그리고 목표를 완성하여 모든 항목에 'v'를 받고 **score**가 300 점을 넘었을 때, 벽에 박았을 때 종료되는 것을 알 수 있다. 또한 게임이 종료되면 다른 장애물이 생성된 다음 스테이지 맵에서 플레이가 가능하다. 총 4 가지 맵에서 플레이를 한후의 최종 점수를 볼 수 있도록 결과가 출력된다.

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

### 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.

#### 프로젝트 수행에 사용한 라이브러리

- **NCURSES** : 터미널 기반의 그래픽 사용자 인터페이스를 개발하기 위한 라이브러리이다. 커서를 움직이거나, 윈도우를 생성하고, 색깔을 만들고, 마우스 관련 함수를 제공하며 텍스트 모드에서 UI를 구현할 수 있는 프레임워크를 제공한다. 이 프로젝트에서는 텍스트 모드에서 윈도우 등을 만들 수 있는 함수를 사용하기 위해서 사용되었다.
- **iostream** : C++의 표준 입출력 라이브러리이다. **iostream** 헤더 파일에는 표준 입출력에 필요한 클래스들과 객체들이 정의되어 있다. 따라서 입출력을 위하여 **cin** 과 **cout** 와 같은 객체를 사용하려면 반드시 **iostream** 파일을 포함시켜야 한다.
- **cstdlib** : C++ 표준 라이브러리의 일부로, C 언어에서 제공하는 함수들을 포함하고 있는 헤더 파일이다. 빈 좌표와 특정 문자가 있는 좌표를 생성하기 위해 **std::rand()** 함수를 사용하였는데, 이는 **cstdlib.h** 에 정의되어 있어서 사용하였다.
- **vector** : 동적 배열을 사용하기 위한 헤더이다. 게이트의 위치를 저장하는데 사용되었고, 게임 내에서 그려야 할 객체들을 관리하기 위해 사용되었다.
- **queue** : 요소를 저장하기 위한 큐 기반 인터페이스를 제공하는 컨테이너 어댑터이다. 효율적으로 뱀의 꼬리를 추가하거나 제거 하기 위해 사용되었다.
- **time.h** : 시간과 관련된 함수를 제공하는 헤더 파일이다. 사과와 포도, 게이트의 위치를 랜덤하게 배치하기 위해 **srand()** 함수를 사용하였다. **srand()** 함수를 사용하기 위해선 **time** 헤더가 필요하다.

#### 새롭게 고안한 알고리즘


- **초기 설정** : 뱀의 헤드의 방향에 따라 꼬리의 방향이 결정된다.
- **게임 진행** : 뱀은 사용자의 입력에 따라 이동방향을 변경할 수 있다.

매 시간 단위마다 뱀은 한 칸 씩 이동한다.

사과를 먹으면 뱀의 길이가 한 칸 늘어나고 점수는 100 점 씩 올라가면서 사과는 다시 새로운 위치로 이동된다.

포도를 먹으면 뱀의 길이가 한 칸 줄어들고 점수는 100 점 씩 내려가면서 포도는 다시 새로운 위치로 이동된다.

뱀이 맵의 경계선에 닿거나 몸에 닿거나, 진행방향과 반대편 방향키를 누르면 게임이 종료된다.

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

- **게임 종료** : 몸 길이가 3 보다 작아지거나, 맵의 경계선에 부딪히거나, 머리가 자신의 몸에 닿거나, 진행 방향과 반대 방향키를 누르면 종료된다.

## 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.

#### ○ Stage 중복의 오류

- 오류가 너무 많이 나서 중복된 코드를 더 작성 하는 경우가 있었다. 예를 들어 장애물 구현 초반에 게임화면에 장애물이 나타나지 않아, 여러곳에 장애물 생성함수를 집어넣었었다. 그 결과 나중에 코드가 작동할 때 장애물이 stage1,2,3,4 에 중복되어 나타나는 에러가 발생하였다. 따라서 해결하기 위해 장애물 생성함수가 필요한 곳 1 곳을 제외한 나머지는 다 삭제하였다.

#### ○ score 오류

- score 가 100 점이였다가 0 점으로 감점 되었을 때 000 으로 표기되는 오류가 있었는데 결국 해결하지 못하였다.

●

#### ○ Gate 구현 실패와 해결방안


<Gate.hpp>

```
#pragma once
#include <ncurses.h>
#include "Drawable.hpp"
class Gate : public Drawable {
public:

    Gate(int y, int x) : Drawable(y, x, ACS_DIAMOND) {
        this->icon = 'T';
    }

    void setPosition(int newY, int newX) {
        y = newY;
        x = newX;
    }
}
```



 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

};

- 위 코드와 앞서 적은 **SnakeGame.hpp** 파일에서 볼 수 있듯이 **Gate** 를 완벽히 구현해내지 못하였다. 현재까지의 진행상황이다. 맵의 벽과 장애물에 한쌍의 **Gate** 가 생성이 되는 것까지는 구현하였으나, 실행시켰을 때 **Snake** 가 **Gate** 를 통과하지 못하고 벽으로 인식하게 되는 문제가 있었고, 결국 이 문제를 해결하지 못하였다.
- 이에 우리 팀은 이러한 문제를 해결하기 위해 추가적인 공부를 통해 **Gate** 로직에 대한 이해도를 높인 후 다시 도전하여 **Gate** 까지 구현된 **Snake Game** 만들어 보고자 한다.

## 2.2.5 결과물 목록

### 작성요령 (5 점)

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

- **main.cpp**

스네이크 게임을 실행하는 파일이다.

- **Board.hpp**

스네이크 게임의 게임 보드를 나타내는 'Board'클래스를 정의하고 이 정의와 관련된 기능과 속성을 포함하고 있다.

- **SnakeGame.hpp**

스네이크 게임의 로직과 게임 보드, 사과, 포도, 맵, 뱀, 점수판, 게이트 등을 정의하고 구현하는 파일이다. 스네이크 게임의 핵심 로직을 담당하고 있으며 게임 상태를 업데이트하고 사용자 입력을 처리한다. 게임의 진행 상태를 나타내는 변수들과 게임을 초기화하는 메서드, 사용자 키보드 입력값을 처리하는 메서드, 게임 상태를 업데이트하는 메서드 등이 포함되어 있다.

- **Snake.hpp**


**Snake** 클래스와 **SnakePiece** 클래스가 있는 파일이다. 스네이크의 머리와 꼬리를 반환하는 메서드와 방향을 설정하고 반환하는 메서드, 과일에 따라 꼬리를 제거하거나 추가하는 메서드가 있다.

- **Drawable.hpp**

객체를 생성하고 해당 객체의 위치를 나타내는 x,y 좌표와 객체의 아이콘을 나타내는 icon 을 멤버 변수로 가지고 있다. **getX()**, **getY()**, **getIcon()**메서드를 통해 해당 멤버 변수의 값을 반환한다. 즉, 객체의 위치와 아이콘을 표시하는데 사용되는 파일이다.

- **Apple.hpp**

**Apple** 클래스는 **Drawable** 클래스를 상속받은 클래스로서, 게임에서 나타나는 **Apple ( A )** 객체를 나타낸다. **Drawable** 클래스를 상속받았기 때문에 위치와 아이콘을 가진다.

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

- **Grape.hpp**

Apple 과 마찬가지로 Grape 클래스는 Drawable 클래스를 상속받은 클래스로서, 게임에서 나타나는 Grape ( B ) 객체를 나타낸다. Drawable 클래스를 상속받았기 때문에 위치와 아이콘을 가진다.

- **Empty.hpp**

Apple, Grape 과 마찬가지로 Drawable 클래스를 상속받은 클래스다. 위치와 아이콘을 갖는다. 여기서 아이콘은 공백 문자로 사용되었다.

- **Scoreboard.hpp**

Scoreboard 클래스를 선언하는 파일이다. Scoreboard 클래스는 게임의 점수판을 관리한다. initialize 메서드는 초기 점수를 받아서 점수판을 초기화하고, updateScore 메서드는 점수와 길이, 사과 개수, 포도 개수를 받아서 점수판을 업데이트한다. clear 메서드는 점수판의 내용을 지우고, refresh 메서드는 점수판을 화면에 갱신한다.

- **Map.hpp**

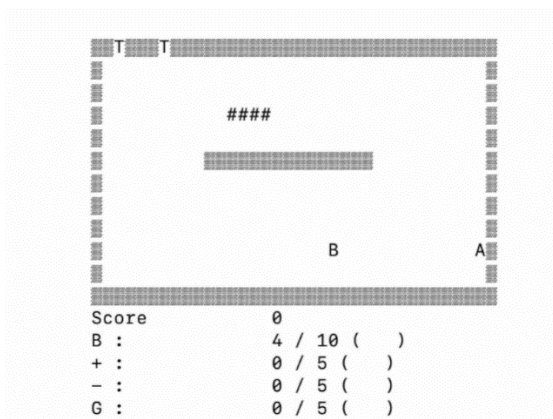
Map 클래스를 선언한 파일이다. Map 클래스는 Drawable 클래스를 상속 받고 있으며 게임 맵을 표현하는 역할을 합니다. 이 클래스는 ncurses 라이브러리의 ACS\_CKBOARD 기호를 이용하여 맵을 그리는 기능을 한다. 즉 게임에서 맵을 나타내기 위해 사용된다.


- **Gate.hpp**

Gate 클래스를 선언하는 파일이다. 이 클래스는 Drawable 클래스를 상속받고 있으며, 게임에서 통로를 표현하는 역할을 한다.

- **게임 구현 동영상**

[https://youtu.be/nZpEQ13R\\_B4](https://youtu.be/nZpEQ13R_B4)



 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

### 3 자기평가

#### 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.

#### ● 이다현 (20212006)

나의 역할은 맵의 stage 4 단계 구현, (장애물) gate 생성, poison item 을 먹었을때의 오류를 개선하였다. 프로젝트 수행시에는 ncurses 를 처음 써보았기 때문에 뭐든지 찾아서 함수를 사용해야 하는 점이 어려웠다. 또한 코드를 작성할 때 오류가 뜨면 함께 고민하고 해결해 나갔다. 그중에서 나는 poison item 을 먹었을때의 오류를 개선하기 위해 노력했다. 우리 팀은 깃헙을 쓰지 않고 구글 드라이브를 통해 코드를 공유했다. 그러다보니 어떤 코드가 업데이트 되었는지 구분이 가지 않았다. 따라서 다음 프로젝트를 진행할 때는 깃헙 사용법을 배우고 다같이 깃헙을 사용하면 코드 공유에 용이할 것 같다.

#### ● 김채현 (20221935)

기본 구조와 Grape, 진행방향과 반대로 움직이면 게임이 끝나는 것을 구현하였다. 처음 접해보는 ncurses 라이브러리가 생소했으며, 이해하는데 많은 시간이 소요되었다. 수업시간에 ncurses 에 대해 배우지 않아 수업에서 배운 내용으로만 코딩하기에 많이 어려웠다. 하지만 팀원들이 서로 도와주면서 다같이 코딩한 결과 개인적으로 만족스러운 결과물을 낼 수 있었다고 생각한다. 또한 누군가와 공유하며 코딩을 해야하기 때문에 github 활용 능력이 필요하다는 것을 깨달았다.

#### ● 김민승 (20203038)

스네이크 게임에서 점수와 게임 오버에 관한 코드를 작성하였다. 시험기간과 겹치고 게임 같은 프로젝트는 처음 진행해 보기에 어려움이 많았다. 오류를 찾아 디버깅하는 것은 물론 깃허브를 사용하지 않아 조원끼리 코드를 공유하는 것 또한 쉽지 않았다. 아쉬운 점이 있다면 100 점이었다가 0 점이 될때 0 이 아닌 000 으로 출력되는 오류가 있었는데 이 부분을 수정하지 못한 것이다. 단순히 숫자 하나를 출력하는 부분이지만 오류를 수정하는 과정은 쉽지 않았다. 하지만 조원들과 소통하고 문제를 찾고 수정하는 과정은 재미있었고 앞으로 프로젝트를 진행할 때 좋은 경험이 될 것 같다. 다음에는 깃허브도 사용하고 좀 더 공부하여 완성도 있는 프로젝트가 되었으면 좋겠다.

 <b>국민대학교</b> <b>소프트웨어부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

## ● 한윤구 (20190423)

SnakeGame 프로젝트를 진행하며 나는 Gate 구현과 보고서 초안을 담당하였다. 처음 사용하는 리눅스 환경과 코딩실력 부족으로 인해 Gate의 구현을 결국 완성하지 못하였다. 온전히 내힘으로 해결하지 못하고 불완전한 결과물이 나와 아쉽지만, 전체적으로 나 스스로의 코딩 실력의 현주소를 돌아볼 수 있어 나의 부족한 부분을 확실히 인지하게 해줬다는 점에서 도움이 되었던 것 같다. 또한 github를 조금이나마 활용해 볼 수 있어 의미있었다. 이 프로젝트를 진행해보며 github를 통해 소통했으면 더욱 좋았겠다는 생각이 들었다.

## 4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	서적	어서와 C++은 처음이지?	인피니티박스	2023	천인국	
	기사					

## 5 부록

### 작성요령 (15 점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

### 5.1 사용자 매뉴얼

#### 게임 규칙

- SnakeGame game (BOARD\_ROWS, BOARD\_COLS, 1000); 코드에 따라 스네이크는 1 초에 한 칸씩 움직인다.
- 게임이 시작되면 좌측 상단에 스네이크가 생성되고, 방향키와 w,a,s,d를 통해 스네이크 방향을 조절할 수 있다.
- 길이를 늘려주는 아이템(사과)과, 길이를 줄여주는 아이템(포도)이 맵 내부에서 무작위로 하나씩 생성된다.
- 스네이크는 벽을 통과할 수 없다.
- Score Board에는 다음과 같이 표시된다.

B : 현재 길이 / 최대 길이

+ : 획득한 Growth items (사과) 의 개수

- : 획득한 Poison items (포도) 의 개수

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

G : Gate 사용 횟수

목표가 달성 되면 괄호 사이에 체크 표시가 나타난다.

## 게임 플레이

- 스네이크는 방향키와 **w,a,s,d**로 움직일 수 있다.
- 점수를 올리기 위해 사과를 먹어야 한다.

## 게임 아이템 설명

‘A’ = Growth item 으로 몸 길이를 한 칸 증가시키며, 점수가 +100 점이 된다.

‘B’ = Poison item 으로 몸 길이를 한 칸 감소시키며, 점수가 -100 점이 된다.

‘T’ : Gate 로 한 개의 게이트로 들어가면 다른 게이트로 나오게 된다.

## 게임 종료


- 맵의 벽에 닿는 경우, 자신의 몸에 닿는 경우, 현재 진행 방향과 반대편 방향키를 누르면 게임이 종료된다.
- Score board 의 항목별로 목표치 도달시 게임이 종료된다.

## 5.2 설치 방법

### 윈도우에서 G++설치하기

1. <https://www.mingw-w64.org/>로 접속하여 downloads 선택
2. SourceForge 링크로 접속
3. 설치진행 (설치로 넘어가기전 경로를 복사해두기)
4. 설치가 완료되고 나면 윈도우 설정중 시스템 속성-> 고급-> 환경변수로 이동
5. 설치중 복사해놓은 경로를 \mingw64\에 추가하기  
(예) \Program Files\mingw-w64\x86\_64-8.1.0-posix-seh-rt\_v6-rev0\mingw64\bin  
)
6. 이 경로를 유저의 환경변수 경로에 추가하기  
Path 를 선택하고 편집 클릭  
조금전 완성하였던 경로 문자열을 적고 확인을 누르기

### 맥 OS 에서 G++ 설치하기

 <b>국민대학교</b> <b>소프트웨어학부</b> <b>C++프로그래밍</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	Snake Game	
	<b>팀 명</b>	소행소식	
	Confidential Restricted	Version 1.3	2023-06-18

1. brew 패키지 설치하기  
Homebrew 홈페이지(<https://brew.sh/>)에 접속하여 명령어를 복사한다.  
/bin/bash -c "\$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)"  
터미널에 붙여넣기 해준다.
2. Next steps 따라하기  
#echo '# Set PATH,MANPATH, etc., for Homebrew.'>> /Users/유저명/.zprofile  
#echo 'eval "\$(/opt/homebrew/bin/brew shellenv)" >> /Users/유저명/.zprofile  
#eval "\$(/opt/homebrew/bin/brew shellenv)"  
위의 세 명령어를 입력한다.  
\*M1 맥을 사용하고 있다면  
#vi ~/.zchrc  
를 입력한다.  
export "PATH=/opt/homebrew/bin:\$PATH"  
를 .zshrc 파일의 맨 아래 부분에 아래의 내용을 추가해주고 저장해준다.
3. 패키지가 설치되었는지 확인하기  
#brew --version  
을 입력하여 패키지가 정상적으로 설치 되었는지 확인
4. gcc 패키지 설치  
>brew install gcc  
\$gcc --version