




국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	Snake game 제작
팀 명	채승표
문서 제목	결과보고서

Version	1.2
Date	2023-05-26

팀원	채승표
----	-----

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26


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 "채승표"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "채승표"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	최종보고서-Snake game.doc
원안작성자	채승표
수정작업자	채승표

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2023-05-26	채승표	1.0	최초 작성	

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

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	12
2.2.3	활용/개발된 기술	32
2.2.4	현실적 제한 요소 및 그 해결 방안	32
2.2.5	결과물 목록	33
3	자기평가	34
4	참고 문헌	34
5	부록	35
5.1	사용자 매뉴얼	35
5.2	설치 방법	35

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

1 개요

1.1 개발 방법

- 개발OS : Ubuntu
- 언어: C++

1.2 사용 라이브러리

- Ncurses 라이브러리

1.3 설치 방법

- wget <https://ftp.gnu.org/pub/gnu/ncurses/ncurses-6.2.tar.gz> -P ~/Downloads
- tar xzfv ncurses-6.2.tar.gz
- sudo apt install make
- ./configure
- make
- sudo make install
- ls -la/opt/ncurses

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2 개발 내용 및 결과물

2.1 목표

작성요령 (10점)

프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.


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

1 단계 Map의 구현 목표

- Map의 크기는 30*30으로 구현한다.
- 외벽 부분은 검은색과 회색인 WALL과 Immune_WALL로 채운다.
- item이나 gate, snake에 각각의 색을 입혀 Map에 표현한다.
- Stage마다 Wall의 배치를 바꿔 여러 개의 Map을 제작한다.

2 단계 Snake 표현 및 조작 목표

- Snake의 기본 크기는 3이다.
- Snake는 Head 방향으로 이동한다.
- Snake는 방향키의 입력마다 Head의 방향이 설정된다.
- Head방향과 정반대로 방향을 입력하면 (tail 방향) 게임이 종료된다. (fail)
- Snake가 벽과 자신의 몸에 닿으면 게임이 종료된다. (fail)
- Snake는 일정 시간마다 이동한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

3 단계 Item 요소의 구현 목표

- Item 은 일정 시간마다 3 개씩 Snake 나 Wall 의 위치가 아닌 랜덤 좌표에 리스폰된다.
- Item 이 재생성 될 때 이미 생성된 Item 은 사라진다.
- Growth 와 Poison 각각의 개수는 랜덤으로 합 3 개의 item 이 리스폰된다.
 - Growth: Snake 길이 1 증가
 - Poison: Snake 길이 1 감소
- 아이템은 획득 시 Map 에서 사라진다.

4 단계 Gate 요소의 구현 목표

- 랜덤한 Wall 2 곳에 한 쌍의 Gate 가 출현하며 Snake 의 Head 가 Gate 진입하면 다른 쪽의 Gate 로 이동한다.
- Gate 로 진입하는 Snake 의 방향과 Gate 의 좌표에 따라 다른 쪽의 Gate 에서 나오는 Snake 의 방향을 설정한다.

5 단계 점수 요소의 구현 목표

- Map 의 우측에 ScoreBoard 와 Mission 을 나타내는 화면을 구성한다.
- 특정 요소의 (ex) grow_item) Mission 을 달성하면 ScoreBoard 의 요소 옆에 v 체크 표시가 된다.
- 모든 미션을 성공하면 문구가 뜨고 다음 stage 로 이동한다.
- 모든 stage 를 성공하면 문구가 뜨고 게임이 종료된다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

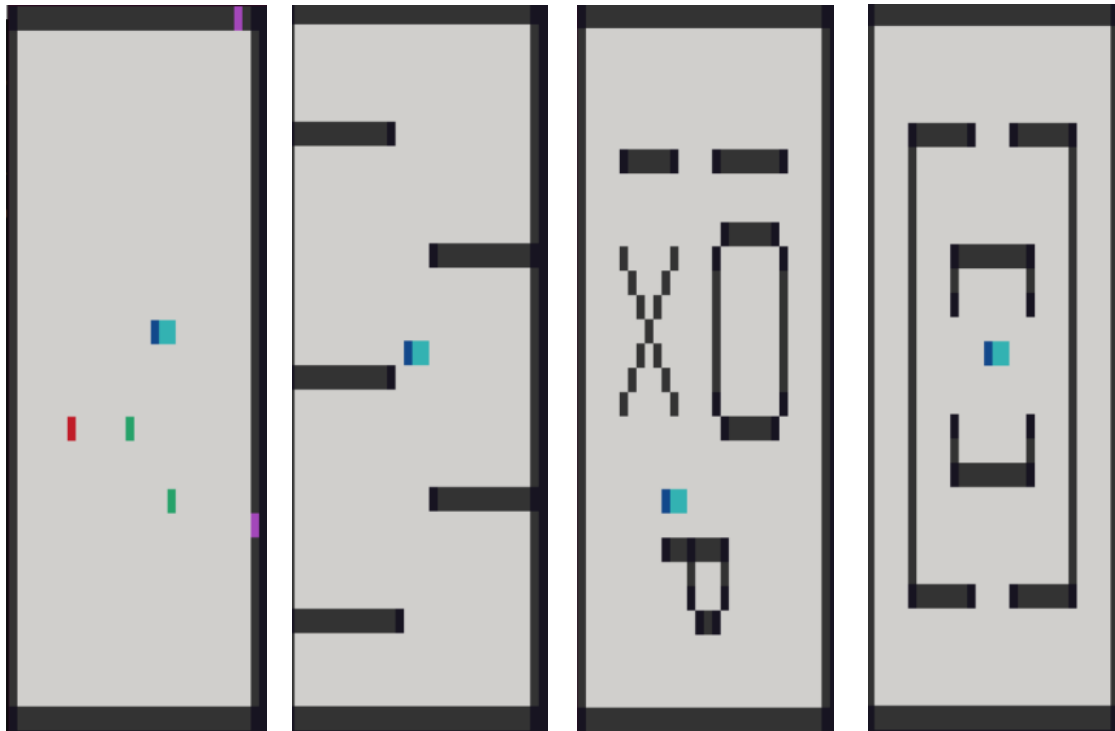
2.2 개발 내용 및 결과물

2.2.1 개발 내용

1단계 Map 의 구현

- Map 을 30*30으로 제작하기 위해 newwin(30,30,0,0)으로 새로운 윈도우를 만들었습니다.
- Map 은 3차원 배열로 설정하여 4개의 stage 에 각각 다른 map 을 구현하였습니다.
- 마우스의 커서와 입력을 콘솔에 보이지 않기 위해 curs_set(0)과 noecho() 함수를 사용하였습니다.
- Map 의 요소들에 색을 입히기 위해 init_color 와 init_pair 함수로 팔레트를 생성할 color_init() 함수를 제작하였습니다.
- 2중 for 문을 이용해 map 배열의 값에 따라 요소들을 표현하게 구현하였습니다.
- Map 의 생성 후 wrefresh 함수로 map 이 콘솔에 보이게 구현하였습니다.

Stage 별 map 이미지



 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2단계 Snake 표면 및 조작

2.1 Snake.cpp

- Snake.cpp에서는 Snake 생성, 방향키에 따른 Head 방향 설정, Snake의 움직임에 따른 상호작용, 자신의 길이를 측정하는 기능들을 가진 함수를 제작하였습니다.

2.2 Snake 생성


- Snake는 int x,y 2개의 변수를 가진 클래스 타입의 vector로 설정하였습니다.
- 처음 head 방향은 왼쪽 방향으로 설정하였습니다.
- stage별 Snake의 첫 스폰 좌표를 제작하였습니다.
- 스폰 장소의 x,y 좌표를 이용하여 snake 배열에 값을 넣어 주었습니다.
- for문을 이용하여 map에 stage 레벨과 snake 값을 입력하여 map에 snake가 생성되도록 구현하였습니다.

2.3 head 방향 설정

- 입력을 받을 wgetch 함수를 원활하게 사용하기 위해 keypad 함수로 방향키의 입력을 허용하였고 noecho 함수로 입력이 없어도 출력이 되게끔 구현하였습니다.
- Wgetch로 입력을 받고 switch문을 이용해 입력에 따라 head의 방향을 설정하였습니다. 이 때 현재의 head 방향과 입력 방향이 정 반대면 게임이 종료됩니다.
- 방향키와 같은 방향으로 head 방향이 설정됩니다.
-

2.4 Snake의 몸 길이

- Score 보드 판에 구현할 Snake의 최대 몸 길이를 구하기 위해 제작했습니다.
- 현재 자신의 몸길이를 .size()를 이용하여 구한 뒤 자신의 최대 몸 길이를 update합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2.5 Snake의 움직임

- Snake가 각 방향으로 1칸씩 움직이기 위해 4개의 값(상하좌우)을 가진 2차원 배열을 생성하였습니다. 이 배열을 이용하여 head 방향에 따라 Snake의 다음 좌표를 구합니다.
- Snake의 다음 위치가 벽이나 자신의 몸이면 게임이 종료됩니다.
- Snake의 다음 위치가 growth item이면 몸 길이가 1 증가하고 growth_item이 1 증가합니다.
- Snake의 다음 위치가 posion item이면 몸 길이가 1 감소하고 posion_item이 1 증가합니다. 이 때 자신의 몸 길이가 3인 상태에서 다음 좌표가 posion item이면 게임이 종료됩니다.
- Snake의 다음 위치가 gate라면 head 방향을 재설정하고 snake의 다음 위치를 구합니다.
- Snake의 다음 위치에 아무런 요소가 없다면 head 방향으로 이동합니다.
- 각각의 이동은 usleep함수를 이용하여 약 0.15초씩 움직이도록 설정하였습니다.

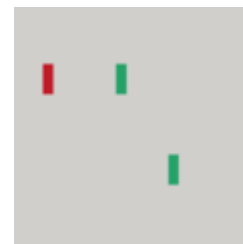
Snake의 head 부분은 파란색, body 부분은 민트색으로 구현하였습니다.



3단계 Item 요소의 구현

- Rand 함수를 이용하여 item의 생성 위치를 받아오게 설정하였습니다. 이 때 와일문을 이용하여 item이 벽이나 snake가 아닌 장소에 생성되도록 구현하였습니다.
- Rand 함수를 이용하여 총 3개의 item 중 growth와 posion의 각각의 개수를 합이 3개가 되는 랜덤 값으로 받아옵니다.
- 3개의 item은 위에 snake를 틱마다 움직이기 위해 사용했던 usleep 함수에 for문을 이용하여 약 7.5초마다 재생성되고 없어지길 반복합니다.

Growth item=초록색, posion item=빨간색으로 구현하였습니다

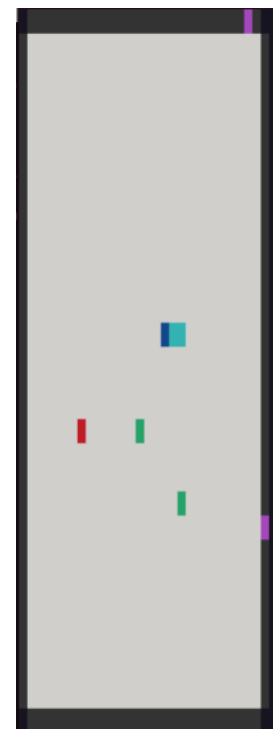


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

4 단계 Gate 요소의 구현

- gate 는 진입하면 다른 쪽의 gate 로 나오게 됩니다. 이를 위해 게이트는 한 쌍이 존재하게끔 size 가 2인 배열로 구현하였습니다. 이를 이용하여 어느 한 gate 에 진입 시 나가는 gate 와 진입 gate 를 구분할 수 있습니다.
- Gate 의 진입 시 snake 의 다음 이동 좌표는 gnext_x 와 gnext_y 의 전역 변수에 저장됩니다.
- Gate 에서 나올 시 Head 방향이 재설정 되는데 이 때 Gate 가 외곽 벽이면 가운데로 오는 방향으로 head 방향을 설정하였고 진출 방향이 좌우 방향만 존재한다면 현재 Head 방향이 오른쪽과 위쪽일 경우 Head 방향이 오른쪽으로 바뀌게 되고 왼쪽과 아래쪽일 경우 Head 방향이 왼쪽으로 바뀌게 된다. 진출 방향이 상하 방향만 존재한다면 현재 Head 방향이 오른쪽과 위쪽일 경우 Head 방향이 위쪽으로 바뀌게 되고 왼쪽과 아래쪽일 경우 Head 방향이 아래쪽으로 바뀌게 된다. 진출 방향이 자유로운 경우 Head 방향은 자유롭게 되고 예외적인 상황인 진출방향이 1개인 경우나 3개인 경우의 벽은 Immune Wall 로 설정하여 gate 가 출현하지 않게 설정하였습니다.
- Gate 는 Immun_Wall 이 아닌 일반적인 벽에서 생성됩니다. Item 과 같이 약 7.5초마다 재생성되고 없어지길 반복하며 게이트가 사라질 때는 gate.clear 를 해줬습니다.
- Gate 에 입장 시 gnext_x 와 y 의 값이 snake 배열의 마지막 요소 값과 같을 때 게이트를 모두 통과 하였으므로 gate 를 Wall 로 바꾸고 clear 해주었습니다. gnext_x ,y 변수를 초기화해주고 gate_cnt 가 1증가합니다.
- Gate 가 통과 도중 여부를 알 수 있는 gate_pass 변수를 사용하여 gate 를 통과 중 일 시 gate 가 재생성되지 않고 유지되게끔 설정하였습니다..

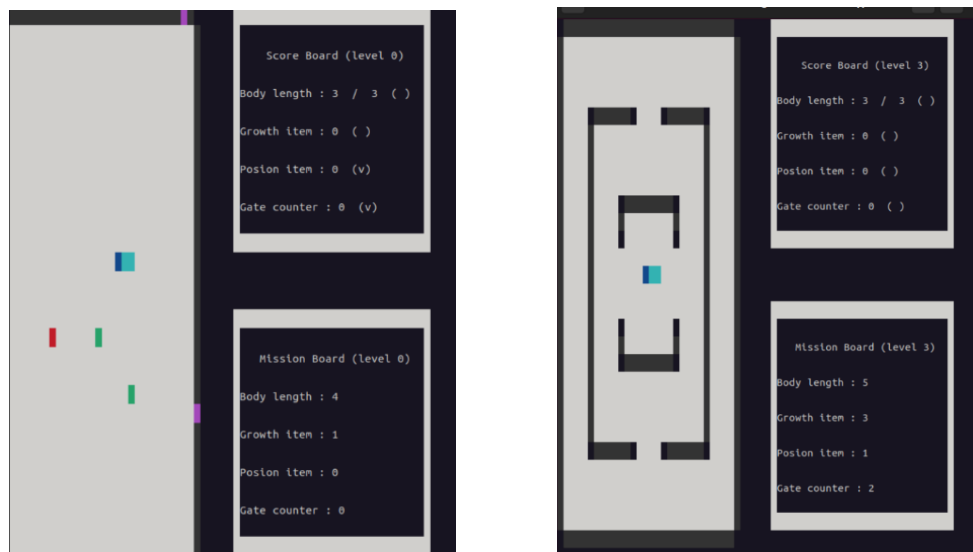
Gate 는 보라색으로 구현하였습니다.



 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

5단계 점수 요소의 구현

- 점수판과 미션판은 newwin 을 통해 각각의 윈도우로 제작하였습니다. 이 때 생성 좌표는 게임 윈도우의 우측으로 설정하였습니다.
- 점수판은 게임을 진행하는 도중 획득한 아이템의 개수, 게이트의 통과 횟수, 자신의 몸길이를 기록합니다. 각각의 요소들은 게임 상황에 따라 실시간으로 update 되도록 설정하였습니다,
- 미션판은 목표 아이템의 개수, 게이트의 통과 횟수, 자신의 몸길이를 나타냅니다. 미션은 2차원 배열로 이루어져 있으며 스테이지별 미션이 기록되어 있습니다.
- 각각의 미션을 충족했다면 점수판의 충족한 요소에 if 문을 이용해 v 체크표시를 합니다.
- 미션 전부 클리어 시 sucess 라는 문구가 화면에 출력되며 다음 스테이지로 이동합니다. 만약 마지막 stage 의 미션을 clear 한 상황이라면 game clear 라는 문구가 화면에 출력됩니다.



 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2.2.2 시스템 구조 및 설계도

0. main 문과 헤더파일

프로그램의 전체적인 동작을 알아보기 위하여 헤더파일로 제작한 함수들의 대략적인 기능을 소개하겠습니다.

0.1 사용 할 함수와 전역 변수를 선언해둔 function.h

```
#pragma once

#include <ncurses.h>
#include <iostream>
#include <locale.h>
#include <time.h>
#include <string>
#include <vector>
#include <unistd.h>
#include "map.h"
using namespace std;

#define COLOR_GRAY 8
#define COLOR_MINT 9

const int BLANK = 0;
const int WALL = 1;
const int Immune_WALL = 2;
const int HEAD = 3;
const int BODY = 4;
const int GROWTH = 5;
const int POISON = 6;
const int GATE = 7;

const int UP = 0;
const int LEFT = 1;
const int RIGHT = 2;
const int DOWN = 3;
```

- 헤더파일의 중복 선언을 방지하기 위해 pragma once 를 선언한 뒤 헤더파일을 작성하였습니다.
- 코드의 가독성을 위해 map 에 존재하는 요소들과 Head 의 방향을 const int 형으로 선언하였습니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

```

class SNAKE{
public:
    SNAKE();
    SNAKE(int a, int b);

public:
    int x, y;
};
SNAKE::SNAKE() : x(0) ,y(0)
{
}
SNAKE::SNAKE(int a, int b) : x(a), y(b)
{
}

```

- Snake class 를 제작하고 생성자를 구현하였습니다.

```

void map_set();
void color_init();
void item_gate_clear();

```

- Map 관련 함수입니다. map_set() 함수는 map 을 화면에 표현하는 기능을 합니다.
- color_init() 함수는 map 에 필요한 색상들을 팔레트에 저장하는 기능을 합니다.
- item_gate_clear() 함수는 map 에 생성되어 있는 item 과 gate 를 지우는 기능을 합니다.

```

void make_s();
void set_direction();
void move_s();
void check_body();

```

- Snake 관련 함수입니다. make_s() 함수는 stage 가 시작될 때 snake 의 초기 형태를 구현하는 기능을 합니다.
- set_direction() 함수는 방향키에 따라 Head 의 방향을 설정해주고 입력이 Head 의 방향과 반대일 때 게임을 종료시키는 기능을 합니다.
- Item_gate_clear() 함수는 map 에 생성되어 있는 item 과 gate 를 지우는 기능을 합니다.
- move_s() 함수는 snake 의 다음 이동 좌표를 출력하고 item 과 gate 같은 map 의 요소들과의 상호 작용을 구현하는 기능을 합니다.
- check_body() 함수는 snake 의 size 를 실시간으로 알아내서 score 판에 표기할 최대 snake 의 길이를 update 하는 기능을 합니다.

```

void add_item();

```

- item 관련 함수입니다. add_item() 함수는 map 에 item 을 랜덤 좌표생성하는 기능을 합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

```
void add_gate();
void gate_direction();
```

- gate 관련 함수입니다. add_gate() 함수는 랜덤한 Wall 에 gate 한 쌍을 랜덤 생성하는 기능을 합니다.
- gate_direction() 함수는 snake 가 gate 진입 하는 순간에 Head 의 방향을 정하는 기능을 합니다.

```
void sub_win();
void game_set();
void gameover();
void set_up();
```

- score 판과 mission 판, 인터페이스 관련 함수입니다. sub_win() 함수는 score 판과 mission 판을 구현하였고 정보들을 실시간으로 update 하는 기능을 합니다.
- game_set() 함수는 stage clear 시 sucess 문구를, game clear 시 game clear 문구를 윈도우에 출력하는 기능을 합니다.
- gameover() 함수는 특정 조건 달성 시 fail 문구를 윈도우에 출력하고 종료되는 기능을 합니다.
- set_up() 함수는 game 의 start 화면을 출력합니다.

```
vector<SNAKE> snake;
vector<SNAKE> gate;

WINDOW *my_win,*sub1,*sub2;

int
head_direction,stage_level,growth_item,posion_item,g_flag,out_gate,gate_cnt,gnext_x,gnext_y;
int clear_stage,gate_pass,body_length=3;
```

- 전역 변수입니다. Snake 와 gate 는 vector 로 구현하였으며 window 는 게임용, 미션판용 스코어판용으로 3 개의 변수를 선언하였습니다. Int 형 변수로는 score 판에 표기할 변수들, head 의 방향, flag 들을 선언하였습니다.

```
int dx[4][2] = {{0, -1}, {-1, 0}, {1, 0}, {0, 1}};
int goal[4][4] = {{4,1,0,0},{4,1,1,1},{5,3,1,1},{5,3,1,2}};
```

- dx 배열은 상하좌우를 표현한 값으로 snake 의 현 좌표와 head 방향에 따른 dx 배열을 더해 snake 의 다음 위치를 구할 수 있게 선언하였습니다.
- goal 배열은 미션창에 미션을 의미합니다. 4 개의 stage 에 맞는 4 가지 미션을 선언하였습니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

0.2 map 을 구현할 map.h

- Map 은 3 차원 배열로 제작하였습니다. Map 배열은 4 개의 stage 와 1 개의 시작화면을 구현할 수 있도록 설정하였습니다.
- Map 에 0 은 빈 공간, 1 은 wall, 2 는 immune wall 로 표현합니다. 각각의 색은 흰색, 회색, 검은색으로 설정하였습니다.

0.3 main 문

```
#include "map.cpp"
#include "snake.cpp"
#include "item.cpp"
#include "gate.cpp"
#include "ui.cpp"
```

각각의 기능별로 작성해둔 cpp 파일을 include 합니다.

```
int main(){
    initscr();
    start_color();
    color_init();
    noecho();
    curs_set(FALSE);

    my_win=newwin(30,30,0,0);
    wrefresh(my_win);

    keypad(my_win, TRUE);

    sub1=newwin(13,30,0,35);
    sub2=newwin(13,30,16,35);

    nodelay(my_win,TRUE);
    set_up();
```

- main 문의 초기 설정입니다. 커서, 키입력 등 각종 설정, 윈도우 제작, 시작화면을 띄웁니다.



```
- while(1){  
-     head_direction=LEFT;  
-     map_set();  
-     sub_win();  
-     make_s();  
-     map_set();  
-     while(1){  
-         item_gate_clear();  
-         map_set();  
-         add_item();  
-         add_gate();  
-         map_set();  
-         for(int i=0;i<50;i++){  
-             set_direction();  
-             move_s();  
-             map_set();  
-             sub_win();  
-             if(clear_stage)  
-                 break;  
-             usleep(150000);  
-         }  
-         if(clear_stage)  
-             break;  
-     }  
-     game_set();  
- }  
- return 0;  
- }
```

- main 문의 와일문입니다. 3 중 반복문중 가장 끝에있는 for 문에서는 약 0.15 초마다 Head 의 방향을 정하고 snake 의 움직임을 구현하며 그 정보들을 score 판에 update 시킵니다. Stage 가 clear 되거나 50 번 반복하여 약 7.5 초가 지나면 종료됩니다.
- 2 번째 와일문입니다. 약 7.5 초가 지나 for 문을 탈출하면 item 과 gate 를 지우고 재생성합니다. Stage 가 clear 될 시 종료됩니다.
- 1 번째 와일문입니다. Snake 의 초기 방향을 정하고 snake 를 map 에 생성합니다. Stage 가 clear 될 시 다음 stage 로 넘어가거나 게임이 clear 됩니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

1 단계 map 의 구현 (map.cpp)

1.1 color

```
void color_init(){
    // 글씨 배경 순서
    init_color(COLOR_GRAY, 200, 200, 200);        // 회색으로 설정
    init_color(COLOR_MINT, 200, 700, 700);
    init_pair(1, COLOR_GRAY, COLOR_GRAY);          // Wall
    init_pair(2, COLOR_BLACK, COLOR_BLACK);        // Immune Wall
    init_pair(3, COLOR_BLUE, COLOR_BLUE);          // Snake Head
    init_pair(4, COLOR_MINT, COLOR_MINT);          // Snake Body
    init_pair(5, COLOR_GREEN, COLOR_GREEN);        // Growth Item
    init_pair(6, COLOR_RED, COLOR_RED);            // Poison Item
    init_pair(7, COLOR_MAGENTA, COLOR_MAGENTA);    // Gate
    init_pair(8, COLOR_WHITE, COLOR_WHITE);        // Blank
    init_pair(9, COLOR_WHITE, COLOR_BLACK);
    init_pair(10, COLOR_BLACK, COLOR_WHITE);
}
```

Map 에 색상을 입히기 위해 init_pair 함수로 각 팔레트를 지정해 줬습니다. 이 때 mint 와 gray 색상은 기본적으로 제공되지 않는 색이기에 init_color 함수를 사용하여 색을 지정해주었습니다.

1.2 item 과 gate 를 map 에서 지우기

```
void item_gate_clear(){
    for(int i=0;i<30;i++){
        for(int j=0; j<30; j++){
            if(map[stage_level][i][j] == GROWTH || map[stage_level][i][j] == POISON)
                map[stage_level][i][j] = 0;
            if(map[stage_level][i][j] == GATE&&gate_pass==0)
                map[stage_level][i][j] = WALL;
        }
        if(gate_pass==0)
            gate.clear();
    }
}
```

Item 과 gate 가 재생성 될 시 기존의 item 과 gate 를 제거하는 함수입니다. 기존 item 의 좌표는 0 의 값을 주었고 gate 는 snake 의 gate 통과 도중 여부를 알 수 있는 gate_pass 변수를 이용하여 gate 통과 중이 아닐 시에만 wall 로 바꾸도록 구현하였습니다.

1.3 map 표현



```
void map_set(){
for(int i=0;i<30;i++){
for(int j=0;j<30;j++){
if(map[stage_level][i][j] == 0){
    wattron(my_win, COLOR_PAIR(8));
    mvprintw(my_win, i, j, "%d", map[stage_level][i][j]);
    wattroff(my_win, COLOR_PAIR(8));
}
else if(map[stage_level][i][j] == WALL){
    wattron(my_win, COLOR_PAIR(1));
    mvprintw(my_win, i, j, "%d", WALL);
    wattroff(my_win, COLOR_PAIR(1));
}
else if(map[stage_level][i][j] == Immune_WALL){
    wattron(my_win, COLOR_PAIR(2));
    mvprintw(my_win, i, j, "%d", Immune_WALL);
    wattroff(my_win, COLOR_PAIR(2));
}
else if(map[stage_level][i][j] == HEAD){
    wattron(my_win, COLOR_PAIR(3));
    mvprintw(my_win, i, j, "%d", HEAD);
    wattroff(my_win, COLOR_PAIR(3));
}
else if(map[stage_level][i][j] == BODY){
    wattron(my_win, COLOR_PAIR(4));
    mvprintw(my_win, i, j, "%d", BODY);
    wattroff(my_win, COLOR_PAIR(4));
}
else if(map[stage_level][i][j] == GROWTH){
    wattron(my_win, COLOR_PAIR(5));
    mvprintw(my_win, i, j, "%d", GROWTH);
    wattroff(my_win, COLOR_PAIR(5));
}
else if(map[stage_level][i][j] == POISON){
    wattron(my_win, COLOR_PAIR(6));
    mvprintw(my_win, i, j, "%d", POISON);
    wattroff(my_win, COLOR_PAIR(6));
}
else if(map[stage_level][i][j] == GATE){
    wattron(my_win, COLOR_PAIR(7));
    mvprintw(my_win, i, j, "%d", GATE);
    wattroff(my_win, COLOR_PAIR(7));
}
}
wrefresh(my_win);
}
```

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

```
}

```

map_set() 함수입니다. 2 중 for 문을 사용하여 각 배열의 값이 특정 요소와 같다면 watttron 함수를 이용해 글자와 배경색상을 가진 팔레트를 불러와 값에 특정 색을 표현합니다. 값 출력 후 map 의 update 를 위해 wrefresh 함수를 사용하였습니다.

2 단계 snake 표면 및 조작 (snake.cpp)

2.1 snake 생성

```
void make_s(){
    int s_location[4][1][2] = {
        {{14, 14}},
        {{14, 14}},
        {{10, 20}},
        {{14, 14}}
    };
    snake.clear();
    for(int i=0;i<3;i++)
        snake.push_back(SNAKE(s_location[stage_level][0][1],s_location[stage_level][0][0]+i));
    map[stage_level][snake[0].x][snake[0].y] = HEAD;
    for(int i=1; i<3; i++)
        map[stage_level][snake[i].x][snake[i].y] = BODY;
    map_set();
    nodelay(my_win,FALSE);
    wgetch(my_win);
    nodelay(my_win,TRUE);
}
```

Snake 를 생성하는 함수입니다. Stage 첫 진입 시에만 작동합니다.

먼저 이전 snake 벡터를 초기화합니다.

Snake 의 생성 좌표를 s_location 배열로 구현하였고 이 배열을 이용하여 배열의 값을 head 로 설정 후 몸 길이가 3 이 되도록 snake 에 push_back 하였습니다.

이 후 snake 벡터 값을 이용하여 map 에 head 와 body 를 생성시켰고 map_set 을 이용해 map 을 update 시켰습니다.

Map 이 update 되고 난 후 게임이 즉시 시작하지 않도록 nodelay 함수를 0 으로 주어 키의 입력이 있을 때까지 시작하지않고 대기하도록 설정하였습니다.

입력이 들어오면 이 후 게임이 원활하게 진입되도록 nodelay 함수의 값을 1 로 바꿔주었습니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2.2 snake 의 head 방향 설정

```

void set_direction(){

    int ch = wgetch(my_win);
    switch(ch){
        case KEY_UP:
            if(head_direction==DOWN)
                gameover();

            head_direction=UP;
            break;

        case KEY_DOWN:
            if(head_direction==UP)
                gameover();
            head_direction=DOWN;
            break;

        case KEY_RIGHT:
            if(head_direction==LEFT)
                gameover();

            head_direction=RIGHT;
            break;

        case KEY_LEFT:
            if(head_direction==RIGHT)
                gameover();

            head_direction=LEFT;
            break;
    }
}

```

Snake 의 방향은 wgetch 를 이용하여 방향키의 입력으로 설정됩니다. nodelay 값을 1 로 설정하였기에 입력이 없어도 게임이 멈추지 않고 진행됩니다.

조작 키에 맞게끔 head 방향을 설정하였고 만약 현방향과 조작방향이 정반대일시 gameover 됩니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2.3 snake 의 움직임 구현

```
void move_s(){
    int next_x= snake[0].x + dx[head_direction][1];
    int next_y= snake[0].y + dx[head_direction][0];
```

snake 의 다음 좌표는 앞서 선언해둔 dx 배열의 값과 head 의 방향을 이용하여 구합니다.

```
if(map[stage_level][next_x][next_y]==WALL||map[stage_level][next_x][next_y]==IMMUNE_WALL||map[stage_level][next_x][next_y]==BODY)
    gameover();
```

snake 의 다음 좌표가 wall, body, immune wall 일 시 game over 됩니다.

```
else if(map[stage_level][next_x][next_y]==GROWTH){
    growth_item++;
    snake.insert(snake.begin(),SNAKE(next_x,next_y));
    map[stage_level][snake[0].x][snake[0].y]=HEAD;
    map[stage_level][snake[1].x][snake[1].y]=BODY;
    check_body();
    wrefresh(my_win);
}
```

snake 의 다음 좌표가 growth item 일 시 growth item 변수가 증가합니다. Snake 벡터에 다음 좌표 값을 insert 하고 다음 좌표가 head 가 되도록 update 합니다.

또 최대 몸길이를 구하기 위해 growth item 을 먹었을 시 check body 함수를 불러옵니다.

```
else if(map[stage_level][next_x][next_y]==POISON){
    if(snake.size()==3)
        gameover();
    posion_item++;
    snake.insert(snake.begin(),SNAKE(next_x,next_y));
    map[stage_level][snake[snake.size()-1].x][snake[snake.size()-1].y]=0;
    snake.pop_back();
    map[stage_level][snake[snake.size()-1].x][snake[snake.size()-1].y]=0;
    snake.pop_back();
    map[stage_level][snake[0].x][snake[0].y] = HEAD;
    map[stage_level][snake[1].x][snake[1].y] = BODY;
    wrefresh(my_win);
}
```

snake 의 다음 좌표가 growth item 일 시 posion item 변수가 증가합니다. Snake 벡터에 다음 좌표 값을 insert 하고 꼬리와 그 다음 꼬리를 pop 하여 snake 를 update 해줍니다.

또 몸길이가 3 일 시 posion item 을 먹으면 gameover 됩니다.



```
else if(map[stage_level][next_x][next_y]==GATE){
    if(next_x==gate[0].x&&next_y==gate[0].y)
        out_gate=1;

    else if(next_x==gate[1].x&&next_y==gate[1].y)
        out_gate=0;


    gate_direction();
    gnext_x=gate[out_gate].x+dx[head_direction][1];
    gnext_y=gate[out_gate].y+dx[head_direction][0];
    map[stage_level][snake[snake.size()-1].x][snake[snake.size()-1].y]=0;
//맨 끝 delete
    snake.pop_back();
    snake.insert(snake.begin(), SNAKE(gnext_x, gnext_y));
    map[stage_level][snake[0].x][snake[0].y] = HEAD;
    map[stage_level][snake[1].x][snake[1].y] = BODY;
    wrefresh(my_win);
    gate_pass=1;
}
```

snake 의 다음 좌표가 gate 일 시 out gate 를 설정합니다. Gate_direction 함수를 사용해 gate 통과 시 head 방향을 update 해주고 그에 맞게끔 out gate 의 좌표를 이용해 gate 통과 시 나올 좌표를 설정해줍니다. 이 때 gate 가 통과중임을 알 수 있게 gate_pass 값을 1 로 설정합니다. 이 후 head 를 update 하고 tail 을 지웁니다.

```
else{

    map[stage_level][snake[snake.size()-1].x][snake[snake.size()-1].y]=0;
//맨 끝 delete
    snake.pop_back();
    snake.insert(snake.begin(), SNAKE(next_x, next_y));
    map[stage_level][snake[0].x][snake[0].y] = HEAD;
    map[stage_level][snake[1].x][snake[1].y] = BODY;
    wrefresh(my_win);
}
```

Snake 가 아무런 요소도 만나지 않을 경우입니다. Snake 의 head 를 update 하고 tail 을 지웁니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

```
if(snake[snake.size()-1].x==gnext_x&&snake[snake.size()-1].y==gnext_y){
    map[stage_level][gate[0].x][gate[0].y] = WALL;
    map[stage_level][gate[1].x][gate[1].y] = WALL;
    gate_cnt++;
    gate.clear();
    gnext_x=gnext_y=0;
    gate_pass=0;
}
```

Snake의 꼬리 부분이 out 게이트로 나왔을 때의 좌표와 같아진다면 gate 통과가 완료 된 것으로 gate_cnt를 1 증가시키고 gate의 통과가 끝났다는 것을 알리기 위해 gate_pass를 0으로 설정합니다.

2.4 snake의 최대 몸 길이

```
void check_body(){
    if(snake.size()>body_length)
        body_length=snake.size();
}
```

Snake의 현재 길이가 최대 몸길이 보다 클 시 최대 몸길이를 update 합니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

3 단계 item 요소의 구현 (item.cpp)

3.1 item 생성

```
void add_item(){
    int n,x,y;
    n=x=y=0;
    srand(time(NULL));
    int cnt = rand() % 4;

    while(n<3){
        x = rand()%30;
        y = rand()%30;

        if(map[stage_level][x][y] == 0){
            if(n<cnt){
                map[stage_level][x][y] = GROWTH;
                n++;
            }
            else if(n!=3&& n>=cnt){
                map[stage_level][x][y] = POISON;
                n++;
            }
        }
    }
    wrefresh(my_win);
}
```

총 3 개의 item 중 growth 와 posion 각각의 item 들의 개수를 랜덤으로 정하기 위해 cnt 의 값을 0~3 의 랜덤 값으로 설정하였습니다.

Item 의 총 개수인 n 이 3 개가 될 때까지 while 문으로 좌표를 랜덤으로 입력받고 입력받은 좌표가 빈 공간일 시 item 이 생성되고 n 이 1 증가합니다.

이 때 0~3 값을 지닌 cnt 값이 현재 n 값보다 크다면 growth item 을, 작다면 poison item 을 출력하도록 설정하여 어느 정도의 랜덤성을 부여하였습니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

4 단계 gate 요소의 구현 (gate.cpp)

4.1 gate 생성

```
void add_gate(){
    int x1,x2,y1,y2;
    x1=x2=y1=y2=0;
    srand(time(NULL));
    while(1){
        x1=rand()%30;
        y1=rand()%30;
        x2=rand()%30;
        y2=rand()%30;
        if(map[stage_level][x1][y1]==WALL&&map[stage_level][x2][y2]==WALL)
            if(x2!=x1|y2!=y1)
                break;
    }
    if(gate_pass==0){
        gate.push_back(SNAKE(x1,y1));
        gate.push_back(SNAKE(x2,y2));
        map[stage_level][x1][y1]=GATE;
        map[stage_level][x2][y2]=GATE;}
}
```

Gate 를 생성하는 함수다. 무한와일문을 돌려 2 개의 gate 가 특정 좌표값을 가질 시 생성되도록 구현하였습니다. 랜덤 좌표 값이 wall 이고 이 두개의 좌표가 서로 겹치지 않아야 와일문이 종료되며 gate 가 생성됩니다.

이 때 snake 가 gate 를 통과 중이라면 gate 의 재생성이 안되게끔 gate_pass 변수를 사용하였습니다.

4.2 gate 통과 시 head 의 방향

```
void gate_direction(){
    if(gate[out_gate].x==0)
        head_direction=DOWN;
    else if(gate[out_gate].x==29)
        head_direction=UP;
    else if(gate[out_gate].y==0)
        head_direction=RIGHT;
    else if(gate[out_gate].y==29)
        head_direction=LEFT;
```

out gate 가 가장자리 라면 가운데로 오게끔 head 방향을 설정하였습니다.

out gate 가 왼쪽 가장자리라면 head 의 방향은 오른쪽, 오른쪽일 시 왼쪽, 위쪽일 시 아래쪽, 아래쪽일 시 위쪽을 향하도록 head 방향을 설정하였습니다.



```
else{
    if((map[stage_level][gate[out_gate].x-1][gate[out_gate].y]>0&&map[stage_level][gate[out_gate].x-1][gate[out_gate].y]<3)
        &&(map[stage_level][gate[out_gate].x+1][gate[out_gate].y]>0&&map[stage_level][gate[out_gate].x+1][gate[out_gate].y]<3)){
        if(head_direction==RIGHT||head_direction==UP)
            head_direction=RIGHT;
        else if(head_direction==LEFT||head_direction==DOWN)
            head_direction=LEFT;
    }
    else if((map[stage_level][gate[out_gate].x][gate[out_gate].y-1]>0&&map[stage_level][gate[out_gate].x][gate[out_gate].y+1]>0)){
        if(head_direction==RIGHT||head_direction==UP)
            head_direction=UP;
        else if(head_direction==LEFT||head_direction==DOWN)
            head_direction=DOWN;
    }
}
}
```

out gate 가 가장자리가 아니고 진출방향이 상하 또는 좌우일 경우입니다.

out gate 의 상하 또는 좌우가 막혔는지의 여부는 out gate 의 좌표에 x 축, y 축으로 1 을 더하거나 빼 그 여부를 판단하였습니다.

만약 out gate 의 상하가 1 이나 2 값을 가진다면, 즉 wall 이나 immune wall 로 막혀있다면 현재 head 방향이 오른쪽이나 위쪽일 경우 head 의 방향을 오른쪽으로 재설정하고 현재 head 의 방향이 왼쪽이나 아래쪽일 시 head 의 방향을 왼쪽으로 재설정합니다.

out gate 의 좌우가 벽으로 막혀있다면 head 의 방향이 오른쪽이나 위쪽일 경우 위쪽으로 재설정, 왼쪽이나 아래쪽일 경우 아래쪽으로 재설정합니다.

만약 진출 방향이 자유롭다면 head 의 방향을 건들지 않고 현재 head 의 방향을 유지하도록 구현하였습니다.

진출방향이 상하 또는 좌우가 아니거나 가장자리 또한 아니고 자유롭지 않은 경우는 immune wall 으로 설정하여 gate 가 생성되지 못하도록 하여 조건을 맞추었습니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

5 단계 점수 요소의 구현 (ui.cpp)

5.1 score 판과 mission 판 생성

```
void sub_win(){
    wattron(sub1, COLOR_PAIR(8));
    wborder(sub1, '1', '1', '1', '1', '1', '1', '1', '1');
    wattroff(sub1, COLOR_PAIR(8));
```

score 판으로는 전역 변수로 선언한 sub1 윈도우를 이용하였습니다. 팔레트로 흰 색을 지정한 다음 테두리에 1 을 찍어 하얀 테두리로 둘러싸인 윈도우를 구현하였습니다.

```
mvwprintw(sub1,2,5,"Score Board (level %d)",stage_level);
    mvwprintw(sub1,4,1,"Body
length : %d / %d ( )",snake.size(),body_length);
    if(body_length>=goal[stage_level][0])
        mvwprintw(sub1,4,1,"Body
length : %d / %d (v)",snake.size(),body_length);

    mvwprintw(sub1,6,1,"Growth item : %d ( )",growth_item);
    if(growth_item>=goal[stage_level][1])
        mvwprintw(sub1,6,1,"Growth item : %d (v)",growth_item);

    mvwprintw(sub1,8,1,"Posion item : %d ( )",posion_item);
    if(posion_item>=goal[stage_level][2])
        mvwprintw(sub1,8,1,"Posion item : %d (v)",posion_item);

    mvwprintw(sub1,10,1,"Gate counter : %d ( )",gate_cnt);
    if(gate_cnt>=goal[stage_level][3])
        mvwprintw(sub1,10,1,"Gate counter : %d (v)",gate_cnt);

    wrefresh(sub1);
```

mvprintw 함수를 사용해 윈도우의 적절한 위치에 문자를 작성하였습니다.

Game 도중 update 해 둔 body_lenth, growth item, posion item, gate cnt 변수를 적절한 위치에 작성하였으며 mission 정보가 담겨져 있는 goal 배열과 각각의 값을 비교하여 mission 을 달성하였다면 변수 옆에 v 체크표시를 하였습니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

```

wattron(sub2, COLOR_PAIR(8));
wborder(sub2, '1', '1', '1', '1', '1', '1', '1', '1');
wattroff(sub2, COLOR_PAIR(8));
mvwprintw(sub2, 2, 4, "Mission Board (level %d)", stage_level);
mvwprintw(sub2, 4, 1, "Body length : %d ", goal[stage_level][0]);
mvwprintw(sub2, 4, 1, "Body length : %d ", goal[stage_level][0]);
mvwprintw(sub2, 6, 1, "Growth item : %d", goal[stage_level][1]);
mvwprintw(sub2, 8, 1, "Posion item : %d", goal[stage_level][2]);
mvwprintw(sub2, 10, 1, "Gate counter : %d", goal[stage_level][3]);

wrefresh(sub2);

```

sub2 윈도우는 mission 판으로 구현하였다. sub1 과 같은 방식으로 테두리를 구현한 뒤 goal 배열의 값으로 각각의 mission 을 작성하였습니다.

```

if(body_length>=goal[stage_level][0]&&growth_item>=goal[stage_level][1]
    &&posion_item>=goal[stage_level][2]&&gate_cnt>=goal[stage_level][3])
    clear_stage=1;
}

```

모든 mission 을 달성했을 시 cleat_stage 값이 1 로 설정되며 이 때 main 문의 반복문에서 탈출하게 됩니다.

5.2 game over 구현

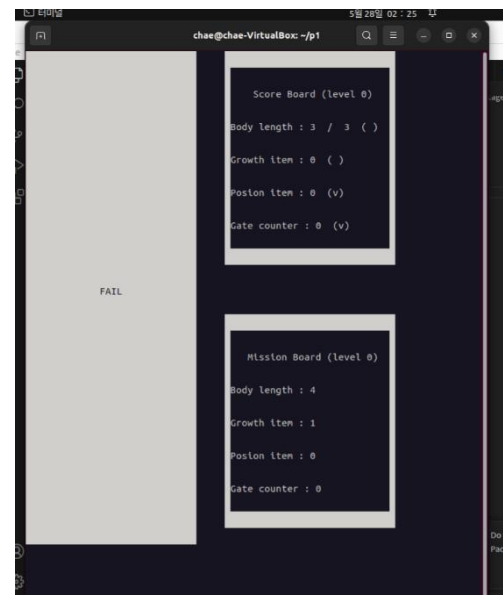
```

void gameover(){
    nodelay(my_win, FALSE);
    wattron(my_win, COLOR_PAIR(8));
    for(int i=0; i<30; i++)
        for(int j=0; j<30; j++)
            mvwprintw(my_win, i, j, "1");

    wattroff(my_win, COLOR_PAIR(8));

    wattron(my_win, COLOR_PAIR(10));
    mvwprintw(my_win, 14, 13, "FAIL");
    wattroff(my_win, COLOR_PAIR(10));
    wgetch(my_win);
    endwin();
    exit(0);
}

```



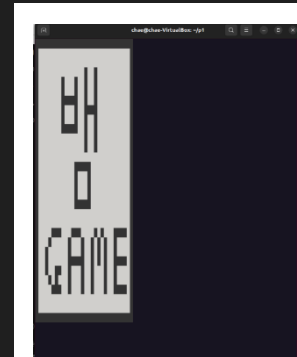
 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

게임 윈도우에 모든 값을 흰색 팔레트로 이용하여 흰색 윈도우를 구현한 뒤 mvprint 함수를 사용하여 배경색은 white, 문자 색은 black 인 팔레트를 불러와 화면 가운데 fail 이라는 문자를 띄웠습니다.

nodelay 함수와 wgetch 를 이용하여 입력이 있을 시까지 화면을 유지하도록 하였다. 입력이 들어온다면 endwin 함수로 win 를 종료한 뒤 exit 함수로 콘솔창에서 나가게 됩니다.

5.3 game start 화면

```
void set_up(){
    nodelay(my_win, FALSE);
    for(int i=0; i<30; i++){
        for(int j=0; j<30; j++){
            if(map[4][i][j] == 0){
                wattron(my_win, COLOR_PAIR(8));
                mvwprintw(my_win, i, j, "%d", map[stage_level][i][j]);
                wattroff(my_win, COLOR_PAIR(8));
            }
            else {
                wattron(my_win, COLOR_PAIR(1));
                mvwprintw(my_win, i, j, "%d", WALL);
                wattroff(my_win, COLOR_PAIR(1));
            }
        }
    }
    wrefresh(my_win);
    wgetch(my_win);
    nodelay(my_win, TRUE);
}
```



map 배열의 4 번째 2 차원 배열은 게임 시작화면을 구현하도록 값을 설정하였습니다.

이를 사용하여 배열의 0 값은 white 색, 배열의 1 값은 black 으로 설정하여 시작화면을 그려냈습니다. 이 때 nodelay 함수와 wgetch 를 이용하여 입력이 있을 시까지 화면을 유지하도록 하였습니다.

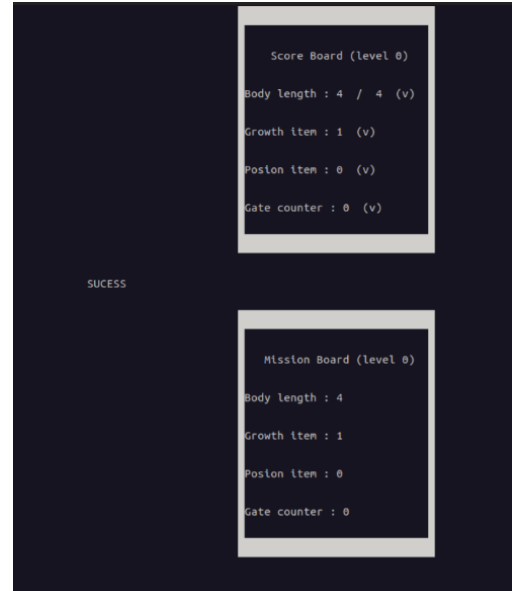
 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

5.3 game clear 화면

```
void game_set(){
    nodelay(my_win, FALSE);
    if(stage_level<3){
        wattron(my_win, COLOR_PAIR(2));
        for(int i=0;i<30;i++)
            for(int j=0;j<30;j++)
                mvwprintw(my_win,i,j,"1");

        wattroff(my_win, COLOR_PAIR(2));

        wattron(my_win, COLOR_PAIR(9));
        mvwprintw(my_win,14,12,"SUCESS");
        wattroff(my_win, COLOR_PAIR(9));
        wrefresh(my_win);
    }
}
```

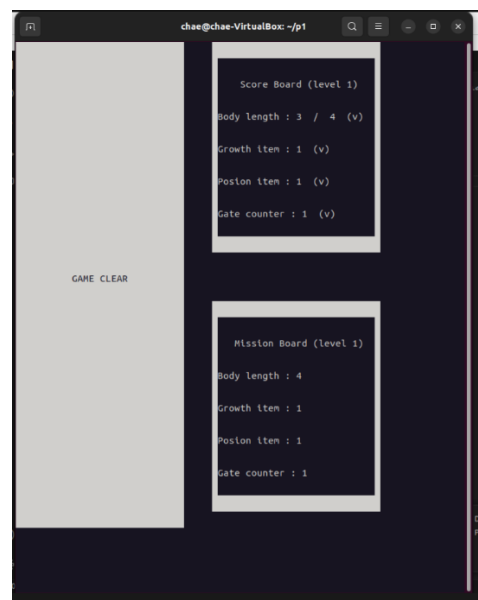


Stage 가 clear 되었을 시 기능하는 함수입니다. 이 때 현재 stage_level 이 마지막 레벨이 아닐 경우 map 에 sucess 라는 문구를 작성합니다.


```
else{
    wattron(my_win, COLOR_PAIR(8));
    for(int i=0;i<30;i++)
        for(int j=0;j<30;j++)
            mvwprintw(my_win,i,j,"1");

    wattroff(my_win, COLOR_PAIR(8));

    wattron(my_win, COLOR_PAIR(10));
    mvwprintw(my_win,14,10,"GAME CLEAR");
    wattroff(my_win, COLOR_PAIR(10));
    wrefresh(my_win);
    wgetch(my_win);
    endwin();
    exit(0);
}
```



Stage 가 마지막 level 일 시 game clear 라는 문구를 작성하고 게임이 종료됩니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

```

clear_stage=growth_item=posion_item=gate_cnt=0;
body_length=3;
while(snake.size()!=3){
    snake.pop_back();
}
stage_level++;
wrefresh(my_win);

wgetch(my_win);
nodelay(my_win,TRUE);
}

```

이 후 다음 stage가 원활히 진행되도록 score판에 사용되는 요소들을 초기화하고 snake또한 size를 3으로 만듭니다. 이 후 stage 레벨을 1증가시키며 입력이 있을 시 까지 콘솔 화면을 유지합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2.2.3 활용/개발된 기술

- rand 함수를 사용하기 위해 time.h 라이브러리가 사용되었습니다.
- usleep 함수를 사용하기 위해 unistd.h 라이브러리가 사용되었습니다.
- Snake 를 동적 배열로 설정하기 위해 함수를 사용하기 위해 vector 라이브러리가 사용되었습니다.

2.2.4 현실적 제한 요소 및 그 해결 방안

1단계

- Map 의 표현이 어려웠음. Map 제작까진 쉬웠지만 특정 요소에 특정 색을 입히는 방식을 생각해 내는 것이 어려웠음. 자료를 읽다가 inpair 함수를 이용해 배경과 글자색을 같게끔 설정하여 해결.

2단계

- Ncurses 윈도우의 사용법에 익숙치 않아 어려웠음.

3단계

- 없음

4단계

- Gate 진입 시 방향설정 상황이 상하나 좌우로 막혀 있을 경우에만 특정되어 있었음. 해결을 위해 자유로운 진출 방향, 가장자리에서의 진출 방향, 상하나 좌우가 막혀 있을 경우 진출 방향, 이 3 가지 경우의 수를 제외한 경우가 나오지 않게 immune wall 을 사용하여 해결.

5단계

- 없음

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

2.2.5 결과물 목록

main.cpp

- Snakegame 을 구현한 파일

snake.cpp

- snake 에 필요한 함수를 작성한 파일

item.cpp

- item 에 필요한 함수를 작성한 파일

gate.cpp

- gate 에 필요한 함수를 작성한 파일

ui.cpp

- 미션판, 스코어판과 여러 화면 구성에 필요한 함수를 작성한 파일

map.cpp

- map 구성에 필요한 함수를 작성한 파일

map.h

- 초기 map 을 저장한 파일.

function.h

- snakegame 에 사용될 변수들과 함수, 클래스를 선언한 파일.

makefile

- snakegame 실행을 위한 makefile.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

3 자기평가

이번 과제를 처음 받았을 때는 막막했었습니다. 알고리즘 자체는 간단하였지만 저에게 익숙치 않은 우분투를 사용하여야 한다는 점과 한 번도 사용해보지 않았던 ncurses 라이브러리를 사용해야 했기 때문입니다. 하지만 이러한 점 때문인지 과제를 해나가면서 저에게 많은 도움이 된 것 같습니다. 이번 과제로 ncurses 라이브러리 사용법뿐만 아니라 chat gpt의 사용법이나 자료 조사의 중요성 같은 중요한 요소들을 깨달았습니다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	Webpage	Ncourse programming how to	KLDPWiki : NCURSE S-Programming-HOWTO			
2	Webpage	C++Ncurses- Snake Game Play Video	C++ Ncurses - Snake Game Play Video - YouTube			

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake game	
	팀 명	채승표	
	Confidential Restricted	Version 1.2	2023-05-26

5 부록

5.1 사용자 매뉴얼

1. Snakegame 은 map 이 30*30 이기에 터미널 크기가 30*30 이상이어야 한다. 나는 80*43 환경에서 테스트하였다.
2. Terminal 에서 파일이 설치된 폴더로 이동한다.
3. make 명령어를 입력하여 snakegame 실행 파일을 생성한다.
4. make가 없을 시 sudo apt install make (make install 명령어) 설치한다.
5. ./snakegame 을 입력하면 게임이 시작된다.
6. 시작 화면은 아무 키나 누르면 넘어간다.
7. fail 하지 않고 steage 4 까지 mission clear 할 시 성공한다.
8. make 가 정상적으로 작동 하지 않을 시 g++ -o main main.cpp -lncurses 로 컴파일 한 후 ./main 으로 실행

5.2 설치 방법

Zip 파일을 다운하고 원하는 경로에 압축을 푼다.