



C++ 프로그래밍 프로젝트

프로젝트 명	스네이크 프로젝트
팀 명	기승전스
문서 제목	결과 보고서

Version	1.2
Date	2022-JUN-15

팀원	신 윤재
	이 세희

CONFIDENTIALITY/SECURITY WARNING

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

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

문서 정보 / 수정 내역

Filename	최종보고서-스네이크프로젝트.docx
원안작성자	신윤재, 이세희
수정작업자	신윤재, 이세희

수정날짜	대표수정자	Revisio n	추가/수정 항목	내 용
2022-06-10	이세희	1.0	최초 작성	수정된 연구내용 추가(Stage 1, 2)
2022-06-10	신윤재	1.1	내용 수정	수정된 연구내용 추가(Stage 2, 3)
2022-06-13	이세희	1.2	내용 수정	수정된 연구내용 추가(Stage 4), 마무리
2022-06-13	신윤재	1.3	내용 수정	수정된 연구내용 추가(Stage 5), 마무리

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

목 차

1	개요	4
2	개발 내용 및 결과물	5
2.1	목표	5
2.2	개발 내용 및 결과물	6
2.2.1	개발 내용	6
2.2.2	시스템 구조 및 설계도	6
2.2.3	활용/개발된 기술	6
2.2.4	현실적 제한 요소 및 그 해결 방안	6
2.2.5	결과물 목록	7
3	자기평가	8
4	참고 문헌	8
5	부록	8
5.1	사용자 매뉴얼	8
5.2	설치 방법	8

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

1 개요

이번 스네이크 프로젝트에서 저희가 사용한 라이브러리는 **ncurses**입니다. **ncurses**는 **curses**의 새로운 버전으로 리눅스 기반 라이브러리입니다. 이 라이브러리는 일반 터미널의 콘솔 화면에서 사용자 친화적인 **Console** 윈도우 환경을 만드는 일을 쉽게 하기위한 라이브러리입니다. 이 라이브러리를 통해 **Window**창을 띄우고, 키보드로 글자를 입력받은 것을 처리하는 등 다양한 일들을 함수를 통해 편리하게 사용할 수 있습니다.

ncurses를 사용하기 위해서는 먼저 라이브러리 설치가 필요합니다. 따라서 터미널 창에 “**sudo apt-get install libncurses5-dev libncursesw5-dev**”를 입력해 설치해 줍니다. 최신 리눅스 버전의 경우 **gcc**등의 개발프로그램을 설치하면 자동으로 설치되기도 합니다. 이렇게 설치된 라이브러리를 사용하기 위해서는 소스코드 속 **#include <ncurses.h>**를 추가해줘야 합니다. **ncurses**라이브러리를 사용한 소스코드를 컴파일 하기 위해서는 **-lncurses** 옵션을 맨 마지막에 추가해줘야 원하는 실행결과물을 도출할 수 있습니다.

```
//라이브러리 설치( 터미널창)
sudo apt-get install libncurses5-dev libncursesw5-dev

//라이브러리 사용(소스코드 속)
#include <ncurses.h>

//소스코드 컴파일
g++ -std=c++11 -o 실행파일이름 소스코드이름 -lncursesw
```

팀원과의 협업 및 프로그램 진행을 위해서 사용한 도구는 **Git & Github, Visual Studio Code** 의 **live Share**, 구름 IDE가 있습니다.

Git의 경우 변경된 내용들을 확인할 수 있고, 프로젝트를 서로의 상황에서 유연하게 진행할 수 있도록 관리의 편의성을 위해 사용하였습니다. **Git**은 버전 관리 시스템으로 프로젝트를 진행할 때 소프트웨어 프로그램의 개발을 용이하게 해줍니다 . 또한 이러한 기능을 다른 개발자와 쉽게 이용할 수 있도록 해주는 **Github**를 이용함으로써 팀원과의 협업 능률을 향상시킬 수 있었습니다. 또한 프로젝트의 버전관리를 진행함으로써 소프트웨어 버전관리에 대한 경험을 해볼 수 있었습니다.

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

저희가 사용한 **ncurses** 라이브러리의 경우 현재 리눅스에서만 설치 및 사용이 가능하여 다른 OS를 사용하고 있는 팀원은 프로젝트를 구현하고 실행하기 어렵다는 한계가 존재했습니다. 이를 해결하기 위해 리눅스 계열의 운영체제인 우분투를 기반으로 운영되는 **cloud IDE**를 사용하였습니다. 이 **cloud IDE**의 경우 **Git & Github**와 연동하여 쉽게 사용할 수 있다는 장점이 있었습니다.

하지만, 이러한 **cloud IDE**의 경우에는 **ncurses**의 창(window)를 구현하기에는 한계가 존재하였고, 실행시간의 버퍼링 및 끊김 현상 등 다양한 문제로 인해서 프로젝트 협업에 원활한 진행이 어렵다고 생각하였습니다. 따라서 저희는 **Visual Studio Code**의 **live Share** 기능을 사용하였습니다. **Visual Studio Code**에서 제공하는 이 기능은 페어코딩으로 작업할 수 있도록 하는 기능입니다. 이를 통해 실시간 협업, 실시간 페어 코딩으로 프로젝트 진행에 있어 편리성을 제공할 수 있었습니다. **live Share** 기능과 더불어 **Google meet** 화상통화를 통해 프로젝트를 함께 진행하였습니다.

 		
Git & GitHub	cloud IDE : goorm	Live Share
코드 관리	리눅스 OS를 다른 OS에서 사용 가능 (개인)	프로젝트 실시간 협업 및 진행상황 분석

저희 팀의 **Snake-game** 구조는 바탕이 되는 **map**과 그 위에서 움직이는 **Snake**, 그리고 **Snake**을 움직이는 사용자로 구성되어 진행되어도록 하였습니다.

Game에 바탕이 되는 **Map**은 총 4단계로 구성이 되어져 있습니다. 각 단계마다 실패할 수 있는 확률이 높아지도록 설계를 하였고 이는 2차원 배열을 이용하여 구현하였습니다. **Map**위에는 바탕과 벽, 그리고 **Item**, **Gate**로 구성되어져 있으며 **Item**은 스네이크의 길이를 늘리거나 줄여주고, **Gate**는 다른 위치로 옮겨주는 역할을 합니다.

이러한 바탕에서 움직이는 **Snake**은 머리와 몸 부분으로 이루어져 있고 총 4개의 방향(왼쪽, 오른쪽, 위, 아래)으로 움직일 수 있도록 하였습니다. 스네이크의 머리 부분이 벽에 달게 되면 게임은 실패하고, 원하는 조건이 넘고 **Stage**를 모두 깨면 성공하도록 구성하였습니다.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

사용자는 키보드의 화살표 키를 이용하여 스네이크를 조작할 수 있도록 했습니다. 예를 들어, 오른쪽 키보드의 경우 스네이크의 진행 방향의 오른쪽 방향이 아닌, Map상에서의 오른쪽으로 갈 수 있도록 구현하였습니다. 이를 통해 만약 진행방향에 반대방향의 키보드를 누르게 된다면 실패하게 됩니다.

이러한 구성을 구현하기 위해서 각각의 함수를 구현하여 사용하였습니다. 각 함수들을 main함수에서 호출 후 관리하도록 하였습니다.

핵심 구성을 위한 함수		
Map 구현 및 시각화	void show_map();	정해진 Map 표현
	void show_gate();	Gate 구현
	void show_item();	item 구현
Snake 움직임 구현 및 Map 속 구현	void show_snake();	Map에 Snake 표현
	bool move();	Snake에 관한 움직임 구현 및 Snake 점수 구현
사용자 입력 처리	getch();	ncurses 라이브러리 속 함수

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

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

1) 1단계: Map의 구현

1단계의 목표는 **ncurses** 라이브러리를 통해 맵을 구현하고 그 위에 스네이크를 구현하는 것입니다. 1단계에서는 21(높이), 42(너비) 크기의 맵을 만들었습니다. 1은 일반 벽, 2는 영구 벽, 0은 스네이크가 지날 수 있는 공간으로 구성하여 이를 **stage** 별로 서로 다른 맵을 구현했습니다. 맵은 txt 파일로 저장되어 있어 실행시 파일을 읽어서 **array**로 저장을 합니다. 그 후 **ncurses** 라이브러리를 이용해 값(0,1,2)에 따라 속성을 부여하여 일반 벽, 영구 벽 등을 구분할 수 있도록 하였습니다. 또한 **snake**의 머리와 몸통은 **vector** 2개를 이용하여 x, y 위치를 표현하고 속성을 부여하여 머리와 몸통을 구분할 수 있도록 하였습니다.

2) 2단계: Snake 표현 및 조작

2단계의 목표는 상, 하, 좌, 우 키를 이용하여 스네이크를 원하는 방향으로 조작하는 것입니다. 앞서 1단계에서 구현했던 요소와 더불어 스네이크의 방향을 알려주는 **direction**, 스네이크가 움직이는 시간 단위인 **tick**, 스네이크가 진행방향의 반대방향으로 가거나 벽으로 갔을 때를 나타내는 **fail** 등의 변수를 추가하여 **Snake**가 움직일 때의 환경을 조성하였습니다. 그 후 **while**문을 **main** 함수에 구현하여 그 안에서 매 **tick** 동안 스네이크가 진행방향으로 한칸 씩 가도록 구현을 하였습니다. 또한 진행방향은 상, 하, 좌, 우 키보드의 입력을 받아 변경할 수 있도록 구현하였습니다. 스네이크는 머리가 먼저 진행방향으로 움직이고 그 이후 몸통들은 머리와 바로 앞의 몸통 칸의 위치를 따라서 x, y 위치를 수정해주어 진행방향으로 움직이도록 구현하였습니다.

3) 3단계: Item 요소의 구현

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

3단계의 목표는 스네이크의 몸길이를 조절하는 **growth, poison 아이템의 기능을 구현하고 이를 맵의 임의의 위치에 놓는 것입니다.** item은 growth item, poison item 두 가지 아이템이 있습니다. growth item은 스네이크의 몸통의 길이를 한 칸 늘려주는 역할을 합니다. 스네이크의 머리가 이 아이템에 닿으면 snake의 몸통과 머리 위치를 담고 있는 x, y vector에 한칸이 더 push 됩니다. poison 아이템은 스네이크의 몸통을 한칸 줄여줍니다. 스네이크의 머리가 이 아이템에 닿으면 스네이크 x, y vector의 맨 끝 값을 pop 해줍니다. item의 위치는 1, 2에 해당하는 벽과 겹치지 않으면서 랜덤으로 나타날 수 있도록 while loop와 랜덤 함수를 사용했습니다.

4) 4단계: **Gate** 요소의 구현

4단계의 목표는 스네이크의 공간이동을 도와주는 한 쌍의 **gate를 임의의 위치에 구현하는 것입니다.** 앞서 구현한 item과 같이 while loop와 랜덤 함수를 이용하여 2개의 gate가 위치가 겹치지 않고 임의의 위치로 나타나도록 구현하였습니다. 또한 item과는 다르게 일반 벽에도 나타날 수 있도록 구현하였습니다. gate의 위치를 중심으로 주변에 벽이 있는지를 판단하여 스네이크가 gate를 통과한 후 나아갈 방향과 위치를 구현했습니다. 우선 기준 방향을 기준으로 시계방향으로 상황에 따라 진행방향을 설정해줍니다. 또한 맵 바깥 부분으로 스네이크가 나가지 않도록 맵의 경계 부분도 고려하여 설정해주었습니다.

5) 5단계: 점수 요소의 구현

5단계의 목표는 스네이크 게임의 전반적인 점수현황과 각 스테이지의 목표 점수 달성여부를 보여주고, 목표치를 달성하면 다음 스테이지로 넘어가는 것입니다. 그후 게임이 끝난 후 게임 진행시간과 성공여부를 알려주는 것입니다. 점수요소를 구현하기 위해 두 개의 서브 윈도우를 만들었습니다. 하나는 현재의 점수 현황을 보여주는 윈도우이고, 하나는 스테이지 목표 점수 달성여부를 나타내는 윈도우입니다. 여기에서는 점수를 나타내기 위해 현재의 스네이크 길이, 스네이크가 growth 아이템을 획득한 횟수, poison 아이템을 획득한 횟수, gate를 통과한 횟수 등을 전역변수로 설정하여 스테이지마다 보여지게 구현하였습니다. 또한 최대 길이, growth 획득 횟수, poison 획득 횟수, gate 통과 횟수 등을 고정된 목표 값으로 설정하여 이 목표값을 달성하면 다음 스테이지로 넘어가도록 구현하였습니다. 이후 for 문을 통해 다음 스테이지로 넘어갈 수 있습니다. 다음 스테이지로 넘어가면 처음에는 스네이크의 위치, 길이, 점수 등을 초기화 한 후 초기 상태로 돌아와 새로운 맵을 시작할 수 있게 됩니다. 그 후 스테이지를 끝까지 완료하거나 실패하게 되면 화면이 종료되고 터미널 창에 게임 진행 시간과 게임의 성공여부가 나타나게 됩니다.

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

2.2 개발 내용 및 결과물

2.2.1 개발 내용

1) 1단계: Map의 구현

1단계의 목표는 ncurses 라이브러리를 통해 맵을 구현하고 그 위에 스네이크를 구현하는 것입니다.

구현 함수 이름 (파란색은 현재 단계에서 나오는 새로운 함수)	함수 기능 및 주요 설명 (초록색은 현재 단계에서 내용이 수정됨)
show_map()	<p>1. 맵 텍스트 파일 입력 스테이지 별 맵파일을 불러온다. 불러온 맵 파일을 한 줄씩 읽어 integer array로 저장하여 놓는다. 맵파일은 txt 파일로 되어있고 0, 1, 2로 구현되어있다. 0은 스네이크가 지나갈 수 있는 통로 1은 일반 벽(게이트가 나타날 수 있는 벽), 2는 영구벽이다.</p> <p>2. 맵 속성 설정 맵은 ncurses 라이브러리의 attron 함수를 이용해 각 0,1,2 값마다 속성을 부여해 주고 이를 mvprintw 함수를 이용해 화면에 나타날 위치를 지정해 주었다.</p> <p>3. 속성이 적용된 맵을 화면에 구현 ncurses 라이브러리의 refresh 함수를 이용해 속성이 부여된 맵을 구현한다.</p>
show_snake()	<p>1. 스네이크 벡터 x, y 벡터 구현 vector 라이브러리를 이용해 스네이크의 머리, 몸통의 위치 정보를 순서대로 담는다.</p> <p>2. for문을 통한 스네이크를 화면에 구현 스네이크 머리와 몸통을 다른 속성으로 부여하여 for 문을 통해 스네이크를 화면에 출력한다.</p>
main()	<p>1. 스네이크 위치 초기화</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>vector 라이브러리의 <code>push_back</code> 메소드를 통해 스네이크의 머리, 몸통의 위치를 초기화 한다.</p> <p>2. 스네이크와 맵을 화면에 구현</p> <p><code>show_map()</code>, <code>show_snake()</code> 함수를 통해 구현한 맵과 스네이크 구현 함수를 실행한다.</p>
--	--

2) 2단계: Snake 표현 및 조작

2단계의 목표는 상, 하, 좌, 우 키를 이용하여 스네이크를 원하는 방향으로 조작하는 것입니다.

구현 함수 이름 (파란색은 현재 단계에서 나오는 새로운 함수)	함수 기능 및 주요 설명 (초록색은 현재 단계에서 내용이 수정됨)
show_map()	<p>1. 맵 텍스트 파일 입력</p> <p>스테이지 별 맵파일을 불러온다. 불러온 맵 파일을 한 줄씩 읽어 <code>integer array</code>로 저장하여 놓는다. 맵파일은 <code>txt</code> 파일로 되어있고 0, 1, 2로 구현되어있다. 0은 스네이크가 지나갈 수 있는 통로 1은 일반 벽(게이트가 나타날 수 있는 벽), 2는 영구벽이다.</p> <p>2. 맵 속성 설정</p> <p>맵은 <code>ncurses</code> 라이브러리의 <code>attron</code> 함수를 이용해 각 0,1,2 값마다 속성을 부여해 주고 이를 <code>mvprintw</code> 함수를 이용해 화면에 나타날 위치를 지정해 주었다.</p> <p>3. 속성이 적용된 맵을 화면에 구현</p> <p><code>ncurses</code> 라이브러리의 <code>refresh</code> 함수를 이용해 속성이 부여된 맵을 구현한다.</p>
show_snake()	<p>1. 스네이크 벡터 <code>x</code>, <code>y</code> 벡터 구현</p> <p><code>vector</code> 라이브러리를 이용해 스네이크의 머리, 몸통의 위치 정보를 순서대로 담는다.</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>2. for문을 통한 스네이크를 화면에 구현</p> <p>스네이크 머리와 몸통을 다른 속성으로 부여하여 for 문을 통해 스네이크를 화면에 출력한다.</p>
move()	<p>1. 스네이크 머리 위치 이동</p> <p>스네이크의 머리를 direction 변수의 값에 따라 위치를 지정하여 한 칸 이동한다.</p> <p>2. 상, 하, 좌, 우 키 입력값 저장</p> <p>getch() 함수를 이용하여 다음 while loop에서 스네이크의 머리가 이동할 방향을 지정해준다. 현재 진행 방향의 반대 방향으로 이동하는 경우에는 fail 함수가 true가 되도록 설정하여 스네이크 게임이 while 문을 빠져나와 종료 되도록 한다.</p> <p>3. 스네이크 몸통 이동</p> <p>이전 loop의 스네이크 머리 위치를 기반으로 머리와 가까운 몸통부터 꼬리까지의 몸통 위치를 for 문을 이용해 변경해준다.</p> <p>4. while문과 usleep() 함수를 통한 일정시간 이동 반복</p> <p>usleep 함수를 이용하여 tick을 0.3초로 설정하고 이 일정 시간동안 스네이크가 움직이도록 설정한다.</p>
main()	<p>1. 스네이크 위치 초기화</p> <p>vector 라이브러리의 push_back 메소드를 통해 스네이크의 머리, 몸통의 위치를 초기화 한다.</p> <p>2. 스네이크와 맵을 화면에 구현</p> <p>show_map(), show_snake(), move() 함수를 통해 구현한 맵과 스네이크 구현 함수를 실행한다.</p>

3) 3단계: Item 요소의 구현

3단계의 목표는 스네이크의 몸길이를 조절하는 **growth**, **poison** 아이템의 기능을 구현하고 이를 맵의

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

임의의 위치에 놓는 것 입니다.

구현 함수 이름 (파란색은 현재 단계에서 나오는 새로운 함수)	함수 기능 및 주요 설명 (초록색은 현재 단계에서 내용이 수정됨)
show_map()	<p>1. 맵 텍스트 파일 입력 스테이지 별 맵파일을 불러온다. 불러온 맵 파일을 한 줄씩 읽어 integer array로 저장하여 놓는다. 맵파일은 txt 파일로 되어있고 0, 1, 2로 구현되어 있다. 0은 스네이크가 지나갈 수 있는 통로 1은 일반 벽(게이트가 나타날 수 있는 벽), 2는 영구벽이다.</p> <p>2. 맵 속성 설정 맵은 ncurses 라이브러리의 attron 함수를 이용해 각 0,1,2 값마다 속성을 부여해 주고 이를 mvprintw 함수를 이용해 화면에 나타날 위치를 지정해 주었다.</p> <p>3. 속성이 적용된 맵을 화면에 구현 ncurses 라이브러리의 refresh 함수를 이용해 속성이 부여된 맵을 구현한다.</p>
show_snake()	<p>1. 스네이크 벡터 x, y 벡터 구현 vector 라이브러리를 이용해 스네이크의 머리, 몸통의 위치 정보를 순서대로 담는다.</p> <p>2. for문을 통한 스네이크를 화면에 구현 스네이크 머리와 몸통을 다른 속성으로 부여하여 for 문을 통해 스네이크를 화면에 출력한다.</p>
move()	<p>1. 스네이크 머리 위치 이동 스네이크의 머리를 direction 변수의 값에 따라 위치를 지정하여 한 칸 이동한다.</p> <p>2. 상, 하, 좌, 우 키 입력값 저장 getch() 함수를 이용하여 다음 while loop에서 스네이크의 머리가 이동할</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>방향을 지정해준다. 현재 진행 방향의 반대 방향으로 이동하는 경우에는 fail 함수가 true가 되도록 설정하여 스네이크 게임이 while 문을 빠져나와 종료 되도록 한다.</p> <p>3. 스네이크 몸통 이동</p> <p>이전 loop의 스네이크 머리 위치를 기반으로 머리와 가까운 몸통부터 꼬리까지의 몸통 위치를 for 문을 이용해 변경해준다.</p> <p>4. 아이템 습득 시 스네이크 몸통 길이 수정</p> <p>1) growth 아이템을 습득한 경우</p> <p>growth 아이템의 위치와 스네이크 머리의 위치가 일치하는 경우, 이전 loop의 꼬리 부분의 위치에 새로운 몸통이 들어가도록, 몸통의 길이가 한 칸 늘어나도록 push_back 함수를 이용하여 스네이크의 x, y 좌표 vector에 새로운 몸통 위치 정보를 넣는다.</p> <p>2) poison 아이템을 습득한 경우</p> <p>poison 아이템의 위치와 스네이크 머리의 위치가 일치하는 경우, 현재 마지막 꼬리 위치에 대한 정보를 스네이크의 x, y 좌표 vector에서 pop 메소드를 이용해 삭제해준다.</p> <p>5. while문과 usleep() 함수를 통한 일정시간 이동 반복</p> <p>usleep 함수를 이용하여 tick을 0.3초로 설정하고 이 일정 시간동안 스네이크가 움직이도록 설정한다.</p>
get_item()	<p>1. 랜덤 함수와 while 문을 이용한 아이템 위치 설정</p> <p>랜덤 함수를 이용하여 아이템이 임의의 위치에 구현되도록 설정한다. 또한 growth 아이템과 poison 아이템의 위치가 겹치지 않도록, 영구벽과 일반벽의 위치와 겹치지 않도록 while 문을 통해 위 두 사항을 만족하는 x, y 좌표를 생성한다.</p> <p>2. 아이템의 종류에 따른 속성 부여 및 화면에 출력</p> <p>growth와 poison 아이템을 구분할 수 있도록 각자 다른 속성을 부여하여 지정된 위치에 출력한다.</p>
main()	<p>1. 스네이크 위치 초기화</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>vector 라이브러리의 <code>push_back</code> 메소드를 통해 스네이크의 머리, 몸통의 위치를 초기화 한다.</p> <p>2. 스네이크와 맵을 화면에 구현</p> <p><code>show_map()</code>, <code>show_snake()</code>, <code>move()</code>, <code>get_item()</code> 함수를 통해 구현한 맵과 스네이크 구현 및 이동 함수를 실행한다.</p>
--	---

4) 4단계: Gate 요소의 구현

4단계의 목표는 스네이크의 공간이동을 도와주는 한 쌍의 **gate**를 임의의 위치에 구현하는 것입니다.

구현 함수 이름 (파란색은 현재 단계에서 나오는 새로운 함수)	함수 기능 및 주요 설명 (초록색은 현재 단계에서 내용이 수정됨)
<code>show_map()</code>	<p>1. 맵 텍스트 파일 입력</p> <p>스테이지 별 맵파일을 불러온다. 불러온 맵 파일을 한 줄씩 읽어 <code>integer array</code>로 저장하여 놓는다. 맵파일은 <code>txt</code> 파일로 되어있고 0, 1, 2로 구현되어있다. 0은 스네이크가 지나갈 수 있는 통로 1은 일반 벽(게이트가 나타날 수 있는 벽), 2는 영구벽이다.</p> <p>2. 맵 속성 설정</p> <p>맵은 <code>ncurses</code> 라이브러리의 <code>attron</code> 함수를 이용해 각 0,1,2 값마다 속성을 부여해 주고 이를 <code>mvprintw</code> 함수를 이용해 화면에 나타날 위치를 지정해 주었다.</p> <p>3. 속성이 적용된 맵을 화면에 구현</p> <p><code>ncurses</code> 라이브러리의 <code>refresh</code> 함수를 이용해 속성이 부여된 맵을 구현한다.</p>
<code>show_snake()</code>	<p>1. 스네이크 벡터 <code>x</code>, <code>y</code> 벡터 구현</p> <p><code>vector</code> 라이브러리를 이용해 스네이크의 머리, 몸통의 위치 정보를 순서대로 담는다.</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>2. for문을 통한 스네이크를 화면에 구현</p> <p>스네이크 머리와 몸통을 다른 속성으로 부여하여 for 문을 통해 스네이크를 화면에 출력한다.</p>
move()	<p>1. 스네이크 머리 위치 이동</p> <p>스네이크의 머리를 direction 변수의 값에 따라 위치를 지정하여 한 칸 이동한다.</p> <p>2. 상, 하, 좌, 우 키 입력값 저장</p> <p>getch() 함수를 이용하여 다음 while loop에서 스네이크의 머리가 이동할 방향을 지정해준다. 현재 진행 방향의 반대 방향으로 이동하는 경우에는 fail 함수가 true가 되도록 설정하여 스네이크 게임이 while 문을 빠져나와 종료 되도록 한다.</p> <p>3. 스네이크 몸통 이동</p> <p>이전 loop의 스네이크 머리 위치를 기반으로 머리와 가까운 몸통부터 꼬리까지의 몸통 위치를 for 문을 이용해 변경해준다.</p> <p>4. 아이템 습득 시 스네이크 몸통 길이 수정</p> <p>1) growth 아이템을 습득한 경우</p> <p>growth 아이템의 위치와 스네이크 머리의 위치가 일치하는 경우, 이전 loop의 꼬리 부분의 위치에 새로운 몸통이 들어가도록, 몸통의 길이가 한 칸 늘어나도록 push_back 함수를 이용하여 스네이크의 x, y 좌표 vector에 새로운 몸통 위치 정보를 넣는다.</p> <p>2) poison 아이템을 습득한 경우</p> <p>poison 아이템의 위치와 스네이크 머리의 위치가 일치하는 경우, 현재 마지막 꼬리 위치에 대한 정보를 스네이크의 x, y 좌표 vector에서 pop 메소드를 이용해 삭제해준다.</p>
get_item()	<p>1. 랜덤 함수와 while 문을 이용한 아이템 위치 설정</p> <p>랜덤 함수를 이용하여 아이템이 임의의 위치에 구현되도록 설정한다. 또한 growth 아이템과 poison 아이템의 위치가 겹치지 않도록, 영구벽과 일반벽의 위치와 겹치지 않도록 while 문을 통해 위 두 사항을 만족하는 x, y</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>좌표를 생성한다.</p> <p>2. 아이템의 종류에 따른 속성 부여 및 화면에 출력 growth와 poison 아이템을 구분할 수 있도록 각자 다른 속성을 부여하여 지정된 위치에 출력한다.</p>
in_gate()	<p>1. gate의 위치에 따른 gate를 통과하는 스네이크 몸통의 주소 및 방향 변환</p> <p>gate의 좌표를 기반으로 주변에 벽이 없는지를 확인하고, 스네이크의 기존 방향을 고려하여 스네이크가 gate를 통과할 때의 위치 및 방향을 설정해준다.</p>
show_gate()	<p>1. 랜덤 함수와 while 문을 이용한 gate 위치 설정 랜덤 함수를 이용하여 gate가 임의의 위치에 구현되도록 설정한다. 또한 두 gate의 위치가 겹치지 않도록, 영구벽과 겹치지 않도록 while 문을 통해 위 두 사항을 만족하는 x, y 좌표를 생성한다.</p> <p>2. gate 속성 부여 및 화면에 출력 gate를 구분할 수 있도록 각자 다른 속성을 부여하여 지정된 위치에 출력한다.</p>
main()	<p>1. 스네이크 위치 초기화 vector 라이브러리의 push_back 메소드를 통해 스네이크의 머리, 몸통의 위치를 초기화 한다.</p> <p>2. 스네이크와 맵을 화면에 구현 show_map(), show_snake(), move(), get_item(), in_gate(), show_gate() 함수를 통해 구현한 맵과 스네이크 구현 및 이동 함수를 실행한다.</p> <p>3. while문과 usleep() 함수를 통한 일정시간 이동 반복 move() 함수 안의 usleep 함수를 이용하여 tick을 0.3초로 설정하고 이 일정 시간동안 스네이크가 움직이도록(화면이 새로 렌더링 되도록) 설정한다.</p>

5) 5단계: 점수 요소의 구현

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

5단계의 목표는 스네이크 게임의 전반적인 점수현황과 각 스테이지의 목표 점수 달성여부를 보여주고, 목표치를 달성하면 다음 스테이지로 넘어가는 것입니다. 그후 게임이 끝난 후 게임 진행시간과 성공여부를 알려주는 것입니다.

구현 함수 이름 (파란색은 현재 단계에서 나오는 새로운 함수)	함수 기능 및 주요 설명 (초록색은 현재 단계에서 내용이 수정됨)
show_map()	<p>1. 맵 텍스트 파일 입력 스테이지 별 맵파일을 불러온다. 불러온 맵 파일을 한 줄씩 읽어 integer array로 저장하여 놓는다. 맵파일은 txt 파일로 되어있고 0, 1, 2로 구현되어있다. 0은 스네이크가 지나갈 수 있는 통로 1은 일반 벽(게이트가 나타날 수 있는 벽), 2는 영구벽이다.</p> <p>2. 맵 속성 설정 맵은 ncurses 라이브러리의 attron 함수를 이용해 각 0,1,2 값마다 속성을 부여해 주고 이를 mvprintw 함수를 이용해 화면에 나타날 위치를 지정해 주었다.</p> <p>3. 속성이 적용된 맵을 화면에 구현 ncurses 라이브러리의 refresh 함수를 이용해 속성이 부여된 맵을 구현한다.</p>
show_snake()	<p>1. 스네이크 벡터 x, y 벡터 구현 vector 라이브러리를 이용해 스네이크의 머리, 몸통의 위치 정보를 순서대로 담는다.</p> <p>2. for문을 통한 스네이크를 화면에 구현 스네이크 머리와 몸통을 다른 속성으로 부여하여 for 문을 통해 스네이크를 화면에 출력한다.</p>
move()	<p>1. 스네이크 머리 위치 이동 스네이크의 머리를 direction 변수의 값에 따라 위치를 지정하여 한 칸 이동한다.</p>

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>2. 상, 하, 좌, 우 키 입력값 저장</p> <p><code>getch()</code> 함수를 이용하여 다음 <code>while</code> loop에서 스네이크의 머리가 이동할 방향을 지정해준다. 현재 진행 방향의 반대 방향으로 이동하는 경우에는 <code>fail</code> 함수가 <code>true</code>가 되도록 설정하여 스네이크 게임이 <code>while</code> 문을 빠져나와 종료 되도록 한다.</p> <p>3. 스네이크 몸통 이동</p> <p>이전 <code>loop</code>의 스네이크 머리 위치를 기반으로 머리와 가까운 몸통부터 꼬리까지의 몸통 위치를 <code>for</code> 문을 이용해 변경해준다.</p> <p>4. 아이템 습득 시 스네이크 몸통 길이 수정</p> <p>3) growth 아이템을 습득한 경우</p> <p><code>growth</code> 아이템의 위치와 스네이크 머리의 위치가 일치하는 경우, 이전 <code>loop</code>의 꼬리 부분의 위치에 새로운 몸통이 들어가도록, 몸통의 길이가 한 칸 늘어나도록 <code>push_back</code> 함수를 이용하여 스네이크의 <code>x, y</code> 좌표 <code>vector</code>에 새로운 몸통 위치 정보를 넣는다.</p> <p>4) poison 아이템을 습득한 경우</p> <p><code>poison</code> 아이템의 위치와 스네이크 머리의 위치가 일치하는 경우, 현재 마지막 꼬리 위치에 대한 정보를 스네이크의 <code>x, y</code> 좌표 <code>vector</code>에서 <code>pop</code> 메소드를 이용해 삭제해준다.</p>
<code>get_item()</code>	<p>1. 랜덤 함수와 while 문을 이용한 아이템 위치 설정</p> <p>랜덤 함수를 이용하여 아이템이 임의의 위치에 구현되도록 설정한다. 또한 <code>growth</code> 아이템과 <code>poison</code> 아이템의 위치가 겹치지 않도록, 영구벽과 일반벽의 위치와 겹치지 않도록 <code>while</code> 문을 통해 위 두 사항을 만족하는 <code>x, y</code> 좌표를 생성한다.</p> <p>2. 아이템의 종류에 따른 속성 부여 및 화면에 출력</p> <p><code>growth</code>와 <code>poison</code> 아이템을 구분할 수 있도록 각자 다른 속성을 부여하여 지정된 위치에 출력한다.</p>
<code>in_gate()</code>	<p>1. gate의 위치에 따른 gate를 통과하는 스네이크 몸통의 주소 및 방향 변환</p>

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	gate의 좌표를 기반으로 주변에 벽이 없는지를 확인하고, 스네이크의 기존 방향을 고려하여 스네이크가 gate를 통과할 때의 위치 및 방향을 설정해준다.
show_gate()	<p>1. 랜덤 함수와 while 문을 이용한 gate 위치 설정 랜덤 함수를 이용하여 gate가 임의의 위치에 구현되도록 설정한다. 또한 두 gate의 위치가 겹치지 않도록, 영구벽과 겹치지 않도록 while 문을 통해 위 두 사항을 만족하는 x, y 좌표를 생성한다.</p> <p>2. gate 속성 부여 및 화면에 출력 gate를 구분할 수 있도록 각자 다른 속성을 부여하여 지정된 위치에 출력한다.</p>
show_score()	<p>1. 현재 게임의 점수 화면에 구현 서브 원도우를 하나 구현하고 스네이크의 현재 길이, 최대 길이, growth 아이템 및 poison 아이템 획득 횟수, gate 통과 횟수를 화면에 구현하도록 한다.</p> <p>2. 스테이지의 목표 점수 달성여부 화면에 구현 서브 원도우를 하나 구현하고 스네이크의 최대 목표길이, 아이템의 목표 획득 횟수, gate 목표 통과 횟수를 표현하고 그에 대한 목표 달성 여부를 표시한다.</p>
init()	<p>1. 새로운 스테이지가 시작 될 때마다 게임 환경 초기화 이전 스테이지의 목표를 달성하면 다음 스테이지로 이동 할 수 있다. 이동하고 나면 스네이크의 x, y 위치 벡터 값, 각종 점수(최대 길이, 아이템 획득 횟수, 게이트 통과 횟수 등)를 초기화 한다.</p> <p>2. 스테이지 별 목표 점수 설정 스테이지마다 다른 목표점수를 설정하여 다양한 게임 환경을 구현한다.</p>
main()	<p>1. 스네이크 위치 초기화 init() 함수를 통해 스네이크의 머리, 몸통의 위치를 초기화 한다.</p> <p>2. 스네이크와 맵을 화면에 구현</p>

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

	<p>show_map(), show_snake(), move(), get_item(), in_gate(), show_gate(), show_score() 함수를 통해 구현한 맵과 스네이크 구현 및 이동 함수를 실행한다.</p> <p>3. while문과 usleep() 함수를 통한 일정시간 이동 반복 move() 함수 안의 usleep 함수를 이용하여 tick을 0.3초로 설정하고 이 일정 시간동안 스네이크가 움직이도록(화면이 새로 렌더링 되도록) 설정한다.</p> <p>4. 스테이지 변경 각 스테이지의 목표점수를 달성하면 while loop를 빠져나와 다음 for문으로 이동한다.</p> <p>5. 게임 플레이 시간과 게임의 성공여부 스테이지를 모두 통과하거나 게임 도중 스네이크가 죽는 경우, 화면이 종료되고 터미널에 게임 진행시간과 게임의 성공여부가 나오게 된다.</p>
--	---

2.2.2 시스템 구조 및 설계도

1단계 : Map의 구현 - 신윤재, 이세희

전역변수

```
#include<ncurses.h>
#include<fstream>
#include<iostream>
#include <vector>

using namespace std;
// Size define
#define map_height 21
#define map_width 42

//file
ifstream map_file;
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```
//map
int map[map_height][map_width];

//snake
vector<int> snake_x;
vector<int> snake_y;
```

- Window를 구현하기 위한 ncurses 라이브러리를 사용.
- 파일 Read/Write를 위한 함수인 fstream 라이브러리를 사용함.
- 뱀의 몸을 구현하기 위해 vector를 사용함.
- 뱀의 몸은 map의 x좌표와 y좌표를 따른 vector를 통해 저장하면서 관리하게 편하도록 함.
- Map배열을 전역변수로 선언하여 다른 공간에서도 모두 사용할 수 있도록 함.

show_map() 함수

```
void show_map()
{
    init_pair(1, COLOR_WHITE, COLOR_WHITE); // 0: background
    init_pair(2, COLOR_CYAN, COLOR_CYAN); // 1: wall
    init_pair(3, COLOR_BLACK, COLOR_BLACK); // 2: Immune Wall
    string filename = "./snakemap/snake1.map";
    map_file.open(filename);
    string line;
    for (int i=0; i<map_height; i++)
    {
        getline(map_file,line);
        for(int j=0; j<map_width; j++)
        {
            map[i][j] = line.at(j) - '0'; // char to int: out of range
            // Set & Print color
            switch (map[i][j])
            {
                case 0: // background
                    attron(COLOR_PAIR(1));
                    mvprintw(i, j, " ");
                    attroff(COLOR_PAIR(1));
            }
        }
    }
}
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

          attron(COLOR_PAIR(1));
          break;
        case 1: // wall
          attron(COLOR_PAIR(2));
          mvprintw(i, j, " ");
         attroff(COLOR_PAIR(2));
          break;
        case 2: // Immune Wall
          attron(COLOR_PAIR(3));
          mvprintw(i, j, " ");
         attroff(COLOR_PAIR(3));
          break;
        }
      }
    }
  }
map_file.close();
refresh();
}
}

```

- txt 파일로 21 X 42 Map을 수기로 작성.
- Map 파일을 읽고 화면에 표시.
- Map에서 표현을 하기 위해서는 여러가지의 색상을 통해서 각 구성요소를 구별함.
 - Background(배경) : WHITE
 - WALL 1(가변 벽) : CYAN
 - WALL 2(불변 벽) : BLACK
- 총 4개의 Map을 구현하였으나, 1단계의 세부 목표는 Map의 여러 모드를 변경하는 것이 아닌 Map의 시각화(구현)을 더 중점적으로 구현했기 때문에 map1만 먼저 사용함.
- TXT 파일은 구현을 위해 21X42 배열로 변경되어 구현됨.
- map[21][42] 는 파일에서 읽어들인 문자열 값들을 배열에 넣음.
- 각 문자열에 맞도록 색상을 지정하여 만들어놓은 window에 표현.
 - switch & case 를 사용하면서 읽어들인 문자열을 숫자로 바꿔서 표현

Show_snake함수

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

void show_snake()
{
    init_pair(4, COLOR_YELLOW, COLOR_YELLOW); // snake head
    init_pair(5, COLOR_GREEN, COLOR_GREEN); // snake body
    // print head
    attron(COLOR_PAIR(4));
    mvprintw(snake_x[0], snake_y[0], " ");
   attroff(COLOR_PAIR(4));

    //print body
    for(int i=1; i<snake_x.size(); i++)
    {
        attron(COLOR_PAIR(5));
        mvprintw(snake_x[i], snake_y[i], " ");
       attroff(COLOR_PAIR(5));
    }
    refresh();
}

```

- 뱀의 머리 색상은 노란색, 몸체의 색상은 녹색으로 설정함.
- 라이브러리 속 함수인 `mvprintw` 함수를 이용하여 위치를 설정하여 공백문자를 출력하여 색상이 보이도록 함.
- `mvprintw()`를 사용하기 위해 이 색상을 설정하기 위해서는 앞 뒤로 `attron()` 함수와 `attroff()` 함수를 사용해 줘야함.
- 화면 출력을 변경하였으면 `refresh()` 함수를 사용해야 변경된 출력이 가능함.

main 함수

```

int main()
{
    // init snake place
    for (int i=0; i<4; i++)
    {
        snake_x.push_back(11);
        snake_y.push_back(11+i);
    }
}

```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

initscr();
resize_term(21, 42);
start_color();
init_pair(6, COLOR_GREEN, COLOR_GREEN); // background
bkgd(COLOR_PAIR(6));

show_map();
// getch();

show_snake();
getch();
endwin();

return 0;
}

```

- 스네이크의 초기 위치는 11, 11로 설정한 후 아래쪽으로 뻗어있는 방향으로 설정함.
- ncurses 라이브러리 속 initscr()을 통해 main 창을 만들고 사이즈를 21,42으로 설정해줌
- 창의 색상을 사용하기 위해서는 ncurses 속 start_color() 함수 사용.
- 색상의 쌍을 정해주기 위해 (배경색 , 글자색) init_pair를 사용함.
- 전체적인 background를 지정해주기 위해서는 bkfd 함수를 사용하여 전체적인 색상을 결정함.
- 앞서 만들었던 show_map()을 이용하여 불변 벽, 가벽, background 의 색상을 각각 입혀줌.
- show_snake() 함수를 통해 뱀의 색상을 결정해줌.
- 마지막은 endwin()를 통해 만들어준 창을 닫아줌.
- 다음 단계에서 뱀의 움직임을 결정함.

2단계 스네이크 표현 및 조작 : 신윤재 & 이세희

전역변수 추가

```

#include <unistd.h>
#include <time.h>

#define tick 100000

```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

bool fail = false; // game over, init = false

int direction = 0; // 0: up, 1: right, 2: down, 3: left

```

- tick 변수를 설정하여 설정한 시간마다 움직일 수 있도록 설정. 현재 설정된 상태는 1초로 ns단위로 표현해야 함.
- 게임 실패와 성공을 판단할 수 있는 전역 변수 fail을 boolean 변수로 선언함.
- 스네이크의 진행 방향을 파악하고 진행방향을 변경할 수 있도록 하기 위해 direction변수 선언
- direction의 초기값은 0으로 설정하여 처음에는 위쪽 방향을 향하여 갈 수 있도록 설정함.

move() 함수

```

void move()
{
    int a = 0;
    int key ;

    while(!fail)
    {
        show_map(); show_snake();

        int cur_x = snake_x[0];
        int cur_y = snake_y[0];

        if(direction == 0){snake_x[0]--;}
        else if(direction == 1){snake_y[0]++;}
        else if(direction == 2){snake_x[0]++;}
        else if(direction == 3){snake_y[0]--;}

        key = getch();
        // 0: up, 1: right, 2: down, 3: left
        switch(key){
            case KEY_RIGHT : // 오른쪽
                if (direction == 3) {fail = true;} // left
                else {direction = 1;}
                break;
        }
    }
}

```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

    case KEY_LEFT : // 왼쪽
        if(direction == 1){fail = true;} // right
        else {direction = 3;}// up
        break;
    case KEY_DOWN : // 아래
        if(direction == 0){fail = true;} // up
        else {direction = 2;}
        break;
    case KEY_UP : // 위
        if (direction == 2) {fail = true;}// left
        else {direction = 0;}
        break;
    }

    for(int i = 1; i <= snake_y.size(); i++){

        int y = snake_y[i];
        int x = snake_x[i];

        snake_y[i] = cur_y;
        snake_x[i] = cur_x;

        cur_y = y; cur_x = x;
    }

    usleep(tick*3);
}
}

```

- snake 게임에 가장 핵심 구현인 move()함수의 경우 direction에 따라 snake의 머리 부분을 변경해주고, 몸체 부분을 하나씩 앞에 있던 칸으로 옮기는 방식으로 구현을 함.
- while문을 통해서 fail이 뜰 때까지 스네이크가 움질 일 수 있도록 구현함.
- show_snake() 함수와 show_map()함수를 while문이 한번 돌 때마다 실행되어 snake의 변경된 좌표값을 변환시켜 보여줌.
- direction의 값에 따라 head부분을 변경해줌.
 - 위쪽 : head의 x값만 위로 변경 (--)

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

- 아래쪽 : **head**의 x값만 아래로 변경(+)
- 오른쪽 : **head**의 y값을 오른쪽으로 변경 (+)
- 왼쪽 : **head**의 y값을 왼쪽으로 변경(-)
- 키보드의 입력 값을 받아오기 위해서 **getch()**라는 **ncurse** 라이브러리 속 함수를 사용함.
 - **getch()** 함수는 키보드의 입력값을 int 값이나 char값으로 모두 받아들일 수 있으며, 특정 키보드의 경우는 이름을 정해놓은 값도 존재함.
 - **move()**함수의 구현을 위해서는 라이브러리에서 정해놓은 키보드 화살표 입력값 이름을 사용함.
 - 오른쪽: **KEY_RIGHT** / 왼쪽 : **KEY_LEFT** / 아래 : **KEY_DOWN** / 위 : **KEY_UP**
- **getch()**를 통해 받아온 입력값을 **key**라는 변수로 받아서 사용함.
- **key** 변수를 **switch & case** 에서의 입력변수 값으로 사용하여 **direction**을 변경해줌.
 - 입력받은 키 값과 현재의 **direction**에 따라서 게임 종료가 되는 경우가 존재함.
 - 현재 **direction**과 입력받아서 움직이려고 하는 **direction**값이 서로 반대 방향이면 게임이 종료되게 됨. 따라서 이를 비교해주기 위해서 **if**문을 사용하여 만약 이 반대 방향 조건이 만족한다면 게임을 종료 할 수 있도록 구현함
 - 그렇지 않다면 **direction**의 값을 변경해줌.
- 변경된 x,y **head** 좌표에 몸체 부분이 따라올 수 있도록 하여 한칸씩 옮겨주도록 코드를 작성함.
- 시간에 맞게 변경시켜주기 위해서 **usleep()** 함수를 이용함.
 - 이는 **micro** 단위로 초를 받아와서 그만큼 잠깐 멈췄다가 실행시켜주는 함수로 이 프로젝터에서 시간 지연 구현시 사용함.

main함수 구현 및 수정

```
int main()
{
    int init_x = 11;
    int init_y = 11;
    // init snake place
    for (int i=0; i<4; i++)
    {
        snake_x.push_back(init_x + i);
        snake_y.push_back(init_y);
    }

    initscr();
    start_color();
    nodelay(stdscr, true);
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

noecho();
cbreak();
keypad(stdscr, TRUE);
// move snake
move();

getch();
endwin();

return 0;
}

```

수정

- `snake`의 x, y 초기값을 변수로 설정.
- 창의 사이즈를 지정하지 않고, 다른 창을 위에 더 표현 할 수 있도록 변경
- 배경 색상은 `show_map()`에서 지정해주기 때문에 삭제함.

구현 추가

- `nodelay()` 함수 추가
 - `getch()`함수는 키가 눌려질 때까지 기다리기 때문에 키가 눌리지 않는 경우에는 `snake`가 움직이지 않음.
 - 따라서 이러한 문제를 해결하기 위해서 `ncurses` 라이브러리에 있는 `nodelay`함수를 추가함
 - 이 함수의 경우 `blocking` 모드로 지정하지 않는 것을 뜻하고 입력이 오지 않으면 입력을 지속적으로 기다리지 않고 `ERR`를 반환하도록 함.
- `cbreak()` 함수 추가
 - `cbreak()` 함수도 `ncurses` 라이브러리에 있으며 이 함수는 `cbreak` 모드를 `on`으로 전환시키는 함수이다.
 - `on`으로 전환시면 사용자로부터 오는 입력을 즉시 프로그램에서 사용 가능하도록 함. 만약 이 함수가 추가되지 않는다면, 입력이 `newline`이 발생할때까지 버퍼에 저장 될 것이므로 필요한 함수로 추가함.
- `keypad()` 함수
 - `keypad()` 함수도 동일하게 `ncurses` 라이브러리에 포함된 함수.
 - 이 함수를 통해서 키패드의 입력을 활성화함.

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

- **move()**함수
 - 앞서 구현했던 **move()** 함수를 추가하여 실행 될 수 있도록 함.

3단계 Item 요소 구현 - 신윤재 & 이세희

전역변수 추가

```
//item
int poison_x, poison_y, growth_x, growth_y;

int item_timer = 0;
```

- 스네이크의 길이를 증가시킬 수 있는 item인 growth item의 x, y 좌표를 계산 할 수 있는 변수 선언.
- 스네이크의 길이를 감소시킬 수 있는 item인 poison item의 x, y 좌표를 계산 할 수 있는 변수 선언 .
- item의 변경 시간을 나타내는 timer 변수 선언.

get_item() 함수 구현

```
void get_item()
{
  init_pair(6, COLOR_RED, COLOR_RED); // poison
  init_pair(7, COLOR_BLUE, COLOR_BLUE); // growth

  // srand(NULL);
  srand(time(NULL));
  if(item_timer == 0){
    poison_x = rand() % 18 + 2;
    poison_y = rand() % 39 + 2;
    do{
      growth_x = rand() % 18 + 2;
      growth_y = rand() % 39 + 2;
    }
    while(poison_x == growth_x && growth_x == poison_x );
    item_timer = 30;
  }
  else {item_timer--;} }
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```
// show Item

attron(COLOR_PAIR(6));
mvprintw(poison_x, poison_y, " ");
attroff(COLOR_PAIR(6));
attron(COLOR_PAIR(7));
mvprintw(growth_x, growth_y, " ");
attroff(COLOR_PAIR(7));
}
```

- item 색상
 - poison의 경우는 빨간색, growth 아이템은 파란색으로 설정해줌
- item의 위치 설정
 - random으로 설정하기 위해서 rand()함수를 사용해줌.
 - 만약 두개의 아이템이 위치가 동일하다면 다시 random하게 위치를 지정해줌.
- item timer
 - get_item 함수는 move() 함수 속에 들어가 있기 때문에 tick의 영향을 받게 됨.
 - 스네이크가 한번 움직일 때마다 아이템을 변경시키면 안되기 때문에 timer 기능을 만들어서 구현함
 - 초기 timer를 전역변수에서 0으로 지정한 후 한번 위치를 설정해주면 timer를 30으로 설정해줌.
 - timer 설정 후 한 tick마다 -1을 해주며 다시 0이 된다면 위치를 item의 위치를 변경시켜줌.
- Map 속 구현
 - 아이템을 각자 정해진 색상에 맞도록 window에 표시함.

move() 함수의 수정 (수정된 부분만 볼께)

```
void move()
{
    while(!fail)
    {
        get_item();

        if (cur_x == growth_x && cur_y == growth_y) {
            int lastx = snake_x[-1];
            int lasty = snake_y[-1];
        }
    }
}
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

    if(direction == 0)
        {snake_x.push_back(++lastx); snake_y.push_back(lasty);}
    else if(direction == 1)
        {snake_x.push_back(lastx);snake_y.push_back(--lasty);}
    else if(direction == 2)
        {snake_x.push_back(--lastx);snake_y.push_back(lasty);}
    else if(direction == 3)
        {snake_x.push_back(lastx);snake_y.push_back(++lasty);}
    }

    else if (cur_x == poison_x && cur_y == poison_y)
    {
        snake_x.pop_back();snake_y.pop_back();
        if(snake_x.size()<3) {fail = true;}
    }
}

}
}
}

```

- 앞서 구현한 `get_item`함수를 추가함
- `item`을 `get` 했는지 확인하기 위해 스네이크의 `head` 부분을 확인함.
 - 현재의 스네이크 머리와 `growth item` & `poison item`과 x,y 좌표가 동일하면 아이템을 먹은것으로 판단
- `growth item`의 경우 가는 방향에 따라 추가되는 몸체 부분의 x,y 좌표가 변경됨.
 - `push`를 통해서 `vector` 값을 추가해줌.
 - 위쪽 방향으로 움직이고 있는 경우 : 마지막 x좌표에 +1 을 해준 값을 추가해주고, y좌표는 동일한 것을 추가해줌.
 - 아래쪽 방향으로 움직이고 있는 경우 : 마지막 x좌표는 동일한 값을 추가해주고, 마지막 y 좌표를 -1 한 값을 추가해줌.
 - 오른쪽 방향으로 움직이고 있는 경우 : 마지막 x 좌표의 -1 한 값을 추가해주고, 마지막 y좌표에 동일한 값을 추가해줌
 - 왼쪽 방향으로 움직이고 있는 경우 : 마지막 x 좌표는 동일한 값을 추가해주며, 마지막 y 좌표는 +1 한 값을 추가해줌.
- `poison item`을 먹는 경우 현재 가는 방향과는 관련 없이 맨 뒤에 있는 값을 제거시켜줌.

4단계 : Gate 요소의 구현 : 신윤재 & 이세희

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

전역변수 추가

```
int gate1_x, gate1_y, gate2_x, gate2_y;
int gate_timer = 0;
```

- 두개의 **gate**의 x,y를 선언 함(다른 함수에서도 쓰이기 때문에 전역변수로 선언)
- item_timer**와 동일하게 **gate**도 tick에 따라 움직이면 안되기 때문에 **timer**을 사용하여 새롭게 만들어지는 **gate**들의 시간을 늦추는 방식을 사용함.

main 함수 변경 및 추가

```
// move snake
curs_set(0);

while(!fail) {
    fail = move();
}
```

- curs_set()** 함수를 통해서 현재 커서의 위치를 보이지 않도록 설정함.
- move**에서의 **while**문을 **main** 함수로 변경하여 돌아갈 수 있도록 하여 보기 쉽도록 변경.

move 함수 추가

```
show_gate(); in_gate();
```

- 구현한 함수를 **move**함수에 추가하여 함께 실행될 수 있도록 함.

show_gate()함수 구현

```
void show_gate()
{
    init_pair(8, COLOR_MAGENTA, COLOR_MAGENTA);
    srand(time(NULL));
    if(gate_timer == 0) // not immune wall
    {
        do{
            gate1_x = rand() % 21;
            gate1_y = rand() % 42;
            do{
                gate2_x = rand() % 21;
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

        gate2_y = rand() % 42;
    }
    while(gate1_x == gate1_y && gate1_x == gate2_y);
}
while
(
    (gate1_x == 20 && gate1_y == 41) ||
    (gate2_x == 20 && gate2_y == 41) // (20, 41)
    (gate1_x == 20 && gate1_y == 0) ||
    (gate2_x == 20 && gate2_y == 0) // (20, 0)
    (gate1_x == 0 && gate1_y == 41) ||
    (gate2_x == 0 && gate2_y == 41) // (0, 41)
    (gate1_x == 0 && gate1_y == 0) ||
    (gate2_x == 0 && gate2_y == 0) // (0, 0)
); // when value = 2 position
gate_timer = 70;
}
else {gate_timer--;

// show gate
attron(COLOR_PAIR(8));
mvprintw(gate1_x, gate1_y, " ");
mvprintw(gate2_x, gate2_y, " ");
attroff(COLOR_PAIR(8));
}

```

- **gate**의 색상은 magenta 색상으로 설정함.
- **gate** 1과 **gate2**가 서로 동일한 곳에 있으면 안되기 때문에 **do while**문을 통해 만약 서로가 동일한 좌표값을 가지고 있다면 다시 **gate 2**의 좌표를 바꿔주는 형식을 사용함.
- 또한, **gate**는 가변 벽에는 생성될 수 있지만, 불변 벽에는 생성될 수 없는 특징을 가짐.
- 따라서 현재 읽어들인 **map1**의 불변벽인 4방향 모서리에 **gate**가 생성되면 다시 생성 될 수 있도록 구현함.
- **gate timer**의 경우는 앞서 말했던 **item timer**와 동일하게 전역변수를 통해서 처음에는 0으로 설정을 해주고, 그 후 초기 **gate**들이 설정되면 **timer**를 70으로 맞춰줌.
- 그 후 0이 될때까지 **time**을 하나씩 줄여주는 방식으로 **timer**를 구현함.
- 이렇게 설정된 좌표들을 가지고 창에 표시해줌.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

in_gate() 함수 구현

```

void in_gate()
{
    int cur_x = snake_x[0];
    int cur_y = snake_y[0];
    if((cur_x == gate1_x && cur_y == gate1_y) || (cur_x == gate2_x &&
cur_y == gate2_y))
    {
        if(cur_x == gate1_x) {cur_x = gate2_x; cur_y = gate2_y;}
        else {cur_x = gate1_x; cur_y = gate1_y;}

        switch (direction)
        {
        case 0:
            if(map[cur_x - 1][cur_y] ==0 && cur_x - 1 > 0 )
                {snake_x[0] = cur_x - 1
                 ;snake_y[0] = cur_y;
                 direction = 0 ;}
            else if (map[cur_x] [cur_y + 1] ==0 && cur_y + 1 < 42)
                {snake_x[0] = cur_x ;
                 snake_y[0] = cur_y + 1;
                 direction = 1;}
            else if (map[cur_x] [cur_y - 1]==0 && cur_y - 1> 0 )
                {snake_x[0] = cur_x ;
                 snake_y[0] = cur_y - 1;
                 direction = 3;}
            else if (map[cur_x + 1] [cur_y ] ==0 && cur_x + 1< 21)
                {snake_x[0] = cur_x + 1 ;
                 snake_y[0] = cur_y;
                 direction = 2;}
            break;
        case 1:
            if (map[cur_x] [cur_y + 1] ==0 && cur_y + 1 <42)
                {snake_x[0] = cur_x ;snake_y[0] = cur_y + 1;
                 direction = 1;}
            else if (map[cur_x + 1] [cur_y ] ==0 && cur_x + 1 < 21)

```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

    {snake_x[0] = cur_x+1 ;
     snake_y[0] = cur_y; direction = 2;}
else if (map[cur_x - 1][cur_y] ==0 && cur_x - 1 > 0 )
    {snake_x[0] = cur_x - 1 ;
     snake_y[0] = cur_y; direction = 0 ;}
else if (map[cur_x][cur_y - 1]==0 && cur_y - 1> 0 )
    {snake_x[0] = cur_x ;
     snake_y[0] = cur_y - 1; direction = 3;}
break;

case 2:
if (map[cur_x + 1][cur_y ] ==0 && cur_x + 1< 21)
    {snake_x[0] = cur_x + 1 ;
     snake_y[0] = cur_y; direction = 2;}
else if (map[cur_x][cur_y - 1]==0 && cur_y - 1> 0 )
    {snake_x[0] = cur_x ;snake_y[0] = cur_y - 1;
     direction = 3;}
else if (map[cur_x][cur_y + 1] ==0 && cur_y + 1 < 42)
    {snake_x[0] = cur_x ;snake_y[0] = cur_y + 1;
     direction = 1;}
else if (map[cur_x - 1][cur_y] ==0 && cur_x - 1 > 0 )
    {snake_x[0] = cur_x - 1 ;snake_y[0] = cur_y;
     direction = 0 ;}
break;

case 3:
if (map[cur_x][cur_y - 1]==0 && cur_y - 1> 0 )
    {snake_x[0] = cur_x ;snake_y[0] = cur_y - 1;
     direction = 3;}
else if (map[cur_x - 1][cur_y] ==0 && cur_x - 1 > 0 )
    {snake_x[0] = cur_x - 1 ;snake_y[0] = cur_y;
     direction = 0 ;}
else if (map[cur_x + 1][cur_y ] ==0 && cur_x + 1< 21)
    {snake_x[0] = cur_x + 1 ;snake_y[0] = cur_y;
     direction = 2;}
else if (map[cur_x][cur_y + 1] ==0 && cur_y + 1 < 42)
    {snake_x[0] = cur_x ;snake_y[0] = cur_y + 1;
     direction = 1;}

```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

      direction = 1; }

    break;
  }
}

}

```

- 이 함수는 스네이크가 만들어놓은 **gate**에 들어갔을 때 어디로 움직여야 하는지를 구현해 놓은 함수로 이는 **head**의 x,y 좌표만 변경하고 그 후 스네이크의 바디는 **move**함수에서 위쪽 단계에서 구현해 놓은 것을 동일하게 사용함.
- **gate**에서 들어가고 나오는 방향은 들어가는 진행 방향과 동일한 방향으로 나오게 되지만 만약 막혀있다면 다음과 같은 순서를 갈 수 있는 방향을 체크한다.
 - 들어온 진행 방향과 같은 방향
 - 진행 방향의 오른쪽 방향
 - 진행 방향의 왼쪽 방향
 - 진행방향의 반대방향
- 이를 구현하기 위해서 먼저 들어온 **gate**가 1번 **gate**인지, 2번 **gate**인지 확인해 준 후 **cur_x** 와 **cur_y**변수에 들어온 **gate**의 x, y 좌표를 넣어줌.
- **switch & case** 함수를 통해 매개변수로 현재 진행 방향을 받아오고 이에 따라 알맞게 구현
 - 먼저 앞서 말한 순서대로 스네이크가 나가려고 하는 방향에 **map**의 값이 0인지 확인함.
 - 0이 맞다면 벽이 없는 것이기 때문에, 스네이크의 머리의 방향을 변경해주고 진행 방향도 변경시켜줌.
 - 예를 들어, 만약 현재 진행방향이 위쪽이였는데, **gate 2**의 위쪽 방향이 막혀있다면, 오른쪽 방향을 탐색함. 만약 오른쪽 방향도 막혀있다면 왼쪽 방향을 탐색하고 이 쪽이 0값이랑 동일하면 **direction**을 3(왼쪽)으로 변경해주고 스네이크 머리의 값도 변경시켜줌.

5단계 :점수 요소의 구현 및 **stage**변화 신윤재 & 이세희

전역 변수 추가 및 변경

```

WINDOW *score_board;
WINDOW *goal_board;

int stage = 1;

int cur_x; int cur_y;

```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```
// score
int get_growth, get_poison, get_gate, max_length = 4;
int goal_growth = 1, goal_poison = 1, goal_gate = 0, goal_length = 5;
float play_time;
```

- 두개의 새로운 창을 만들어 주기 위해 사용함.
- 현재의 **score**를 알려주는 창과, 어떤 미션이있는지 알려주는 창 총 두개의 창을 선언해줌.
- **stage** 변수를 선언하면서 **stage**를 변경시켜줌.
- 현재의 머리 위치를 전역변수로 선언해줌. (**cur_x**, **cur_y**)
- 각 **item**의 목표값과 현재 내가 취득한 **item**의 수도 각각 변수로 선언해줌.
- 길이의 경우 처음 **init**가 4이기 때문에 4로 초기화를 시켜줌.
- 현재 얼만큼의 시간이 초과되어있는지 알수 있도록 **play_time** 변수를 선언해줌.

show_map() 함수 변경

```
void show_map(int stage)
{
    ifstream map_file("./snakemap/snakeMap" + to_string(stage) + ".txt");
```

- 앞선 **stage**까지는 만들어놓은 1가지의 **map**만 사용했지만, **stage**를 변경할 때마다 **map**의 구현을 바꿔줘야 하기 때문에 **stage**에 맞게 **map**을 변경시켜줌.
- 그러기 위해 **show_map** 함수의 매개변수로 **stage**를 입력받아서 활용함.

move() 함수 변경

```
bool move()
{
    show_map(stage);
}
```

- **show_map** 함수를 변경해서 넣어줌.

get_item() 함수 변경

```
void get_item()
{
    init_pair(6, COLOR_RED, COLOR_RED); // poison
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

init_pair(7, COLOR_BLUE, COLOR_BLUE); // growth

srand(time(NULL));
if(item_timer == 0){
  do{
    poison_x = rand() % 18 + 2;
    poison_y = rand() % 39 + 2;
    do{
      growth_x = rand() % 18 + 2;
      growth_y = rand() % 39 + 2;
    }
    while(poison_x == growth_x && growth_x == poison_x );
  }
  while (map[poison_x][poison_y] == 1 ||
         map[poison_x][poison_y] ==2 ||
         map[growth_x][growth_y] == 1 ||
         map[growth_x][growth_y] == 2); // when value = 2 position

  item_timer = 50;
}
else {item_timer--;}

// show Item
show_item();
}
  
```

- 위에 단계까지에서는 `item`을 을 표현을 할때 단지 가장자리만 고려하여 만들어 줬지만, 어디에 있을지 모를 벽들을 위해 `while`문을 통해서 이를 탐지하고 다시 구현해줌.
- 또한, 게임을 실행하면서 `timer`를 적절하게 변경.

show_gate() 함수 변경

```

void show_gate()
{
  init_pair(8, COLOR_MAGENTA, COLOR_MAGENTA);
  srand(time(NULL));
  if(gate_timer == 0) // not immune wall
  {
    
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

    do{
        gate1_x = rand() % 21;
        gate1_y = rand() % 42;
        do{
            gate2_x = rand() % 21;
            gate2_y = rand() % 42;
        }
        while(gate1_x == gate1_y && gate1_x == gate2_y);
    }
    while
        (map[gate1_x][gate1_y] == 2 || map[gate2_x][gate2_y] ==2); //when value = 2 position
        gate_timer = 50;
    }
    else {gate_timer--;}

    // show gate
    attron(COLOR_PAIR(8));
    mvprintw(gate1_x, gate1_y, " ");
    mvprintw(gate2_x, gate2_y, " ");
   attroff(COLOR_PAIR(8));
}

```

- `get_item()`과 동일하게 `stage`마다 벽이 변경되기 때문에 이에 맞게 `gate`도 변경시켜줘야함.
- 따라서 네 모퉁이만을 탐지하는 것이 아닌, 현재 위치로 설정된 곳이 불변의 벽이 맞는지 즉 값을 2가 맞는지 확인을 하고 이가 맞다면 다시 값을 추출하도록 구현

show_score() 구현

```

void show_score(){
    init_pair(9, COLOR_BLACK, COLOR_WHITE);

    score_board = newwin(10, 20, 0, 43);
    goal_board = newwin(10, 20, 11, 43);
    wbkgd(score_board, COLOR_PAIR(9));
    wbkgd(goal_board, COLOR_PAIR(9));
    wattron(score_board, COLOR_PAIR(9));
    mvwprintw(score_board, 1, 1, "Score board");
}

```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

mvwprintw(score_board, 2, 1, "B: %d / %d", snake_x.size(),
max_length);

mvwprintw(score_board, 3, 1, "+: %d", get_growth);
mvwprintw(score_board, 4, 1, "-: %d", get_poison);
mvwprintw(score_board, 5, 1, "G: %d", get_gate);
wborder(score_board, '@', '@', '@', '@', '@', '@', '@', '@');

wattroff(score_board, COLOR_PAIR(9));
wrefresh(score_board);

wattron(goal_board, COLOR_PAIR(9));
mvwprintw(goal_board, 1, 1, "Mission");
mvwprintw(goal_board, 2, 1, "B: %d", goal_length);
if (max_length < goal_length) {wprintw(goal_board, " ( )");} else
{wprintw(goal_board, " (V)");}
mvwprintw(goal_board, 3, 1, "+: %d", goal_growth);
if (get_growth < goal_growth) {wprintw(goal_board, " ( )");} else
{wprintw(goal_board, " (V)");}
mvwprintw(goal_board, 4, 1, "-: %d", goal_poison);
if (get_poison > goal_poison) {wprintw(goal_board, " ( )");} else
{wprintw(goal_board, " (V)");}
mvwprintw(goal_board, 5, 1, "G: %d", goal_gate);
if (get_gate < goal_gate) {wprintw(goal_board, " ( )");} else
{wprintw(goal_board, " (V)");}
wborder(goal_board, '@', '@', '@', '@', '@', '@', '@', '@');
wattroff(goal_board, COLOR_PAIR(9));
wrefresh(goal_board);
}

```

- **score**를 보여주는 두개의 창을 만들어줌.
- **score**를 보여주는 창과 미션창 두개를 만들어서 위쪽에 현재 사용자가 획득한 **score** 창이 위치하도록 하였고 그 바로 밑에 미션창이 뜨도록함.
- **score_board**
 - **score_board** 에 맨위에는 현재 이 창이 어떤 창인지를 확인하기 위해 “Score board”를 출력해줌.
 - “B:” 현재 스네이크의 바디의 길이가 어떤지 출력

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

- “+:” : 현 스테이지에서 먹은 growth item의 수를 출력
- “-:” : 현 스테이지에서 먹은 poison item 수를 출력
- ‘G:’ : 현 스테이지에서 통과한 gate 수를 출력
- goal_board
 - goal_board 에 맨위에는 현재 이 창이 어떤 창인지를 확인하기 위해 “Mission”을 출력해줌.
 - “B:” 스네이크의 바디의 길이의 미션길이를 출력.
 - “+:” : 현 스테이지에서 먹어야 하는 growth item의 수를 출력
 - “-:” : 현 스테이지에서 먹어야하는 poison item 수를 출력
 - ‘G:’ : 현 스테이지에서 통과해야 하는 gate 수를 출력
 - 만약에 이 미션들을 각각 통과하게 되면 괄호 안에 각각 V체크를 하도록 구성.

init() 함수 구현

```
void init() {
    while(snake_x.size() != 0)
    {
        snake_x.pop_back();
        snake_y.pop_back();
    }

    int init_x = 15;
    int init_y = 17;

    for (int i=0; i<4; i++)
    {
        snake_x.push_back(init_x + i);
        snake_y.push_back(init_y);
    }

    direction =0;

    get_growth=0; get_poison= 0; get_gate= 0; max_length= 4;
    switch(stage){
        case 1:
            goal_growth = 2;
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

goal_poison = 0;
goal_gate = 2;
goal_length = 6;
break;

case 2:
    goal_growth = 3;
    goal_poison = 0;
    goal_gate = 3;
    goal_length = 7;
    break;

case 3:
    goal_growth = 4;
    goal_poison = 1;
    goal_gate = 4;
    goal_length = 8;
    break;

case 4:
    goal_growth = 7;
    goal_poison = 2;
    goal_gate = 6;
    goal_length = 9;
    break;
}

```

- `init()` 함수는 `stage`가 바뀔 때마다 초기화를 해주는 함수이다.
- `snake` 초기화
 - 만약에 전 `stage`에서 `snake`가 길이가 늘어났다면 이를 다시 없애줌.
 - 머리의 위치도 초기화 시켜줌.
 - 바디의 위치도 초기화 시켜서 위쪽 방향을 보도록 만들어줌.
 - 방향의 위치도 0으로 초기화.
- 점수의 초기화
 - 지금까지 획득했던 점수들을 모두 다시 초기화 시켜줌.
 - `stage`에 맞도록 목표치를 서로 다르게 만들어서 점점 더 단계를 깨기 어렵게 구현.

main 함수 변경

```
int main()
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

```

{
    resize_term(100, 100);

for(; stage< 5; stage++)
{
    init();
    show_map(stage);
    // show_score();

    while(!fail)
    {
        play_time += 0.1;
        fail = move();
        show_score();
        if ((max_length >= goal_length) && (get_growth >= goal_growth)
&& (get_poison <= goal_poison) && (get_gate >= goal_gate) ) {break;}
    }

    if(fail == 1)
        break;

    getch();
}

sleep(4);
clear();
getch();

endwin();
    if (fail){cout << "\n\nFailed!\nPlay time: "<< play_time <<"sec\n\n";}
    else{cout << "\n\nSuccessed!\nPlay time: "<< play_time <<"sec\n\n";}
    return 0;
}

```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

}

- 기본적인 원도우 창을 늘려서 **score** 창과 **goal** 창이 들어갈 수 있도록 설정함. (100,100)으로 사이즈 설정.
- **for**문을 통해서 만약에 따로 실패를 해서 빠져나오지 않는다면 **stage**를 하나씩 늘려줌.
- **init()**함수를 통해서 매 **stage**마다 초기화해줌.
- **show_map(stage)**로 **map**에 **stage** 변수를 줌에 따라 **stage**에 맞는 **map**을 그려줄 수 있도록 함.
- **while**문을 통해서 스네이크가 계속 움직이고, 미션 창이 업데이트 되도록 함.
- **while** 문이 돌갈때마다 0.1초씩 늘려주며 **play time**을 확인함
- **while** 문을 빠져나올 수 있는 방법은 총 2가지의 방법이 존재함.
 - **move()**에서 **fail**을 **true**로 만드는 경우(반대 방향키를 눌렀을 때, 벽에 부딪혔을 때 등)
 - 목표치를 도달했을 때.. : **if** 문을 통해서 **break** 해줌.
- **while** 문을 빠져나온 후 만약 **move()**의 **fail**을 통해서 빠져나왔는지 확인하고 만약 그렇다면 완전히 게임을 끝낼 수 있도록 함.
- 게임이 끝나면 4초정도 기다리게 한 후 창을 닫아줌.
- 그후 터미널 창에 성공 여부와 게임시간을 출력해줌.

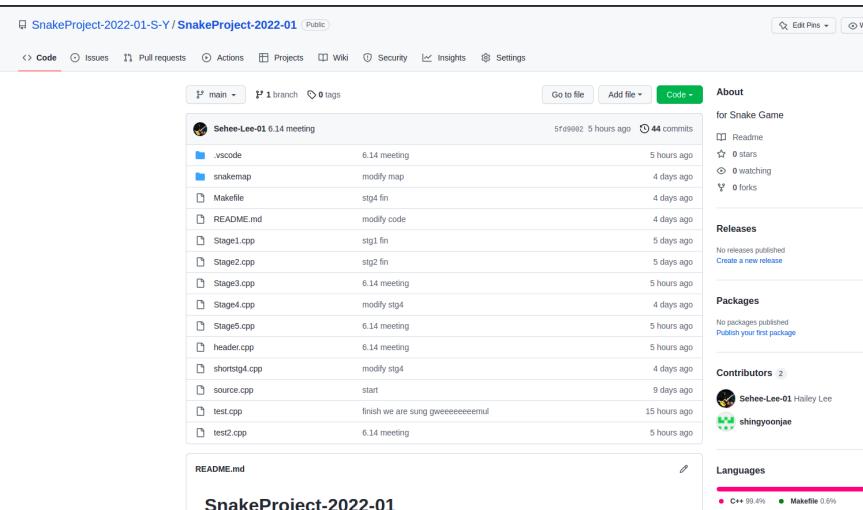
 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

2.2.3 활용/개발된 기술

1) 사용 라이브러리 설명

사용 라이브러리 및 클래스	설명
<ncurses.h>	ncurses는 텍스트 사용자 인터페이스를 터미널 독립 방식으로 기록할 수 있도록 API를 제공하는 프로그래밍 라이브러리이다. 스네이크 게임을 화면에 구현하는데에 사용했다.
<unistd.h>	POSIX 운영 체제 API에 대한 액세스를 제공하는 헤더 파일의 이름이다.
<fstream>	파일에서 읽어오거나 파일에 쓰는 클래스이다. 파일을 불러오고 읽는데에 사용하였다.
<iostream>	C++ 표준 라이브러리의 하나이다. 기본적인 C++ 기능을 사용할 수 있도록 하였다.
<vector> (STL)	벡터(queue)를 구현한 STL 라이브러리이다. 스네이크의 머리 및 몸통 위치 좌표를 저장하고 몸통의 길이를 변화시킬 때 사용하였다.
<time.h>	날짜와 시간 조작 명령을 구현하는 C 프로그래밍 언어의 표준 라이브러리이다. usleep 함수를 이용하여 tick을 구현해주었다.

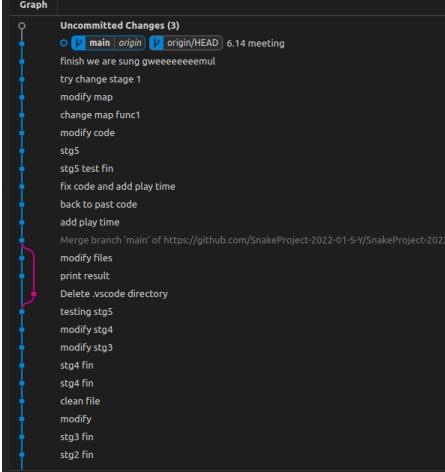
2) 기타 활용 프로그램



The screenshot shows a GitHub repository page for "SnakeProject-2022-01-S-Y / SnakeProject-2022-01". The repository has 44 commits from user "Sehee-Lee-01". The commits are listed in descending order of age, mostly from 6 hours ago. The repository includes files like vscode, snakemap, Makefile, README.md, Stage1.cpp, Stage2.cpp, Stage3.cpp, Stage4.cpp, Stage5.cpp, header.cpp, shortstage4.cpp, source.cpp, test.cpp, and test2.cpp. The README.md file contains the text "SnakeProject-2022-01". The repository has 0 stars, 0 forks, and 0 releases. It also lists contributors Sehee-Lee-01 and shingyoonjae, and languages C++ (99.4%) and Makefile (0.6%).

스네이크 프로젝트 **Github** 저장소

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	스네이크 프로젝트		
	팀 명		
	Confidential Restricted		Version 1.3
			2022-JUN-05



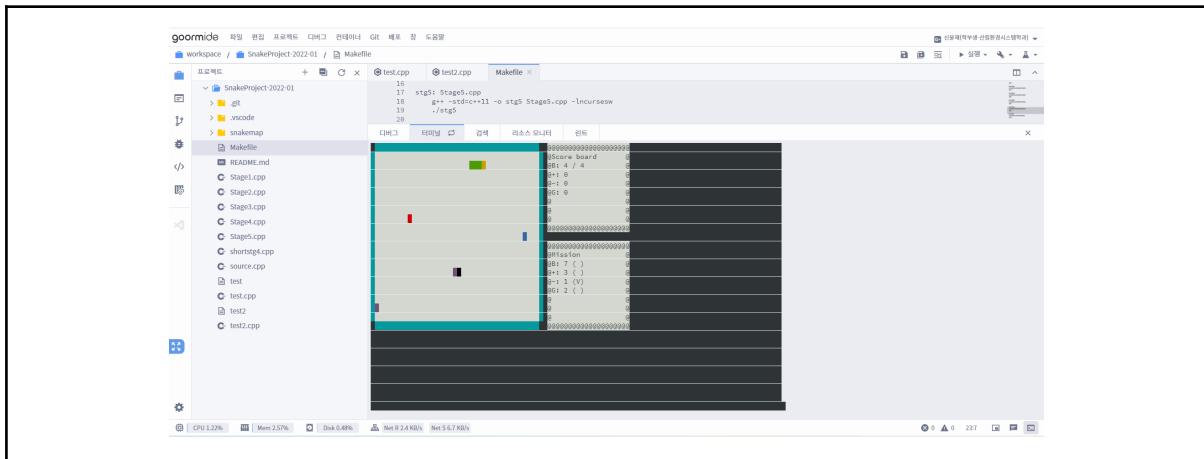
```

Graph
Uncommitted Changes (3)
o [ ] main [origin] [origin/HEAD] 6.14 meeting
finish we are sung gweeeeeemul
try change stage 1
modify map
change map func1
modify code
stg5
stg5 test fin
fix code and add play time
back to past code
add play time
Merge branch 'main' of https://github.com/SnakeProject-2022-01-S/Y/SnakeProject-2022-01-S
modify files
print result
Delete .vscode directory
testing stg5
modify stg4
modify stg3
stg4 fin
stg4 fin
clean file
modify
stg3 fin
stg2 fin

```

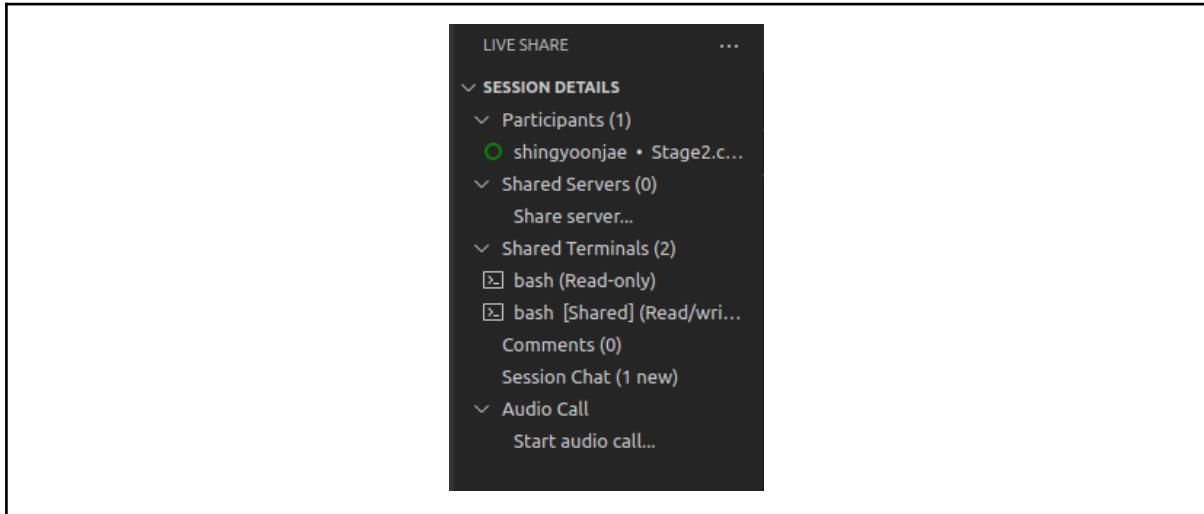
Git log로 확인한 commit들

Git과 Github를 이용해 프로젝트의 단계별 구현을 기록해왔다. 단계별로 어떤 기능을 추가해왔고 어떤 기능을 수정했는지 알기에 편리하여 프로젝트 진행 및 관리에 많은 도움을 주었다.



구름 IDE 사용 환경

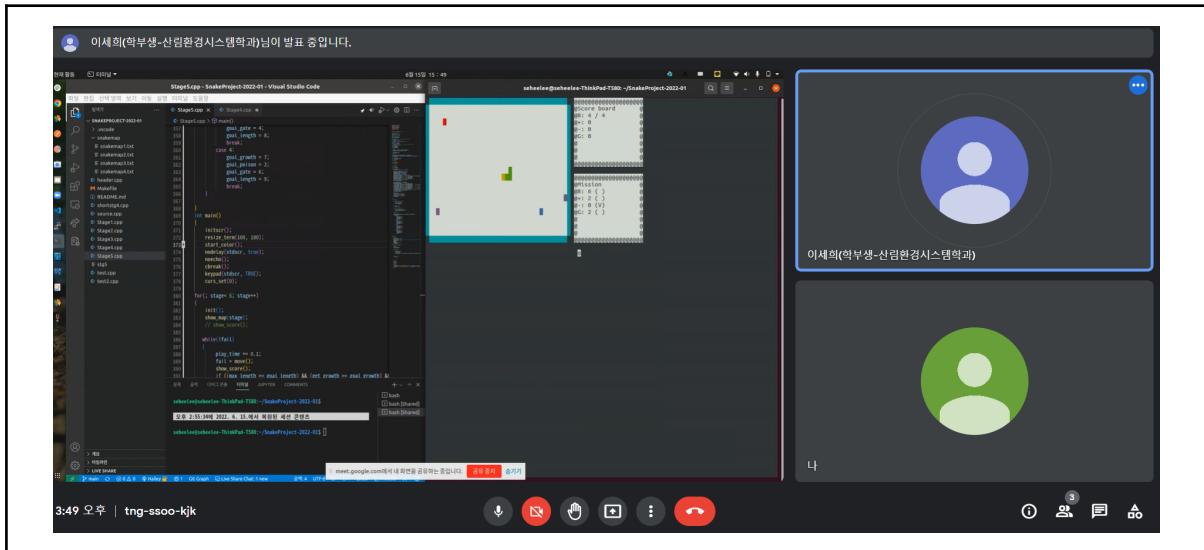
 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05



Live Share 접속 확인창

팀원과 사용하는 운영체제가 다르고 게임 구현에 필요한 `ncurses` 라이브러리가 우분투 환경에서만 실행되어 동시 접속이 되고 작업이 가능한 구름 IDE를 이용해 프로젝트 진행을 시도해 보았다. 하지만 실시간 업데이트가 조금 느리고 스네이크 게임을 실행하기에는 적합하지 않은 환경이었다.

그래서 동시 접속이 가능하고 동시 작업이 가능한 다른 방법을 찾아야 했다. 그래서 Visual Studio Code의 확장 도구 중 Live Share를 이용하여 로컬 IDE를 공유하여 작업을 진행했다. 터미널도 공유가 가능하기 때문에 공동 작업을 할 때 편리한 도구이다.



구글 미트 실행 캡쳐

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

또한 구글 미트를 이용하여 실시간 피드백 및 기능 구현 함수의 입출력 결정 등을 진행하였다.

2.2.4 현실적 제한 요소 및 그 해결 방안

- **3단계, 4단계: tick 중복 적용의 어려움**

스네이크, 아이템, 게이트에 각각 같은 **tick**이 적용 되다보니 아이템과 게이트의 위치 변화 속도가 빨라 게임을 구현하기 어려웠다. **tick**의 텀을 늘리면 스네이크가 느려지고, 텀을 좁히면 아이템 및 게이트의 변화속도가 빨라 이를 획득하거나 통과하기 어려웠다.

이러한 문제를 해결하기 위해 **integer** 전역변수를 사용하였다. 이는 **timer**와 같은 역할을 한다. 텀을 주고 싶은 만큼 **timer**의 최대치를 설정해주고 그 최대치에 **timer** 변수가 도달하면 이때 아이템과 게이트의 위치를 바꾸어준다. 이를 통해 스네이크의 속도와 아이템 및 게이트의 위치변화 속도를 적절하게 조절할 수 있었다.

2.2.5 결과물 목록

1. 파일 구조

파일 구조도
<pre> ├── snakemap │ ├── snakemap1.txt │ ├── snakemap2.txt │ ├── snakemap3.txt │ └── snakemap4.txt ├── Makefile ├── README.md └── Stage1.cpp ├── Stage2.cpp ├── Stage3.cpp ├── Stage4.cpp └── Stage5.cpp </pre>

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

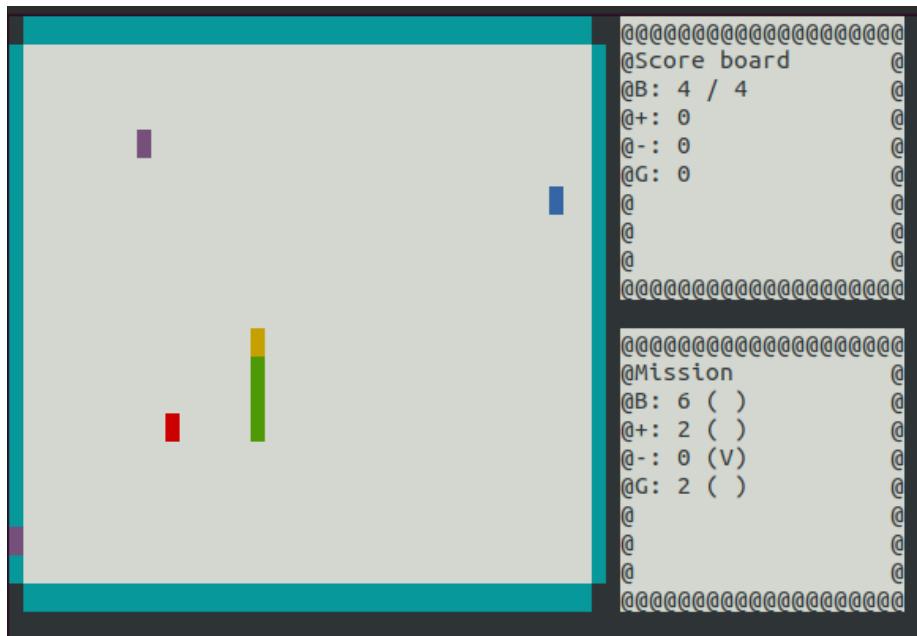
 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

2. 파일 설명

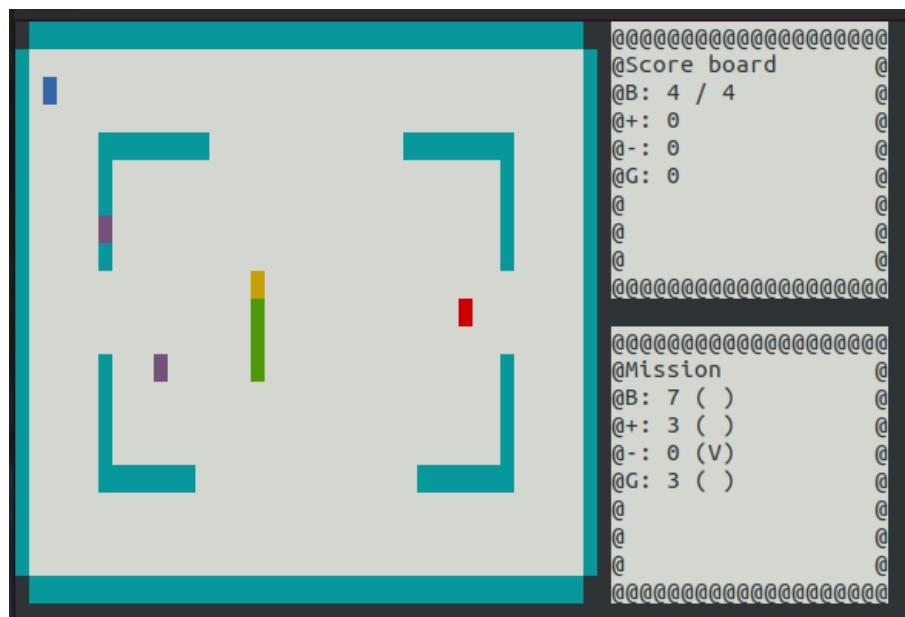
파일명	파일 설명
snakemap*.txt	스네이크 맵의 정보를 담고 있는 파일이다.
Makefile	각 단계별 Stage*.cpp 파일을 컴파일과 동시에 실행 파일을 실행시켜주는 명령어를 정리해놓은 파일이다.
README.md	프로젝트에 대한 정보를 담고 있는 파일이다.
Stage1.cpp	1단계를 구현한 파일이다.
Stage2.cpp	2단계를 구현한 파일이다.
Stage3.cpp	3단계를 구현한 파일이다.
Stage4.cpp	4단계를 구현한 파일이다.
Stage5.cpp	5단계를 구현한 파일이다.

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

2) 스네이크 게임 캡쳐

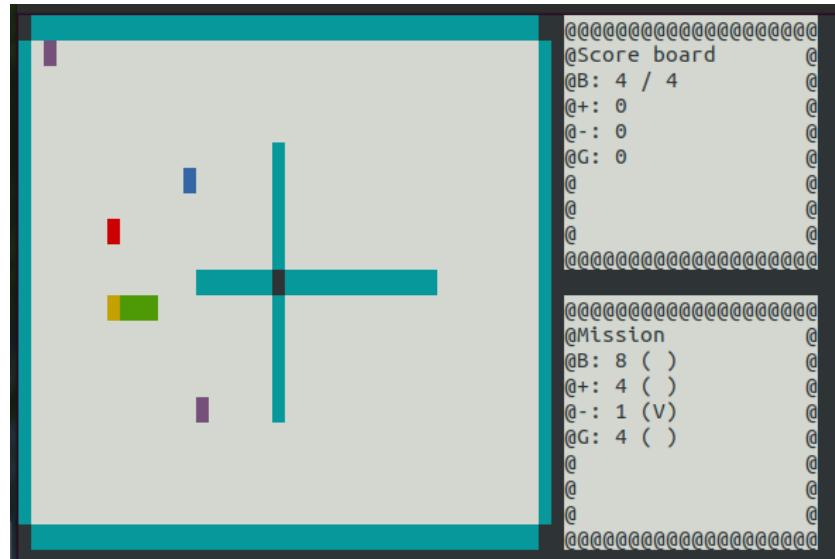


스네이크 게임 화면(1단계)

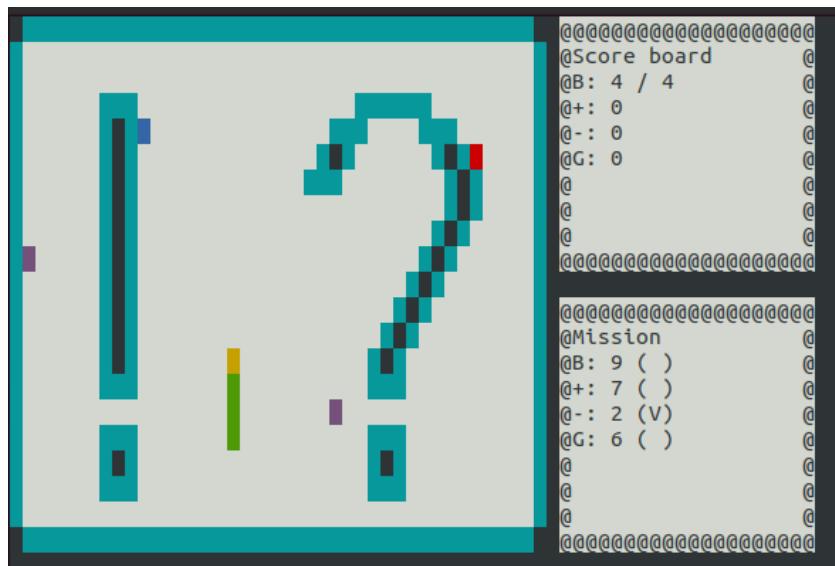


스네이크 게임 화면(2단계)

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05



스네이크 게임 화면(3단계)



스네이크 게임 화면(4단계)

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

3 자기평가

1. 신윤재

1) 역할

역할을 정확하게 정해놓기 보다는 한 기능을 구현할 때 각자 어떤 부분을 맡아서 할지 정한 후 실시간으로 어려움을 공유하고 해결해 나가는 방식으로 구현했다.

2) 프로젝트 수행 시 어려운 점

리눅스 환경이 아닌 공간에서 리눅스 기반 라이브러리를 쓰려다 보니, 이를 어떻게 구현하고 실행해야하는지에 대한 어려움이 가장 커던 것 같다. 하지만 여러 기능들을 찾아보고 조언을 구하면서 구름 IDE를 찾아냈고 완벽한 환경은 아니었지만 그 속에서 실행 및 구현해 나갈 수 있었다.

처음에는 어떤 기능을 가진 라이브러리가 있고 이에 대한 사용방법을 알아가는 것이 어려웠다. 내가 기능을 구현함에 있어서 다양한 라이브러리의 사용이 큰 도움이 된다는 점을 깨달았고 점차 여러 라이브러리를 찾아보게 되고 C++에는 어떤 기능이 있는지 등을 알아갈 수 있었다.

같이 코드를 수행하다 보니 서로가 어떤 부분을 고치고 있고, 어떤식으로 해결해 나가는지 어려울 수 있다고 생각했는데, 라이브로 함께 하다 보니 빠르게 이를 파악해나갈 수 있었다.

3) 프로젝트 운영에 개선이 필요하다고 생각하는 점

만약 다양한 OS에서 사용이 편리한 라이브러리를 사용했다면 조금 더 다양한 방법으로 코드를 구현할 수 있었을 것이라고 생각한다. 나의 경우 Git 를 더 배워서 더 잘 정리할 수 있으면 좋을 것이라고 생각했다. 프로젝트를 딜레이 없이 좀 더 개선이 필요하다. 또한, 더 많은 아이디어를 가지고 더 재미있게 만들 수 있다면 이 또한 개선이 필요하다고 생각한다.

4) 팀원 평가

다양한 프로젝트 경험이 있는 팀원과 함께 하여 수월하게 프로젝트를 진행할 수 있었다. 다양한 경험으로 인해 배울점이 많은 팀원으로 더 배울 점이 많고 재미있는 프로젝트를 할 수 있었다.

2. 이세희

1) 역할

구글 미트와 라이브 쉐어를 통해 팀원과 소통하면서 동시에 게임 구현을 진행했다. 1단계부터 5단계까지 실시간으로 함께 화면을 보며 토론을 통해 기능을 구현해 나갔다.

2) 프로젝트 수행 시 어려운 점

새로운 라이브러리를 사용하다보니 진입 장벽이 조금 크게 느껴졌다. 하지만 관련 내용을 찾아보고 예시 문서를 보면서 수월하게 라이브러리를 이용할 수 있었다. 또한 스네이크 단계 구현이 많아질 수록 게임의 딜레이가 점점 심해져갔다. 이를 개선하기 위해 코드의 위치를 팀원과 계속 수정해 나가보았다. 코드를 짤 때 제 기능을 하는 것도 중요하지만 성능도 중요하다는 것을 깨닫게 되었다.

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

3) 프로젝트 운영에 개선이 필요하다고 생각하는 점

프로젝트가 단계별로 갈수록 딜레이가 많아지는 현상이 발생하였다. 맵을 계속 매 틱마다 랜더링을 해야할지 스네이크가 지나간 부분만 랜더링을 해야할지를 팀원과 함께 계속 고민해왔던 것 같다. 시간상 스네이크가 지나간 부분만 되돌리는 것은 구현하지 못했지만 추후 여건이 된다면 수정해 나가고 싶다.

또한 스네이크 게임이 하나의 파일에 함수, 변수, 메인함수 등이 다 담겨 있었는데 나중에는 좀 더 세밀한 구조화를 통해 구현해 나가도 괜찮겠다는 생각이 들었다.

4) 팀원 평가

활발한 팀원과의 의사소통, 다양한 소통관련 응용프로그램(라이브 쉐어, 구글미트) 사용을 통해 오프라인뿐만 아니라 비대면 환경에서도 원활한 팀 프로젝트를 진행할 수 있었다. 특히 팀원의 뛰어난 역량과 활발한 상호간 소통 덕분에 프로젝트를 즐겁게 참여할 수 있었다.

4 참고 문헌

- <https://blackinkaj.github.io/ncurses/>
- <https://psman2.tistory.com/entry/ncurses-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D>
- <https://widian.tistory.com/58>
- <https://minwook-shin.github.io/basic-ncurses/>
- <https://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO>

5 부록

5.1 사용자 매뉴얼

1) 실행

실행 명령어 1	\$ make stg5
실행 명령어 2	\$./stg5

실행파일 ‘stg5’가 없는 경우, 실행명령어 1을 터미널에 위와 같은 명령어를 입력해준다. 이미 컴파일되어 실행파일이 있는 경우에는 실행 명령어2를 입력해준다.

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

2) 방법

- (1) 실행 파일을 실행하면 스네이크 게임 화면이 나타난다.
- (2) 이후 스네이크가 매 틱마다 자동으로 앞으로 나아간다.
- (3) 상, 하, 좌, 우 키를 이용해 스네이크의 방향을 전환해준다.
- (4) 오른쪽의 현재 항목별 점수와 목표 점수를 확인하여 게이트 통과 및 growth 아이템 획득을 진행한다.
- (5) 목표 점수를 달성하면 다음 스테이지에서 새로운 맵으로 게임이 시작된다.
- (6) 마지막 스테이지로 가서 최종 목표치를 달성하거나 게임 중간에 스네이크가 죽는 경우 화면이 자동으로 종료되고 게임 플레이 시간과 성공여부가 나타난다.

5.2 설치 방법

1) 깃허브 파일 다운로드

```
$ git clone https://github.com/SnakeProject-2022-01-S-Y/SnakeProject-2022-01.git
```

git이 설치되어 있는 환경에서 위 명령어를 터미널에 입력해준다.

2) 필수 라이브러리 설치

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

우분투 환경에서 위 명령어를 입력하여 게임 파일을 실행하기 위해 필요한 기본 환경 설정을 해준다.

3) 파일 컴파일 및 실행

Makefile에서 설정된 명령어

```
stg5: Stage5.cpp
g++ -std=c++11 -o stg5 Stage5.cpp -lncursesw
./stg5
```

 국민대학교 소프트웨학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	스네이크 프로젝트	
	팀 명	기승전스	
	Confidential Restricted	Version 1.3	2022-JUN-05

Stage5.cpp을 컴파일 하여 실행하면 게임을 진행할 수 있다. Make 파일에서 이와 같은 명령어를 설정해주어 터미널에서 ‘make stg5’를 Stage5.cpp가 있는 디렉토리로 들어가 입력해주면 게임이 실행된다.