


C++프로그래밍 프로젝트

프로젝트 명	Snake game
팀 명	ESC
문서 제목	결과보고서

Version	1.5
Date	2021-06-20

팀원	김한림(20203055) (팀장)
	김영광(20203042)
	이다은(20203112)

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20


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

문서 정보 / 수정 내역


Filename	ESC최종보고서- Snakegame.doc
원안작성자	이다은, 김한림, 김영광
수정작성자	이다은, 김한림, 김영광

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2020-06-18	이다은	1.0	최초 작성	보고서 초안
2020-06-19	김영광	1.1	2.2.1, 2.2.4 항목	2.2.1 개발 내용: map 구현, Gate 구현, snake 점수요소, 메인화면 2.2.4 현실적 제한 요소 및 해결 방안 작성
2020-06-20	김한림	1.2	2.2.1, 2.2.2의 1,2,3,6번 항목	2.2.1 개발내용 : 2단계, 5단계 2.2.2 개발상세 : 게임로직, 1단계,2단계,5단계
2020-06-20	이다은	1.3	2.2.2 2.2.3 2.2.5 항목	2.2.2 개발내용 추가 2.2.3 활용/개발된 기술 내용 추가 2.2.5 결과물 목록
2020-06-20	김영광	1.4	2.2.2항목	2.2.2 개발상세 : 4단계
2020-06-20	이다은	1.5	5.1 5.2 항목 및 보고서 마무리	5.1 사용자 매뉴얼 5.2 설치 방법 보고서 마무리

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

목 차

1	개요	4
2	개발 내용 및 결과물	6
2.1	목표	6
2.2	개발 내용 및 결과물	9
2.2.1	개발 내용	9
2.2.2	시스템 구조 및 설계도	12
2.2.3	활용/개발된 기술	31
2.2.4	현실적 제한 요소 및 그 해결 방안	32
2.2.5	결과물 목록	32
3	자기평가	33
4	참고 문헌	36
5	부록	37
5.1	사용자 매뉴얼	37
5.2	설치 방법	39

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

1 개요

평가기준 (10점)

프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.

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

이번 프로젝트에 사용한 외부 라이브러리는 총 한가지로, ncurses 라이브러리를 사용했습니다. 조원 모두 Ubuntu를 공통개발환경으로 하였고, 이를 기준으로 터미널에서 `wget https://ftp.gnu.org/pub/gnu/ncurses/ncurses-6.2tar.gz -P ~/Downloads`의 명령문을 사용하여 설치하였습니다. 다운로드 파일은 ncurses-6.2.tar.gz의 명령어로 압축을 해제하였고, ncurses를 빌드하기 위해 `make`, `sudo make install`의 명령어를 사용하였습니다. 그 후 코드 내에서 ncurses 라이브러리를 사용하기 위해 c++ 소스 파일 내부에 `#include <ncurses.h>`를 추가하였습니다.

게임의 각 파트는 기능을 구현하기 위해 여러 클래스로 세분화되어 있습니다. 저희 팀은 각각의 세분화된 파트별 기능을 구현하기 위해 Git을 활용하였습니다. 각자 맡은 기능을 구현한 뒤 코드 리뷰 과정을 거쳐 merge하는 방식으로 개발하였습니다.

다음은 프로젝트의 단계별로 나누어 서술한 전체적인 구조 및 개발 내용입니다.

1단계. Map


Map에는 기본적으로 빈 공간, wall, immune wall, snake의 head와 body, Growth Item, Poison Item, Gate 등 한 번에 다양한 항목이 나타나야 합니다. 그래서 Map을 char형태의 이차원 배열로 선언 후, 각 항목을 나타내는 한 자리 문자를 정하였고, 생성되어야 할 좌표에 해당 문자를 대입하기로 하였습니다. 공백은 '0', wall은 '1', immune wall은 '2', snake의 head는 '4', body는 '3', gate는 '5', growth item은 '6', poison item은 '7'로, 각각의 문자가 다양한 항목을 나타냅니다. 이후 switch 문을 통해 map을 시각화 하였습니다.

2단계. Snake

우선 snake를 map상에 나타나게 하기 위해, snake의 초기 생성 위치를 정해주고, map상의 snake head위치, body위치를 각각 '4', '3'으로 업데이트하였습니다.

snake는 방향 키를 입력 받아 움직입니다. 이때 snake는 진행방향의 반대방향으로 이동할 수 없기 때문에 진행방향과 반대 방향의 입력이 주어지면 게임이 끝나도록 만들었습니다. 또한 진행 방향과 같은 방향의 입력은 무시하도록 바로 전 입력을 변수에 저장한 채로 switch문을 통해 현재 입력과 비교하였습니다.

입력 받은 방향이 정상적이라면, 방향에 맞게 snake head의 좌표를 움직이고, 동시에 stage도 업데이트 하였습니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

3단계. Item

Item은 총 2가지로 snake의 몸길이가 1 증가하는 growth item과, snake의 몸길이가 1 감소하는 poison item이 있습니다. 아이템은 빈 공간의 임의의 위치에 나타내야 하기 때문에, 아이템을 생성하려는 위치의 문자가 공백을 나타내는 '0'이 아니라면 생성될 수 없도록 하였습니다. 또, 아이템은 출현 후 일정시간(5초)이 지나면 사라지고 다른 위치에 생성되어야 하기 때문에 생성 당시의 시간을 저장해두고, 현재 시간과 계속 비교하여 아이템 시간을 체크하였습니다. 마지막으로, 아이템이 random한 위치에 나타날 수 있도록 random 함수를 import하여 srand함수를 사용하였고, 동시에 나타날 수 있는 아이템의 수를 3개로 제한하기 위해, 한 번에 세 개 이상의 아이템이 나타나지 않도록 조건을 걸어주었습니다.


4단계. Gate

Gate는 immune wall 아닌 wall의 임의의 위치에서 생성되어야 합니다. 따라서 gate의 위치를 초기 생성할 때, rand함수로 랜덤 하게 생성한 좌표의 문자가 '1'(wall의 위치를 나타내는 문자)인 곳에 gate가 생성되도록 하였습니다. gate가 생성될 위치를 map에서 '5'로 바뀌 주었습니다.

5단계. Score/mission board

Score board에는 게임 중 몸의 최대 길이 계산(B), 획득한 Growth item(+), 획득한 poison item(-), gate 사용 횟수(G)를 나타냈습니다. Mission board에는 주어진 미션의 달성 여부를 나타냈습니다.

scored board와 mission board는 각각 window를 선언하여 창을 구성하였습니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

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
6단계	메인 화면 구현	추가 적용

1 단계. Map의 구현


1 단계의 주 목표는 NCurses Library 함수들을 사용하여 2 차원 배열로 된 Map 을 Game 화면으로 표시 하는 프로그램을 완성하는 것입니다.

- stage 를 4 단계로 구분하고, stage 별로 모양이 다른 맵을 구현한다.
- Map 의 세로길이는 30, 가로길이는 110 으로 설정한다.
- Wall 과 Immune Wall 을 잘 구분해야 한다(wall 은 map 에서 '1'로 표현, immune wall 은 map 에서 '2'로 표현)

2 단계. Snake 표현 및 조작

2 단계의 주 목표는 1 단계에서 구현한 맵 위에 Snake 를 표시하고, 화살표를 입력 받아 방향에 맞게 Snake 가 움직이도록 프로그램을 완성하는 것입니다.

- Snake 는 규칙 #1 을 준수하여야 한다.
 - 방향키는 사용자가 직접 정한다.
 - Snake 는 진행방향의 반대방향으로 이동할 수 없다. 반대 방향 키를 입력할 경우 게임이 종료된다. (head 가 진행방향이며, tail 방향으로 이동할 수 없다)
 - 진행방향과 같은 방향키 입력은 무시한다. (head 방향 이동은 무시한다)
 - Snake 는 일정 시간(틱)에 의해 이동한다.
- Snake 의 x 좌표, y 좌표를 묶어서 vector 에 저장한다.
- Map 에서 Snake 의 head 와 body 를 구분해야 한다(head 는 map 에서 '4', body 는 '3'으로 표현)
- 이 외의 실패 조건
 - Snake 는 자신의 body 를 통과할 수 없다.
 - Snake 는 벽(wall)을 통과할 수 없다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

3 단계. Item 요소의 구현


3 단계의 주 목표는 Map 위에 Growth Item 와 Poison Item 을 출현하도록 프로그램을 완성하는 것입니다.

- 게임 규칙#2 를 준수해야 한다.
 - Snake 의 이동방향에 item 이 놓여 있는 경우 snake 가 item 을 획득한다.
 - Growth item 의 경우 몸의 길이가 진행방향으로 1 증가한다.
 - Poison item 의 경우 몸의 길이가 1 감소한다.
 - 몸의 길이가 3 보다 작아지면 게임이 종료된다.
- Growth item 과 Poison item 의 출현
 - Item 은 snake body 가 있지 않은 임의의 위치에 출현해야 한다.
 - 출현 후 일정시간(5 초)가 지나면 사라지고, 랜덤한 다른 위치에 나타나야한다.
 - 동시에 출현할 수 있는 item 의 개수는 3 개로 제한한다.
- Growth Item 과 Poison Item 을 Map 배열에 표현할 때 값을 정한다.
 - map 에서 Growth Item 은 '6', Poison Item 은 '7'로 표현한다.
- 화면상에 표현시, 색이나 기호를 달리하여 구분할 수 있도록 한다.

4 단계. Gate 요소의 구현

4 단계의 주 목표는 Map 의 Wall 위의 임의의 위치에 한 쌍의 Gate 가 출현할 수 있도록 변경하고, 각 Gate 를 Snake 가 통과할 수 있도록 프로그램을 완성하는 것입니다.

- 게임 규칙 #3, #4, #5 를 준수해야 한다.
 - Gate 는 두 개가 한 쌍이다.
 - Gate 는 겹치지 않는다.
 - Gate 는 벽(wall)위의 임의의 위치에서 나타난다.
 - Gate 에 snake 가 진입하면, 다른 Gate 로 진출한다.
 - Gate 는 한번에 한 쌍만 나타난다.
 - Gate 에 snake 가 진입중인 경우, Gate 는 사라지지 않아야 하며, 동시에 다른 위치에서 Gate 가 나타나면 안 된다.
- Gate 가 나타나는 벽이 가장자리에 있을 때
 - 항상 Map 의 안쪽 방향으로 진출한다.(고정)
 - 상단 벽 => 아래 방향
 - 하단 벽 => 위 방향
 - 좌측 벽 => 오른쪽 방향
 - 우측 벽 => 왼쪽 방향
- Gate 가 나타나는 벽이 Map 의 가운데 있을 때 다음의 순서로 진출
 - 진입 방향과 일치하는 방향이 우선
 - 진입 방향의 시계방향으로 회전하는 방향

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

- 진입 방향의 역시계 방향으로 회전하는 방향
- 진입 방향과 반대방향
- 화면상에 표현 시, Gate 는 Wall 과 구분될 수 있도록 한다.
- Wall(Immune Wall 포함)과 Gate 를 Map 배열에 표현할 때 값을 결정한다.
 - map 에서 wall '1', immune wall 은 '2', Gate 는 '5'로 표현한다.


5 단계. 점수 요소의 구현

5 단계의 주 목표는 우측에 게임 점수를 표시하는 화면을 구성하는 것입니다.

- score board 와 mission board 를 각자 다른 윈도우로 구현하여 우측에 띄운다.
- 게임 점수는 게임 규칙 #6 을 준수한다.
 - 게임 중 몸의 최대 길이 계산 (B : 현재 길이/최대 길이)
 - 게임 중 획득한 Growth Item 의 수 (+)
 - 게임 중 획득한 Poison Item 의 수 (-)
 - 게임 중 Gate 사용 횟수 (G)
- mission 은 구성된 Map 의 정의에 고정 값을 주는 방식으로 수행한다.
- stage 마다 Mission 을 달성하면 다음 Map 으로 진행하도록 프로그램을 완성한다.
- Stage 는 최소 4 개로 구성하고, 각 Stage 의 Map 은 서로 달라야 한다.

6 단계. 메인 화면 구현

ESC 팀에서 추가적으로 구현한 기능으로, 게임을 시작했을 때, 메인 화면이 나오도록 하여 게임이 바로 시작되지 않도록 하였습니다. 메인 화면에는 Snake game 의 썸네일을 띄우도록 하여 사용자가 랜덤한 키를 입력할 경우 게임이 시작되도록 구성하였습니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

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.

1단계 개발 내용: map의 구현

- 저희 게임의 Map은 기본적으로 세로 110, 가로 30으로 최대 길이를 설정해 놓고 define으로 정의해 놓았습니다.
- Map 클래스는 gate, item을 통합하여 관리합니다.
- Map 클래스의 생성자에서 각 stage가 시작될 때마다 맵 배치도인 txt파일을 불러옵니다.
- Map 클래스는 총 7가지의 클래스 함수를 가지고 있습니다.
- **Init()**: 가져온 Map의 txt파일을 이용해 maps라는 char형 2차원 배열에 초기화를 하는 작업을 해줍니다.
- **Render()**: 초기화된 maps 배열을 가지고 실질적으로 콘솔 화면에 맵을 보여줄 수 있도록 배경색을 입힌 문자들을 배치하는 역할을 합니다. '0': 공백, '1': 벽, '2': immune 벽, '3': 스네이크 몸, '4': 스네이크 머리, '5': 게이트, '6': Growth item, '7': Loss item
- **Update()**: 다른 클래스에서 접근할 수 있는 Map클래스의 멤버함수로 원하는 시점에서 maps배열의 값을 전달한 인자 값으로 바꿀 수 있습니다.
- **make_item()**: item의 위치를 랜덤 값으로 뽑아내 maps배열에 전달하는 Map클래스의 멤버 함수입니다.
- **check_item()**: 실시간으로 item의 위치를 확인해 make_item함수를 호출하여 item의 위치를 초기화해주는 함수입니다. 5초이상 지나갔을 경우 item의 위치를 랜덤하게 바꿔줍니다.
- **set_Gatepos()**: gate의 위치를 랜덤 값으로 뽑아내 maps배열에 전달하는 Map클래스의 멤버 함수입니다.
- **GateUpdate()**: 실시간으로 gate의 위치를 확인해 set_Gatepos함수를 호출하여 gate의 위치를 초기화해주는 함수입니다. 4초이상 지나갔을 경우 gate의 위치를 랜덤하게 바꿔줍니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

- stage의 클리어 여부에 따라 총 4단계 stage의 맵이 구성되어 있습니다.

2단계 개발 내용: Snake 표현 및 조작

- 스네이크의 구조는 벡터의 형태로 되어있으며, vector의 0번째 원소는 헤드를 가리키고 나머지는 바디를 가리킵니다. 또한 vector의 원소의 타입은 pair이며 이 때 pair는 각각 x, y로 각 부분 하나하나의 좌표 위치를 가리킵니다.
- 이러한 벡터 정보를 통해 맵 상에 초반에 나타날 위치에 Map의 Update함수를 통해 맵에 표시합니다.
- 이 때 2차원 배열의 맵 상에서 스네이크의 헤드는 '4' 바디는 '3'으로 구현하였습니다.
- 스네이크의 이동은 move함수에서 GameScene.cpp에서 getch를 통해 받아온 키 값을 통해 조작합니다.
- 스네이크가 한 번 움직일 때마다 for문을 통해 vector의 각 바디의 위치값을 바로 직전의 바디의 위치값으로 업데이트 해줍니다.
- 스네이크의 멤버함수인 direction을 구현하여 받아온 키 값에 따라 direction을 상하좌우로 바꿀 수 있게합니다. 이 때 진행방향과 정반대 방향으로 키 값이 입력되면 dead변수를 true로 바꾸고 함수를 종료합니다.
- 그 다음 업데이트된 direction값을 바탕으로 헤드의 좌표 값을 변경시키게 하고, usleep 함수를 통해 일정 틱 마다 움직이게끔 제한을 두었습니다.
- 또한 업데이트된 헤드의 좌표 값을 통해 그 좌표의 값이 Map배열에서 어떤 부분과 만나냐에 따라 상호작용을 하게끔 Collision함수에서 구현을 하였습니다.
- Collision함수에서는 헤드 좌표의에 위치한 Map의 요소가 어떤 것이냐에 따라 각각 작용을 하는데, 6, 7번은 아이템, 5번은 게이트, 0번은 빈공간, 1,2번은 벽으로 처리합니다.
- 이 중 1,2,3번인 경우는 dead를 true로 처리합니다.
- 6,7번 아이템인 경우에는 각각 Growth함수, Reduce함수를 호출하여 스네이크의 길이를 조정합니다.
- 5번의 경우 게이트구현에 관한 부분이며 게이트가 맵 가장자리에 나타날 경우, 맵의 중간에 나타난 경우 둘로 나눠 처리합니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

3단계 개발 내용: Item 요소의 구현


- 두 가지 Item은 map상에서 서로 다른 변수로 지정하였습니다. 아이템의 모양은 종류에 따라 Growth item이라면 '6'으로, Poison item이라면 '7'로 Map에 저장합니다.
- 동시에 나타날 수 있는 item의 개수를 3개로 제한합니다. 현재 아이템의 개수를 저장해 놓고, 아이템을 생성할 때마다 1씩 증가하여 3이상이 될 경우 아이템 생성을 중단합니다.
- random함수를 import하여 Item의 위치를 random하게 생성합니다. Item이 생성될 당시의 시간을 저장하고 이를 현재 시간과 비교하여 5초이상 지났을 경우, 기존의 item은 사라지고 아이템 생성 함수를 호출하여 Item을 재 생성하도록 함수를 구현하였습니다. 이때 생성 좌표의 map값이 '0'일 경우 (빈 공간일 경우)에만 해당 위치에 아이템이 생성될 수 있도록 하였습니다.
- Snake의 이동 중에 Snake Head와 Item의 좌표가 같은 지 함수로 비교하고, Item의 종류를 체크해서 Snake의 Size를 1씩 줄이거나 늘립니다.
- 부가적으로 화면상에 아이템의 종류를 표시하기 위해 COLOR_PAIR(), init_pair() 등을 적절히 활용하여 아이템과 벽, Snake의 색상을 달리 하였습니다.

4단계 개발 내용: Gate 요소의 구현

- gate관련 변수와 함수는 기본적으로 Map 클래스에서 관리됩니다. Gate의 초기 생성은 Map 클래스의 생성자와 set_Gatepos함수를 통해 이루어집니다.
- gate의 랜더링은 Map 클래스의 Render함수에서 switch문을 통해 '5'값으로 지정되어 있습니다.
- gate 위치는 random하게 생성됩니다. Gate가 생성되고 4초이상 지났을 경우 현재 위치와 다른 random한 위치에서 재생성 됩니다.
- 실질적인 gate의 기능은 snake 클래스 Collision함수에서 구현됩니다. gate1과 gate2의 위치를 파악하고 gate와 닿았을 때 snake의 방향과 다른 gate의 상하좌우의 빈 공간을 통해 경우의 수를 가정하여 snake가 gate로 워프합니다.

5단계 개발 내용: 점수 요소의 구현

- GameScene.cpp파일에서 점수요소를 구현하였습니다. 먼저 UI가 게임화면, 점수화면, 미션화면으로 세 개의 서브윈도우로 나누어져있습니다.
- InitScoreboard, InitMissionboard 함수를 통해 첫 실행 때의 UI초기화를 해줍니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

- 그 후 UpdateScoreboard, UpdateMissionboard를 통해 게임실행의 While문 안에 넣어 매 업데이트마다 정보가 업데이트되게끔 합니다.
- mvwprintw함수를 통해 메시지를 표시했으며 각 점수마다 변수를 만들어 Snake의 변수를 가져와 변수의 내용이 바뀔때마다 업데이트 되게끔 했습니다.
- const char 배열로 형변환을 위해 to_char함수를 구현하였습니다.
- 변수의 값은 Snake클래스의 게터함수를 통해 가져왔습니다
- 미션의 점수가 달성되면 체크표시가 되게끔 isAchieve함수를 구현하였는데, 현재 스코어와 목표스코어를 인자로 받아 체크표시 스트링을 리턴하게끔 구현하였습니다.

6단계 개발 내용: 메인 화면 구현

- StartScene.cpp를 만들어 첫 실행 때 나타나게끔 하였습니다. Initscr로 stdscr을 만든 후 getch()로 아무 입력이나 받으면 endwin을 하고 함수를 종료하고 게임화면으로 넘어가게 합니다.
- attron으로 블링크 효과를 줍니다.

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.


1. 게임 로직

먼저 게임의 시작점이 되는 main.cpp의 구조입니다.

```
#include "SnakeGame.h"

int main() {
    SnakeGame *game = new SnakeGame(4);
    game->startGame();
}
```

SnakeGame이라는 게임 하나를 통칭하는 객체를 선언하고 startGame()함수를 통해 게임을 시작합니다. 당연히 startGame 함수가 종료되면 프로그램은 종료됩니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```
class SnakeGame {
    bool isOver;
    StartScene *intro;
    GameScene *inGame;
    GameoverScene *overScene;
    GameclearScene *clearScene;
    EndScene *endScene;
    int curLevel = 1;
    int objLevel;
public:
    SnakeGame(int objLevel);
    void startGame();
};
```


SnakeGame.h 이다 isOver 은 스네이크게임의 종료여부를 나타내는 bool 변수입니다. 그리고 각각 시작화면 intro, 게임화면 inGame, 게임오버화면 overScene, 게임클리어화면 GameclearScene, 그리고 게임 종료화면 EndScene 이 있습니다. 현재 레벨을 나타내는 curLevel 과 목표 레벨을 나타내는 objLevel 이 있습니다.

```
SnakeGame::SnakeGame(int objLevel) {
    intro = new StartScene();
    clearScene = new GameclearScene();
    overScene = new GameoverScene();
    endScene = new EndScene();
    this->objLevel = objLevel;
}
```

SnakeGame 클래스의 생성자입니다. 각 Scene 마다 객체를 생성하고 objLevel 을 인자값으로 초기화합니다.

```
void SnakeGame::startGame() {
    intro->Draw();
    while(true) {
        inGame = new GameScene(curLevel);
        inGame->Run();
        if(inGame->isClear) {
            clearScene->Draw();
            curLevel += 1;
            if(curLevel > objLevel) break;
        }
        else {
            overScene->Draw();
        }
    }
    endScene->Draw();
}
```

게임을 시작하는 함수인 startGame 입니다. 먼저 intro 화면을 그린다음 while 문에 진입하여 목표레벨만큼 스테이지를 진행합니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

매 게임마다 GameScene 을 생성하여. GameScene 내 Run 에서 다시 while 루프에 들어가 게임을 진행합니다. 이 때 Run 이 종료되고 isClear 가 true 면 현재 GameScene 을 종료시키고 레벨을 증가시켜 다음 루프로 진행합니다. 이 때 isClear 가 false 면 gameOverScene 을 그려 레벨을 증가시키지 않고 다시 진행합니다.

마지막으로 현재레벨이 목표레벨을 달성하면 while 문에서 벗어나 EndScene 을 호출한 다음 함수를 종료시켜 최종적으로 프로그램을 종료합니다.


```
class GameScene {
    WINDOW *gameboard;
    WINDOW *scoreboard;
    WINDOW *missionboard;
    Map *stage;
    Snake *player;
public:
    bool isClear;
```

GameScene 클래스의 멤버변수입니다. 다음 gameboard, scoreboard, missionboard 는 각각 서브윈도우로 게임화면, 스코어, 미션화면입니다. stage 변수는 Map 타입의 스테이지 정보를 담는 변수입니다. player 는 Snake 타입의 플레이어(스네이크)의 정보를 담는 변수입니다. isClear 변수는 한 스테이지의 클리어 여부를 나타내는 bool 형 변수입니다. GameScene 클래스 자체는 ‘한’ 게임의 진행과 정보를 담고 있는 클래스입니다. 객체의 생성과 소멸이 스테이지의 시작과 끝을 나타냅니다.

```
GameScene::GameScene(int level) {
    gameboard = newwin(LINES, COLS/2+40, 0, 0);
    scoreboard = newwin(LINES/2, COLS/2-40, 0, COLS/2+40);
    missionboard = newwin(LINES/2, COLS/2-40, LINES/2, COLS/2+40);
    stage = new Map(level);
    stage->Init();
    player = new Snake(*stage, 5);
    player->init_snake_pos(1, 7);
    isClear = false;
}
```

생성자에서는 게임에 필요한 초반 작업을 합니다. 먼저 세 윈도우(게임,스코어,미션) 객체를 newwin 함수를 통해 생성합니다. 그 다음 새로운 맵 객체를 생성해 스테이지 변수에 할당 후 Init 함수로 초기화작업을 합니다. 다음도 똑같이 player 객체의 생성과 초기화 작업을 합니다.

```
void GameScene::InitWindow() {
    initscr();
    start_color();
    curs_set(false);
    noecho();
    cbreak();
    init_pair(1, COLOR_BLACK, COLOR_WHITE);
```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

InitGameboard();
InitScoreboard();
InitMissionboard();
UpdateGameBoard();
UpdateScoreBoard();
UpdateMissionBoard();
stage->make_item(); //시작할 때 아이템 배치
stage->set_Gatepos();
nodelay(gameboard, TRUE);
}

```

다음으로 InitWindow 함수입니다. 처음 initscr 을 통해 메인윈도우를 생성한 후 여러 윈도우 초기화 함수(InitOOBoard())를 이용해 초반 UI 세팅을 합니다. 또한 맵의 아이템과 게이트의 초기 위치를 설정합니다.

마지막으로 nodelay 함수를 통해 키입력을 받으려 할 때 윈도우가 멈추지 않게 설정을 합니다.


```

void GameScene::InitGameboard() {
    int row, col;
    getmaxyx(gameboard, row, col);
    box(gameboard, 0, 0);
    attron(COLOR_PAIR(1));
    wbkgd(gameboard, COLOR_PAIR(1));
    mvwprintw(gameboard, 0, col/2-col/10, "GAME");
    keypad(gameboard, true);
}

void GameScene::InitScoreboard() {
    int row, col;
    getmaxyx(scoreboard, row, col);
    box(scoreboard, 0, 0);
    attron(COLOR_PAIR(1));
    wbkgd(scoreboard, COLOR_PAIR(1));
    mvwprintw(scoreboard, 0, col/2-col/5, "SCORE BOARD");
    mvwprintw(scoreboard, 2, 2, "B : 6");
    mvwprintw(scoreboard, 4, 2, "+ : 0");
    mvwprintw(scoreboard, 6, 2, "- : 0");
    mvwprintw(scoreboard, 8, 2, "G : 0");
}

void GameScene::InitMissionboard() {
    int row, col;
    getmaxyx(missionboard, row, col);
    box(missionboard, 0, 0);
    attron(COLOR_PAIR(1));
    wbkgd(missionboard, COLOR_PAIR(1));
    mvwprintw(missionboard, 0, col/2-col/5, "MISSION BOARD");
}

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

다음은 각 윈도우를 초기화하는 함수입니다. ncurses 내 함수를 적절히 사용하여 UI 를 꾸몄습니다

```
void GameScene::Run() {
    InitWindow();
    while(true) {
        int ch = wgetch(gameboard);
        if(ch == KEY_F(1)) break;
        player->move(ch);
        stage->check_item(); //5 초 이상 지났는지 체크해서 아이템 재배치
        if(!player->inGate) {
            stage->GateUpdate();
        }
        UpdateGameBoard();
        UpdateScoreBoard();
        UpdateMissionBoard();


        if(checkClear()) {
            isClear = true;
            break;
        }
        else if(player->isDead()) break;
    }
    endwin();
}
```

다음으로 중요한 부분인 Run 함수 입니다. 이 함수에서 게임의 진행에 관한 부분을 담당합니다. 먼저 InitWindow 함수를 통해 UI 를 설정한 뒤 while(true)문에 진입하여 게임이 끝나기 전까지 게임 로직이 돌아가게끔 설정하였습니다. wgetch 를 통해 gameboard 에 키입력을 받고 Snake 클래스 객체의 move 멤버 함수의 인자값으로 주어 움직임을 표현합니다.(자세한 내용은 2 번항목) 또한 매 업데이트(while 문 한번)마다 check_item 을 통해 아이템 정보를 업데이트합니다.(자세한 내용은 3 번항목)

게이트의 업데이트 또한 위와 같이 이루어집니다.(이 때 플레이어가 게이트에 진입할 때는 업데이트가 이루어지지 않도록 if 문과 inGate bool 변수를 사용해 업데이트 제한함)

마지막으로 매 업데이트마다 맵의 정보를 업데이트하고 렌더할 수 있도록 UpdateOOOboard 함수를 호출합니다. 그리고 이 때 checkClear 의 값이 참이 되거나 플레이어의 dead 변수가 true 가 되면 break 로 while 루프를 빠져나와 윈도우를 닫고 함수를 종료시킵니다.

```
bool GameScene::checkClear() {
    if(player->getCurLen() >= stage->getObjLen() && player->getGrowthNum() >=
stage->getObjGrowth() && player->getReduceNum() >= stage->getObjReduce() &&
player->getGateNum() >= stage->getObjGate()) {
        return true;
    }
    else {
```


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

    return false;
}
}

```

게임 클리어 여부를 확인하는 checkClear 함수입니다. 스테이지 객체의 목표 점수와 Snake 객체의 현재 점수를 비교하여 클리어 여부를 리턴합니다.

2. 맵 구현

```

#define Width 110
#define Height 30

class Map{
    int objLen;
    int objGrowth;
    int objReduce;
    int objGate;
    string mapUrl;

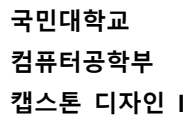
public:
    ifstream map_txt;
    char maps[HEIGHT][WIDTH];
    time_t nowT;
    time_t startT;

    int gate_X[2];
    int gate_Y[2];
    time_t gate_nowT;
    time_t gate_startT;

```

맵 클래스의 멤버 변수는 다음과 같습니다. objLen, Growth, Reduce, Gate 는 맵의 목표 점수를 담는 변수이고 mapUrl 은 Stage 폴더에서 가져올 맵 파일의 경로를 저장할 스트링 변수입니다. map_txt 는 fstream 으로 받아올 파일의 정보를 담을 변수입니다. maps 변수는 텍스트파일의 맵구조를 2 차원형태로 받는 char 형 2 차원 배열입니다. Width 와 Height 는 맵의 크기를 담는 상수를 선언했습니다. 우분투 터미널창 폴스케일을 기준으로 하여 110x30 의 크기로 정하였습니다. 이외에 맵에 나타날 게이트의 위치 정보를 저장할 크기 2 의 gate_X, gate_Y 배열을 선언합니다. 이외에 time_t 타입으로 선언된 변수는 매 시간마다 게이트와 아이템 생성을 다르게 하기 위해 선언한 변수입니다.

먼저 맵의 구조는 텍스트파일형태로 저장 되어있습니다. 그리고 각각의 스테이지는 Stage 폴더 안에 저장되어있습니다. 아래 사진은 맵 텍스트파일 예시입니다.



결과보고서		
프로젝트 명	Snake game	
팀 명	ESC	
Confidential Restricted	Version 1.5	2021-06-20

23
20
10
5

[illegible]


이 때 위의 4 개 줄에 적힌 숫자는 순서대로 해당 맵의 목표 길이, Growth Item, Reduce Item, Gate 사용횟수 입니다.

이렇게 저장되어 있는 텍스트 파일을 Map.cpp 파일(이하 파일 생략)의 생성자에서 받아옵니다.

```
Map::Map(int level) {
    this->mapUrl = "stage/stage" + to_string(level) + ".txt";
    map_txt.open(this->mapUrl);
    for(int i = 0; i < 2; i++){
        gate_X[i] = 0;
        gate_Y[i] = 0;
    }
}
```

fstream 의 open 함수를 통해 텍스트 파일을 읽어옵니다. 여러 맵을 받아오기 위해
스테이지 번호별로 맵파일의 경로를 저장할 변수 mapUrl 을 선언하여 할당했습니다.

또한 Map.h 에 정의된 게이트의 좌표 위치를 초기화해줍니다. (자세한 내용은 Gate 파트에서 설명)

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

void Map::Init(){
    int map_height = 0;
    while (!map_txt.eof())
    {
        char temp[200];
        map_txt.getline(temp, 200);
        if(map_height == 0) {
            string tmp;
            for(int i = 0; i < 3; i++) {
                tmp += temp[i];
            }
            objLen = stoi(tmp);
        }
        else if(map_height == 1) {
            string tmp;
            for(int i = 0; i < 3; i++) {
                tmp += temp[i];
            }
            objGrowth = stoi(tmp);
        }
        else if(map_height == 2) {
            string tmp;
            for(int i = 0; i < 3; i++) {
                tmp += temp[i];
            }
            objReduce = stoi(tmp);
        }
        else if(map_height == 3) {
            string tmp;
            for(int i = 0; i < 3; i++) {
                tmp += temp[i];
            }
            objGate = stoi(tmp);
        }
        else {
            for (int i = 0; i < WIDTH; i++)
            {
                maps[map_height-4][i] = temp[i];
            }
        }
        map_height++;
    }
}


```

Init 함수는 불러온 map_txt 파일을 적절한 형태로 Map 의 멤버변수에 정보를 할당하는 역할을 합니다. 0,1,2,3 번째 줄은 맵의 목표 점수들을 읽어들입니다. stoi 함수를 이용하여 string 형태의 글자를 변수에 읽어옵니다. 이외에 맵 전체 부분은 for 을 이용해 maps 2 차원 배열 변수에 저장합니다.

```

void Map::Render(WINDOW *board){

```

	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20


```

init_pair(5, COLOR_WHITE, COLOR_WHITE);
init_pair(2, COLOR_GREEN, COLOR_GREEN);
init_pair(3, COLOR_RED, COLOR_RED);
init_pair(4, COLOR_BLUE, COLOR_BLUE);
init_pair(8, COLOR_YELLOW, COLOR_YELLOW);
init_pair(9, COLOR_MAGENTA, COLOR_MAGENTA);
init_pair(7, COLOR_CYAN, COLOR_CYAN);
init_pair(10, COLOR_BLACK, COLOR_BLACK);
for(int i = 0; i < HEIGHT; i++){
    for(int j = 0; j < WIDTH; j++){
        //attron(COLOR_PAIR(maps[i][j]));
        switch (maps[i][j])
        {
            case '0':
                mvwaddch(board, i+1, j+1, ' ' | COLOR_PAIR(5)); // 공백
                break;
            case '1':
                mvwaddch(board, i+1, j+1, '#' | COLOR_PAIR(2)); // 벽
                break;
            case '2':
                mvwaddch(board, i+1, j+1, '@' | COLOR_PAIR(10)); // 벽
                break;
            case '4':
                mvwaddch(board, i+1, j+1, 'O' | COLOR_PAIR(3)); // 스네이크 머리
                break;
            case '3':
                mvwaddch(board, i+1, j+1, 'X' | COLOR_PAIR(4)); // 스네이크 몸
                break;
            case '5':
                mvwaddch(board, i+1, j+1, '^' | COLOR_PAIR(7)); // 게이트
                break;
            case '6':
                mvwaddch(board, i+1, j+1, 'G' | COLOR_PAIR(8)); // Growth item
                break;
            case '7':
                mvwaddch(board, i+1, j+1, 'L' | COLOR_PAIR(9)); // Loss item
                break;
        }
        //attroff(COLOR_PAIR(maps[i][j]));
    }
}
wrefresh(board);
}

```

Render 함수는 maps 변수를 바탕으로 UI 에 그려주는 역할을 합니다. 각 번호별로 여러 오브젝트를 표시하며 COLOR_PAIR 를 통해 배경색을 채워 UI 상에서 색깔있는 네모픽셀로 표현하게 했습니다. 마지막으로 wrefresh 를 통해 윈도우를 업데이트 해줍니다. 이 때 이 함수의 인자값은 맵을 표시할 WINDOW 객체를 받습니다.

```
void Map::Update(int x, int y, char value) {
```

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```
maps[x][y] = value;
}
```

맵의 정보를 업데이트 해줄 함수입니다. 인자값으로 업데이트할 xy 좌표값과 업데이트할 내용을 받습니다.


이제 Map.cpp 파일에서 맵에 대한 기본 설정을 끝냈다면 GameScene.cpp 에서 맵의 형태를 Map 의 Render 함수를 통해 렌더합니다.

3. Snake 표현 및 조작

```
class Snake {
    vector<pair<int, int> > snake;
    int size;
    int maxSize;
    int growthNum;
    int reduceNum;
    int gateNum;
    int cnt;
    char direction;
    int headPosX, headPosY;
    int tailPosX, tailPosY;
    bool dead;
    Map *stage;
public:
    bool inGate;
```

Snake.h 의 멤버 변수입니다. 순서대로 스네이크 각 부위의 위치 값을 담는 vector 를 snake 를 선언합니다. 그 다음 스네이크의 길이 size, 최대길이 maxSize, 현재 GrowthNum, reduceNum, gateNum 점수가 각각 있고, (cnt 는 게이트의 생성 제한을 위해 선언한 임시 변수입니다.) 스네이크의 이동방향 direction, 스네이크 헤드의 위치값, 꼬리 위치값 headPosX,Y tailPosX,Y 가 있고, 죽음 유무 dead, 그리고 플레이어가 생성될 스테이지인 stage, 플레이어가 gate 를 지나고있는지 확인하는 inGate 변수가 있습니다.

```
Snake::Snake(Map &map, int bodyNum) {
    stage = &map;
    size = bodyNum + 1;
    maxSize = size;
    direction = 'r';
    dead = false;
    growthNum = 0;
    gateNum = 0;
    inGate = false;
    reduceNum = 0;
    cnt = 0;
}
```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

Snake 클래스의 생성자입니다. 변수의 초기화를 합니다.

```
void Snake::init_snake_pos(int x, int y) {
    for(int i = 1; i < snake.size(); i++) {
        if(stage->maps[x][y-i] == 1) {
            return;
        }
    }
    for(int i = 0; i < size; i++) {
        snake.push_back(make_pair(x, y-i));
    }
    headPosX = snake[0].first;
    headPosY = snake[0].second;
    tailPosX = snake[size-1].first;
    tailPosY = snake[size-1].second;


    stage->Update(snake[0].first, snake[0].second, '4');
    for(int i = 1; i < size; i++) {
        stage->Update(snake[i].first, snake[i].second, '3');
    }
}
```

스테이지 상에서 스네이크의 초기 위치를 설정하고 렌더하는 init_snake_pos 함수입니다. for 문을 통해 스네이크의 생성 위치를 검사하는데 이 때 스네이크의 길이가 맵의 범위를 넘어가면 함수가 종료되게끔 하였습니다.

그 다음 문제가 없다면 스네이크의 길이만큼 push_back 을 통해 바디원소들을 추가해줍니다. 이 때 각 바디의 위치 값도 설정해줍니다.(pair 의 형태로 x,y 가 저장되어있음) 그 다음 head, tail 의 위치값을 할당해줍니다.

마지막으로 맵의 Update 함수를 통해 Map 의 maps 에 스네이크를 그려줍니다.

```
void Snake::move(int ch) {
    for(int i = size-1; i > 0; i--) {
        snake[i].first = snake[i-1].first;
        snake[i].second = snake[i-1].second;
    }
    switch (ch) {
        case KEY_LEFT:
            if(direction != 'r') {
                direction = 'l';
            }
            else {
                dead = true;
                return;
            }
            break;
    }
```


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

case KEY_RIGHT:
    if(direction != 'l') {
        direction = 'r';
    }
    else {
        dead = true;
        return;
    }
    break;
case KEY_UP:
    if(direction != 'd') {
        direction = 'u';
    }
    else {
        dead = true;
        return;
    }
    break;
case KEY_DOWN:
    if(direction != 'u') {
        direction = 'd';
    }
    else {
        dead = true;
        return;
    }
    break;
}
if(direction == 'l') {
    headPosY -= 1;
    usleep(100000);
}
else if(direction == 'r') {
    headPosY += 1;
    usleep(100000);
}
else if(direction == 'u') {
    headPosX -= 1;
    usleep(200000);
}
else if(direction == 'd') {
    headPosX += 1;
    usleep(200000);
}
Collision(stage->maps[headPosX][headPosY]);
if(!dead) {
    snake[0].first = headPosX;
    snake[0].second = headPosY;

    stage->Update(tailPosX, tailPosY, '0');
    tailPosX = snake[size-1].first;
    tailPosY = snake[size-1].second;

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

stage->Update(headPosX, headPosY, '4');

for(int i = 1; i < size; i++) {
    stage->Update(snake[i].first, snake[i].second, '3');
}
}
if(inGate) {
    if(cnt > size) {
        inGate = false;
    }
    cnt++;
}
}
}

```

다음으로 중요한 부분인 move 함수입니다. for 문을 통해 매 움직임마다 자신 앞에 있는 바디의 위치정보를 계승하면서 모든 바디들이 Head 의 위치를 따라서 움직이게 됩니다. 이 때 각 바디의 x,y pair 값은 직전 앞에 있던 바디의 xy pair 값이 됩니다. 따라서 head 의 위치만 바꿔준다면 자연스럽게 바디 또한 따라오게 됩니다.

다음으로 위에서 설명했듯이 GameScene 클래스의 Run 함수에서 getch 로 받은 키 값을 통해 스네이크의 이동방향을 결정합니다. 이 때 현재 스네이크의 direction, 이동방향과 반대 방향의 키를 누른다면 죽게끔 dead 를 true 로 바꾸고 함수를 종료했습니다.


그 다음 문제가 없다면 headPos 의 값을 수정하여 headPos 를 따라 바디들이 움직이게끔 합니다.

이제 업데이트된 headPos 가 Map 객체에서 어떤 부분을 만났냐에 따라 다른 상호작용을 하게끔 Collision 함수를 통해 구현하였습니다. 이 부분은 move 함수 나머지 아래를 설명하고 다시 설명하겠습니다. Collision 함수를 통해 상호작용을 처리했다면 플레이어가 안죽었다면, 업데이트 된 snake 의 벡터 좌표정보값들을 통해 headPos 와 tailPos 를 업데이트 한 후 맵상에도 Update 를 통해 정보를 갱신합니다. 마지막으로 inGate 부분은 플레이어가 게이트에 들어갔을 때 게이트가 바뀌지 않도록 하는 부분인데, cnt 변수를 통해 처음 게이트에 들어간 이후 바디사이즈만큼 틱이 지나지 않았으면 게이트에 진입하는 중이므로 바뀌지 않게 inGate 를 true 로 바꿨습니다. 일정 틱이 지나면(gate 를 통과하면) 생성자에 의해 inGate 는 false 로 바뀌어 게이트가 다시 바뀝니다.

```

void Snake::Collision(char type) {
    if(type == '1' || type == '2' || type == '3') {
        dead = true;
    }
    else if(type == '5') {
        inGate = true;
        gateNum += 1;
        int gx, gy;
        bool isup, isdown, isleft, isright;
        if(headPosY == stage->gate_X[0] && headPosX == stage->gate_Y[0]){

```



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

    gx = stage->gate_X[1];
    gy = stage->gate_Y[1];
}
else{
    gx = stage->gate_X[0];
    gy = stage->gate_Y[0];
}

if(gx == 0){
    changeDirection('r', gx, gy);
}
else if(gx == WIDTH-1){
    changeDirection('l', gx, gy);
}
else if(gy == 0){
    changeDirection('d', gx, gy);
}
else if(gy == HEIGHT-1){
    changeDirection('u', gx, gy);
}
else{
    if(direction == 'r') {
        if(stage->maps[gy][gx+1] != '1' && stage->maps[gy][gx+1] != '2')
            changeDirection('r', gx, gy);
        else if(stage->maps[gy+1][gx] != '1' && stage->maps[gy+1][gx] != '2')
            changeDirection('d', gx, gy);
        else if(stage->maps[gy-1][gx] != '1' && stage->maps[gy-1][gx] != '2')
            changeDirection('u', gx, gy);
        else if(stage->maps[gy][gx-1] != '1' && stage->maps[gy][gx-1] != '2')
            changeDirection('l', gx, gy);
    }
    else if(direction == 'd') {
        if(stage->maps[gy+1][gx] != '1' && stage->maps[gy+1][gx] != '2')
            changeDirection('d', gx, gy);
        else if(stage->maps[gy][gx-1] != '1' && stage->maps[gy][gx-1] != '2')
            changeDirection('l', gx, gy);
        else if(stage->maps[gy][gx+1] != '1' && stage->maps[gy][gx+1] != '2')
            changeDirection('r', gx, gy);
        else if(stage->maps[gy-1][gx] != '1' && stage->maps[gy-1][gx] != '2')
            changeDirection('u', gx, gy);
    }
    else if(direction == 'l') {
        if(stage->maps[gy][gx-1] != '1' && stage->maps[gy][gx-1] != '2')
            changeDirection('l', gx, gy);
        else if(stage->maps[gy-1][gx] != '1' && stage->maps[gy-1][gx] != '2')
            changeDirection('u', gx, gy);
        else if(stage->maps[gy+1][gx] != '1' && stage->maps[gy+1][gx] != '2')
            changeDirection('d', gx, gy);
        else if(stage->maps[gy][gx+1] != '1' && stage->maps[gy][gx+1] != '2')
            changeDirection('r', gx, gy);
    }
    else if(direction == 'u') {

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

        if(stage->maps[gy-1][gx] != '1' && stage->maps[gy-1][gx] != '2')
changeDirection('u', gx, gy);
        else if(stage->maps[gy][gx+1] != '1' && stage->maps[gy][gx+1] != '2')
changeDirection('r', gx, gy);
        else if(stage->maps[gy][gx-1] != '1' && stage->maps[gy][gx-1] != '2')
changeDirection('l', gx, gy);
        else if(stage->maps[gy+1][gx] != '1' && stage->maps[gy+1][gx] != '2')
changeDirection('d', gx, gy);
    }
}
}
else if(type == '6') {
    Growth();
}
else if(type == '7') {
    Reduce();
}
}
}

```

먼저 기준은 업데이트된 headPos 이다. 이 headPos 와 해당하는 maps 의 좌표의 값을 비교해 아이템, 게이트, 벽, 허공 처리를 한다. 아이템과 게이트의 처리부분은 3,4 번에서 설명하겠다. 1,2 번에 headPos 가 위치한경우 벽에 부딪힌 경우이므로 dead 를 true 로 바꾸고 리턴한다.

4. 아이템 요소 구현

```

void Map::make_item(){
startT = time(NULL);


srand(time(NULL));
int num = 0;
int growth_Item = rand() % 3;
int grow_num = 0;
int tmpx, tmpy;

while(num<3){
    tmpx = rand()%WIDTH;
    tmpy = rand()%HEIGHT;

    if(maps[tmpy][tmpx] != '0')
        continue;

    if (grow_num < growth_Item) {
        maps[tmpy][tmpx] = '6';
        num++;
        grow_num++;
        continue;
    }
}

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

        maps[tmpy][tmpx] = '7';
        num++;
    }
}

```

Make_item 함수는 아이템을 생성하는 역할을 합니다. 일정 시간 뒤에 아이템이 다시 생성될 수 있도록 startT 변수에 아이템 생성 당시의 시간을 저장해두었습니다.

Growth 아이템과 Poison 아이템의 총 개수는 3 개로 제한해야 하기 때문에 num 이라는 변수에 현재 아이템의 개수를 저장하였고, num 이 3 보다 커지면 아이템 생성을 종료하도록 하였습니다.

또한 rand()%3 을 이용하여 growth 아이템의 개수를 0~2 의 값 중 랜덤하게 결정되도록 하였고, growth_item 이라는 변수에 개수를 저장했습니다. 동시에 현재 growth 아이템의 개수를 grow_num 이라는 변수에 저장하여 생성되어야 할 growth 아이템의 개수를 체크해주었습니다.

rand 함수를 이용하여 아이템의 생성 위치를 랜덤 하게 생성하였고, 이를 tmpx, tmpy 라는 변수에 저장했습니다. 이때 빈 공간에 아이템이 생성되어야 하기 때문에 만약 map[tmpy][tmpx]값이 '0'이 아니라면 위치를 다시 정하도록 하였습니다.

만약 tmpx, tmpy 의 좌표위치가 빈공간이라면, 해당 위치에 아이템을 생성할 수 있도록 growth 아이템의 경우 maps[tmpy][tmpx]값을 '6'으로 바꿔주었고, poison 아이템의 경우 maps[tmpy][tmpx]값을 '7'로 바꿔주었습니다.

```

void Map::check_item() {
    nowT = time(NULL);

    if(nowT - startT > 5){ //5 초 이상 지났을 경우
        for (int i=0; i<30; i++) { //아이템의 위치 지움
            for (int j=0; j<110; j++) {
                if (maps[i][j]=='6' || maps[i][j]=='7') maps[i][j]='0';
            }
        }

        make_item();
    }
}

```

Check_item 함수에서는 아이템을 생성한 뒤 얼마의 시간이 지났는지를 체크하여, 만약 5 초 이상 지났을 경우 기존의 아이템을 지우고 아이템을 새로 생성하는 역할을 합니다.

Make_item 함수에서 startT 에 저장해뒀던 생성 시간과, nowT 에 저장한 현재 시간을 비교합니다. 이때 두 변수에 담긴 시간차이가 5 초 이상일 경우 기존의 아이템을 지워야 하기 때문에 map 에서 '6'(growth item), '7'(poison item)인 좌표를 찾아 그곳을

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

'0'(빈공간)으로 바꿔줍니다. 그리고 make_item 함수를 호출하여 아이템을 새로 생성합니다.

이 함수는 GameScene 파일에 있는 Run 함수에서 지속적으로 호출하여 아이템의 시간을 체크하도록 하였습니다.

5. Gate 요소 구현

```
void Map::set_Gatepos(){
    gate_startT = time(NULL);

    srand(time(NULL));
    int x = 0;
    int y = 0;
    int gateCount = 0;
    while(gateCount < 2){
        x = rand() % WIDTH;
        y = rand() % HEIGHT;
        if(maps[y][x] == '1'){
            maps[y][x] = '5';
            gate_X[gateCount] = x;
            gate_Y[gateCount] = y;
            gateCount++;
        }
    }
}
```

set_Gatepos()함수는 gate의 위치를 랜덤 값으로 뽑아내 maps배열에 전달하는 Map클래스의 멤버 함수입니다.

랜덤 값을 뽑기 위해 srand(time(NULL))함수를 먼저 호출해주고, 랜덤 값을 담을 x, y 변수를 선언해줍니다.

그 후 gateCount변수를 선언해 while문의 조건을 통해 gate가 최대 2개만 생성될 수 있도록 해줍니다.

While문 내부에서는 x, y를 랜덤 값으로 초기화해주고, gate가 될 수 있는 유일한 칸인 '1'(wall)을 조건으로 두고 랜덤 좌표의 위치가 '1'이면 그곳을 gate로 바꿔줍니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

void Map::GateUpdate(){
    gate_nowT = time(NULL);
    if(nowT - startT > 4) {
        for (int i=0; i<30; i++) {
            for (int j=0; j<110; j++) {
                if (maps[i][j]=='5') maps[i][j]='1';
            }
        }
        set_Gatepos();
    }
}

```

GateUpdate()함수는 실시간으로 gate의 위치를 확인해 **set_Gatepos()**함수를 호출하여 gate의 위치를 초기화해주는 함수입니다.

set_Gatepos()함수내에서 초기화된 Map클래스의 멤버변수인 startT는 게이트가 생성되자마자 타이머가 시작됩니다. 즉 gate_nowT변수, 현재 시간과 빼 주면 게이트 생성직후부터 타이머를 구현할 수 있습니다. 그 시간이 4초가 넘어가면 조건문 안으로 진입합니다.

조건문 안에서는 전체 맵을 for문으로 돌면서 '5'(gate)인 maps의 부분을 '1'(wall)으로 바꿔 주어 gate를 삭제해줍니다.

반복문을 빠져나간 후 gate 생성을 거쳤으니 다시 **set_Gatepos()**함수를 호출하여 gate를 생성해줍니다.

6. 점수 요소 구현


점수는 Map 객체의 objOOO 변수의 값과 Snake 객체의 curOOO 변수의 값을 각각 가져와 이를 활용합니다.

우선 1 번의 마지막 부분 checkClear 함수 부분에서도 확인했듯이 위의 정보를 통해 스테이지 클리어 여부를 결정합니다. 이제 이러한 점수와 목표를 각각 Scoreboard 와 Missionboard 에 표시해야하는데 이는 GameScene 클래스 내에서 구현되어 있습니다.

```

void GameScene::InitScoreboard() {
    int row, col;
    getmaxyx(scoreboard, row, col);
    box(scoreboard, 0, 0);
    attron(COLOR_PAIR(1));
    wbkgd(scoreboard, COLOR_PAIR(1));
    mvwprintw(scoreboard, 0, col/2-col/5, "SCORE BOARD");
    mvwprintw(scoreboard, 2, 2, "B : 6");
    mvwprintw(scoreboard, 4, 2, "+ : 0");
}

```

	국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
		프로젝트 명	Snake game	
		팀 명	ESC	
		Confidential Restricted	Version 1.5	2021-06-20

```

mvwprintw(scoreboard, 6, 2, "- : 0");
mvwprintw(scoreboard, 8, 2, "G : 0");
}

void GameScene::InitMissionboard() {
    int row, col;
    getmaxyx(missionboard, row, col);
    box(missionboard, 0, 0);
    attron(COLOR_PAIR(1));
    wbkgd(missionboard, COLOR_PAIR(1));
    mvwprintw(missionboard, 0, col/2-col/5, "MISSION BOARD");
}

void GameScene::UpdateGameBoard(){
    stage->Render(gameboard);
}


void GameScene::UpdateScoreBoard() {
    mvwprintw(scoreboard, 2, 6, to_char(to_string(player->getCurLen()) + "/" +
to_string(player->getMaxLen())));
    mvwprintw(scoreboard, 4, 6, to_char(to_string(player->getGrowthNum())));
    mvwprintw(scoreboard, 6, 6, to_char(to_string(player->getReduceNum())));
    mvwprintw(scoreboard, 8, 6, to_char(to_string(player->getGateNum())));
    wrefresh(scoreboard);
}

void GameScene::UpdateMissionBoard() {
    string bm = "B : " + to_string(stage->getObjLen()) + isAchieve(player-
>getCurLen(), stage->getObjLen());
    string gm = "+ : " + to_string(stage->getObjGrowth()) + isAchieve(player-
>getGrowthNum(), stage->getObjGrowth());
    string rm = "- : " + to_string(stage->getObjReduce()) + isAchieve(player-
>getReduceNum(), stage->getObjReduce());
    string gam = "G : " + to_string(stage->getObjGate()) + isAchieve(player-
>getGateNum(), stage->getObjGate());
    mvwprintw(missionboard, 2, 2, to_char(bm));
    mvwprintw(missionboard, 4, 2, to_char(gm));
    mvwprintw(missionboard, 6, 2, to_char(rm));
    mvwprintw(missionboard, 8, 2, to_char(gam));
    wrefresh(missionboard);
}

char *GameScene::to_char(string s) {
    char *tmp = new char[100];
    strcpy(tmp, s.c_str());
    return tmp;
}

string GameScene::isAchieve(int cur, int goal) {
    string res = "";
    if(cur >= goal) {
        res = "(v)";
    }
}

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

```

}
else {
    res = "( )";
}
return res;
}

```

위와 같이 InitWindow 에서 처음 Scoreboard 와 Missionboard 에 대한 UI 를 설정 후 초기화합니다.

이제 게임 진행상황에 따라 점수들을 업데이트하고 이를 scoreboard, missionboard 에 보여줘야 하는데 그 함수들은 UpdateScoreboard, UpdateMissionboard 함수입니다. 이 함수들은 GameScene 클래스의 Run 함수 while 루프 안에 호출되며 매 순간마다 객체 정보에 따라 업데이트됩니다.

mvwprintw 함수를 통해 ncurses UI 내 글자를 표현하였으며, Snake, Map 의 게터 함수를 통해 멤버 변수 값을 읽어오고 이를 to_string, 직접 디자인한 to_char 함수를 통해 mvwprintw 함수의 인자 값으로 넣었습니다. 마지막으로 중요한 wrefresh 를 통해 윈도우 화면을 업데이트 합니다.

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.

이 프로젝트에 사용한 외부 기술/라이브러리는 ncurses와 내장 STL인 vector가 있습니다.

• STL


Snake의 구조를 표현할 때 vector를 사용하였습니다. vector의 0번째 원소는 snake의 head 를 가리키고 나머지는 snake의 body를 가리킵니다. 이때 x, y 좌표를 나타내기엔 pair타입이 적합하다고 생각되어 vector의 원소 타입을 pair로 하였고, 원소 각각은 x, y로 좌표 위치를 가리킵니다.

• Ncurses

전체적인 게임 화면을 출력하는 데 ncurses를 사용하였습니다.

• fstream

각 stage의 txt기반 맵을 불러오기 위해 fstream의 ifstream을 주로 활용하였습니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

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.


1. Snake를 배열로 만들어 머리와 몸통을 관리하기에는 배열의 값을 실시간으로 삽입삭제 하는 데에 어려움이 있었습니다. 하지만 이것을 c++ 프로그래밍수업에서 배운 vector라이브러리를 이용해 Snake를 vector로 만들었고 push_back과 pop_back함수를 이용해 짧은 시간에 쉽게 많은 구현을 할 수 있게 되었습니다.

2.2.5 결과물 목록

작성요령 (5 점)

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

- **stage1.txt, stage2.txt, stage3.txt, stage4.txt**
각 stage 에 정해진 맵의 구조, mission 을 저장해놓은 텍스트 파일입니다.
- **EndScene.cpp**
사용자가 게임을 클리어했을 때 나타날 화면을 구현해놓은 cpp 파일입니다.
- **EndScene.h**
게임 클리어 화면이 나타나도록 구현한 클래스가 정의되어있는 헤더파일입니다.
- **GameScene.cpp**
게임의 전반적인 윈도우를 다루고, 게임 보드 및 score 보드, mission 보드를 구성하는 기능이 구현되어있는 cpp 파일입니다. Run()함수를 통해 게임의 실행을 관리합니다.
- **GameScene.h**
게임의 화면 구성을 관리하도록 구현한 클래스가 정의되어있는 헤더파일입니다. ncurses 라이브러리를 사용합니다.
- **GameclearScene.cpp**
사용자가 각 단계를 클리어 했을 때 나타날 화면을 구현해놓은 cpp 파일입니다.
- **GameclearScene.h**
클리어 화면이 나타나도록 구현한 클래스가 정의되어있는 헤더파일입니다.
- **GameOverScene.cpp**
여러 이유로 게임이 종료되었을 때 게임 종료 화면이 나타나도록 하는 기능이 구현되어있는 cpp 파일입니다.
- **GameOverScene.h**

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

게임이 종료되었을 때 게임 종료 화면이 나타나도록 구현한 클래스가 정의되어있는 헤더파일입니다.

- **Map.cpp**

ncurses 라이브러리를 이용하여 전반적인 map 을 만드는 기능이 구현되어있는 cpp 파일입니다. Stage 텍스트 파일로부터 읽어온 값들을 이용해 맵을 구성하고, ncurses 라이브러리의 COLOR_PAIR(), init_pair 등을 통해 색을 나타냅니다. 이 외에도 아이템 생성, 함수, 게이트 위치 생성 함수 등을 포함하고 있습니다.

- **Map.h**

게이트 및 아이템 생성 등 전반적으로 맵에 관련된 기능을 수행하도록 구현된 클래스가 정의되어있는 헤더파일입니다.

- **Snake.cpp**

스네이크의 초기 위치를 생성하고, 사용자로부터 키 입력을 받아 방향을 바꾸는 등 스네이크의 전반적인 움직임을 컨트롤하는 기능이 구현되어있는 cpp 파일입니다.

- **Snake.h**

스네이크의 움직임, 위치, 길이 등을 조절하도록 구현한 클래스가 정의되어있는 헤더파일입니다.

- **SnakeGame.cpp**

스네이크 게임의 진행을 담당하는 cpp 파일입니다.

- **SnakeGame.h**

스네이크 게임의 진행을 담당하는 클래스가 정의되어있는 헤더파일입니다.

- **StartScene.cpp**

게임 시작 전 메인화면이 나타나도록 하는 기능이 구현되어있는 cpp 파일입니다. 사용자에게 아무 입력을 받으면 게임화면으로 넘어가도록 합니다.

- **StartScene.h**

게임 시작 전 메인화면을 띄우도록 구현한 클래스가 정의되어있는 헤더파일입니다.


- **Main.cpp**

게임을 시작하는 main 함수를 포함하는 cpp 파일입니다.

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.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

이다은(20203112)

역할 및 프로젝트 과정 수행 시 느낀 점

이번 c++프로젝트에서 규칙에 맞게 기능을 개발하는 역할을 맡아 이를 구현하였고, 보고서의 초안을 작성했습니다. 평소 C++언어를 이용하여 알고리즘 문제를 해결하는 것에 관심이 많았기 때문에 규칙에 맞게 게임 로직을 구현하는 데는 큰 어려움이 없었지만, c++언어를 이용하여 온전히 하나의 프로젝트를 수행한 경험은 없었기 때문에, 구조 설계나 더 효율적인 코드를 만드는 방법, makefile 을 만드는 법, 역할 분배 등에 대해 어려움이 있었습니다. 하지만 이 과정에서 많은 검색을 통해 프로젝트의 구조 설계는 어떻게 해야 하는지, 많이 쓰이는 makefile 의 형식은 어떤지 등 단순 알고리즘 문제 해결만 경험해본 저로서는 이번 기회에 많은 것을 배울 수 있었습니다. 또한 여러 명이 팀을 이루어 역할을 나누고, 코드를 작성하여 하나로 병합하는 법을 알게 되었기 때문에, 앞으로 팀을 만들어 프로젝트를 수행할 일이 있을 때 이번 경험이 도움이 될 것 같습니다. 아쉬웠던 점은 이 프로젝트를 경험하며 아직 저의 실력이 미숙하다는 것을 깨달았습니다. 시간을 투자한 만큼 코드가 원활히 작동하지 않아 아쉬웠지만 팀원들의 도움 및 조언을 통해 프로젝트 운영은 잘 진행되었던 것 같습니다.

특히, 다른 팀원들과 작성한 코드에 대해 코드 리뷰 시간을 가지며, 코드의 효율성을 개선하는데 있어서 많은 도움이 되었습니다. 또한 이번 프로젝트를 통해 STL, class 등 C++프로그래밍 수업에서 배운 개념들을 알맞게 활용하는 법을 배워볼 수 있었고, 동시에 Ncurses 라이브러리를 다뤄볼 수 있었습니다. 특히 Class 나 STL 은 아직 제대로 된 사용법을 익히지 못했다고 생각했었는데 이번 프로젝트의 구조 설계를 통해 class 의 개념에 대해 완벽히 이해할 수 있게 되었고, 기능별로 파일을 분류하는 것의 효율성에 대해 깨달았던 것 같습니다.

결론적으로 수업에서 배운 내용을 전반적으로 적용하고, 또 스스로 학습해보며 C++언어의 기술들을 익혀볼 수 있었던 유익한 프로젝트라고 생각합니다.


개선이 필요한 점

Ncurses 를 조금 더 깊게 분석하여 디자인적인 요소를 추가하거나, 동적인 요소를 추가적으로 구현하였더라면 사용자입장에서 더 보기 좋은 게임을 만들 수 있겠다라는 아쉬움이 들었습니다. 또한 이번 프로젝트를 수행하며 다른 팀원들의 도움을 많이 받았고, 실력적인 면에서 부족함을 느꼈습니다. 그래서 앞으로는 개발 실력을 더 키워서 이와 같은 프로젝트에서 팀원들에게 도움을 주는, 주도적인 역할을 해보고 싶다는 생각이 들었습니다.

김영광(20203042)

역할

프로젝트에서 Map, gate 개발 부분을 주로 맡았습니다. 처음 Ncurses 를 배우면서 어떻게 하면 stage 가 여러 개인 Snake 게임의 Map 을 쉽게 구현할 수 있을까 생각해봤습니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

그러던 중 c++ 파일입출력인 fstream 을 이용해 txt 파일을 불러와 전체적인 Map 을 쉽게 구성해볼 수 있겠다 생각하였고, 팀원들과 방법을 공유하여 함께 만들어갈 수 있었습니다. Gate 개발은 김한림 팀장이 snake 기능을 먼저 잘 만들어 주어 구현하는 데에는 크게 어려움이 없었습니다. 다만 snake 가 gate 를 통과할 때 나오는 방향을 정할 때 생각할 경우의 수가 많아 조금 헷갈렸지만 팀원과 상의하여 잘 구현할 수 있었습니다. 이외에도 프로젝트의 전체적인 구조, snake 의 구현 방식 등 팀원들과 화면공유 방식을 통해 코드를 보며 상의하며 개발에 기여했습니다.

프로젝트 과정 수행 시 느낀 점

개발 프로젝트를 하면서 항상 느끼는 점은 역할을 분담하는 점에서 어려움을 느끼는 것입니다. 하지만 이번 프로젝트는 수행하면서 3 명 모두의 역할이 잘 분배된 것은 물론이고, 팀원의 부족한 점이 있으면 다른 팀원이 보충해주 가면서 서로의 역할에 부담 없이 개발을 진행할 수 있어 팀원 모두가 만족할 만한 프로젝트를 진행했다 라는 생각이 들었습니다. 개발 쪽으로 느낀 점은 snake 게임을 개발하는 것이 주 목적이었지만 그 이전에 ncurses 라이브러리를 이해하고 라이브러리 안에 있는 함수와 기능을 파악하는 것에 어려움이 있었습니다. 아무래도 처음 써보는 라이브러리 이기도 했고, 이전에 게임을 개발할 때는 그래픽 이미지를 이용할 수 있는 라이브러리를 주로 사용했기에 한 번도 다뤄 본적 없는 형태의 라이브러리라 이해하는데 시간이 조금 걸렸습니다. 하지만 그것도 프로젝트의 수행의 과정이라고 생각하며 임했고, 무엇보다 팀원들이 있어 같이 알아갈 수 있었기에 더 빨리 어려움을 극복할 수 있었습니다.


개선이 필요한 점

기본적으로 Snake 게임에 필요한 기능은 모두 구현했고, 팀원들과 협동심도 좋았지만 아무래도 프로젝트라는 명목 하에 조금 더 체계적인 계획을 정하고 프로젝트에 들어갔으면 더 잘 프로젝트를 수행할 수 있지 않았을까 라는 생각이 들었습니다. 하지만 다른 과목의 프로젝트와 과제들도 있었고, 그에 따라 모두의 일정을 조율할 수 없어 계획을 정리하는 데에는 무리가 있었습니다. 그럼에도 만약 다시 프로젝트를 진행한다면 무리가 가지 않는 선에서 조금 더 체계적인 계획을 세우고 프로젝트 수행을 하고싶다는 욕심은 있었습니다.

김한림(20203055)

역할

프로젝트의 팀장을 맡으며 프로젝트의 전반적인 진행을 컨트롤했습니다. 코드의 전반적인 뼈대를 구현하였으며 이외의 자잘한 부분은 팀원과 함께 고민하며 수정해 나갔습니다. 게임의 전반적인 UI 를 설계하였으며 김영광 조원이 미리 만들어놓은 Map 의 뼈대를 적용시켜 게임의 전체적인 골격을 잡았습니다. 김영광 조원과 상의를 통해서 프로젝트의 파일 구조를 설계하였으며, 헤더파일과 cpp 파일로 나누어서 효율적으로 클래스를 관리하였습니다. Snake 의 이동을 구현하였으며 Growth, Reduce 아이템, 벽충돌에 대한

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

처리를 구현하였습니다. 이 밖에 UI 의 직관성을 위하여 GameStartScene, GameoverScene, EndScene 을 추가하였습니다.

수행과정 중 느낀 점과 어려운 점

먼저 기능별로 클래스를 나누고 헤더파일과 cpp 파일로 나누어 전체 프로젝트를 효율적으로 관리하는 방법이 익숙치 않아 초반 게임의 틀을 잡을 때 힘들었습니다. ncurses 라이브러리에 대해 이해하는 과정에서 윈도우의 생성 과정과 터미널 내에서 어떻게 처리되는지 이해하는 과정도 꽤나 시간이 걸렸습니다. 하지만 과제를 진행함에 따라 전반적인 뼈대는 ncurses 를 배제하고 맵을 2 차원배열로 다룸으로써 그 안에서 스네이크의 로직을 구현한 다음 만들어진 로직에 ncurses 의 UI 를 덮어씌운다는 느낌으로 작업을 하니 프로젝트 진행 방향에 대한 길이 보였고, 작업속도 또한 점점 빨라졌습니다. 가장 크게 느낀 점은 한번 뼈대(클래스 구조, 파일 분리)를 만들고 나니 그 이후부터는 기능추가와 구현이 정말 편하다는 것이었습니다. 기능을 추가하고 싶다면 기존에 만들어진 뼈대 위에 변수와 몇 가지 클래스만 새로 추가하여 연동하면 되니 유지보수에 있어 정말 편하다고 느꼈습니다. 그 과정에서 게임 제작에 대한 흥미가 다시 느껴졌습니다. 아마 이번 학기 중 가장 힘들기도 했지만 그와 동시에 가장 재밌기도 한 프로젝트였습니다. 이렇게 뼈대만 제대로 만든다면 더 큰 프로젝트를 만들 수 있을거란 자신감 또한 생겼습니다.


개선이 필요한 점

먼저 프로젝트의 골격을 잡고 파일 이름을 정하며 여러 클래스를 나누는 작업을 할 때 기능이 제대로 쓰이지 않는 경우가 있었습니다. 예를 들어 Map 클래스의 Update 함수를 구현하여 maps 변수의 값을 변경하게 구현하였는데, 다른 부분에서 함수를 쓰지 않고 stage->maps[x][y] = '2' 이런 식으로 직접적으로 멤버변수에 접근하여 값을 변경했습니다. 또 아쉬운 점은 Gate 구현에 있었는데 Gate 가 맵의 중간에서 생성되었을 때 처리하는 부분에서 나오는 게이트의 방향을 결정하는 부분에서 if, else 문으로만 단순하게 구현하여 코드의 효율성이 떨어졌다는 것입니다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
	서적					
	기사					

<https://widian.tistory.com/58> // ncurses의 함수들의 사용법을 참고하였습니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

5 부록

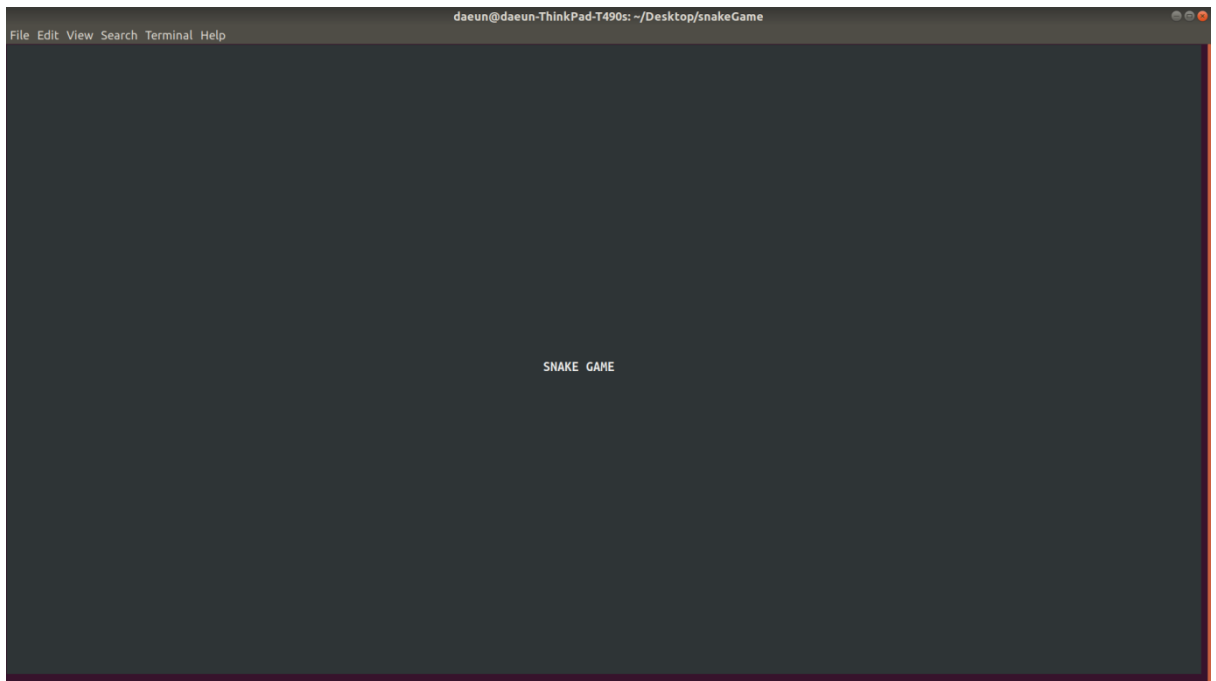
작성요령 (15 점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

5.1 사용자 매뉴얼

아래 5.2 의 설치 방법을 통해 필요한 실행파일을 생성하고 게임을 시작합니다..


• 시작화면

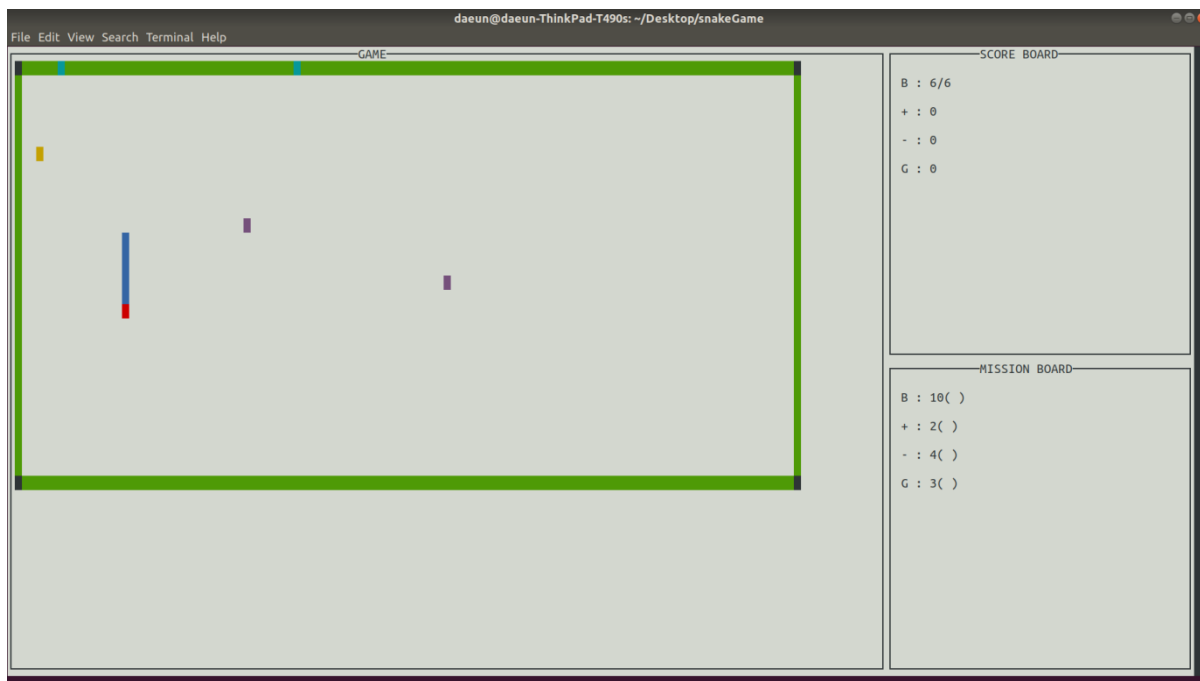


파일을 실행하면 본격적인 게임이 시작되기 전에 위와 같은 화면이 나타납니다.

사용자가 키보드의 아무 키를 누르게 되면 게임을 시작합니다.

• game play 화면

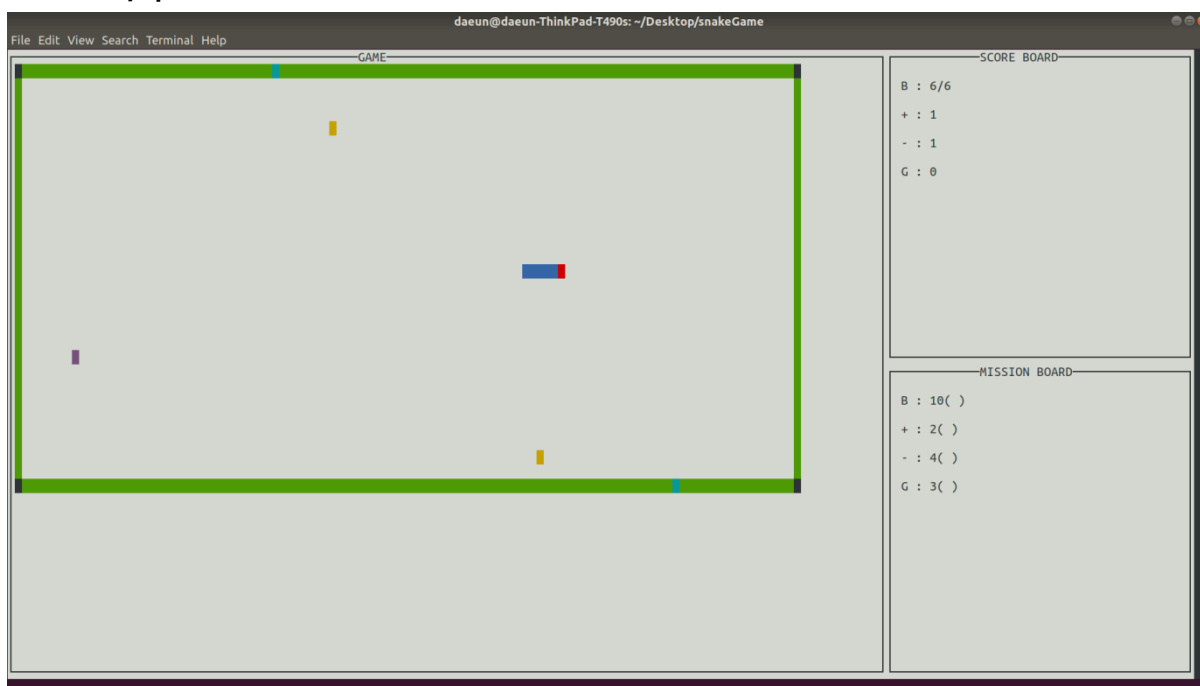
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20




사용자가 키를 누르게 되면 본격적으로 게임이 시작됩니다. 메인화에는 스네이크와 아이템이 보이게 되며 우측에는 현재 스코어를 나타내주는 Score board와 미션 달성 여부를 확인할 수 있는 mission board가 나타납니다.

사용자는 방향키로 Snake를 자유롭게 움직일 수 있습니다.

• item 획득

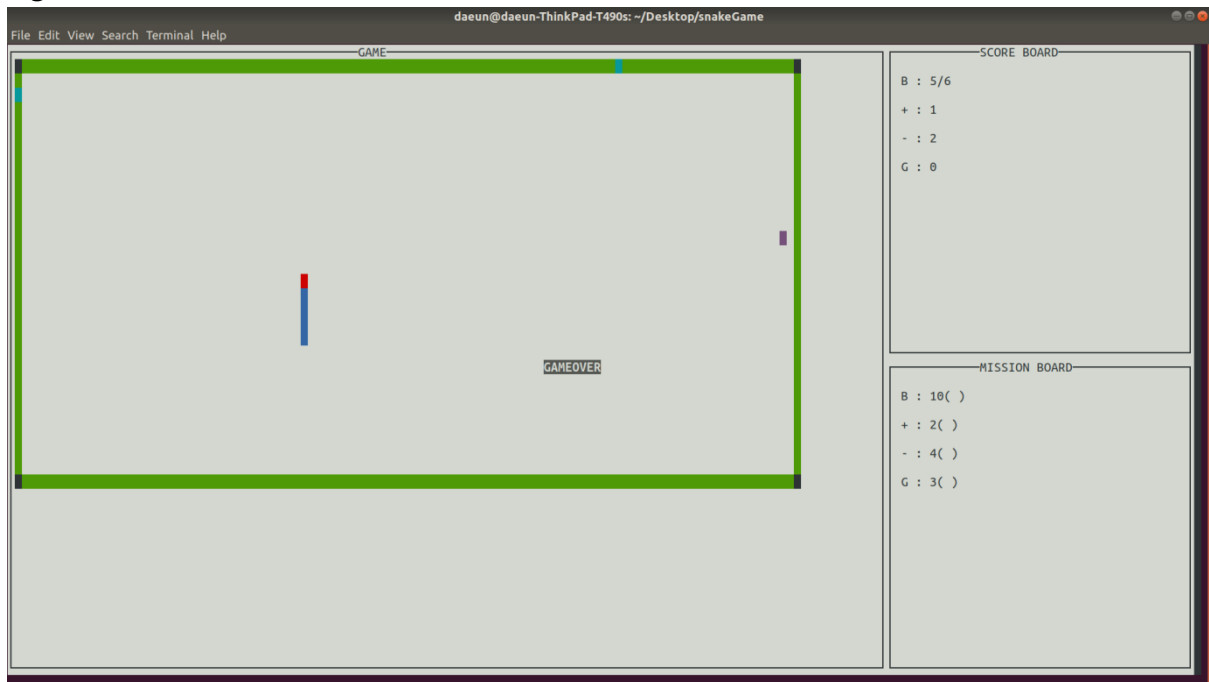


Snake가 아이템을 획득하게 되면 우측의 스코어보드에 획득한 아이템의 개수가 기록됩니다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Snake game	
	팀 명	ESC	
	Confidential Restricted	Version 1.5	2021-06-20

Growth item은 +에 체크되며, poison item은 -에 체크됩니다. 아이템을 획득할 때마다 snake의 길이가 변화하는 모습을 볼 수 있습니다.

• game over 화면



여러 이유(snake의 진행 방향과 반대 방향의 키를 입력했을 때 등)로 게임에 실패했을 경우 스네이크가 멈추고 game over라는 문구가 나타납니다. 사용자가 키보드 위의 아무 키를 누르게 된다면 게임을 재시작 할 수 있습니다.

5.2 설치 방법

• ncurses 라이브러리 설치

• 컴파일 및 실행

터미널에서 make 를 입력하면 실행파일인 main 이 생성됩니다.

실행파일을 만든 후 터미널에서 ./main 명령어를 입력하면 게임이 실행됩니다.