국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	스네이크 게임
팀 명	말하는감자
문서 제목	결과보고서

Version	**1.1**
Date	2024-06-15

팀원	20203104 유 동현 (팀장)
	20212972 김 민찬
	20213074 장 종아

문서 정보 / 수정 내역

CONFIDENTIALITY/SECURITY WARNING

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

Filename	최종보고서-스네이크 게임.doc
원안작성자	유동현, 김민찬, 장종아
수정작업자	유동현, 김민찬, 장종아

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2024-06-10	유동현	1.0.0	최초 작성	문서 생성
2024-06-12	김민찬	1.0.1	내용 추가	
2024-06-13	장종아	1.0.2	내용 추가	
2024-06-14	김민찬	1.0.3	내용 추가	
2024-06-15	장종아	1.0.4	내용 추가	
2024-06-16	유동현	1.1.0	내용 추가	

목 차

1 개요

1.1 목적

본 프로젝트는 2024학년도 ‘C++프로그래밍' 과목의 기말 프로젝트로써 진행되었으며 소규모 프로젝트로 C++ Language의 실력 향상과, 객체 지향 프로그램을 이해하고 ncurses 라이브러리를 사용하여 SnakeGame을 구현하는데 있다. 또한 Github을 통한 협업 또한 경험하는 것에 있다.

1.2 결과물

- **파일 목록**

```
├── Gate.cpp
├── Gate.h
├── Item.cpp
├── Item.h
├── Map.cpp
├── Map.h
├── Mission.cpp
├── Mission.h
├── Score.cpp
├── Score.h
├── ScoreBoard.cpp
├── ScoreBoard.h
├── Snake.cpp
├── Snake.h
├── SnakeGame.cpp
├── SnakeGame.h
├── StageManager.cpp
├── StageManager.h
├── main.cpp
```

- Map.cpp / Map.h

  맵 데이터 저장, 스테이지 번호를 불러와 그에 맞는 맵 데이터 로드.

- Item.cpp / Item.h

  아이템 생성 주기 관리, 생성 시 Growth/Poison 인지 구분, 아이템 생성 위치 설정

- Gate.cpp / Gate.h

게이트 생성 주기 관리, 게이트 생성 위치 판단 및 설정

- Snake.cpp / Snake.h

  스네이크 움직임 설정, 스네이크 길이 설정, 게이트 진입 시 스네이크 위치와 방향 설정,

- SnakeGame.cpp / SnakeGame.h

  스네이크 게임 시작, 스네이크게임 화면 그리기, 아이템 획득, 벽 충돌, 게이트 진입 판단, 게임오버 판단, 키 입력 시 스네이크 방향 전환

- Mission.cpp / Mission.h

  각 스테이지 별 클리어 미션 설정, 스테이지 별 미션 불러오기

- Score.cpp / Score.h

  먹은 아이템 개수, 총 길이, 게이트 사용 횟수 등 스코어 설정

- ScoreBoard.cpp / ScoreBoard.h

  맵 우측에 점수판 그리기, 점수에 따른 점수판 수정, 스테이지 클리어 판단

- StageManager.cpp / StageManager.h

  점수 리셋, 점수 획득 관리, 스테이지 관리

- Credit.cpp / Credit.h

  게임 시작화면, How To Play, 게임 종료 기능 구현

- main.cpp

  스네이크 게임 실행

1.3 클래스 설계

- **하위요소는 파일 별 상호작용을 나타낸다.**

main.cpp

```
├── Credit.cpp
└── SnakeGame.cpp
    ├── ScoreBoard.cpp
    │   ├── StageManager.cpp
    │   │   └── Score.cpp
    │   └── Mission.cpp
    │
    └── Map.cpp
        ├── Snake.cpp
        ├── Gate.cpp
        └── Item.cpp
```

1.4 구현 담당

20203104 유 동현	Map, ScoreBoard, Stage, Mission
20212972 김 민찬	Snake, Item, SnakeGame
20213074 장 종아	Snake, Gate, SnakeGame

1.5 Ncurses

1.5.1 ncurses

- ncurses (new curses)는 프로그래머가 텍스트 사용자 인터페이스를 터미널 독립 방식으로 기록할 수 있도록 API를 제공하는 프로그래밍 라이브러리이다. 단말 에뮬레이터에서 실행하는 GUI같은 응용 소프트웨어를 개발하는 툴킷이다.

1.5.2 ncurses 설치 방법

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

1.6 개발 환경

1.6.1 Ubuntu 20.04LT

- Ncurses 라이브러리 사용을 위한 Linux 환경 제공
- 조원들간의 OS 차이(MacOS ⇔ Windows) 해결

1.6.2 Github

- 버전 관리
- 협업

2 개발 내용 및 결과물

2.1 구현 목표

적용단계	내용	적용 여부
1단계	Map의 구현	적용
	Map의 Data 처리	적용
	Map 클래스와 Snake 클래스의 상호 작용	적용
	Map 클래스와 Item 클래스의 상호작용	적용
2단계	Snake 표현 및 조작	적용
	키 입력에 대한 처리	적용
	Item 획득시 Tick의 변화	적용
3단계	Item 요소의 구현	적용
	Item 요소의 출현	적용
	Item클래스와 Snake 사이의 상호작용	적용
	Fast Item, Slow Item 구현	적용
4단계	Gate 요소의 구현	적용
	Map과 Gate의 상호작용	적용
	Snake와 Gate의 상호작용	적용
5단계	점수 요소의 구현	적용
	Mission의 구현	적용
	Stage의 변화 구현	적용

기본 Incurses와 다르게 Incursesw를 사용하여 유니코드 문자를 집어넣고 컴파일 할 수 있습니다.
ex) wall = L'■'; // (■ : \u25fc)

저희 팀의 협업은 Github를 활용하였으며 각자 Ubuntu 환경에서 제작하였습니다.

SnakeGame은 총 5개의 스테이지로 구성되어 있습니다. 해당 스테이지의 미션 (게이트 통과 수, 스네이크의 몸 길이, 아이템 획득 수 등)을 클리어 해야만 다음 스테이지로 넘어갈 수 있습니다.
Growth Item을 먹었을 때 Snake의 길이가 1 늘어나며 Poison item을 먹을 시 Snake의 길이가 1 감소합니다. Growth Item은 동시에 최대 3개까지 나타나도록 구현 했습니다.
Gate를 통과할 때 마다 Gate 통과 수가 1 늘어나며 통과 시 방향은 벽 반대방향으로 고정합니다..

main.cpp 파일에서 SnakeGame을 실행하며 SnakeGame.cpp 파일에서 외부 클래스들을 활용해 SnakeGame의 전반적인 로직을 관리한다.

전체적인 코드는 **OOP**를 준수하여 작성한다.

2.2 게임 규칙

2.2.1 Rule #1

- Snake는 진행 방향의 반대방향으로 이동할 수 없다.
- Snake는 자신의 Body를 통과할 수 없다.
- Snake는 벽(wall) 을 통과할 수 없다.
- Head의 방향 이동은 일정시간(틱)에 의해 이동한다.

2.2.2 Rule #2

- Snake의 이동 방향에 Item이 놓여 있는 경우
- Growth Item의 경우 몸의 길이(Tail)가 1 증가한다.
- Poison Item의 경우 몸의 길이(Tail)가 1 감소한다.
    - 몸의 길이가 3보다 작아지면 Game Over
- Item의 출현
    - Snake Body와 Wall이 있지 않은 임의의 위치에 출현
    - 출현 후 일정시간이 지나면 사라지고 다른 위치에 나타나야 한다.
    - 동시에 출현할 수 있는 Item의 수는 3개로 제한한다.

2.2.3 Rule #3

- Gate는 두 개가 한 쌍이다.
- Gate는 겹치지 않는다.
- Gate는 임의의 위치에 있는 벽에서 나타난다.
- Gate에 Snake가 진입중인 경우 Gate는 사라지지 않고, 다른 위치에 나타나지 않는다.
- Gate는 한번에 한쌍만 나타난다.

2.2.4 Rule #4

- Gate가 나타나는 벽이 가장자리에 있을 때
    - 항상 Map의 안쪽 방향으로 진출한다.
        - 상단 벽 : 아래 방향
        - 하단 벽 : 위 방향
        - 좌측 벽 : 오른쪽 방향
        - 우측 벽 : 왼쪽 방향
- Gate가 나타나는 벽이 Map의 가운데 있을 때
    - 진입 방향과 일치하는 방향이 우선
    - 진입 방향의 시계방향으로 회전하는 방향
    - 진입 방향의 역시계방향으로 회전하는 방향
    - 진입 방향과 반대방향

2.2.5 Rule #5

- Wall은 Gate로 변할 수 있다.

- Immune Wall은 Gate로 변할 수 없다.
- Snake는 모든 Wall을 통과할 수 없다.
- Snake의 Head가 Wall에 충돌 시 Gameover

2.2.6 Rule #6

- 점수 계산
- 게임 중 몸의 길이 계산
- 게임 중 획득한 Growth Item의 수
- 게임 중 획득한 Poison Item의 수
- 게임 중 Gate의 사용 횟수
- 게임 시간
- Misson
    - 5개의 Stage로 구성 된다.
    - 각 Stage의 Mission은 다르다.
    - 각 Stage의 Mission을 달성시 시각적으로 표시된다.
    - Mission을 달성 시 다음 Stage로 넘어간다.

2.3 구현 내용

2.3.1.1 Map.h / Map.cpp

- 유동현, 김민찬, 장종아

- 변수

접근자	자료형	변수명	비고
private	int	WIDTH	Map의 가로 길이
private	int	HEIGHT	Map의 세로 길이
private	vector<vector<vector<int>>>	map	전체 Map 저장
private	vector<vector<int>>	current_map	현재 Stage의 Map 저장

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	Map	x	생성자
public	int	getWidth	x	Map 가로 길이 반환 함수
public	int	getHeight	x	Map 세로 길이 반환 함수
public	void	setMap	stage, y, x, val	Item, Gate, Snake Map 반영 함수
public	void	setAllMap	stage, newMap	반영된 Map으로 업데이트 함수
public	vector<vector<int>>	getMap	stage	현재 Stage의 Map 반환 함수

(1) Map의 구현

- 21 * 21크기의 2차원 배열을 통해 빈 공간은 0, Wall이 표시되어야 하는 부분은 1, Immune Wall이 표시되어야 하는 곳은 2로 표시하였다.
- Map은 5개의 Stage가 있으며 3차원 vector Map 내에 저장된다

(2) setMap 함수를 통해 Map에 Item, Snake, Gate의 위치를 반영한다

(3) setAllMap 함수를 통해 반영된 Item, Snake, Gate를 현재 지도에 업데이트 한다

(4) getMap 함수를 통해 현재 Stage의 Map 정보를 반환한다.

2.3.1.2 Snakegame.h / Snakegame.cpp

- 유동현, 김민찬, 장종아

- 변수

접근자	자료형	변수명	비고
private	bool	gameOver	Game의 종료 여부 Flag 변수
private	bool	isCleared	Game의 클리어 여부 Flag 변수
private	int	stage	현재 Stage를 저장하는 변수
private	int	timeTick	현재 Map
private	static const int	ITEM_DURATION	Item의 지속시간
private	static const int	GATE_DURATION	Gate의 지속시간
private	static const int	MAX_ITEMS	Item의 최대 수
private	static const int	GATE_COOLDOWN	Gate의 재생성 대기시간

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	SnakeGame	x	생성자
public	void	run	x	게임 실행 위한 함수
private	void	init	x	화면의 초기화 위한 함수
private	void	draw	vector<vector<int>>	전체적인 화면을 그리는 함수
private	bool	input	x	입력을 받고 Snake의 진행방향을 바꾸어주는 함수
private	void	goforward	x	Snake의 위치를 업데이트 해주는 함수
private	bool	checkCollision	x	Snake의 충돌이 있는지

				확인하는 함수
private	void	endGame	x	게임을 종료하는 함수
private	void	manageItems	x	Item의 생성을 담당하는 함수
private	void	manageGate	x	Gate의 생성을 담당하는 함수
private	void	CheckItemCollision	x	Item의 획득을 확인하는 함수
private	void	checkGateEnter	x	Gate를 통과중인지 확인하는 함수
private	void	makeMap	x	Map에 Item, Gate, Snake 반영

(1) Map 클래스. ScoreBoard 클래스에서 Map 정보, ScoreBoard 정보를 얻어와 전체적인 게임의 흐름을 구성하는 클래스이다.
(2) bool 자료형 gameOver 변수에 Game의 종료여부를 저장한다
(3) bool 자료형 isCleared 변수에 Game의 클리어 여부를 저장한다
(4) int 자료형 stage에 ScoreBoard 클래스로부터 현재 Stage를 불러와 저장한다
(5) draw 함수를 통해 전체적인 화면을 그린다. 2차원 vector를 인자로 받으며 Snake, Item, Gate와 상호작용이 끝난 21 * 21 크기의 배열을 출력한다
(6) input 함수를 사용자의 입력을 받고 Snake 클래스의 이동방향을 변경하여 Snake의 이동방향을 제어할 수 있다.
(7) goforward 함수를 통하여 매 틱마다 Snake의 위치를 업데이트하여 Map에 반영한다
(8) checkCollision 함수에서 스네이크의 Head의 위치와 벽의 위치가 같으면 True를 반환하여 게임을 종료한다.
(9) checkItemCollision 함수에서 스네이크의 Head의 위치와 아이템의 위치가 같으면 True를 반환하고 Item의 번호를 매개변수로 ScoreBoard 클래스의 increase함수를 호출하여 점수를 증가시킨다.
(10) checkGateEnter 함수에서 Snake가 Gate에 진입하면, Gate의 시간을 멈춰 사라지지 않게 하고, Gate의 몸체 부분이 다 빠져나가면 다시 시간을 진행시킨다. 또한, Snake의 좌표와 방향을 설정한다.
(11) mangeItem, manageGate함수를 통해 Item 과 Gate를 삭제, 생성 한다.

2.3.1.3 main.cpp

(1) 게임을 실행하는 내용을 담고 있다.

(2) 반복문을 통하여 메인화면에서 Exit를 선택하기 전까지 게임을 실행한다.

2.3.2 2단계

2.3.2.1 Snake.h / Snake.cpp

- 김민찬, 장종아

- 변수

접근자	자료형	변수명	비고
public	Direction	dir	방향을 나타냄 UP, DOWN, LEFT, RIGHT의 방향을 가짐
public	deque<pair<int,int>>	body	snake의 몸 위치 정보를 나타냄 첫번째 인덱스 값은 head를 나타내고 나머지는 body를 나타냄

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	Snake	width, height	생성자 초기 Snake의 몸길이 3을 설정하고 위치를 맵의 가운데로 설정함
public	void	move	x	Snake가 한칸 이동할때의 로직
public	void	grow	x	Snake의 꼬리 길이를 1 늘리고 값 추가
public	int	getLength	x	Snake의 body변수의 size를 return함
public	void	gateEntry	pos, dir	gate 진입 시 헤드 좌표 변경
public	void	resetSnake	width,height	다시 초기 Snake의 상태로 초기화

public	Direction	gateDecisionDir	width,height,dir,pos,map	gate 진출 시 방향 설정

- Snake의 움직임과 방향
  (1) Snake는 초기 map의 width와 height를 받아 중간 위치를 계산 후 중간 위치에 snake를 생성한다.
  (2) Snake의 body좌표들은 body라는 deque자료형 변수에 담겨 있으며 맨 첫번째 인덱스의 값은 head로 표시한다.
  (3) Snake는 Direction자료형의 dir변수를 가지고 있다. Direction 자료형은 enum을 통해 선언했으며 UP, DOWN, RIGHT, LEFT 총 4개의 방향값을 담고 있다. Snake는 움직일때 이 dir방향으로 움직인다.
  (4) Snake는 한칸 움직일때 맨뒤 body값을 pop_back으로 자르고 맨 앞에 push_front를 통해 body를 늘림으로써 한칸씩 움직인것처럼 동작한다.

- Snake와 Gate의 상호 작용 설정
  (5) Gate 진입 시, Rule에 따라 gateDicisionDir 함수에서 방향을 설정.
  (6) Rlue이 복잡해보이지만, 모든 경우를 종합해보았을 때, 진행 방향이 우선이고, 진행 방향의 우측으로 가는 것이 우선이다.
  (7) 따라서 방향 우선 순위대로 진행 방향에 벽이 있는지 확인하고 벽이 없는 방향으로 설정
  (8) Gate 진입 시, gateEntry 함수에서 진입한 Gate 외 다른 Gate의 좌표에서 정해진 방향의 한 칸 앞으로 Snake head의 좌표를 설정.

2.3.3 3단계

2.3.3.1 Item.h / Item.cpp

- 김민찬

- 변수

접근자	자료형	변수명	비고
private	pair<int, int>	pos	item의 위치 좌표
private	int	curTime	item이 생성된 이후 몇 초가 지났는지 나타내는 변수
private	int	itemType	item 종류 1 : GrowthItem 2. : PoisonItem

			3 : FastItem 4 : SlowItem

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	Item	width, height	생성자 item을 랜덤으로 생성함
public	void	generateNew Position	width, height	item의 x., y 좌표값을 랜덤으로 지정함
public	pair<int, int>	getPosition	x	pos 변수값을 리턴함
public	int	getItemType	x	itemType 변수값을 리턴함
public	void	setTime	t	curTime 변수값을 t로 지정함
public	int	getTime	x	curTime 변수값을 리턴함

(1) item은 generateNewPosition함수에 의해 매 순간 랜덤 좌표에서 생성된다. generateNewPosition 함수에선 ctime헤더파일의 rand함수를 통해 랜덤 좌표를 생성한다.

(2) item은 snake와 벽이 없는 부분에서 생성되며 동시에 최대 3개까지만 생성 가능하다.

(3) itemType도 랜덤으로 정해지는데 itemType값에 따른 효과는 아래와 같다.

- 1 : GrowthItem (몸 길이 1증가)
- 2 : PoisonItem (몸 길이 1감소)
- 3 : FastItem (속도 증가=>틱이 2배 빨라짐)
- 4 : SlowItem (속도 감소=> 틱이 2배 느려짐)

(4) Item의 최대 지속시간은 5초이며 5초가 지나면 item이 삭제되고 다른 위치에 생성된다.

2.3.4 4단계

2.3.4.1 Gate.h / Gate.cpp

- 장종아

변수

접근자	자료형	변수명	비고

private	pair <int, int>	g1pos	첫 번째 게이트의 좌표
private	pair <int, int>	g2pos	두 번째 게이트의 좌표
private	int	curTime	게이트 생성 이후 시간을 기록하는 변수
private	int	pausedTime	일시정지 된 시간을 저장하는 변수
private	bool	isPaused	일시정지 상태인지 판단하는 변수

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	Gate	width, height	생성자 curTime을 Gate 재생성 대기시간으로 초기화
public	void	genGate	width, height, map, snakeBody	Gate의 좌표를 랜덤으로 설정하고 Rule에 맞는 좌표인지 확인
public	pair <int, int>	getGate1pos	x	Gate1의 좌표 반환
public	pair <int, int>	getGate2pos	x	Gate2의 좌표 반환
public	void	setTime	t	curTime을 t로 설정
public	int	getTime	x	curTime 값을 반환
public	void	pauseTime	x	pauseTime 변수에 curTime을 저장하고 isPaused 값을 True로 설정
public	void	resumeTime	x	curTime 변수에 pauseTime을 저장하고 isPaused 값을 False로 설정

(1) Gate는 생성 주기에 따라 랜덤한 위치에 genGate에 의해 생성한다.

(2) genGate에서 rand() 함수를 통해 g1pos와 g2pos를 맵 좌표 내에서 랜덤으로 설정한다.

(3) Gate는 벽에 생성되어야 하기 때문에 if문으로 랜덤으로 설정된 좌표가 벽인지 확인한다.
(4) 두 Gate의 좌표가 달라야 하기 때문에, 좌표가 다른지 확인한다.
(5) 위 두 조건을 만족할 때까지 반복해서 좌표 랜덤 생성한다.
(6) Snake가 Gate에 진입 중일때, Gate가 사라지는 것을 방지할 때 사용하기 위해 pauseTime과 resumeTime 함수를 선언했다.

2.3.5 5단계

2.3.5.1 Mission.h / Mission.cpp

- 유동현

- 변수

접근자	자료형	변수명	비고
private	vector<int>	stage1	Stage 1의 미션
private	vector<int>	stage2	Stage 2의 미션
private	vector<int>	stage3	Stage 3의 미션
private	vector<int>	stage4	Stage 4의 미션
private	vector<int>	stage5	Stage 5의 미션
current	vector<int>	current	현재 Stage의 Mission 저장

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	Mission	x	생성자
public	vector<int>	getMission	stage	현재 stage의 Mission 반환 함수

(1) 1차원 vector stage1, 2, 3, 4 ,5에 각각 1, 2, 3, 4, 5 스테이지의 미션을 저장한다
    - {LengthScore, GrowthScore, PoisonScore, GateScore}
(2) getMission 함수를 통해 현재 stage의 Mission을 반환한다.

2.3.5.2 Score.h / Score.cpp

- 유동현

- 변수

접근자	자료형	변수명	비고
private	int	lengthScore growthScore poisionScore gateScore	Snake의 길이와 각 아이템의 사용 횟수
private	vector<int>	score	현재 점수를 저장하는 1차원 배열

- Method

접근자	리턴 타입	함수명	인자	비고
private	void	update	x	점수 업데이트 함수
public	void	increaseLengthScore increaseGrowthScore increasePoisonScore increaseGateScore	x	각 점수 증가 함수
public	void	decreaseLengthScore	x	Snake의 길이 점수 감소 함수
public	void	resetScore	x	점수 초기화 함수
public	vector<int>	getScore	x	현재 점수 반환 함수

(1) 각 점수를 lengthScore, growthScore, poisonScore, gateScore 변수에 저장한다. 초기값은 0

(2) Item을 획득하면 increaseLengthScore, increaseGrowthScore, increasePoisonScore, increaseGateScore, decreaseLengthScore을 호출하여 점수를 증가시킨다.

(3) 다음 Stage로 넘어갈 경우 resetScore을 호출하여 점수를 초기화 한다.

(4) getScore 함수를 통해 현재 점수를 1차원 배열에 담아 반환한다.

2.3.5.3 StageManager.h / StageManager.cpp

- 유동현

- 변수

접근자	자료형	변수명	비고
private	int	stage	현재 Stage 저장

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	StageManager	x	생성자
public	vector<int>	getScore	x	현재 점수 반환 함수
public	void	increaseLengthScore increaseGrowthScore increasePoisonScore increaseGateScore	x	각 점수 증가 함수
public	void	decreaseLengthScore	x	Snake의 길이 점수 감소 함수
public	void	resetScore	x	점수 초기화 함수
public	int	getCurrentStage	x	현재 Stage 반환 함수
public	void	nextStage	x	다음 Stage 업데이트 함수

(1) stage 변수에 현재 진행중인 stage의 정보를 저장한다 (초깃값 0)

(2) increaseLengthScore, increaseGrowthScore, increasePoisonScore, increaseGateScore, decreaseLengthScore, resetScore 함수는 ScoreBoard.cpp 와 Score.Cpp 사이의 상호작용을 위해 추가된 함수이다.

(3) getCurrentStage 함수를 통해 현재 진행중인 Stage의 정보를 반환한다

(4) 모든 Mission이 클리어 되었을 경우 nextStage 함수를 호출하여 stage를 1 증가 시킨다.

2.3.5.4 ScoreBoard.h / ScoreBoard.cpp

- 유동현

- 변수

접근자	자료형	변수명	비고
private	vector<vector<int>>	scoreBoard	ScoreBoard 표시 위한 2차원 배열
private	vector<int>	currentMission	현재 미션 저장 1차원 배열
private	vector<int>	currentScore	현재 점수 저장 1차원 배열
private	wstring	lengthProgressBar growthProgressBar poisonProgressBar gateProgressBar	현재 Stage의 Mission 진행상황 시각적 표시
private	string	LENGTH GROWTH POISON GATE	현재 Stage의 Mission 저장
private	int	lengthScore growthScore poisonScore gateScore	Snake의 길이 점수와 Item의 사용 점수
private	int	currentStage	현재 Stage 저장
private	int	adjust	ScoreBoard의 위치 보정
private	int	duration	게임 진행 시간 저장
private	time_t	TIME timeScore	게임 진행 시간을 계산하기 위한 변수
public	bool	lengthCleared growthCleared poisonCleared gateCleared	Mission의 클리어 여부 저장

- Method

접근자	리턴 타입	함수명	인자	비고

public	None	ScoreBoard	x	생성자
public	void	resetScore	x	점수 초기화 함수
public	void	drawBoard	x	ScoreBoard 템플릿 그리는 함수
public	void	printScore	x	점수 표시 함수
public	int	getCurrentStage	x	현재 스테이지 반환 함수
public	void	increase	item	점수 증가 함수
public	bool	isCleared	length growth poison gate	전체 Mission 클리어 여부 반환 함수

(1) scoreBoard 2차원 vector에 그릴 ScoreBoard의 형태를 저장한다 Wall은 1로, 공백은 0과 2 로 구성되어있다.

(2) Map의 구현 방식과 동일한 방식이다.

(3) currentMission 1차원 vector에 Mission 클래스로부터 현재 Stage의 Mission을 받아와 저장한다.

(4) currentScore 1차 1차원 vector에 Score 클래스로부터 현재 Score을 받아와 저장한다.

(5) LENGTH, GROWTH, POISON, GATE는 currentMission의 Mission을 파싱하여 양식에 맞게 저장하는 문자열이다.

(6) lenghtScore, growthScore, poisonScore, gateScore는 currentScore의 Score을 파싱하여 각 점수를 저장한다.

(7) lengthProgressBar, growthProgressBar, poisonProgressBar, gatePrograssBar는 현재 미션의 진행상황을 시각적으로 표시한다

- 한계 - 반복문을 통하여 각 진행상황에 따라 + 연산을 통해 ProgressBar에 추가하고자 하였으나 wstring 자료형에서 + 연산을 지원하지 않아 Mission에 맞게 수동으로 조건문을 추가하여 ProgressBar 가 증가하도록 하였다.

(8) 게임의 진행 시간을 계산하기 위해 <ctime> 라이브러리를 사용하였다

- duration에 현재 진행중인 시간이 저장된다
- 계산식 duration = ((double)(CURRENT - TIME) / CLOCKS_PER_SEC)*1000000
- 초당 1 씩 증가한다
- Stage 클리어시 0으로 초기화된다.

(9) bool자료형 legnthCleared, growthCleared, poisonCleared, gateCleared 에 각 미션의 클리어 여부를 저장한다

(10)increase 함수는 item의 번호를 인자로 입력받아 조건문을 통하여 StageManager 클래스의 increaseLengthScore, increaseGrowthScore, increasePoisonScore, increaseGateScore, decreaseLengthScore을 호출한다

(11)resetScore 함수는 StageManager 클래스의 resetScore 함수를 호출한다.

(12)isCleared 함수는 lengthCleared, growthCleared, poisonCleared, gateCleared 를 인자로 받아 네 개의 변수 모두가 true이면 Stage가 클리어 된 것으로 간주하고 StageManager 클래스의 nextStage 함수를 호출한다.

2.3.6 추가 구현 사항

2.3.6.1 Credit.h / Credit.cpp

- 유동현

- 변수

접근자	자료형	변수명	비고
private	vector<vector<int>>	creditMap	시작화면 그리기 위한 2차원 배열
private	wstring	title1 title2 title3 title4 title5 title6	SnakeGame ASCII_ART 문자열
private	string	menu1 menu2 menu3	메뉴 문자열
private	wstring	maker maker_end name1 name2 name3	제작자 문자열
private	int	currentSelection	현재 선택되어 있는 메뉴의 번호 저장
private	wchar_t	wall	시작 화면 그리는 기호

		lud rud	

- Method

접근자	리턴 타입	함수명	인자	비고
public	None	Credit	x	생성자
public	void	init	x	화면 초기화
public	void	draw	x	메인 화면 출력 함수
public	void	showHowToPlay	x	HowToPlay 출력 함수
public	int	showMenu	x	Menu 출력 함수

(1) 2차원 vector creditMap 에 시작화면을 그리기위한 템플릿을 공백을 0 으로, 벽을 0이 아닌 숫자로 표시한다.

(2) title1, title2, title3, title4, title5, title6에 SnakeGame 텍스트를 시각적으로 표시하기 위한 아스키아트를 담고있다

(3) init 함수를 통해 화면을 ncurses 라이브러리로 화면을 출력하기위한 초기화를 한다. Map의 방식과 동일하다.

(4) draw 함수를 통해 템플릿을 화면에 출력한다 Map의 방식과 동일하다

(5) showMenu 함수를 통해 메뉴 3가지 (GameStart, How To Play, Exit) 를 출력한다

(6) How To Play가 선택되었을 경우, showHowToPlay 함수를 호출하여 화면에 게임에 대한 정보를 표시한다.

3 자기평가

20203104 유동현 (ydh91026@kookmin.ac.kr)

- https://github.com/DongHyeonYu/SnakeGame

본인이 맡은 역할

- Mission과 ScoreBoard, Stage에 관련된 전반적인 클래스 설계를 진행하였다.

- Map 클래스, Mission 클래스, Score 클래스, ScoreBoard 클래스, StageManager 클래스, Credit 클래스를 작성하였다.

- Map클래스에서 게임 맵의 설계와 SnakeGame 클래스에서 지도정보를 얻어와 Map을 그리는 로직을 구현하였다.

- Mission 클래스에서 게임 미션의 설계를 진행하였고, ScoreBoard클래스에서 Mission에 대한 정보를 얻어와 ScoreBoard에 표시하는 로직을 구현하였다.

- ScoreBoard 클래스에서 Score 정보를 얻어와 Mission 달성여부를 시각적으로 표시하고, 미션의 클리어 여부를 판단하는 로직, Mission이 클리어 되었다면, 다음 Stage로 넘어가는 로직을 구현하였다.

프로젝트 수행시 어려웠던 점

화면 출력에 있어 문자의 크기와 관련하여 가로의 크기와 세로의 크기가 달랐던 점이다.

이번에 사용하였던 유니코드 특수문자의 경우 가로의 크기와 세로의 크기에 차이가 존재하여 21 * 21 크기의 정사각형 배열임에도 불구하고 Snake가 이동하는 것이 가로로 이동할 때, 세로로 이동할 때 속도가 다른 것 처럼 문제가 존재 했다. 이제 문자를 출력할때, y좌표는 그대로, x좌표에는 *2를 하는 로직을 추가하여 해결 가능하였다.

Stage클리어와, nextStage로 넘어가는 로직을 구현하는데 있어 ncurses 라이브러리의 getch() 함수의 특성에 대해 알 수 있었다

구현 목표를 정할 당시, Stage가 클리어 되면, 사용자의 입력을 받기 전까지 클리어 화면을 출력하고 대기하는 것이 목표였다. 하지만, 개발을 진행하는데 있어 Snake의 이동에 사용되었던 timeout()함수의 설정으로 인하여 입력을 대기하지 않고 바로 넘어가는 문제가 존재하였다. 이에 nodelay() 함수를 통해 필요시 입력을 제한 할 수
있었다.

C++ Language에 대한 지식의 부재로 여러 최적화 이슈들이 존재한다.

초기에는 Map에 대한 정보, ScoreBoard에 대한 정보를 가져와 화면에 출력하는데 있어 단순히 반복문 여러개를 통해 수행하였다. 하지만, 클래스 설계를 진행하고 Snake, Item, Gate의 정보를 Map클래스 자체에 반영하고 SnakeGame 클래스에서는 Map의 정보만을 가지고 지도를 그린다면 한번의 Map을 그리는 로직 만으로도 화면 출력이 가능하였다.

이것 외에도 SnakeGame클래스에서 지도정보를 불러올 때, Map에 있는 21 * 21 크기의 2차원 배열을 Call by Reference 로 받게 하는 등의 최적화를 완벽히 수행하지 못하였다.

느낀점

이번 프로젝트를 진행하는데 있어 크게 느낀 점은 **2 가지**이다.

첫 번째로, 소규모 프로젝트임에도 불구하고 클래스로 구분하여 개발하는데 있어 파일의 갯수가 늘어나는 점이다.

이전까지는 파일의 갯수가 많지 않아 떠오르는대로 주먹구구식으로 개발하였던 경험이 많았던 것이

사실이다. 이번 프로젝트 또한 초기 개발진행 당시에는 제대로된 클래스 설계를 진행하지 않고 개발을 진행하다 보니 개발도중에 클래스의 구조가 변경되거나, 연결되는 객체들의 순서가 변경되는 등 큰 변경이 존재했다.

두 번째로, 단순히 학습만 진행 하는 것 보다, 프로젝트기반으로 학습의 장점이었다.
기본적인 c++ language에 대한 지식이 부족한 상태에서 개발을 시작하였다. 초기에는 무수히 많은 Reference, Document, 클래스 설계 등 개발에 필요한 전반적인 부분에 대해 검색을 통하여 진행하였지만, 개발이 진행되어가며 c++ language에 대한 지식이 쌓이고 코드를 작성하며 프로그램의 실행 흐름을 이해할 수 있었다

추가로 이번 기말고사에서 사용하였던 2차원 vector를 사용하는 법 또한 이번 프로젝트를 경험하였기에 해결 할 수 있었다.

20212972 김민찬 (kmc0487@kookmin.ac.kr)

본인이 맡은 역할
제가 이번 프로젝트에서 제작한 기능들은 Snake와 Item구현, 그리고 draw기능 구현입니다.
Snake 클래스를 사용해 Snake의 전반적인 움직임과 벽에 부딪혔을 때 게임이 종료되는 등의 로직을 구현했습니다.
Item 클래스를 사용해 총 4가지의 Item 기능들을 구현했으며 Item이 랜덤 생성되고 시간이 지나면 삭제되는 로직들을 구현했습니다.
snake와 item 그리고 gate 등을 모두 Map 클래스의 map 변수에 정보를 int형 숫자로 표시해뒀으며 draw 함수에서 이 map 배열을 참조하여 게임을 그려 화면에 표시해줍니다.

프로젝트 수행 시 어려웠던 점
프로젝트 수행 시 젤 어려웠던 부분은 SnakeGame 클래스의 구현입니다. 각 클래스를 제작하는건 그렇게 어렵진 않았지만 SnakeGame클래스에 모두 모아서 기능들을 종합하려할때 크거나 작은 충돌들이 발생했고 이를 의도했던 기능들로 작동하도록 배치도 고치고 로직도 수정했던 부분들이 어려웠던 것 같습니다.
특히, 처음에 draw함수에서 snake와 item 및 Gate를 각 클래스의 위치를 나타내는 변수값을 참조해서 draw기능을 수행했지만 마지막에 map클래스의 map변수에 값을 넣고 map변수만을 이용해서 그리도록 로직을 변경할때 각 클래스들의 내용을 전부 변경해줘야해서 힘들었습니다.

프로젝트 운영에 개선이 필요하다고 생각하는 점
해당 게임은 C++의 OOP를 준수하여 제작되었습니다. 하지만 모든 기능들이 아직 OOP를 제대로 준수하지 않은 것 같고 몇몇 코드들은 C++의 기능들을 제대로 사용하지 못하고 있는 것 같습니다.
후에 OOP 규칙에 맞춰 클래스들을 좀 더 세분화 시키고 팀 내에서 코드 작성 규칙을 정해 코드들을 좀 더 깨끗하게 작성 하도록 고칠 필요가 있을 것 같습니다.

20213074 장종아 (jonga1224@kookmin.ac.kr)

본인이 맡은 역할

이번 프로젝트에서 맡은 역할은 **Gate 구현, Gate와 다른 객체들의 상호작용 부분**입니다.

처음에는 랜덤한 위치에 Gate가 생성되어 draw되는 로직을 구현했고, Rule을 다시 한 번 숙지하여 Gate가 Immune wall 이 아닌 벽에만 생기도록 하였습니다. 다음으로 Snake가 Gate에 진입했을 때, Rule에 따라 방향이 정해지도록 설계하였습니다. Rule에 나온 Gate 진출 시 방향 설정이 경우를 많이 나눠 굉장히 많아보이고 복잡해보이지만, 설계 과정에서 진행방향이 우선이고, 그 이후로는 시계방향 순으로 방향 우선순위를 가진다는 것을 파악하고, 방향 설정 로직을 제작하였습니다. 이어서 Snake의 Gate 진출 좌표 변경 로직을 제작하였습니다.

프로젝트 수행 시 어려웠던 점

Gate 생성 시 두 Gate가 같은 좌표에 생성됨

어려웠던 점이라기엔 그리 어렵진 않지만, 이 버그를 기능 구현 마지막에 알아서 적어봤습니다. 사실 Gate 생성이 난수를 이용해서 무작위하게 생성하는 것이었기 때문에, 같은 곳에 생기는 것을 막지 않아도 같은 곳에 생길 확률이 적습니다. 그래서 초반 로직 구현 때는 이것을 생각하지 못하고 만들었다가 다른 팀원들이 운이 좋게도 이 버그를 발견해주어서 고칠 수 있었습니다.

Gate 진출 시 Snake의 방향 설정

주어진 스네이크 게임의 룰에서 Gate 진출 방향 부분이 제일 길고 복잡해보였습니다. 그래서 그 부분을 설계하는데 어려운 점이 있었습니다. 진출하는 Gate가 형성된 벽이 좌우가 뚫려있는 벽인지, 상하가 뚫려있는 벽인지, 가장자리에 있는 벽인지 등등 경우의 수가 굉장히 많아 보여 어려울 것으로 보였습니다. 하지만 이 부분을 규칙이 있을 것이라고 생각하며 설계한 결과, 위치가 어디든 시계방향 순으로 우선 순위를 가진다는 결론이 나왔고, 성공적으로 구현해낼 수 있었습니다.

개선이 필요한 점

일단 제가 맡은 부분인 Gate에서는 좀 더 추가사항을 도입하고 싶습니다. 여유롭게 준비하지 않아 Rule대로만 구현하고 추가사항은 없어서 아쉬운 부분입니다.
전체적인 코드 부분로 봤을 땐, 구분하기 편하고 보기 좋기 위해 코드를 여러 개로 나누어 만들었지만, 실제로 제가 로직 구현을 하면서도, Gate에 대한 코드를 Gate 관련 파일에만 구현하지 못하고, 다른 파일에도 많이 수정을 하였습니다. 물론 모든 로직을 Gate 파일에만 넣는 것은 거의 불가능하겠지만, 레퍼런스 등을 활용하여 최대한 Gate 기능은 Gate 파일에만 넣어 정리하는 것이 코딩하기에 편할 것 같습니다.

4 참고 문헌

번호	종류	URL	기타
1	기술문서	https://cplusplus.com/	C++ 공식 문서, STL 함수 사용방법 참고
2	웹페이지	https://www.youtube.com/watch?v=LGqsnM_WEK4	SnakeGame의 전반적인 제작 방법 참고
3	기술문서	https://www.gnu.org/software/ncurses/	ncurses 라이브러리 Document

4	웹페이지	https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/	ncurses 함수 참고
5	웹페이지	https://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO	ncurses 함수 참고

5 부록

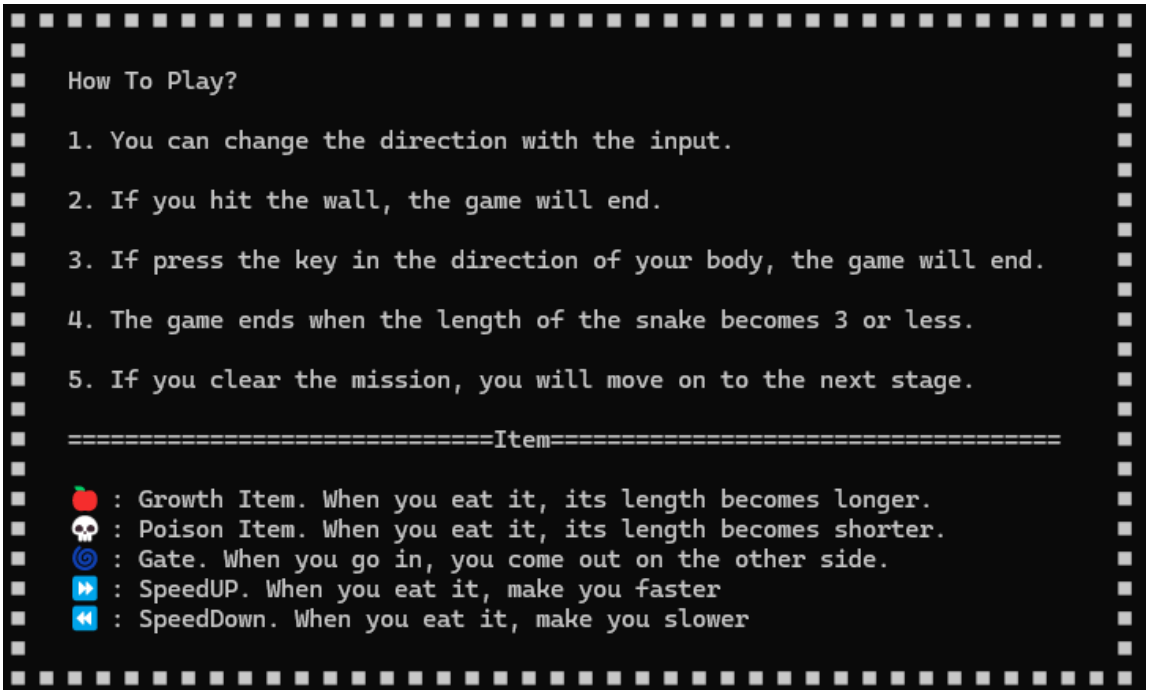
5.1 youtube 시연 영상

https://youtu.be/oFQDgPBdkuY

https://youtu.be/5UV8QhsDFPY

5.2 사용자 메뉴얼

✣ 해당 게임은 Ubuntu 20.04 버전 기준으로 제작되었으며 WSL 및 윈도우나 그 외 환경에서 실행할 시 시연 영상과 똑같이 작동하지 않을 수 있습니다. ✣



기본적인 PLAY 방법은 인게임 내에서 HOW TO PLAY 메뉴얼을 선택시 설명되는 규칙과 똑같습니다

- ↑↓←→ 상하좌우 키보드를 입력해 뱀의 위치를 조종할 수 있습니다.
- 뱀은 기본적으로 매 틱마다 자동으로 한칸씩 앞으로 이동합니다.
- 만약 뱀의 이동 방향과 반대 방향을 입력 시 바로 Game Over가 될 수 있습니다.
- 뱀의 길이가 3 이하로 줄어들면 Game Over가 될 수 있습니다
- Item들의 모양은 위의 사진과 같습니다. 해당 Item들의 모양과 기능들을 숙지하고 플레이하시길 바랍니다.

5.3 설치 방법

```
$ git clone https://github.com/DongHyeonYu/SnakeGame.git
$ sudo apt-get update
$ sudo apt-get install libncurses5-dev libncursesw5-dev
```

5.4 실행 방법

```
$ cd ./SnakeGame/src
$ make
$ ./snake
```