 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

C++프로그래밍 프로젝트

프로젝트 명	Snake Game
팀 명	폰코딩
문서 제목	결과보고서

Version	1.1
Date	2020-JUN-21

팀원	김민정 (팀장)
	김은수

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

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

문서 정보 / 수정 내역


Filename	SnakeGame-폰코딩-결과보고서.doc
원안작성자	김민정, 김은수
수정작업자	김민정, 김은수

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2020-06-20	김은수	1.0	최초 작성	
2020-06-21	김민정	1.1	내용수정	구현내용 추가 및 작성

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

목 차

1	개요	4
1.1	프로젝트 개요	4
1.2	팀 소개	4
1.3	라이브러리 설치방법	4
2	개발 내용 및 결과물	5
2.1	목표	5
2.2	개발 내용 및 결과물	7
2.2.1	개발 내용	7
2.2.2	시스템 구조 및 설계도	31
2.2.3	활용/개발된 기술	35
2.2.4	현실적 제한 요소 및 그 해결 방안	35
2.2.5	결과물 목록	37
3	자기평가	38
4	참고 문헌	39
5	부록	40
5.1	사용자 매뉴얼	40
5.2	실행 방법	41

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

1 개요

1.1 프로젝트 개요

2020년 1학기 C++ 프로그래밍 프로젝트로 진행된 SnakeGame Project입니다.

C++ 언어로 구현하였고 ncurses 라이브러리를 이용하여 터미널에서 text UI를 제공합니다.

1.2 팀 소개

팀장	소프트웨어학부	20191556	김민정
팀원	소프트웨어학부	20191568	김은수

Our GitHub Repository : <https://github.com/minjj0905/kmucs-cpp-snakegame>

개발환경 : Linux Ubuntu 16.04 LTS

Version : C++ 14

1.3 라이브러리 설치방법

Ncurses

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

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

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. Map

NCurses Library 함수들을 사용하여 2차원 배열로 된 Snake Map을 화면으로 표시한다.

2. Snake

- Snake의 초기 길이는 3 이고 초기 방향은 왼쪽이다.
- 사용자가 방향키를 입력하면, Snake의 좌표는 0.4 초마다 진행방향으로 변한다.
- 사용자가 Snake의 방향과 반대 방향 키를 누르면 게임이 종료된다.
- 머리는 짙은 하늘색 원이고, 몸통은 속이 빈 하늘색 원이다.
- 머리가 벽에 닿거나 몸통에 닿으면 게임이 종료된다.
- 머리가 아이템을 먹으면 아이템에 따라 몸이 늘어나거나 줄어든다.
- 머리가 게이트를 통과하면 다른 게이트에 나와야 한다.

3. Item

- 아이템은 약 5 초마다 랜덤으로 생성된다.
- 아이템은 먹지 않으면 랜덤한 시간에 다시 사라진다.
- 아이템은 Snake와 Wall에 겹치지 않도록 생성된다.
- 3 개까지 랜덤으로 생성된다.
- Growth는 초록색이고 Poison은 빨간색이다.

4. Gate

- 게이트는 처음에는 10 초, 사용자가 사용한뒤 10 초로 랜덤으로 생성된다.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

- Snake 가 게이트를 통과하는 동안 게이트는 없어져선 안된다.
- 게이트의 입구가 공유되면 안된다.
- 스네이크의 진행방향에 따라 게이트에서 나오는 스네이크의 방향이 바뀌어야한다.
- 게이트는 한번에 한쌍만 갖는다.
- 게이트는 Immune Wall 에 나타나지 않는다.

5. Score

- Snake 가 게임을 하면서 미션을 수행해야한다.
- 우측에 게임 점수를 표시하는 화면을 구성한다.
- 미션 목록으로는 time, grow item, poison item, gate used 가 있다.
- 목표한 수치에 다 도달했다면 다음 스테이지로 넘어간다.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

2.2 개발 내용 및 결과물

2.2.1 개발 내용

1. Map의 구현

1) Map data

Map은 각 Stage별로 총 4개가 구현되어 있으며, 이 데이터는 각각 level0.txt, level1.txt, level2.txt, level3.txt 파일로 담겨있다. 각 텍스트 파일은 맵의 정보(높이, 너비, 2차원 배열)을 갖고 있다. 맵은 모두 가로 22x21 크기로 구성되어 있으며 맵 좌표값에서 0은 빈 공간, 1과 2는 벽, 3과 4는 Snake를 의미한다.

```

levels > level3.txt
1  21 22
2  2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
3  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
4  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
5  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
6  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
7  1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 2 1 0 0 1
8  1 0 0 0 0 0 0 0 0 1 5 5 1 0 0 0 1 0 0 0 1
9  1 0 0 0 0 0 0 0 0 1 5 5 1 0 0 0 0 0 0 0 1
10 1 0 0 0 0 0 0 0 1 5 5 5 5 1 0 0 0 0 0 0 1
11 1 0 0 0 0 0 0 0 1 5 5 5 5 1 0 0 0 0 0 0 1
12 1 0 0 0 0 0 0 1 5 5 5 5 5 5 1 0 0 0 0 0 1
13 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1
14 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1
15 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1
16 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
17 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
18 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
19 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
20 1 0 0 0 0 0 0 0 0 0 3 4 4 0 0 0 0 0 0 0 1
21 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
22 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2

```

level3.txt에는 값이 5인 좌표가 있는데, 이는 값이 2인 벽과 동일한 의미이다.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

Map – loadMap

```
void Map::loadMap() {
    ifstream in(mapPath);
    in >> mapHeight >> mapWidth;
    map = new int *[mapHeight];
    for (int i = 0; i < mapHeight; i++) {
        map[i] = new int [mapWidth];
        for (int j = 0; j < mapWidth; j++) {
            in >> map[i][j];
        }
    }
}
```


loadMap 함수는 생성자에서 지정된 mapPath를 이용하여 map 정보를 저장한다.

2) Map Visualization

View – drawGameWindow

```
void View::drawGameWindow(Map map) {
    gameWindow = newwin(22, 44, 7, 2);
    wbkgd(gameWindow, COLOR_PAIR('w'));
    watttrn(gameWindow, COLOR_PAIR('w'));

    for(int i=0; i<map.mapHeight; i++) {
        for(int j=0; j<map.mapWidth; j++) {
            int pos = map.getMapValue(i, j);
            if(pos == 0) {
                watttrn(gameWindow, COLOR_PAIR('w'));
                wprintw(gameWindow, " ");
            }
            else if(pos == 1 || pos == 2) {
                watttrn(gameWindow, COLOR_PAIR('w'));
                wprintw(gameWindow, " ");
                mvwprintw(gameWindow, i, 2*j, "■");
            }
            else if(pos == 5) {
                watttrn(gameWindow, COLOR_PAIR('b'));
                wprintw(gameWindow, " ");
                mvwprintw(gameWindow, i, 2*j, "■");
            }
        }
    }
}
```


 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

View의 drawGameWindow 함수는 Map 정보를 인자로 넘겨받아 맵을 시각화 한다. 이때 ubuntu 터미널 특성상 글자가 1:1로 출력되지 않기 때문에 정사각형 모양을 출력되도록 그려지는 text를 조절하였다.



맵의 벽(1, 2) 는 흰색으로 그려지고, 아무것도 없는 빈 공간인 0은 검은색으로 채워지게 된다.

마지막 스테이지인 level3의 경우 5번은 gate가 생성될 수 없는 2와 같은 벽이지만 게임 플레이상 혼란을 막기위해 회색으로 처리하였다.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

2. Snake 표현 및 조작

1) Snake 생성 및 위치 지정

Snake – Snake

```
Snake::Snake(Map& m) {
    direction = '1';
    maxLength = 3;
    curlLength = 3;
    growCount = 0;
    poisonCount = 0;
    gateCount = 0;
    fail = false;
    tailPos.push_back(POSITION());
    for (int i = 0; i < m.mapHeight; i++) {
        for (int j = 0; j < m.mapWidth; j++) {
            if (m.map[i][j] == 3) {
                snakePos.push_back(POSITION(i, j));
                m.map[i][j] = 0;
            } else if (m.map[i][j] == 4) {
                snakePos.push_back(POSITION(i, j));
                m.map[i][j] = 0;
            }
        }
    }
}
```

Snake의 처음 위치를 맵에서 받아와서 벡터에 저장한다. Snake 객체의 안에는 Snake 머리 좌표, 꼬리 좌표가 있고, 방향이 있으며 몸의 길이가 있다.

2) Snake 조작 및 이동

Snake – setDirection

```
void Snake::setDirection() {
    int KeyPressed = getch();
    switch(KeyPressed) {
        case KEY_LEFT:
            if (direction != 'r') {
                direction = 'l';
                break;
            } else {
                setFailed();
                break;
            }
        case KEY_RIGHT:
```



```
        if (direction != 'l') {
            direction = 'r';
            break;
        } else {
            setFailed();
            break;
        }
    case KEY_UP:
        if (direction != 'd') {
            direction = 'u';
            break;
        } else {
            setFailed();
            break;
        }
    case KEY_DOWN:
        if (direction != 'u') {
            direction = 'd';
            break;
        } else {
            setFailed();
            break;
        }
    }
}
```

Snake – moveSnake

```
void Snake::moveSnake() {
    if (direction == 'l') {
        snakePos.insert(snakePos.begin(), POSITION(snakePos[0].y,
snakePos[0].x - 1));
        tailPos.pop_back();
        tailPos.push_back(POSITION(snakePos[snakePos.size()-1].y,
snakePos[snakePos.size()-1].x));
        snakePos.pop_back();
    } else if (direction == 'r') {
        snakePos.insert(snakePos.begin(), POSITION(snakePos[0].y,
snakePos[0].x + 1));
        tailPos.pop_back();
        tailPos.push_back(POSITION(snakePos[snakePos.size()-1].y,
snakePos[snakePos.size()-1].x));
        snakePos.pop_back();
    } else if (direction == 'u') {
        snakePos.insert(snakePos.begin(), POSITION(snakePos[0].y - 1,
snakePos[0].x));
        tailPos.pop_back();
        tailPos.push_back(POSITION(snakePos[snakePos.size()-1].y,
snakePos[snakePos.size()-1].x));
        snakePos.pop_back();
    }
}
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```

    } else if (direction == 'd') {
        snakePos.insert(snakePos.begin(), POSITION(snakePos[0].y + 1,
snakePos[0].x));
        tailPos.pop_back();
        tailPos.push_back(POSITION(snakePos[snakePos.size()-1].y,
snakePos[snakePos.size()-1].x));
        snakePos.pop_back();
    }

    // 스네이크 fail 체크
    checkCorrectPos();
    checkLength();
}

```

Snake는 ncurses 라이브러리의 getch() 함수를 통해서 키를 받아 방향을 바꾼다. 이동 방식은 방향에 따라서 벡터에 저장되어있는 스네이크의 좌표를 갱신시키는 것이다. Snake의 방향과 반대방향키를 누른다면 setFailed()를 실행시켜서 실패가 됐다고 알린다.

3) Snake Visualization

View – draw

```

void View::draw(Map map, Snake snake) {
    drawMainWindow();
    drawGameWindow(map);
    drawSnake(snake);
    drawScoreWindow(snake);
    drawBorder();
    update();
}

```

draw 함수는 전체 게임 화면을 한번에 그려내는 함수이다. 이 함수는 앞으로 개발 단계가 업데이트 될 때마다 각각의 요소를 그려내는 함수가 추가 된다. 이번 단계에서는 drawSnake가, 다음 단계에서는 drawItem이 추가되는 식이다.

View - drawSnake

```

void View::drawSnake(Snake snake) {
    std::vector<POSITION> snakepos = snake.getPosition();

    watttron(gameWindow, COLOR_PAIR(2));
}

```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```

//head
wprintw(gameWindow, " ");
mvwprintw(gameWindow, snakepos[0].y, 2*snakepos[0].x, "●");

//body
for(int i=1; i<snake.getLength(); i++) {
    POSITION pos = snakepos[i];
    wprintw(gameWindow, " ");
    mvwprintw(gameWindow, pos.y, 2*pos.x, "○");
}
}

```

Snake는 gamewindow 내에서 그려야 하기 때문에 ncurses의 window 관련 함수들을 사용하였다. wattron 함수를 이용하여 Snake가 bright cyan 색상으로 그려질 수 있도록 설정하고, wprintw와 mvwprintw 함수를 이용하여 Snake의 좌표에 Unicode를 출력하였다.

4) 틱에 의한 이동 구현

틱에 의한 이동은 한 스테이지 플레이를 담당하는 함수인 Game의 runlevel 함수에서 구현되었다. 시간에 따른 이동을 구현해야 하기 때문에 Timer 기능도 구현하였다.

Timer

```

#include <time.h>
#include <math.h>
#include "Timer.h"
#include <iostream>
using namespace std;

void Timer::startTimer(){
    startTime = clock();
}

void Timer::updateTime() {
    currentTime = clock();
    tick = (double)(currentTime - startTime)/CLOCKS_PER_SEC;
}

unsigned int Timer::getPlayTime() {
    return tick;
}

double Timer::getTick() {
    return tick;
}

```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```
}

```

다음 코드를 통해 Timer 시작 및 측정, 리셋 기능을 구현하였다.

Game – runlevel

```
void Game::runLevel() {
    timer.startTimer();
    int tick = 0;
    int count = 0;

    curMap = level.getCurrentMap();
    curSnake = Snake(curMap);
    view.draw(curMap, curSnake);

    while(!isGameOver()) {
        timer.updateTime();
        double a = timer.getTick();
        curSnake.setDirection();
        if(a > 0.4) {
            curSnake.moveSnake();
            view.draw(curMap, curSnake);
            timer.startTimer();
        }
    }
}
```

앞에서 구현한 Snake의 setDirection 함수를 이용하여 사용자 입력으로 방향을 지정할 수 있도록 하였고, Timer의 초가 0.4초가 될 때마다 한 틱이 실행되도록 하였다. 한 틱이 실행될 때 moveSnake 함수를 이용하여 Snake의 좌표값을 이동한 후 view의 draw 함수를 이용하여 map과 snake를 그리고 tick을 초기화한다. 만약 게임을 실패하였다면 while문을 멈추도록 구현하였다.

5) 실패 판단 – 반대방향 입력, 충돌 확인

Game – isCollision

```
bool Game::isCollision() {
    POSITION headpos = curSnake.getPosition()[0];
    int posvalue = curMap.getMapValue(headpos.y, headpos.x);
    if((posvalue == 1) || (posvalue == 2)) return true;

    return false;
}
```

 <div> 국민대학교 소프트웨어학부 C++ 프로그래밍 </div>	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```
}

```

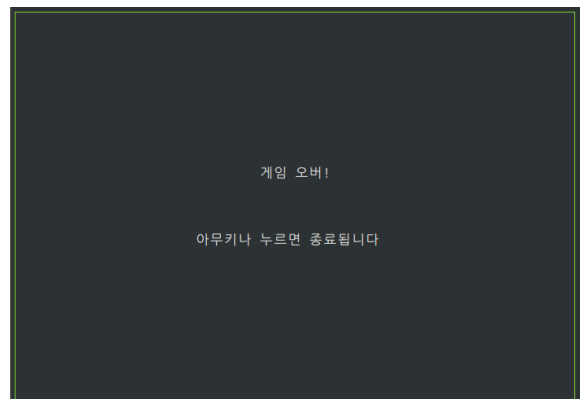
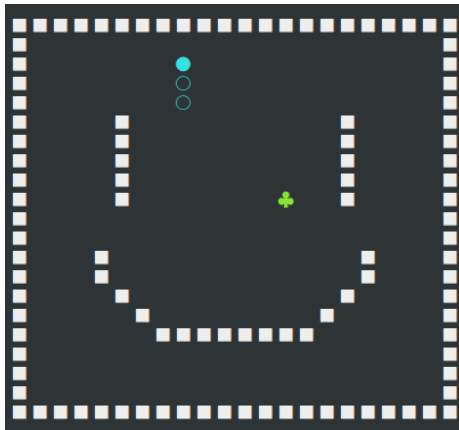
Snake의 좌표와 map의 좌표 value 값을 비교하여 벽인지 판단하여 충돌을 확인한다.

Game – isGameOver

```
bool Game::isGameOver() {
    //Snake 키보드 입력 체크
    if(curSnake.isFailed()) return true;
    if(isCollision()) return true;

    return false;
}
```

User가 이동방향을 반대로 입력하였는지 확인하고, Snake가 벽에 충돌하였는지 확인하여 Gameover를 판단한다.



3. Item 요소의 구현

1) Item 좌표 지정 로직

Item - setItemPos

```
void Item::setItemPos(Map m) {
    srand((unsigned int)time(0));
    int x = (rand() % (m.mapWidth - 1)) + 1;
    int y = (rand() % (m.mapHeight - 1)) + 1;
```



```
if (m.map[y][x] == 0) {
    if (rand() % 2 == 0) {
        pos = POSITION(y, x);
        itemType = 0;
    } else {
        pos = POSITION(y, x);
        itemType = 1;
    }
} else {
    while (m.map[y][x] != 0) {
        x = (rand() % (m.mapWidth - 1)) + 1;
        y = (rand() % (m.mapHeight - 1)) + 1;
        if (m.map[y][x] == 0) {
            if (rand() % 2 == 0) {
                pos = POSITION(y, x);
                itemType = 0;
            } else {
                pos = POSITION(y, x);
                itemType = 1;
            }
        }
    }
}
```

아이템의 좌표를 지정하기 위해 맵을 받아오고 랜덤으로 맵의 좌표를 선택한다.
비어있는 공간이면 랜덤으로 Grow일지 Poison일지 정하고 좌표를 저장한다.

2) Item Visualization

View – drawItem

```
void View::drawItem(std::vector<Item> item) {
    for(int i=0; i<item.size(); i++) {
        POSITION itempos = item[i].getItemPos();
        if(item[i].isGrowItem()) {
            watttron(gameWindow, COLOR_PAIR('g'));
            mvwprintw(gameWindow, itempos.y, 2*itempos.x, "■");
        } else {
            watttron(gameWindow, COLOR_PAIR('r'));
            mvwprintw(gameWindow, itempos.y, 2*itempos.x, "■");
        }
    }
}
```

Item은 gameWindow 내에서 그려진다. Item 벡터를 받아 해당 좌표에 Unicode

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

를 출력한다. 이때 Grow일 경우 bright green 색상을, poison일 경우 bright red 색상으로 출력한다.

3) Item 이용 구현

Game – isGetItem

```
bool Game::isGetItem() {
    POSITION headpos = curSnake.getPosition()[0];
    for(int i=0; i<item.size(); i++) {
        POSITION itempos = item[i].getItemPos();
        if(headpos.x == itempos.x && headpos.y == itempos.y) {
            if(item[i].isGrowItem()) {
                curSnake.eatGrowItem();
            }else {
                curSnake.eatPoisonItem();
            }
            item.erase(item.begin()+i);
            return true;
        }
    }
    return false;
}
```

Snake와 Item의 좌표를 비교하여 획득을 판단한다. 만약 좌표가 같다면 Item이 Grow인지 Poison인지 판별하여 알맞은 함수를 실행시키고, 획득한 Item은 vector에서 삭제한다. 이후 Item을 획득하였는지 여부를 반환해준다.


4) Item 생성, 삭제, 기능구현

Game – runlevel

```
void Game::runLevel() {
    ...
    while(!isGameOver()) {
        ...
        curSnake.setDirection();

        if(tick > 0.4) {
            curSnake.moveSnake();

            if(!isGetItem()) {
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

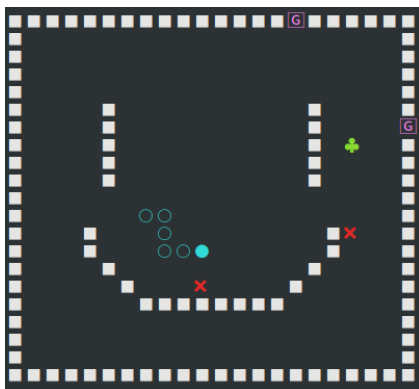
```

        if((erasetime > rand() % 5 + 10) && !item.empty()) {
            item.erase(item.begin());
            eraseTimer.startTimer();
        }
    }
    view.draw(curMap, curSnake, item);
    tickTimer.startTimer();
}

if(itemtime > rand() % 3 + 4) {
    if(item.size() < 3) {
        Item newItem = Item(curMap, curSnake);
        for(int i=0; i<item.size(); i++) {
            if(item[i] == newItem) newItem = Item(curMap,
                                                    curSnake);
        }
        item.push_back(newItem);
    }
    itemTimer.startTimer();
}
}
view.drawGameOver();
}

```

Item이 생성되고 삭제되는 기능은 runlevel함수에 추가로 구현하였다. Item은 약 4~6초 사이에 랜덤한 위치에 생성되고 만약 획득하지 못했을 경우 약 14초 이내로 사라지게 된다. 아이템은 생성된 순서대로 삭제되며 아이템의 획득 기능을 위하여 isGetItem 함수를 사용하였다.



5) 실패 판단 – Snake 길이

Snake – checkLength()

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```
void Snake::checkLength() {
    if (getLength() < 3) setFailed();
}
```

Snake의 길이를 확인하여 3 이하면 fail로 설정되도록 하였다.

4. Gate 요소의 구현

1) Gate 좌표 지정 로직

Gate - setGatePos

```
void Gate::setGatePos(Map m) {
    srand((unsigned int)time(0));
    int x1 = rand() % m.mapWidth;
    int y1 = rand() % m.mapHeight;
    int x2 = rand() % m.mapWidth;
    int y2 = rand() % m.mapHeight;

    if (m.map[y1][x1] == 1) {
        pos1 = POSITION(y1, x1);
    } else {
        while (m.map[y1][x1] != 1) {
            x1 = rand() % m.mapWidth;
            y1 = rand() % m.mapHeight;
            pos1 = POSITION(y1, x1);
        }
    }
    if ((m.map[y2][x2] == 1) && (x1 != x2 || y1 != y2) &&
        (preventShare(x1, y1, x2, y2))) {
        pos2 = POSITION(y2, x2);
    } else {
        while ((m.map[y2][x2] != 1) || (x1 == x2 && y1 == y2) ||
            (!preventShare(x1, y1, x2, y2))) {
            x2 = rand() % m.mapWidth;
            y2 = rand() % m.mapHeight;
            pos2 = POSITION(y2, x2);
        }
    }
    setUdlr(m);
}
```

Gate - preventShare

```
bool Gate::preventShare(int x1, int y1, int x2, int y2) {
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```

int dir[8][2] = { {1, 0}, {1, 1}, {0, 1}, {-1, 1}, {-1, 0}, {-1, -1}, {0, -1}, {1, -1} };
for (int i = 0; i < 8; i++) {
    if ((x2 == x1 + dir[i][1]) && (y2 == y1 + dir[i][0])) {
        return false;
    }
}
return true;
}

```

게이트의 좌표를 지정하기 위해서 맵을 받아오고, 게이트는 한쌍 생겨야 하므로 각각의 게이트들의 좌표를 지정해줘야한다. 그리고 게이트들은 서로 같은 위치에 생겨선 안되고 입구도 공유하면 안된다.

2) Gate Visualization

View – drawGate

```

void View::drawGate(std::vector<Gate> gate) {
    wattron(gameWindow, COLOR_PAIR('p'));
    if(!gate.empty()) {
        POSITION gatepos1 = gate[0].getGatePos(1);
        POSITION gatepos2 = gate[0].getGatePos(2);
        mvwprintw(gameWindow, gatepos1.y, 2*gatepos1.x, "■");
        mvwprintw(gameWindow, gatepos2.y, 2*gatepos2.x, "■");
    }
}

```

Gate는 gameWindow 내에서 그려진다. Gate 벡터를 받아 Gate가 벡터 내에 존재한다면 해당 좌표에 Unicode를 출력한다. Gate는 보라색 계열의 색상으로 그려지도록 설정했다.

3) Gate 이용 구현

Game – isOnGate

```

bool Game::isOnGate() {
    // 게이트 체크해서 방향 바꾸고 pos 위치 변경
    if(gate.empty()) return false;
    std::vector<POSITION> snakepos = curSnake.getPosition();
    Gate curGate = gate[0];
    //나갈 방향 순서 u=0 d=1 l=2 r=3
    int updirSequence[4] = {0, 3, 2, 1};
}

```



```
int downdirSequence[4] = {1, 2, 3, 0};
int leftdirSequence[4] = {2, 0, 1, 3};
int righkdirSequence[4] = {3, 1, 0, 2};

int* dirSequence[4] = {updirSequence, downdirSequence,
                        leftdirSequence, righkdirSequence};
char udlr[4] = {'u', 'd', 'l', 'r'};

//머리좌표 바꾸기
if(snakepos[0] == gate[0].getGatePos(1)) {
    snakepos[0] = gate[0].getGatePos(2);
    char dir = curSnake.getDirection();
    int diridx;
    //dir 인덱스 지정
    for(int i=0; i<sizeof(udlr); i++) {
        if (dir == udlr[i]) {
            diridx = i;
            break;
        }
    }
    int* squence = dirSequence[diridx];

    char newdir;
    for (int i=0; i<4; i++) {
        if(curGate.getUdlr(2, squence[i]) == 0) {
            newdir = udlr[squence[i]];
            break;
        }
    }
    curSnake.makeDirectionThis(newdir);
    curSnake.setHeadPos(snakepos[0]);
    curSnake.usingGate(snakepos[0]);
}
else if(snakepos[0] == gate[0].getGatePos(2)) {
    snakepos[0] = gate[0].getGatePos(1);
    char dir = curSnake.getDirection();
    int diridx;
    //dir 인덱스 지정
    for(int i=0; i<sizeof(udlr); i++) {
        if (dir == udlr[i]) {
            diridx = i;
            break;
        }
    }
    int* squence = dirSequence[diridx];

    char newdir;
    for (int i=0; i<4; i++) {
        if(curGate.getUdlr(1, squence[i]) == 0) {
            newdir = udlr[squence[i]];
            break;
        }
    }
}
```



```
    }
    curSnake.makeDirectionThis(newdir);
    curSnake.setHeadPos(snakepos[0]);
    curSnake.usingGate(snakepos[0]);
}

for(int i=0; i<curSnake.getLength(); i++) {
    if (snakepos[i] == curSnake.gatepos) {
        if (i==curSnake.getLength()-1) {
            curSnake.outGate();
            gate.erase(gate.begin());
        }
        return true;
    }
}
return false;
}
```

Snake가 Gate 좌표에 있는지를 판별하여 만약 Gate에 진입하였다면 반대면 Gate의 위치로 좌표를 지정한다. 이후 상황에 맞추어 Snake의 진행 방향을 설정하고, Snake가 Gate를 이용하고 있음을 Snake의 변수에 저장한다. 이후 Snake의 Gate이용이 끝났다면 Snake가 Gate에서 나왔음을 저장한 후 게이트를 삭제한다.

4) Gate 생성

Game - runlevel

```
void Game::runLevel() {

    ...

    while(play) {

        ...

        curSnake.setDirection();

        // 스네이크 움직이면서 아이템, 게이트 판단 및 그리기
        if(tick > 0.4) {
            curSnake.moveSnake();

            if(!isGetItem()) {

                ...

            }
            if(isOnGate()) {
```



```

        gateTimer.startTimer();
    }
    play = !isGameOver();
    view.draw(curMap, curSnake, item, gate);
    tickTimer.startTimer();
}

...

//게이트 생성
if(gatetime > 10) {
    if(gate.size() < 1) {
        Gate newgate = Gate(curMap);
        gate.push_back(newgate);
    }
    gateTimer.startTimer();
}
}
}

```

Gate는 Item과 같은 방식으로 일정 시간에 따라 생성되도록 하였다. Gate는 한번에 하나만 생성되며 이용하기 전까지는 삭제되지 않는다. 게임 시작 후 10초에 한번씩 생성된다.

5) 충돌 판단 제외

Game - isCollision

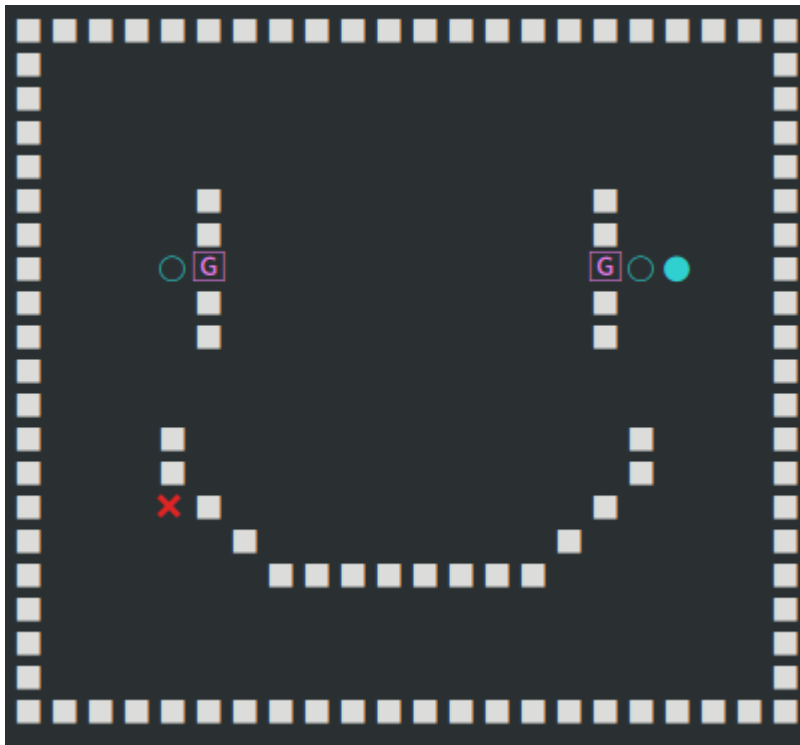
```

bool Game::isCollision() {
    POSITION headpos = curSnake.getPosition()[0];
    int posvalue = curMap.getMapValue(headpos.y, headpos.x);
    if((posvalue == 1) || (posvalue == 2)) {
        if(!gate.empty()) {
            if(headpos == gate[0].getGatePos(1) || headpos ==
                gate[0].getGatePos(2)) {
                return false;
            }
        }
        return true;
    }
    return false;
}

```

Gate가 벽으로 인식되지 않기 위해서 Gate의 좌표는 충돌판단에서 제외하고 판단 할 수 있도록 수정하였다.

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21



5. 점수 요소의 구현

1) Mission data

Mission은 Stage별 txt 파일로 저장되어 있으며, txt파일 내에는 순서대로 목표 시간(초), Grow Item개수, Poison Item 개수, Gate 이용 횟수가 들어있다.

<<사진>>

Mission - loadMission

```
void Mission::loadMission() {
    ifstream in(missionPath);
    for(int i=0; i<4; i++) {
        in >> goal[i];
    }
}
```

Mission data를 map과 같은 방식으로 불러와 저장한다.

2) Mission 생성

Level - createMap

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```
void Level::createMap() {
    for (int i = 0; i < 4; i++) {
        maps[i] = Map(i);
        missions[i] = Mission(i);
        clear[i] = false;
    }
}
```

level에서 map과 mission 데이터를 생성하여 관리할 수 있도록 구현하였다.

3) Mission clear 판단

Game – isLevelClear()

```
bool Game::isLevelClear() {
    Mission curmission = level.getMission();
    if(playtime < curmission.getGoalScore()) return false;
    if(curSnake.getGrowCount() < curmission.getGoalGrow()) return
false;
    if(curSnake.getPoisonCount() < curmission.getGoalPoison()) return
false;
    if(curSnake.getGateCount() < curmission.getGoalGate()) return
false;
    return true;
}
```

해당 Stage가 clear되었는지 mission data와 대조하여 판별한다.

4) ScoreBoard Visualization

Game – isLevelClear()

```
void View::drawScoreWindow(Snake snake, Mission mission, int time) {
    scoreWindow = newwin(21, 30, 7, 48);
    wbkgd(scoreWindow, COLOR_PAIR('s'));
    watttrn(scoreWindow, COLOR_PAIR('s'));
    for(int i=0; i<15; i++) {
        wprintw(scoreWindow, " ");
        mvwprintw(scoreWindow, 0, 2*i, " ");
    }

    //현재 값
    mvwprintw(scoreWindow, 2, 9, "Score Board");

    mvwprintw(scoreWindow, 4, 7, "T I M E : ");
    mvwprintw(scoreWindow, 4, 21, std::to_string(time).c_str());
    mvwprintw(scoreWindow, 5, 7, "Grow Item : ");
    mvwprintw(scoreWindow, 5, 21,
std::to_string(snake.getGrowCount()).c_str());
}
```



```
mvwprintw(scoreWindow, 6, 7, "Poison Item : ");
mvwprintw(scoreWindow, 6, 21,
std::to_string(snake.getPoisonCount()).c_str());
mvwprintw(scoreWindow, 7, 7, "Gate Used : ");
mvwprintw(scoreWindow, 7, 21,
std::to_string(snake.getGateCount()).c_str());

//구분선
for(int i=0; i<15; i++) {
    wprintw(scoreWindow, " ");
    mvwprintw(scoreWindow, 10, 2*i, "▣");
}

//목표값
mvwprintw(scoreWindow, 12, 11, "Mission");

mvwprintw(scoreWindow, 14, 7, "T I M E : ");
mvwprintw(scoreWindow, 14, 21,
std::to_string(mission.getGoalScore()).c_str());
if(mission.getGoalScore() <= time) {
    mvwprintw(scoreWindow, 14, 26, "✓");
}
mvwprintw(scoreWindow, 15, 7, "Grow Item : ");
mvwprintw(scoreWindow, 15, 21,
std::to_string(mission.getGoalGrow()).c_str());
if(mission.getGoalGrow() <= snake.getGrowCount()) {
    mvwprintw(scoreWindow, 15, 26, "✓");
}
mvwprintw(scoreWindow, 16, 7, "Poison Item : ");
mvwprintw(scoreWindow, 16, 21,
std::to_string(mission.getGoalPoison()).c_str());
if(mission.getGoalPoison() <= snake.getPoisonCount()) {
    mvwprintw(scoreWindow, 16, 26, "✓");
}
mvwprintw(scoreWindow, 17, 7, "Gate Used : ");
mvwprintw(scoreWindow, 17, 21,
std::to_string(mission.getGoalGate()).c_str());
if(mission.getGoalGate() <= snake.getGateCount()) {
    mvwprintw(scoreWindow, 17, 26, "✓");
}

//구분선
for(int i=0; i<15; i++) {
    wprintw(scoreWindow, " ");
    mvwprintw(scoreWindow, 20, 2*i, "▣");
}

}
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

해당 스테이지에 알맞은 Score Board를 출력하고, 각 조건이 완성될 때마다 체크 기호를 출력하여 표시한다.

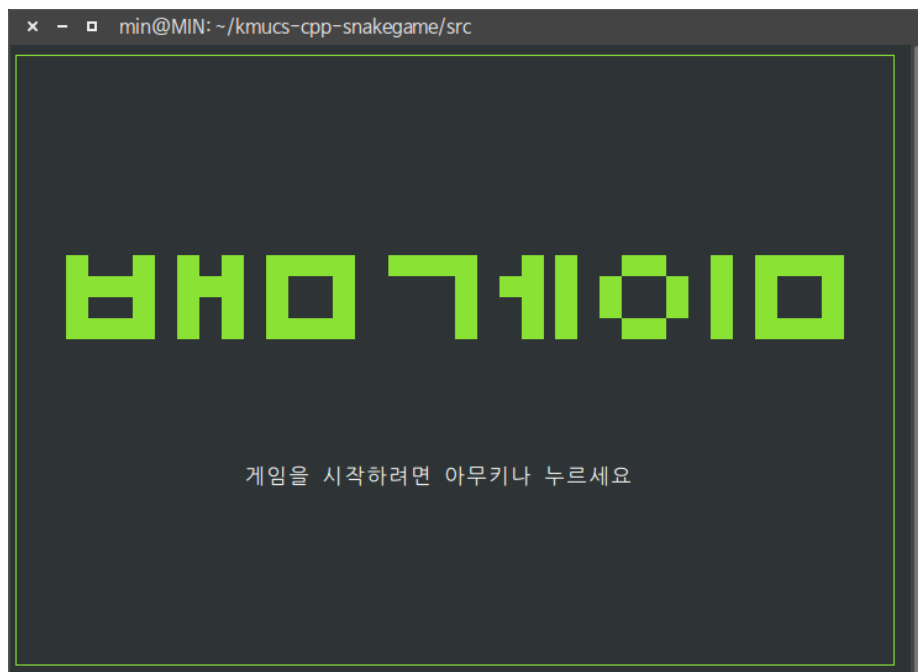
<p>★★★★★★★★★★★★★★★★</p> <p>Score Board</p> <p>T I M E : 14</p> <p>Grow Item : 1</p> <p>Poison Item : 0</p> <p>Gate Used : 1</p> <p>★★★★★★★★★★★★★★★★</p> <p>Mission</p> <p>T I M E : 30</p> <p>Grow Item : 3</p> <p>Poison Item : 1</p> <p>Gate Used : 2</p> <p>★★★★★★★★★★★★★★★★</p>	<p>★★★★★★★★★★★★★★★★</p> <p>Score Board</p> <p>T I M E : 36</p> <p>Grow Item : 3</p> <p>Poison Item : 0</p> <p>Gate Used : 0</p> <p>★★★★★★★★★★★★★★★★</p> <p>Mission</p> <p>T I M E : 30 ✓</p> <p>Grow Item : 3 ✓</p> <p>Poison Item : 1</p> <p>Gate Used : 2</p> <p>★★★★★★★★★★★★★★★★</p>
---	---

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

6. 전체 게임 로직

시작화면

View – drawStartScreen



Game – runGame

```
void Game::runGame() {
    clear();
    while(!isGameClear()) {
        clear();
        runLevel();
        if(level.getClear()) {
            if(level.getCurrentLevel() == 3) {
                break;
            }
            level.upCurrentLevel();
            view.drawNextStage();
        }
        else{
            view.drawGameOver();
            break;
        }
    }
    if(isGameClear()) {
```

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

```

        view.drawGameClear();
    }
}

```

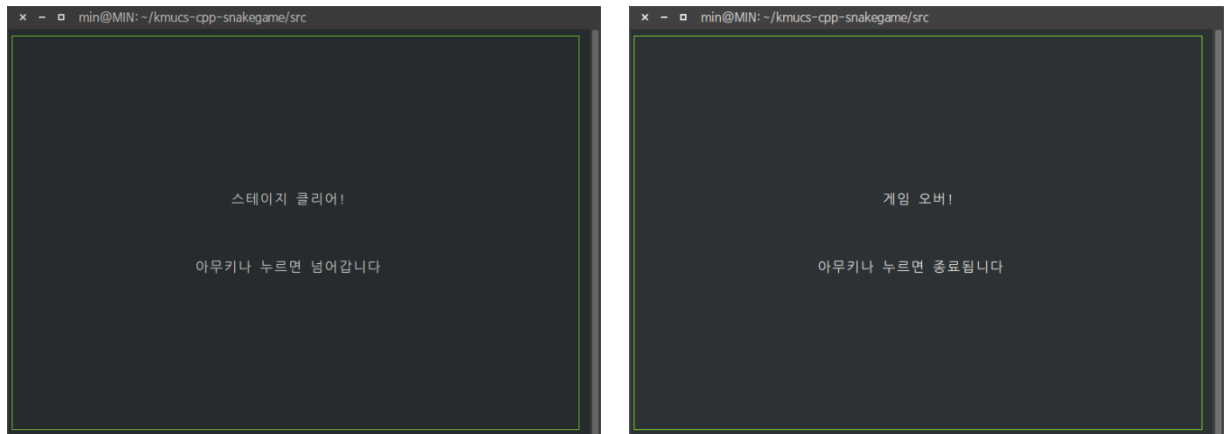
게임 전체가 클리어되기 전까지 각 레벨이 플레이 될 수 있도록 한다. 만약 한 레벨이 끝난 후 게임오버된 상태면 게임오버 화면을 출력하고 게임을 종료한다. 레벨을 클리어 했을 경우 다음 스테이지로 넘어간다. 게임이 정상적으로 클리어 되면 클리어 화면을 출력하고 게임을 종료한다.

게임화면

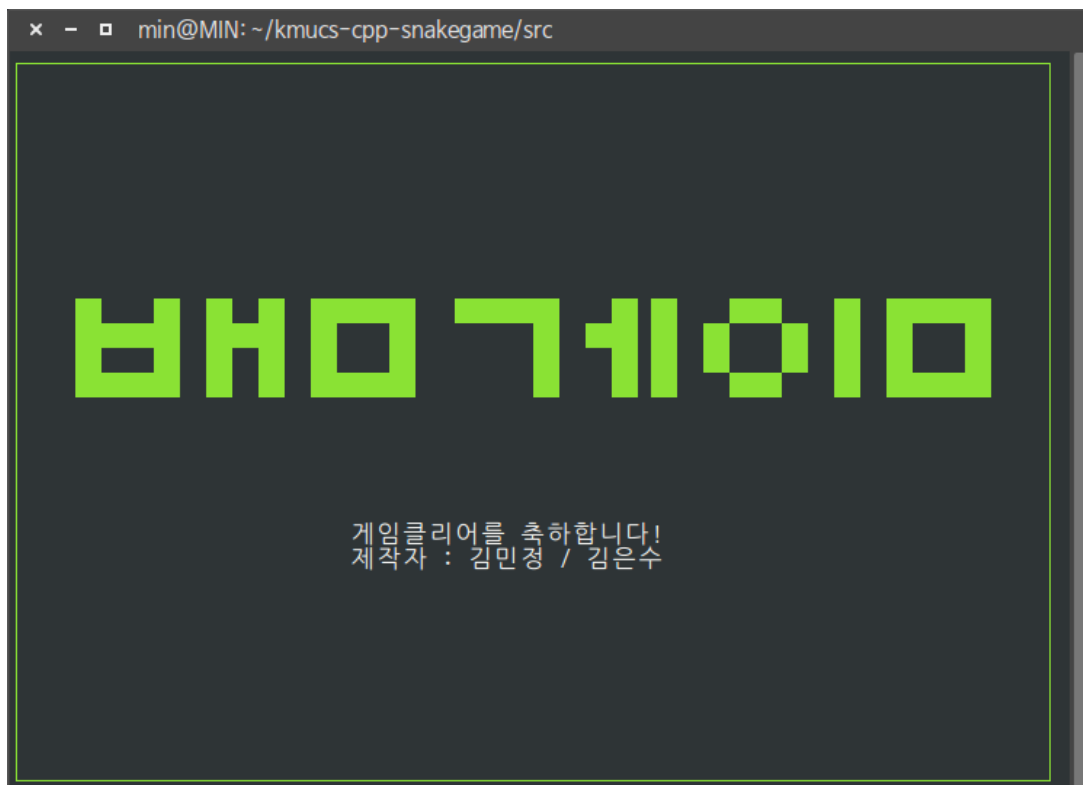


 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

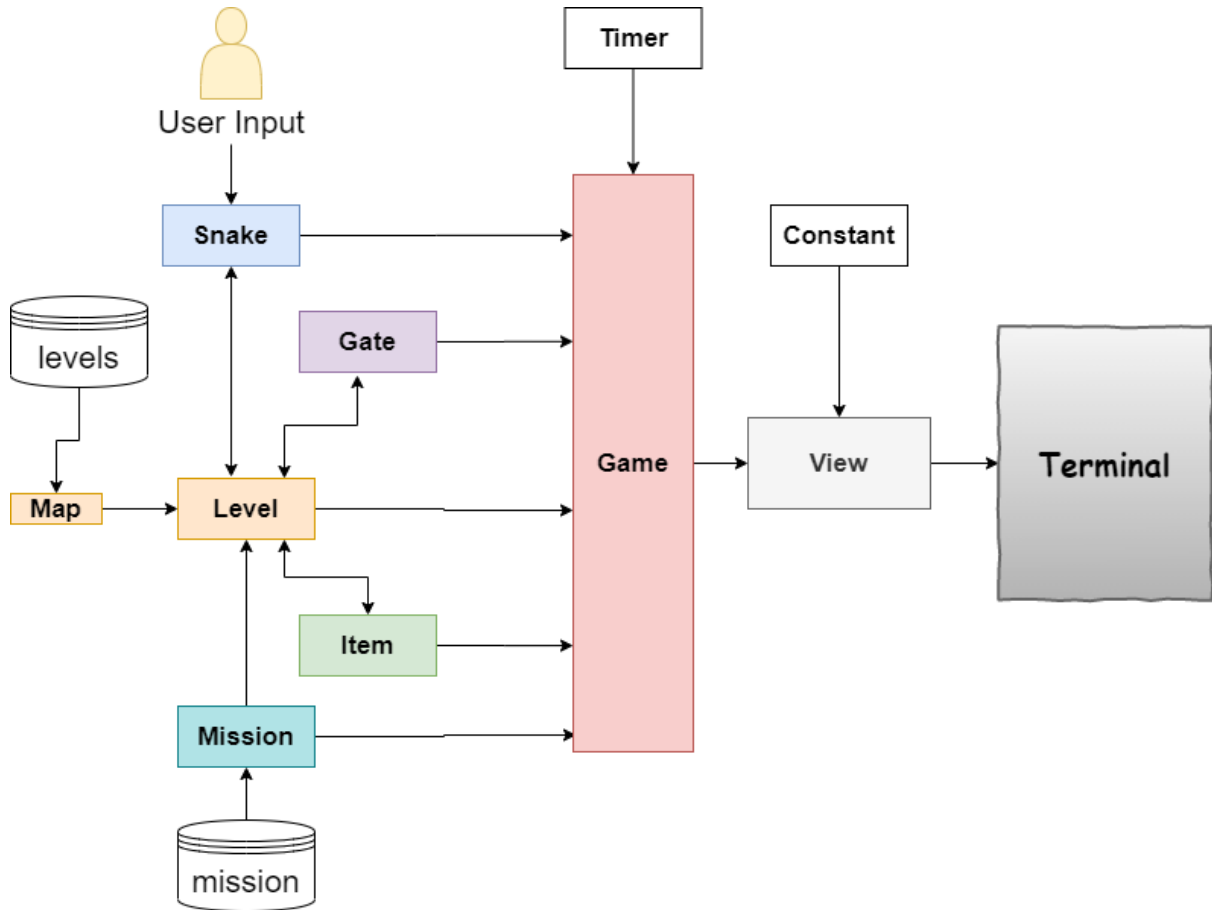
View – drawGameOver / drawNextStage



View – drawClear



2.2.2 시스템 구조 및 설계도



Game

함수명	인자	리턴타입	설명	제작자
Game			생성자	김민정
newGame		Void	새 게임 시작	김민정
runGame		Void	게임 실행	김민정
runLevel		Void	각 스테이지(레벨) 실행	김민정
isGameOver		Bool	게임이 종료되었는지 판단	김민정
isCollision		Bool	Snake 가 벽과 충돌했는지 판단	김민정
isGetItem()		bool	Item 획득 판별	김민정
isOnGate()		bool	Gate 진입 판별	김민정

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

isGameClear()		bool	Game Clear 판별	김민정
isLevelClear()		bool	Level Clear 판별	김민정

Gate

함수명	인자	리턴타입	설명	제작자
Gate			생성자	김민정
setGatePos	Map m		Gate 위치 지정	김민정
setUdlr	Map m		Gate 위치 상하좌우 값 저장	김민정
preventShare	int x1, int y1, int x2, int y2	bool	같은 진입좌표	김민정

Item

함수명	인자	리턴타입	설명	제작자
setItemPos	Map m	Void	아이템의 좌표값 설정	김은수
missSnakePos	Snake s	Bool	아이템의 좌표랑 스네이크의 좌표랑 같은지 확인	김은수

Level

함수명	인자	리턴타입	설명	제작자
Level			생성자 currentLevel 을 0 으로 초기화	김은수
getCurrentLevel		Int	현재 레벨을 가져온다	김은수
upCurrentLevel		Void	레벨을 1 올린다	김은수
getCurrentMap		Map	레벨에 따른 현재 맵을 가져온다	김은수
createMap		Void	Map 객체 array 에 맵을 넣는다	김은수

Map

함수명	인자	리턴타입	설명	제작자
Map	Int stage		생성자	김은수

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

loadMap		Void	맵 불러오기	김은수
---------	--	------	--------	-----

Mission

함수명	인자	리턴타입	설명	제작자
loadMission		Void	미션 데이터 가져와서 저장	김민정

Snake

함수명	인자	리턴타입	설명	제작자
Snake	Map& m		생성자	김은수
setMaxLength		Void	Snake 의 maxLength 갱신	김은수
isFailed		Bool	Snake 의 생사 여부 반환	김은수
setDirection		Void	키를 받고 snake 가 방향전환함	김은수
moveSnake		Void	Snake 를 움직임	김은수
checkCorrectPos		Void	스네이크 head 위치가 올바른지 확인	김은수
checkLength		Void	스네이크 길이가 3 이상인지 확인	김은수
getPosition		Vector<POSITION>	Snake Position	김은수

Snake 하위 Struct

POSITION

함수명	인자	리턴타입	설명	제작자
POSITION			기본 생성자, x = 0, y = 0 으로 초기화	김은수
POSITION	Int y, int x		생성자, x 와 y 를 인자로 받은 값으로 초기화	김은수

Timer

함수명	인자	리턴타입	설명	제작자
-----	----	------	----	-----

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

Timer			생성자	김민정
~Timer			소멸자	김민정
startTimer		Void	Timer 시작	김민정
updateTime		Void	현재 시간을 업데이트	김민정
getPlayTime		Unsigned int	실행된 시간을 정수로 반환 - 화면 표시용	김민정
getTick		Double	실행된 시간을 실수로 반환 - 계산용	김민정

View

함수명	인자	리턴타입	설명	제작자
View			생성자	김민정
drawStartScreen		Void	게임 시작 화면	김민정
drawGameOver		Void	게임 오버 화면	김민정
drawGameClear		Void	게임 클리어 화면	김민정
draw	Map map	Void	메인 플레이 화면을 그림	김민정
drawMainWindow		Void	MainWindow 그리기	김민정
drawGameWindow	Map map	Void	gameWindow 그리기	김민정
drawSnake	Snake snake	Void	화면에 Snake 그리기	김민정
drawScoreWindow	Snake snake, Mission mission, int time	Void	화면에 Scoreboard 그리기	김민정
drawBorder		Void	Window Border	김민정
update		Void	View 를 업데이트	김민정

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

2.2.3 활용/개발된 기술

<cstdlib>와 <ctime>를 이용하여 게이트와 아이템이 랜덤으로 좌표를 가질 수 있게 만들었다. srand()의 시드값을 <ctime> 으로 해결하였다.

<vector>를 이용하여 Snake 의 좌표 등 각종 좌표값을 저장하는 용도로 사용하였다. <vector>를 사용하여 Snake 의 몸이 길어졌다가 줄어들고 벡터의 값을 바꾸는 것으로 스네이크가 움직이는 걸 구현했다.

<fstream>를 이용하여 map 과 mission data 를 파일입출력으로 불러와 저장한다.

<string>를 이용하여 view 에서 점수값을 표시한다.

<time.h>를 이용하여 시간 계산 기능을 구현하였다.

<locale>를 이용하여 유니코드 작성이 가능하도록 하였다.

<wchar.h>를 이용하여 wchar_t 자료형을 사용하도록 하였다.

<ncurses>

ncurses에 게임 로직을 이용하여 얻은 데이터를 바탕으로 snake, map, item, gate, mission 등의 정보를 출력하고 User Interface 를 구현하였다.

2.2.4 현실적 제한 요소 및 그 해결 방안

1. 개발환경

처음에 개발을 시작했을 때 WSL에서 구현하려고 했으나, 우분투 터미널에서는 View가 깨지는 현상이 발생하였다. 결국 개발환경의 통일이 필요했다고 생각하여 wsl 사용을 중지하고 ubuntu로 개발환경을 통합하였다.

2. Gate 진입로 공유 문제

Gate가 생성될 때 두 Gate의 진입로가 같아 이용할 수 없는 Gate가 생성되는 문제가 있었다. Gate는 이용하기 전까지 삭제되지 않도록 했기 때문에 수정이 필요했다. Gate가 생성될 때 우리의 맵 파일에 한해서는 입구를 공유할 수 없도록 수

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

정했다.

3. Snake 중복 키 입력 이슈

Snake에서 방향 전환과 이동을 함께 구현했을 때, getch 함수의 입력값이 쌓여 User가 입력한 방향으로 전환되지 않았다. 키 입력과 이동을 분리하면서 해결할 수 있었다.

4. Item 중복 생성

Item이 같은 좌표 혹은 Snake위에 생기지 않도록 처리를 하였으나, 매우 극히 낮은 확률로 문제 있는 좌표에 생길 가능성이 있다. 완벽하게 차단을 위해서는 올바른 좌표에 생성될때까지 while문을 반복해야 하는데, 그러면 시간복잡도의 부담이 커지기 때문에 게임이 정상적으로 구동되지 않을 가능성이 있어 일정 반복이 지나면 생성되지 않도록 처리하였다.

5. 터미널 크기 조정

ncurses에서 resize_term이라는 함수가 있기는 하지만 실제 터미널 창의 크기를 바꿔주는 것은 아니다 보니, 터미널창이 작으면 제대로 실행되지 않는 문제가 있었다. linux에서 터미널 사이즈를 받고 일정 크기 이하면 실행되지 않도록 처리하려고 했으나, 시간이 모자란 관계로 구현하지 못했다. 프로젝트 제출 이후 방학에 다시 한번 수정하여 구현해보고 싶다

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

2.2.5 결과물 목록

SnakeGame-폰코딩-결과보고서.doc

levels

level0.txt, level1.txt, level2.txt, level3.txt

mission

mission0.txt, mission1.txt, mission2.txt, mission3.txt

src

Constants.h

Game.cpp Game.h

Gate.cpp Gate.h

Item.cpp Item.h

Level.cpp Level.h

Makefile

Map.cpp Map.h

Mission.cpp Mission.h

Snake.cpp Snake.h

Timer.cpp Timer.h

View.cpp View.h

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

3 자기평가

김민정 : 사실 게임 제작은 이번이 처음이 아니기 때문에 잘 만들수 있을 것이라고 생각했다. 그리하여 게임 메인 로직 구성과, Visualization 부분을 맡겠다고 했다. 1학년 2학기에 pygame 라이브러리를 사용하여 2d 탑다운 게임을 개발했었기 때문에 어느정도 게임 로직에 대한 자신감이 있었다. 그러나 ncurses라는 생소한 라이브러리를 사용하고, 이번 학기에 처음 다루게 된 C++로 구현을 하려다 보니 공부해야하는것도 많았고 같은 문제를 몇일씩 잡고 해결하려고 노력하기도 했다. 특히나 이번 프로젝트에서 view를 구성하던 중에, wsl과 우분투와의 개발환경 차이 때문에 코드를 전체수정해야 하기도 했다. 게다가 ubuntu 터미널의 글자비가 정방형이 아니어서, 맵과 글자들을 정방형으로 출력하는 방법도 몇일동안 공부하여 겨우 완성해냈다. 그만큼 어려웠으나 나에게 있어 공부도 많이 되었던 시간인 것 같다.

내가 잘 할 수 있다고 생각하는 분야의 프로젝트를 벌써 두번째 진행해보니, 앞으로도 내가 하고 싶어하는 일이 무엇인지 깨달았다. 이번에 개발한 게임을 발판 삼아 앞으로도 소규모 게임 개발 프로젝트를 진행해보며 게임제작 분야에 대해 더 공부해보고 싶다. ncurses가 아니라 unity나 unreal같은 엔진도 사용해보고 싶은 마음도 들었다.

이번 프로젝트에서 제일 아쉽게 느껴지는 것이 있다면 클래스 구조 설계이다. 사실 처음 계획할때만 해도 이렇게 난잡하게 꼬여있지 않았었는데, 게임의 볼륨이 커지고 로직이 늘어날수록 시간에 쫓기며 작업을 하다보니 코드가 이곳저곳 꼬이기 시작했다. 2학년 1학기나 되어서 구조설계가 엉망인 것은 공부해서 고쳐야 할 점이라고 생각한다.

여러모로 아쉬움도 남고 뿌듯함도 남은 프로젝트였지만 아직 나는 이 프로젝트가 끝났다고 생각하지 않는다. 방학동안 좀 더 다듬어서 자랑할 수 있을만한 나의 Github Repo로 남기고 싶다.


김은수 : 나의 역할은 게임 로직에 필요한 요소들을 만드는 역할이었다. 즉 게임이 돌아갈 때 필요한 것들 예를 들어 스네이크, 아이템 등을 만드는 역할이었다. 메인은 조원이 만들었는데 이제 메인이 필요한 것들을 만들어주는 역할이어서 보조적인 역할을 수행했다. 처음 프로젝트를 받았을 때 이걸 뭐 어떻게 하라는거지 생각하며 막막했다. 제일 처음 맵부터 짜야하고 이걸 어떤식으로 써야 할지 모르겠어서 계속 상의하고 초반에는 매우 이해를 하기 힘들었었다. 게다가 ncurses 라이브러리도 처음 써보는 것이어서 어려운 점이 많았다. 사실 우리가 수업에서 배웠던 씨뽕뽕만 가지고는 만들기

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

어려웠다. 다행히 필요한 것들을 만들어 낼 수 있어서 다행이었고 어떤식으로 게임이 흘러가는지 알 수 있었던 것 같다. 시간이 생긴다면 조금 더 보완해서 좀 더 깔끔하게 짜보고 싶어졌다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	웹 페이지	Unicode-table	https://unicode-table.com/en/sets/check/			
2	웹 페이지	Programming in Color with ncurses	https://www.linuxjournal.com/content/programming-color-ncurses	2018.2.6	Jim Hall	
3	웹 페이지	The UNIX and Linux Forums	https://www.unix.com/programming/253796-ncurses-colors-2.html			
4	웹 페이지	Ncurses 프로그래밍	https://psman2.tistory.com/entry/ncurses-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D	2003.3.10	윤 상배	
5	웹 페이지	NCURSES 란 ?	https://m.blog.naver.com/PostView.nhn?blogId=forc1&logNo=50001923420&proxyReferer=https:%2F%2Fwww.google.com%2F	2006.2.19	벤치	
6	웹 페이지	Ncurses Programming Guide	http://www.cs.ukzn.ac.za/~hug/hm/os/notes/ncurses.html			
7	웹 페이지	NCURSES Programming HOWTO	https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/	2005.6.20		
8	웹 페이지	Writing Programs with NCURSES	https://invisible-island.net/ncurses/ncurses-intro.html			

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

5 부록

5.1 사용자 매뉴얼

:: 주의사항 ::

터미널 창이 작다면 게임이 정상적으로 실행되지 않습니다.
충분히 창 사이즈를 키우고 실행해 주세요.

ncurses 라이브러리를 설치한 후 진행해주세요.

:: 게임 설명 ::

방향키 : ↑ ↓ ← →

■ : 벽

● : Snake 머리

○ : Snake 몸통

♣ : Grow 아이템 (획득시 길이+1)

✕ : Poison 아이템 (획득시 길이-1)

Ⓜ : Gate (진입시 반대 Gate로 진출)

:: 게임 설명 ::

게임은 총 4 스테이지로 구성되어 있습니다.

스테이지의 미션을 모두 클리어하시면 다음 스테이지로 넘어갑니다.

마지막 4스테이지를 완료하시면 게임이 종료됩니다.

gameover

■ 에 충돌시

반대 방향 방향키 입력시

○(몸통)에 ●(머리) 충돌시

Snake의 길이가 3이하로 줄어들 시

 국민대학교 소프트웨어학부 C++ 프로그래밍	결과보고서		
	프로젝트 명	SnakeGame	
	팀 명	폰코딩	
	Confidential Restricted	Version 1.2	2020-JUN-21

5.2 실행 방법

Github : <https://github.com/minjj0905/kmucs-cpp-snakegame>

1. Git clone
2. src directory 로 이동
3. Command : make
4. Command : ./Snake