**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Snake Game* |
| 팀 명 | *나 좀 잘하조* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.2 |
| **Date** | 01 |

|  |  |
| --- | --- |
| **팀원** | 황재일 (팀장) |
|  |
|  |
|  |
|  |

|  |
| --- |
| **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 “나 좀 잘하조”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “나 좀 잘하조”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-SnakeGame.doc |
| **원안작성자** | 황재일 |
| **수정작업자** | 황재일 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2020-06-01 | 황재일 | 1.0 | 최초 작성 | 5단계까지의 전체 구현 내용 작성 |
| 2020-06-01 | 황재일 | 1.1 | 프로그램 수정 | 게이트의 생성 좌표의 사방이 벽으로 둘러쌓여 있으면 좌표를 다시 잡도록 수정.  Grow와 Poison의 생성 확률 비율을 1:1에서 3:2로 수정 |
| 2020-06-01 | 황재일 | 1.2 | 프로그램 설명, 실행화면 사진 | 프로그램 설명과 실행화면 사진 추가 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_Toc73478490)

[2 개발 내용 및 결과물 5](#_Toc73478491)

[2.1 목표 5](#_Toc73478492)

[2.2 개발 내용 및 결과물 6](#_Toc73478493)

[2.2.1 개발 내용 6](#_Toc73478494)

[2.2.2 시스템 구조 및 설계도 15](#_Toc73478495)

[2.2.3 활용/개발된 기술 16](#_Toc73478496)

[2.2.4 현실적 제한 요소 및 그 해결 방안 16](#_Toc73478497)

[2.2.5 결과물 목록 17](#_Toc73478498)

[2.2.6 코드 설명 18](#_Toc73478499)

[3 자기평가 34](#_Toc73478500)

[4 참고 문헌 34](#_Toc73478501)

[5 부록 35](#_Toc73478502)

[5.1 사용자 매뉴얼 35](#_Toc73478503)

[5.2 설치 방법 35](#_Toc73478504)

# 개요

|  |
| --- |
| **평가기준 (10점)**  **프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.**  **또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.** |

C++의 GUI 라이브러리인 ncurses를 이용하여 Snake Game을 구현하였다.

이 외에 내부 기능들을 구현하기 위해서, 시간을 나타내는 chrono, 랜덤 함수를 사용할 수 있는 random, usleep을 사용하기 위한 unistd.h, 맵 파일을 읽어오기 위한 fstream, 데이터를 저장해두기 위한 vector 등의 라이브러리를 추가로 사용하였다.

모든 라이브러리들은 std내에 포함되어 있기 때문에, g++환경만 갖추어져 있다면 사용할 수 있었으며, ncurses는 강의에서 제공된 pdf파일을 통하여 설치할 수 있었다.

$ sudo apt-get update

$ sudo apt-get install libcurses5-dev libcursesw5-dev

위의 명령어를 통하여 ncurses라이브러리를 설치할 수 있다.

# 개발 내용 및 결과물

## 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 |

## 개발 내용 및 결과물

### 개발 내용

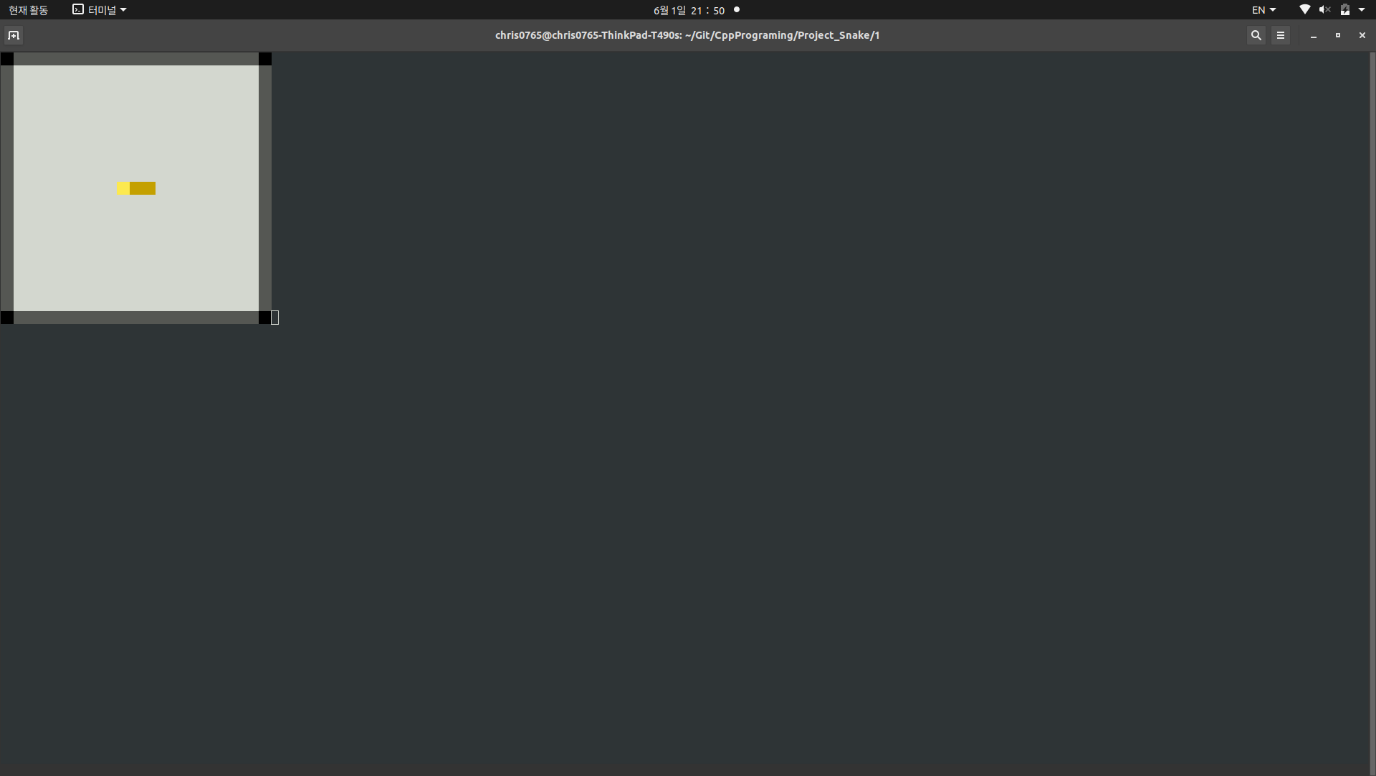
|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.** |

**-1단계 구현**

fstream을 이용하여 map1.txt에 담겨있는 맵에 대한 정보를 vector<vector<int>> 형식으로 불러오고, ncurses를 통해 해당 내용을 출력하는 과정을 구현하였다.

출력은 문자가 아닌 background 컬러를 통하여 출력하여 네모와 네모 사이에 공백이 없도록 하였다. (출력 문자는 “ “ <- 띄어쓰기 2번)

0은 빈 공간, 1은 Wall, 2는 Immune Wall, 3은 Snake Head, 4는 Snake Body, 5는 Grow Item, 6은 Poison Item, 7은 Gate를 표시하도록 COLOR\_PAIR를 설정하였으며, 맵에서는 0, 1, 2로 벽을 표시하고, Snake가 출발할 위치를 표시하기 위해 3, 4를 추가로 지정하였으며, 나머지 값들은 이후 프로그램에서 삽입하도록 구성하였다.



**-2단계 구현**

Snake에 대한 정보들을 통제하는 Snake 객체를 만들었다.

또한 1단계에서 한 부분에 제작했던 map을 불러오는 과정을 새롭게 Map 객체를 생성하여 손쉬운 관리가 가능하게끔 만들었으며, ncurses의 설정에 대한 부분들은 init()함수를 통하여 실행하도록 변경하였다.

Map 객체를 통해 map 파일 데이터를 불러오고, Snake 부분을 따로 추출하여 Snake 객체에 넘겨준다.

getch()를 통하여 방향키를 입력받아 Snake의 진행 방향을 설정하고, 0.5초마다 설정된 진행 방향으로 Snake가 이동하도록 만들었다.

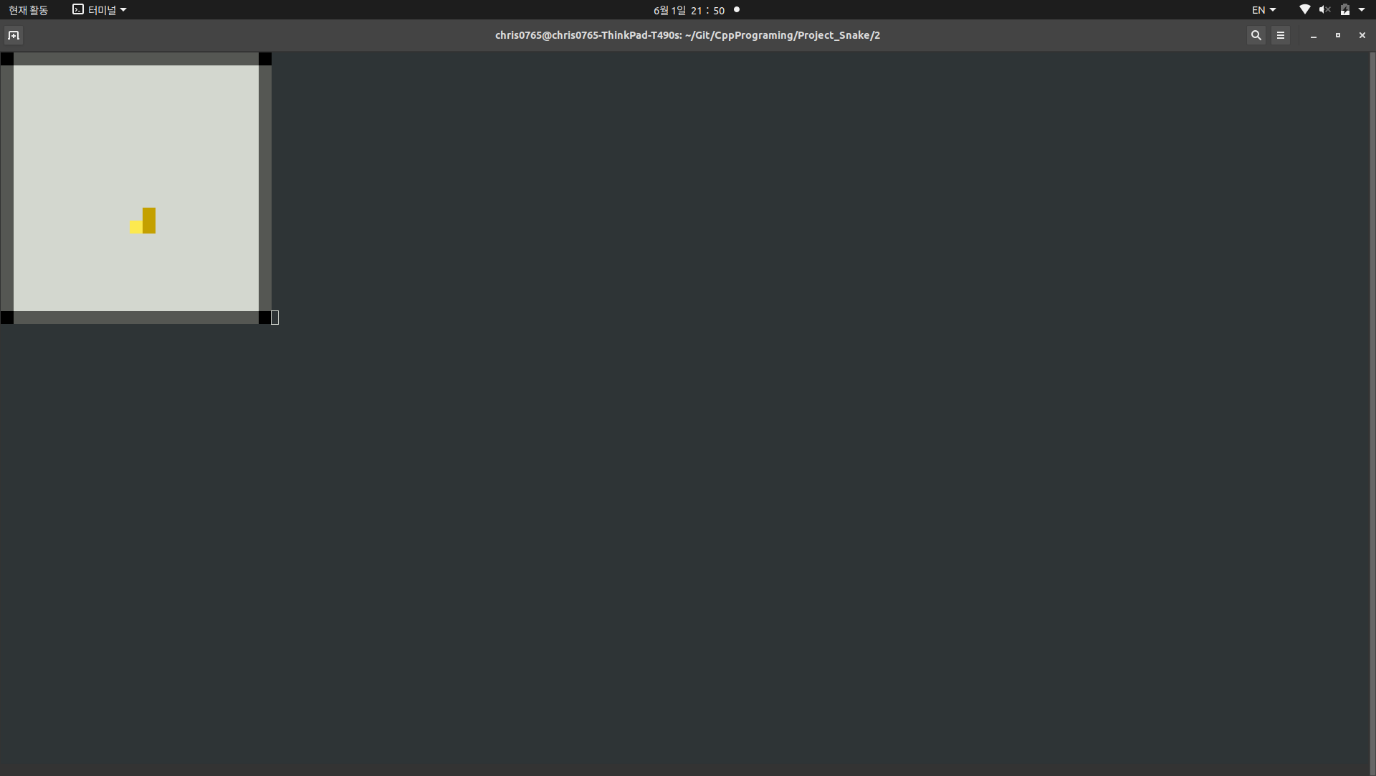
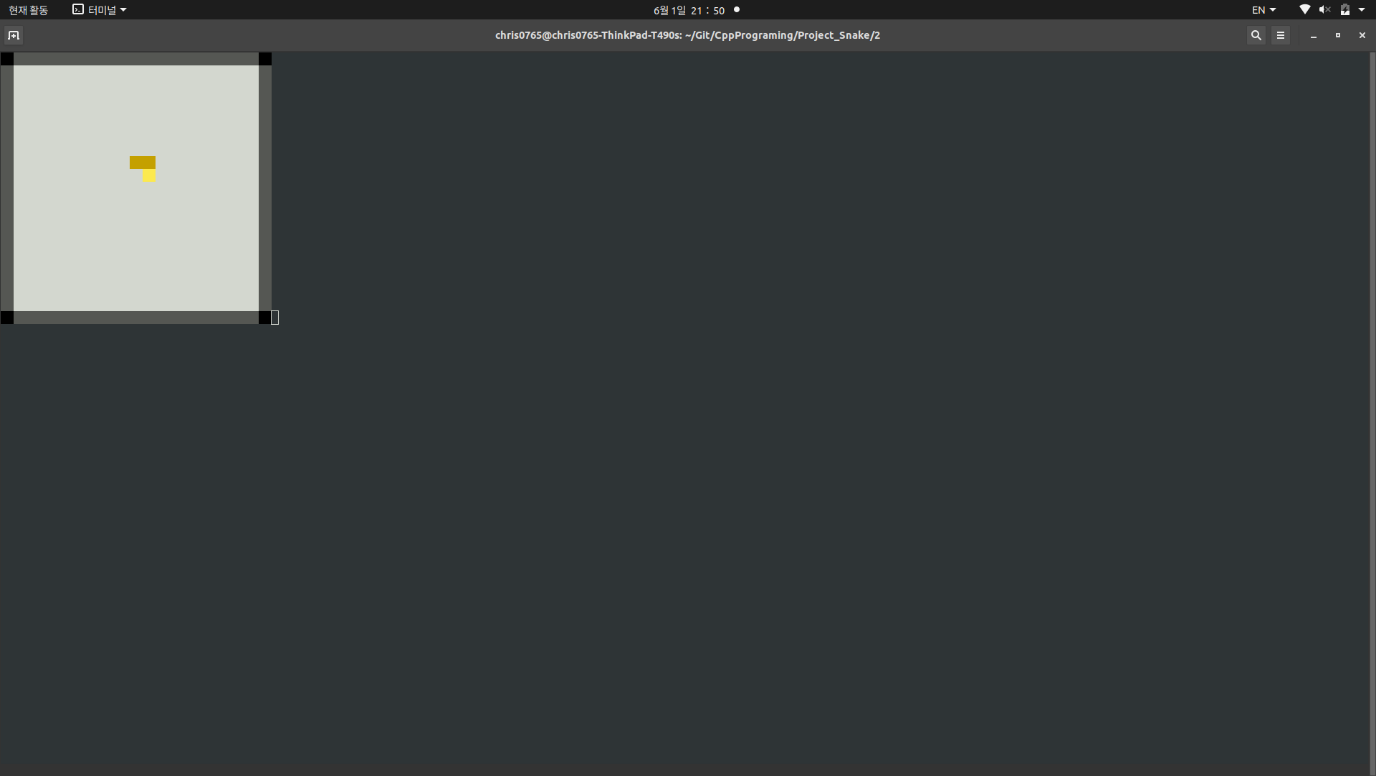
Snake 내에 vector<Pos>형식의 BODY를 통해 [0]에 Head, 나머지에 Body를 저장하도록 하였으며, 이를 통해 다음 머리의 좌표를 계산하여 맨 앞에 insert하고, 맨 뒤의 값을 erase하는 방법으로 Snake의 이동을 구현하였다. (Pos는 정수형 x, y를 저장하는, 좌표를 나타내는 클래스이다.)

또한, getch()를 하는 동안 프로그램이 멈추면 안되므로, nodelay()설정을 하였다.

Snake가 벽이나 자신의 Body에 충돌한다면, 게임 오버 상태를 나타내는 isDie를 통해 게임 오버 여부를 나타낸다.

GameWindow라는 객체를 생성하여 최종 게임 화면을 출력하도록 하였다.

Map 객체에서 맵을, Snake 객체에서 BODY를 넘겨받아 두 정보를 합친 printMap을 생성하고, 이를 화면에 출력하도록 하였다.



**-3단계 구현**

Item에 대한 정보들을 통제하는 Item 객체를 만들었다.

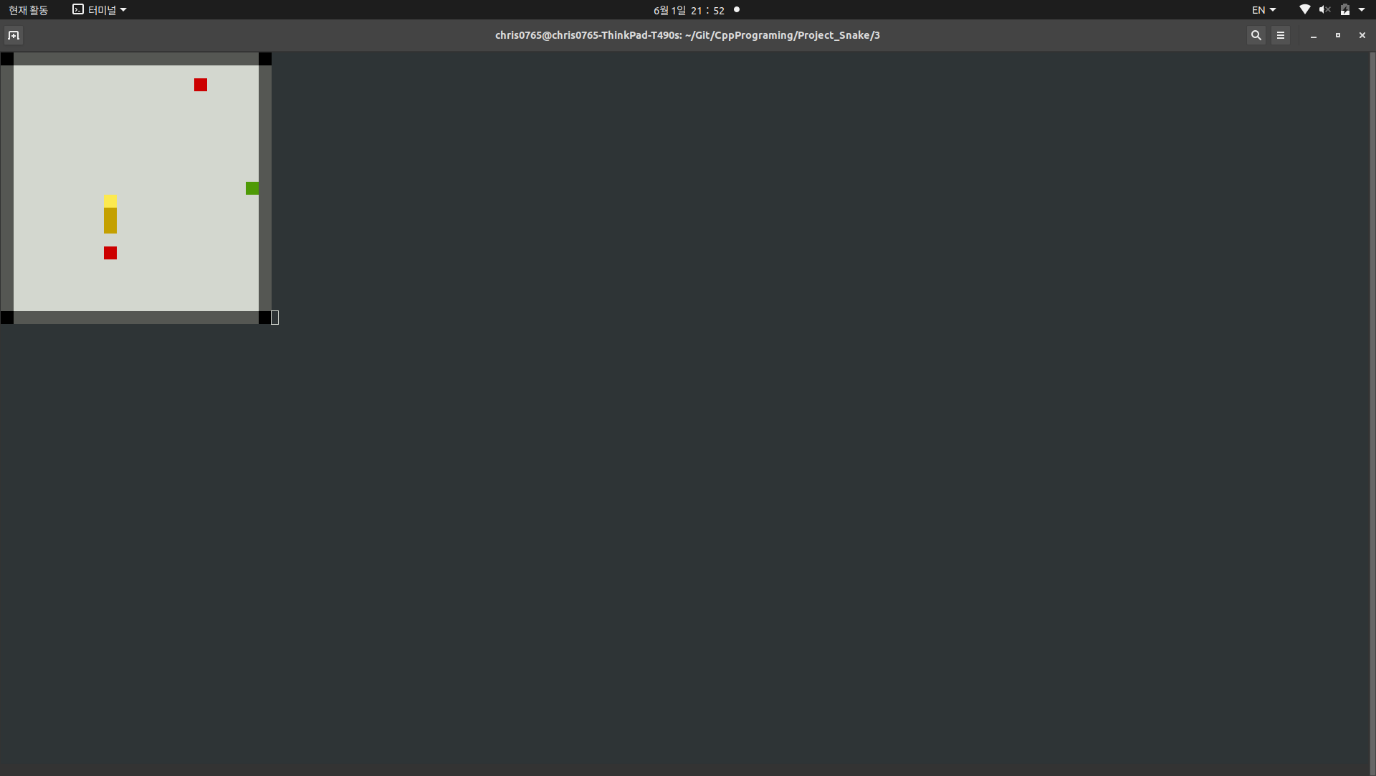
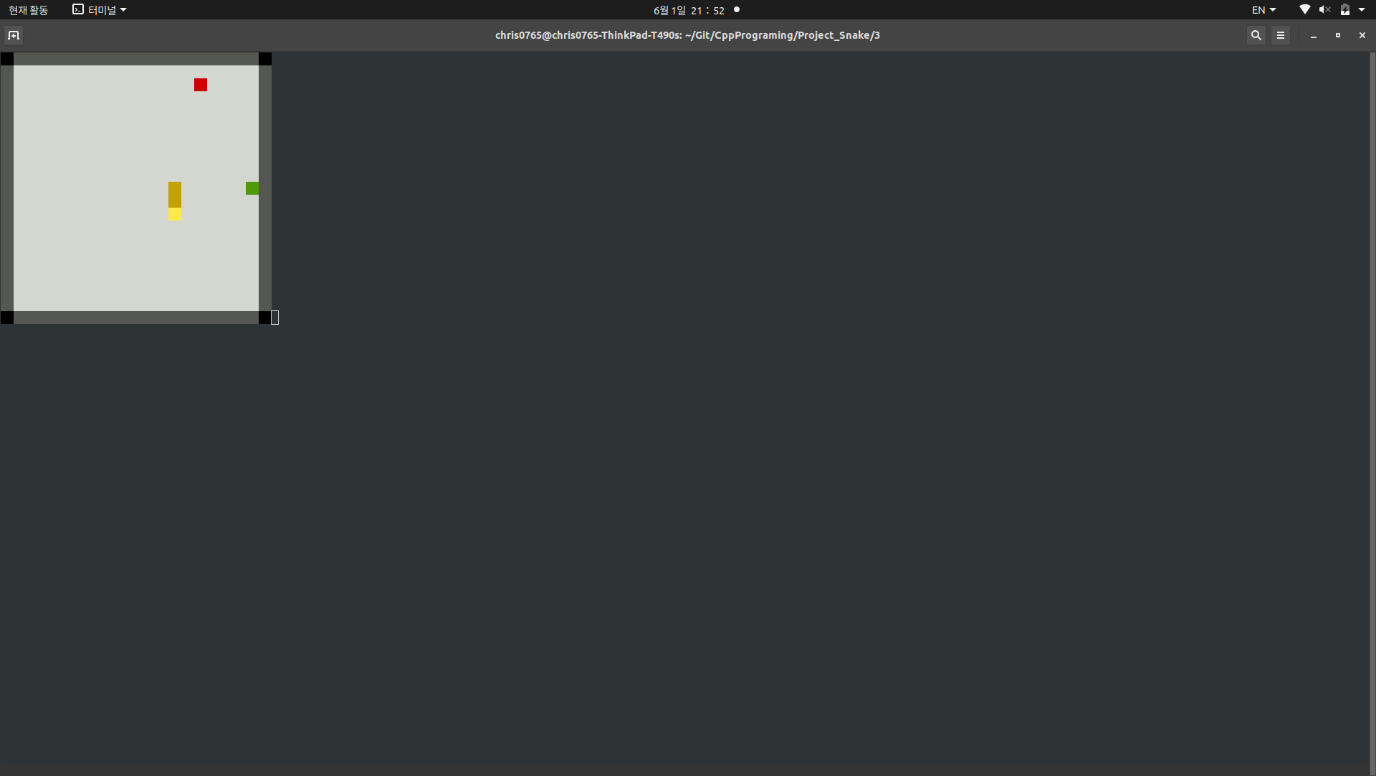
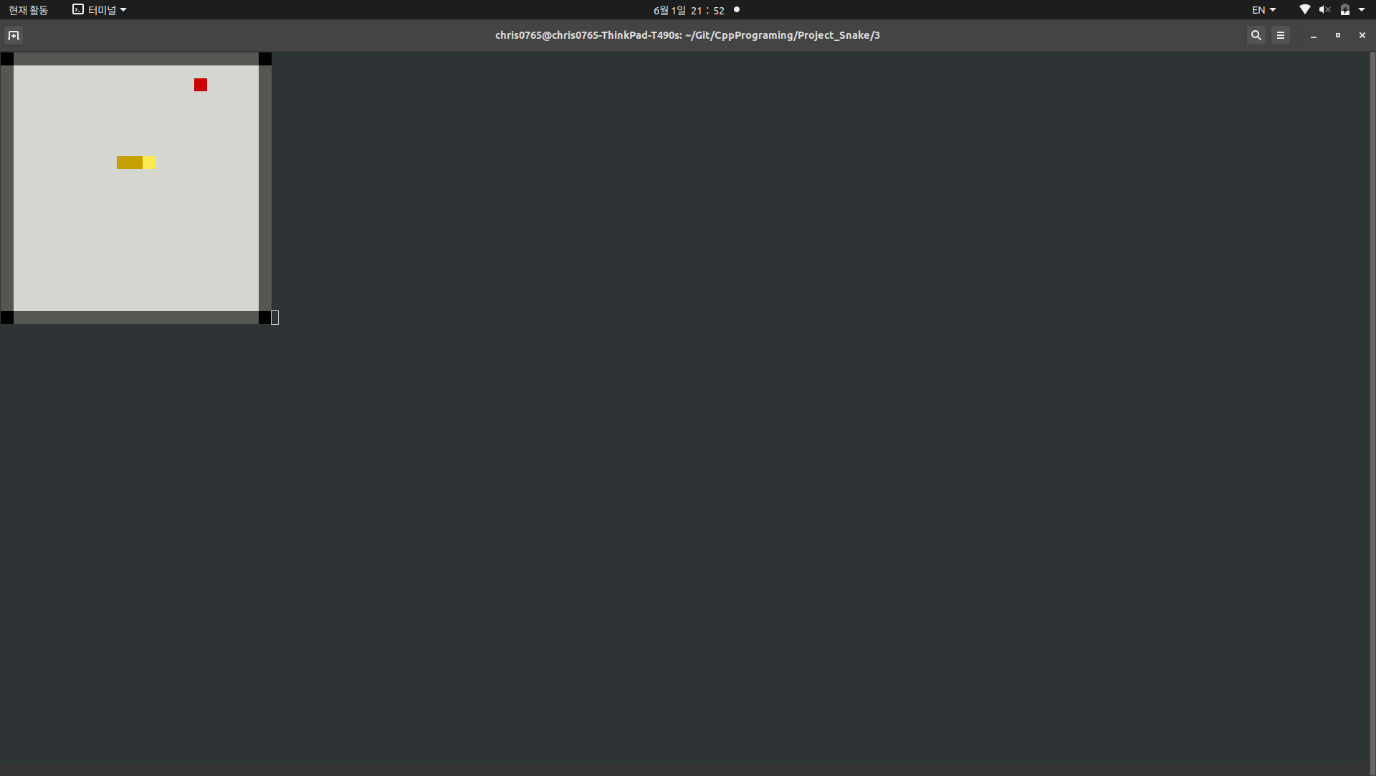
아이템은 최대 3개까지만 화면에 생성되도록 만들었으며, vector<item\_type> items를 통해 item에 대한 정보를 저장한다. (item\_type은 Pos, type, madetime을 멤버로 가지는 struct 구조체이다. Pos로 좌표를 저장하고, type으로 Grow와 Poison을 구별하고, madetime을 통해 만들어진 시각을 저장한다.)

Item 객체는 Map 객체의 벽만 표시해둔 OnlyMap과 Snake 객체의 BODY를 넘겨받으며, 넘겨받은 데이터를 통해 Item 객체가 벽이나 Snake가 위치한 좌표에 생성되지 않도록 한다.

5초마다 새로운 아이템이 생성되도록 하였으며, Snake의 Head가 Item과 충돌 시에, 해당 Item을 삭제하도록 하였다.

Snake에 Grow와 Poison을 이번 틱에 섭취하였는지를 나타내는 isGrow와 isPoison을 생성하였으며, 기존의 Snake의 이동을 표시하던 방법에서 뒷부분을 제거하던 부분을 isGrow의 값 여부를 통하여 실행하도록 바꾸었으며, isGrow가 true, 즉 이번 틱에 Grow 아이템을 획득했다면 꼬리를 제거하지 않고, false라면 제거하도록 하였다. isPoison이 true라면 추가로 꼬리를 제거하는 작업을 수행하였다.

Item 객체 내의 ItemMap을 통해 Map 객체의 맵 데이터에 Items의 정보를 추가로 표시하였고, GameWindow에 Map의 OnlyMap이 아닌 Item 객체의 ItemMap을 넘겨주어 화면에 출력해주도록 하였다.



**-4단계 구현**

Item 객체의 ItemMap을 삭제하였으며, Map 객체에 totalMap을 새롭게 만들어, OnlyMap에 Item과 Gate에 관한 정보들을 추가로 표시하도록 하였다.

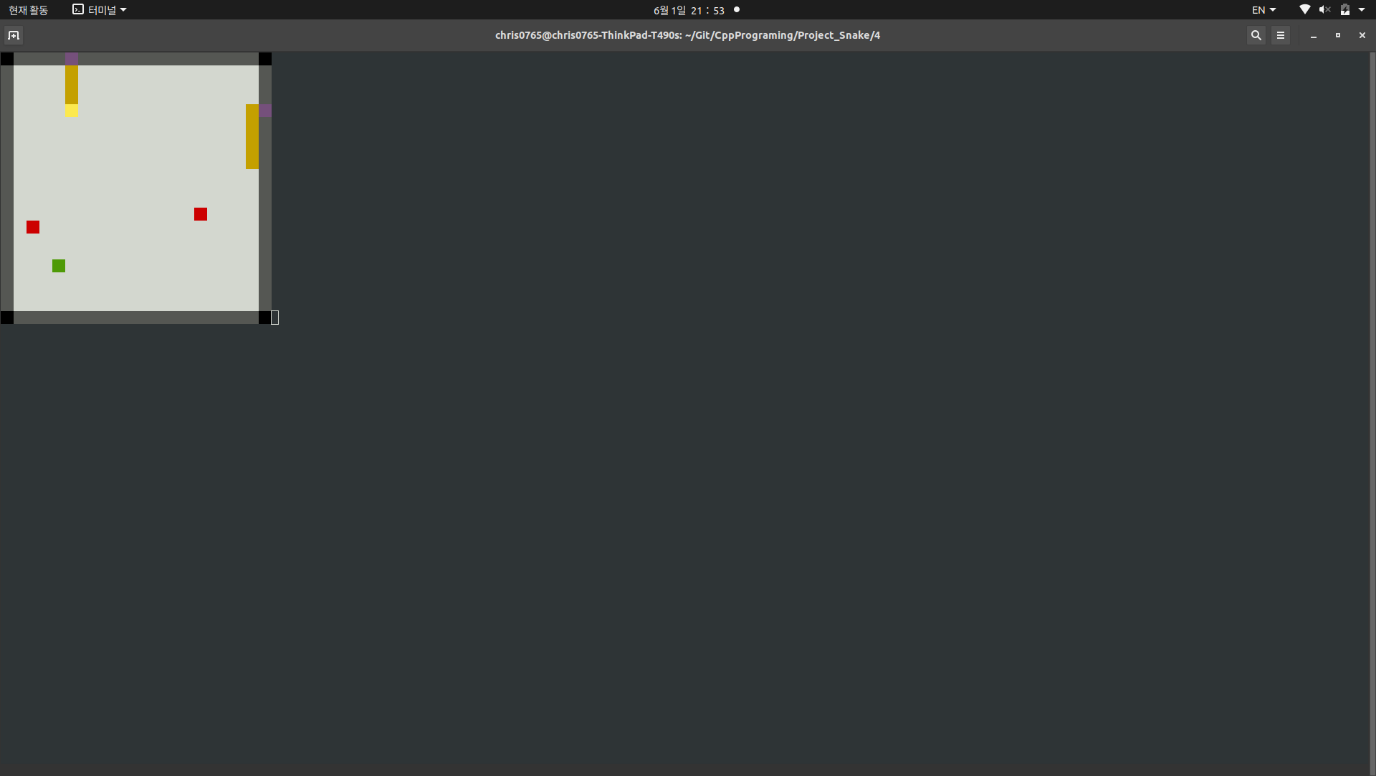
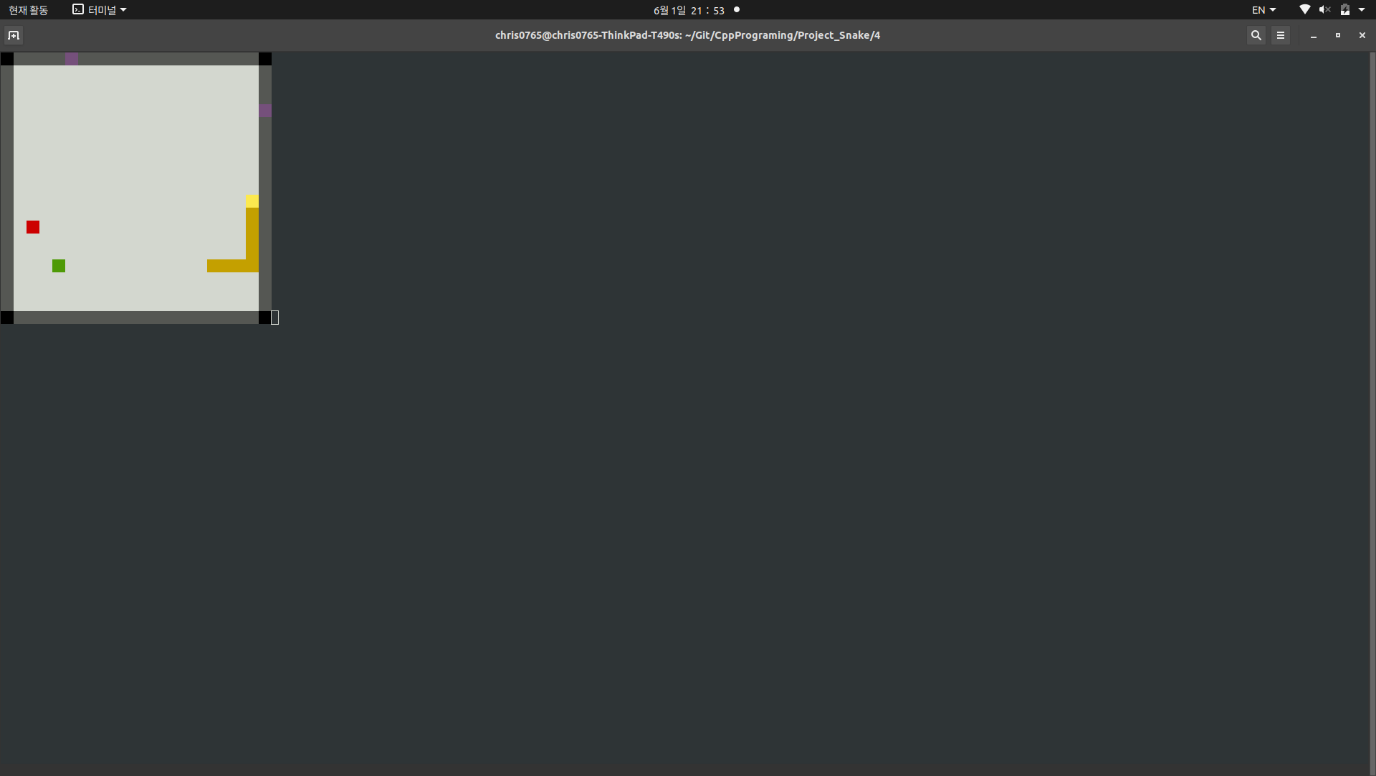
Gate 객체를 만들어 Gate에 대한 정보들을 통제하도록 하였다.

Gate 역시 vector<Pos> 형식의 gates를 통해 생성된 Gate 쌍의 좌표를 저장하며, Snake 객체에 Grow 아이템을 획득한 횟수를 저장하는 변수를 추가하여, 해당 점수가 5점을 넘긴다면 Gate쌍을 생성한다.

gates의 값은 생성된지 15초가 지난다면 삭제하게끔 하였고, 추가로 Snake 객체에 게이트에 진입된 상태를 표시하는 isWarp를 생성하여 해당 변수의 값이 true일 경우 15초가 지난 시점에서도 게이트 쌍을 삭제하지 않도록 하였다.

Snake 객체에 게이트에 진입한지 몇 틱이 지났는지를 저장하는 변수를 생성 후, 해당 변수의 값이 Snake의 length 이상이 된다면 Snake가 게이트를 모두 빠져나온 것이 되므로 isWarp를 false로 만든다.

Map 객체가 Item의 items, Gate의 gates를 넘겨받아 totalMap을 제작하고, GameWindow 객체가 Map의 totalMap과 Snake의 BODY를 넘겨받아 화면에 출력한다.



**-5단계 구현**

스코어보드 구성과 출력을 담당하는 ScoreWindow 객체와 미션보드 구성과 출력을 담당하는 MissionWindow 객체를 생성하였다.

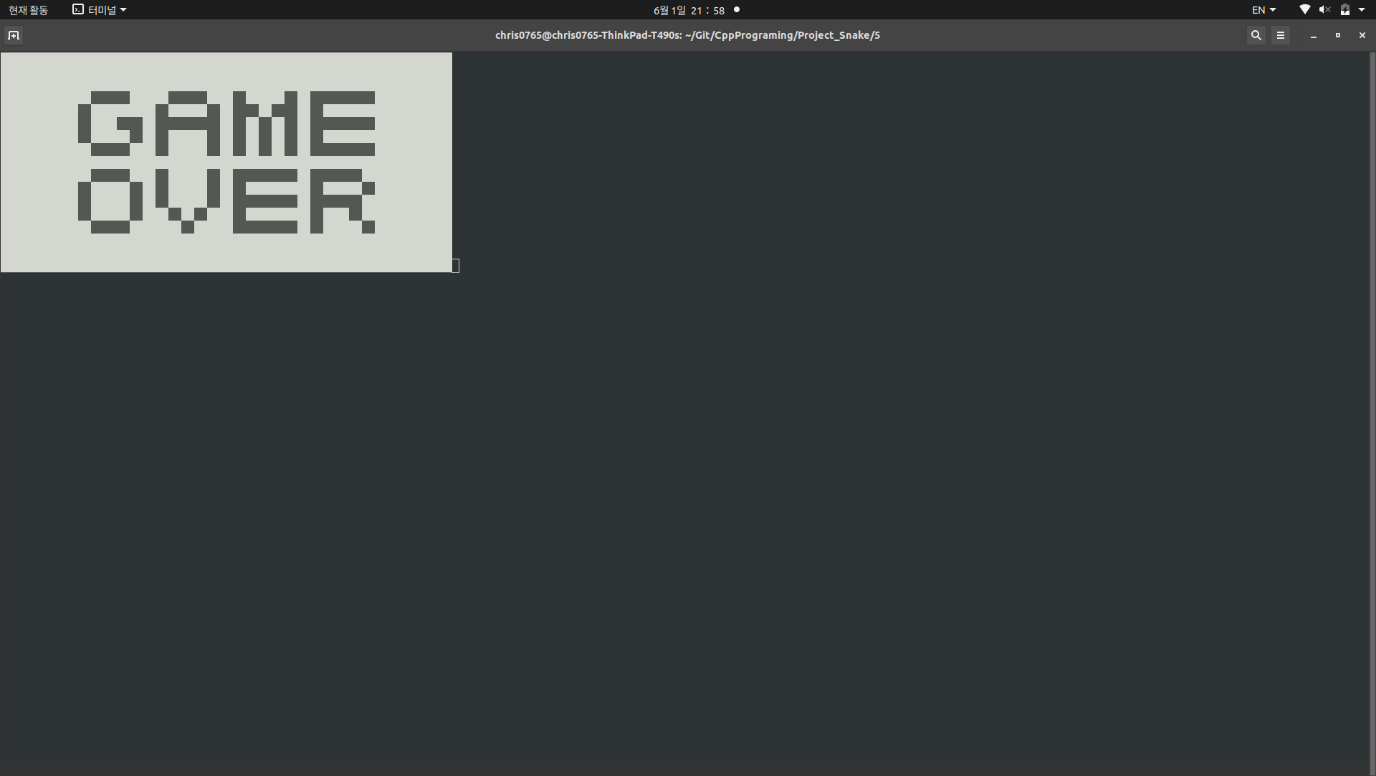
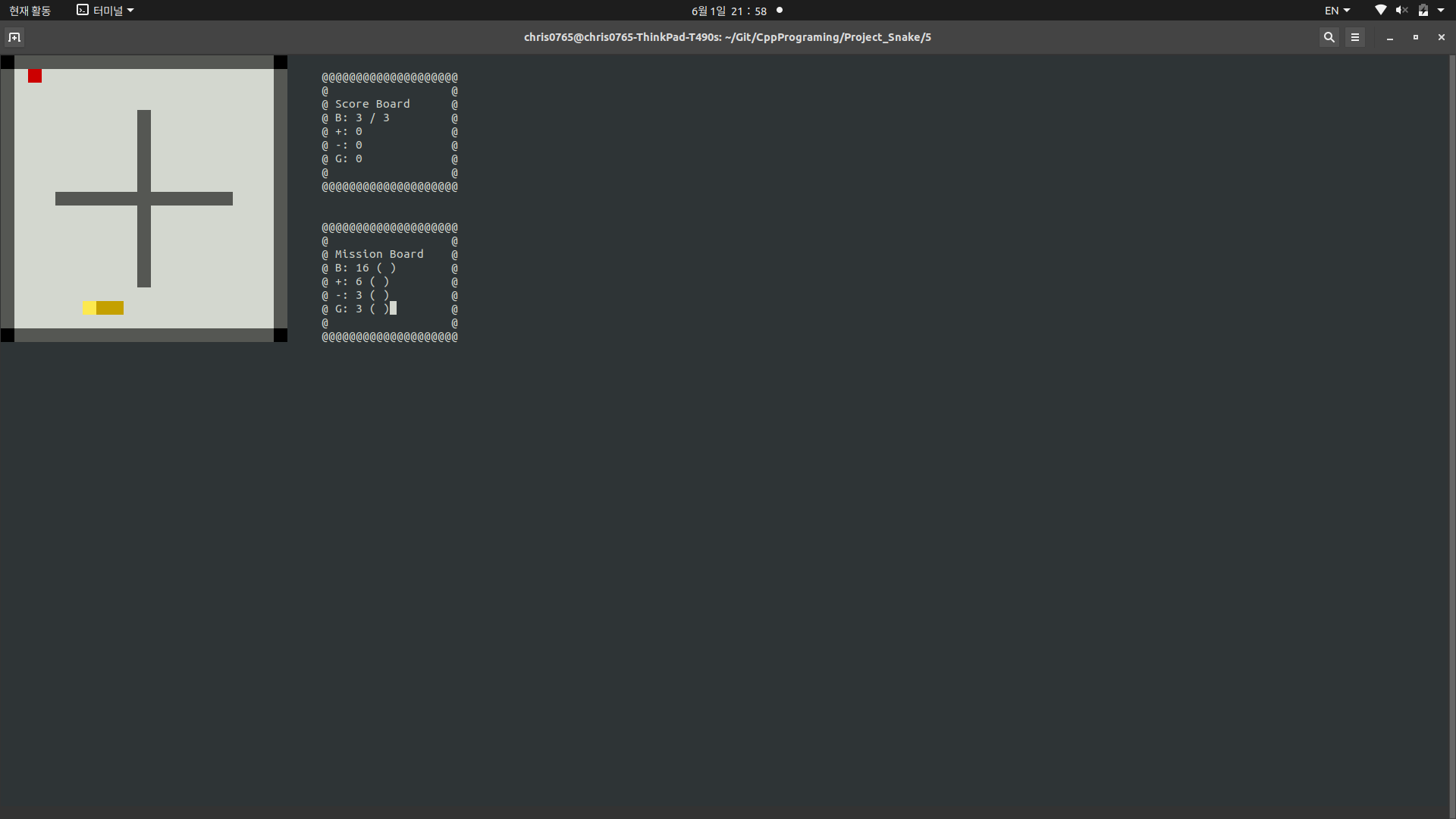
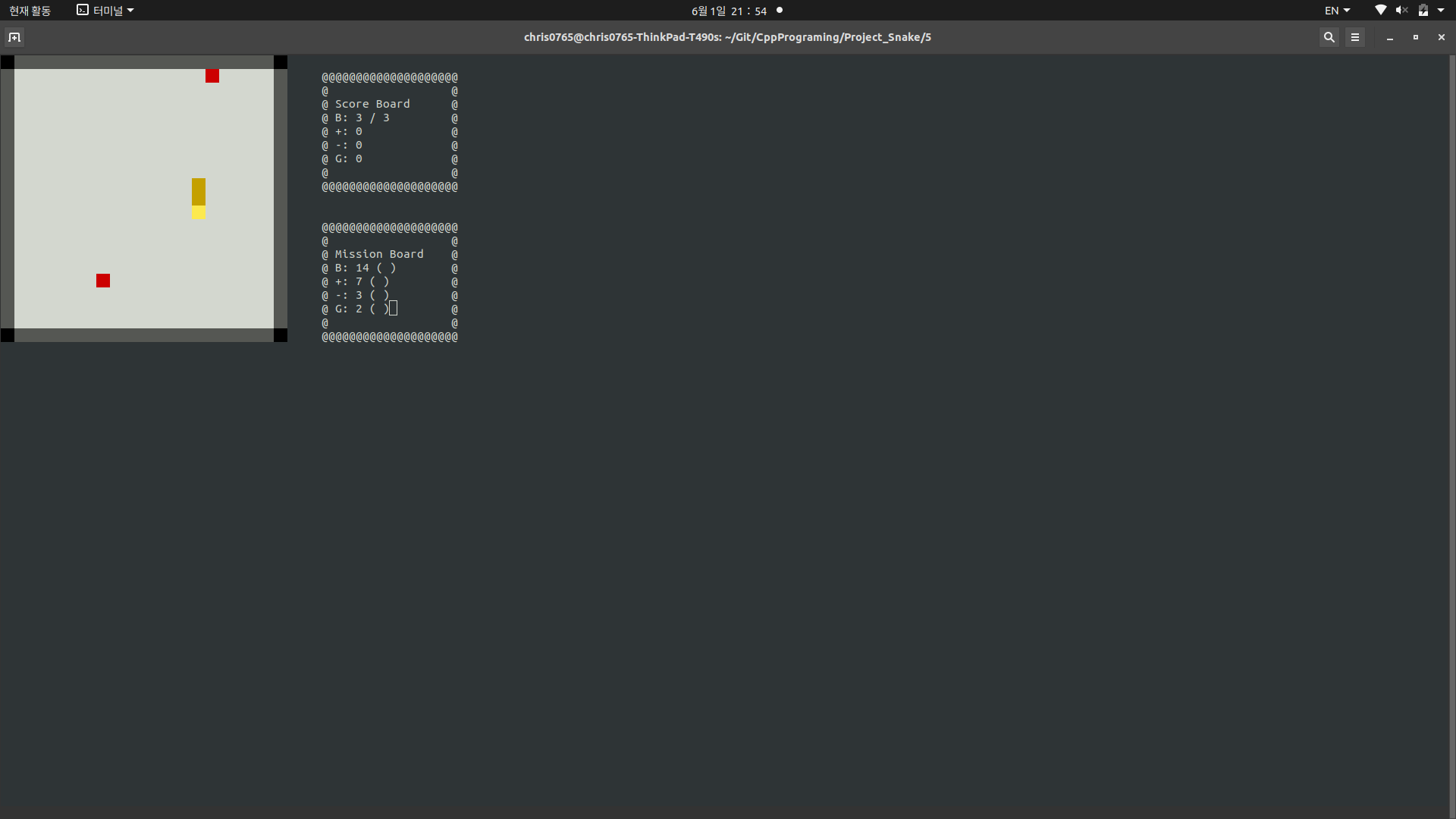
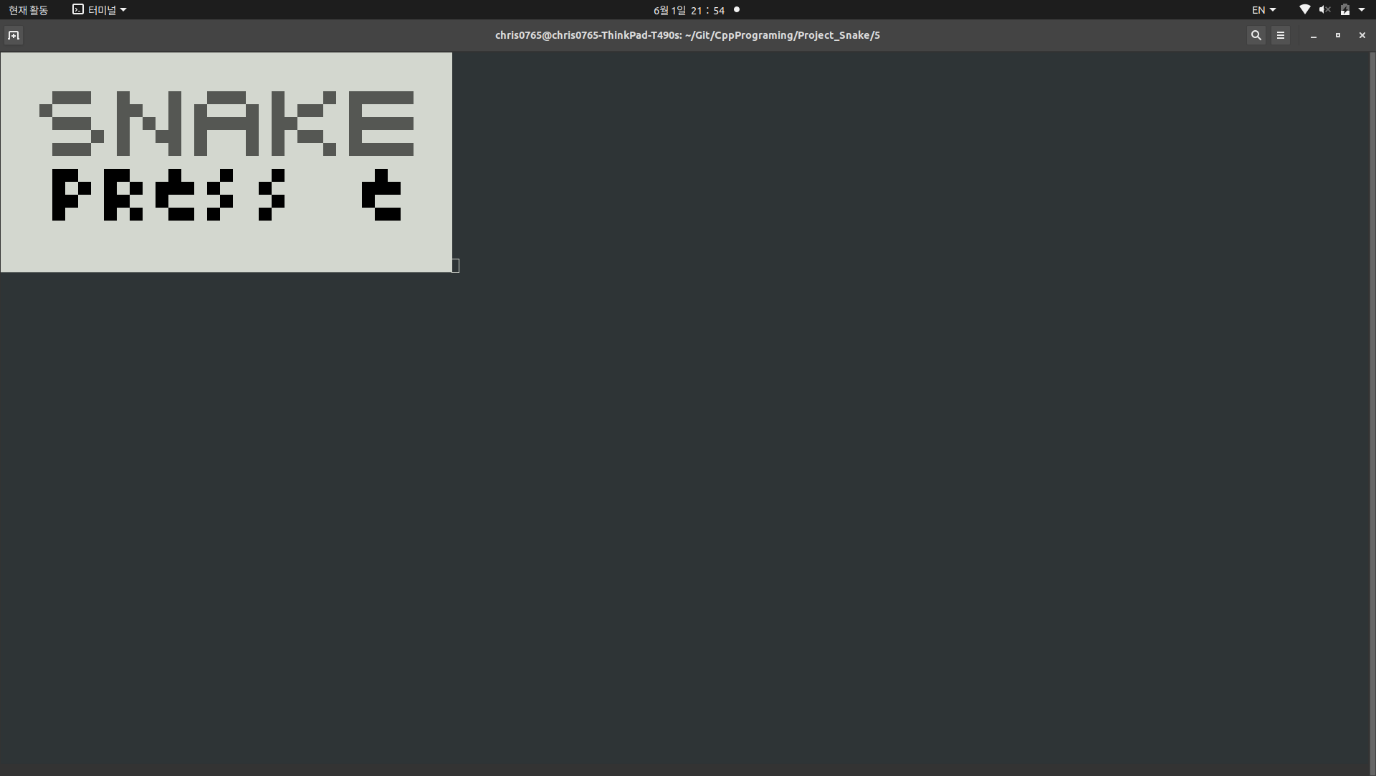
ScoreWindow에서 WINDOW \*ScoreBoard를 생성하여 새로운 출력 윈도우를 제작하였고,

MissionWindow에서 WINDOW \*MissionBoard를 생성하여 새로운 출력 윈도우를 제작하였다.

ScoreBoard에선 매 틱마다 Snake의 길이, Grow 아이템 획득 수, Poison 아이템 획득 수, 게이트 진입 수를 넘겨받아 화면에 pdf에서 제시된 형식대로 출력하였다. Snake의 길이를 계속해서 갱신하는 변수와 최대 길이를 저장하는 변수를 통해 (현재 길이) / (최대 길이) 형식으로 Snake의 길이를 표시하였다.

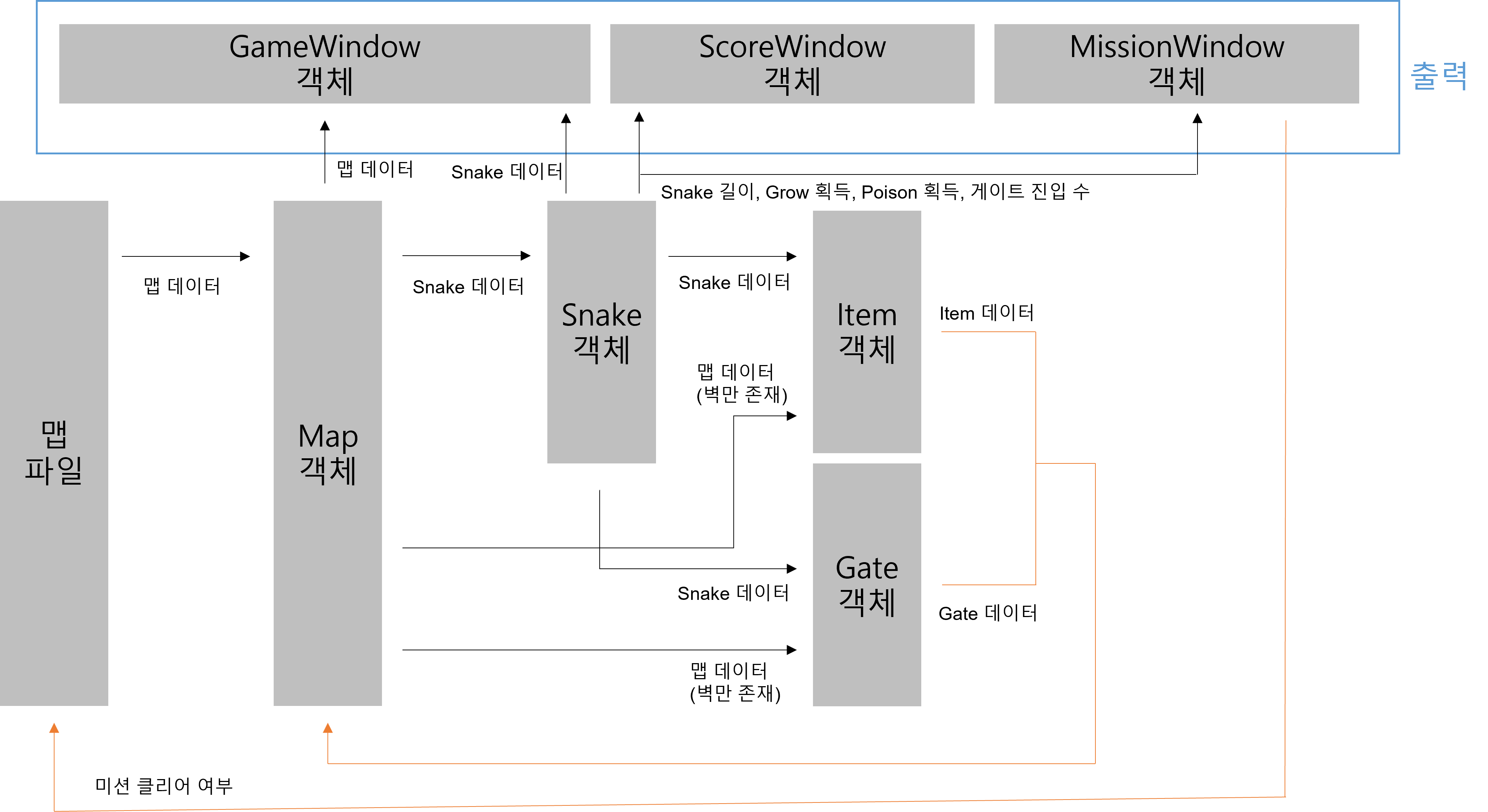
MissionBoard에선 매 틱마다 Snake의 길이, Grow 아이템 획득 수, Poison 아이템 획득 수, 게이트 진입 수를 넘겨받아 레벨에 따라 랜덤한 범위로 설정된 미션을 달성했다면 ‘v’자를 통해 체크하여 표시하였다.

또한, 기존에 map1.txt 한 개만 존재하던 맵 구성에서, 1~5단계까지로 나누어 미션을 달성할 때마다 다음 맵으로 넘어가도록 구성하였으며, 게임 시작화면, 게임 오버화면, 게임 클리어화면을 추가로 구성하였다. 게임 시작화면에선 ‘e’키를 입력 시 게임이 시작되며, 게임 오버, 게임 클리어 시 해당하는 화면을 1초간 노출 후 프로그램이 종료된다.



### 시스템 구조 및 설계도

|  |
| --- |
| **작성요령 (30점)**  **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.** |



### 활용/개발된 기술

|  |
| --- |
| **작성요령 (10점)**  **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.** |

Ncurses : 터미널에 GUI 출력

Vector : 맵 데이터, Pos 데이터 등을 객체에 저장하는 부분에 사용

Chrono : 게임 시작 후 지난 시간을 체크

Random : Item과 Gate를 랜덤한 좌표에 생성하는 부분에 사용, 미션의 값을 랜덤으로 부여하는 부분에 사용

Unistd.h : 틱을 설정하기 위하여 usleep을 사용하는 부분에 사용

Fstream : txt파일에서 데이터를 불러오는 부분에 사용

String : txt파일의 값을 줄 단위 문자열로 받아와 맵 데이터로 변경하는 부분에 사용

### 현실적 제한 요소 및 그 해결 방안

|  |
| --- |
| **작성요령 (5점)**  **제안된 프로젝트의 단계 별 수행에 있어, 제한 요소를 찾아 작성한다. 해당 제한 요소를 해결하기 위해서 어떤 방법으로 해결하였는지 작성한다.** |

오픈소스디자인에서 제공된 ncurses 예제 pdf에서, 터미널 윈도우를 조절하는 방법으로 resize\_window()를 알려주었지만, 실제로 사용해보니 ncurses에서 제공되는 함수가 아니었다.

때문에 맵 크기가 커진다면 그에 맞추어 추가적으로 터미널의 창을 수동적으로 조정해주어야 하는 문제가 있다. 해당 문제를 해결하기 위해 다양한 사이트에 검색을 시도해 보았지만, 마땅한 해결 방법을 찾지 못하였다.

### 결과물 목록

|  |
| --- |
| **작성요령 (5점)**  **결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.** |

|  |  |  |
| --- | --- | --- |
| 게임화면 출력 | GameWindow.h | GameWindow 클래스 정의 |
| GameWindow.cpp | GameWindow클래스 구현 |
| 스코어보드 출력 | ScoreWindow.h | ScoreWindow 클래스 정의 |
| ScoreWindow.cpp | ScoreWindow 클래스 구현 |
| 미션보드 출력 | MissionWindow.h | MissionWindow 클래스 정의 |
| MissionWindow.cpp | MissionWindow 클래스 구현 |
| 맵 데이터 통제 | Map.h | Map 클래스 정의 |
| Map.cpp | Map 클래스 구현 |
| Snake 데이터 통제 | Snake.h | Snake 클래스 정의 |
| Snake.cpp | Snake 클래스 구현 |
| Item 데이터 통제 | Item.h | Item 클래스 정의 |
| Item.cpp | Item 클래스 구현 |
| Gate 데이터 통제 | Gate.h | Gate 클래스 정의 |
| Gate.cpp | Gate 클래스 구현 |
| GUI 화면 초기화 | INIT.h | Init() 함수 정의 |
| INIT.cpp | Init() 함수 구현 |
| 좌표 저장 | Pos.h | Pos 클래스 정의 |
| Pos.cpp | Pos 클래스 구현 |
| 아이템 타입 저장 | Item\_type.h | Item\_type struct 구현 |
| 객체끼리 데이터 교환 | main.cpp | 게임이 실행되는 코어 |
| 컴파일 및 실행파일 제작 | Makefile | 자동 컴파일 |
| 게임 시작 화면 | Start.txt | 게임 시작 화면 데이터 |
| 게임 오버 화면 | GameOver.txt | 게임 오버 화면 데이터 |
| 게임 클리어 화면 | GameClear.txt | 게임 클리어 화면 데이터 |
| 맵 데이터 | map1.txt | 1단계 맵 데이터 |
| map2.txt | 2단계 맵 데이터 |
| map3.txt | 3단계 맵 데이터 |
| map4.txt | 4단계 맵 데이터 |
| map5.txt | 5단계 맵 데이터 |

### 코드 설명

|  |
| --- |
| main.cpp |
| #include <iostream>  #include "INIT.h"  #include "Snake.h"  #include "Map.h"  #include "GameWindow.h"  #include "Item.h"  #include "Gate.h"  #include "ScoreWindow.h"  #include "MissionWindow.h"  using namespace std;  int main(){  chrono::time\_point<chrono::system\_clock> start = chrono::system\_clock::now(); // 프로그램 시작 시간  init(); // 출력 윈도우 설정 // INIT.h  GameWindow GW = GameWindow(); // 게임이 진행되는 화면을 관리하는 객체 생성  int level = 1; // 레벨에 따라 맵과 미션의 난이도 설정  string filename = "Start.txt"; // 게임 시작 화면  Map M = Map(); // 맵 클래스  M.LoadMap(filename);  Snake S = Snake(M.getSnake()); // 맵 데이터에 존재하는 Snake의 데이터 불러와 객체 생성  nodelay(stdscr, false);  GW.ReadyToPrintMap(M.getTotalMap(), S.getBODY());  GW.Print();  do{  char key = getch();  if(key == 'e')  break;  }while(true); // e키를 누르면 시작  nodelay(stdscr, TRUE);  filename = "map1.txt"; // 첫 맵  M.LoadMap(filename); // 텍스트 파일에 있는 맵 데이터 불러옴  S = Snake(M.getSnake()); // 맵 데이터에 존재하는 Snake의 데이터 불러와 객체 생성  ScoreWindow SW = ScoreWindow(S.getLength(), M.getCol()); // 스코어를 보여주는 화면을 관리하는 객체 생성  MissionWindow MW = MissionWindow(M.getCol()); // 미션을 보여주는 화면을 관리하는 객체 생성  MW.setNew(level); // 레벨에 따라 미션 부여  Item I = Item(M.getRow(), M.getCol()); // Grow, Poison 아이템을 관리하는 객체 생성  Gate G = Gate(M.getRow(), M.getCol()); // 게이트를 관리하는 객체 생성  while(true){  I.makeItem(M.getMap(), S.getBODY(), chrono::system\_clock::now()); // 시간에 따라 아이템 생성, Snake, Wall과 겹치지 않는 위치에 생성한다.  G.makeGates(M.getMap(), S.getGrowScore(), chrono::system\_clock::now(), S.Warp()); // 점수에 따라 게이트 생성, Wall 위에 생성한다.  M.MakeMap(I.getItem(), G.getGates()); // 아이템과 게이트를 합쳐서 맵에 표시한다.  Pos nexthead = G.headPath(M.getTotalMap(), S.nextHead(), S.getDirection()); // Snake의 머리가 다음 틱에 어디에 있을지를 구한다.(게이트 통과 포함)  S.update(M.getTotalMap(), nexthead, G.dirPath(M.getTotalMap(), S.nextHead(), S.getDirection())); // Snake를 업데이트하고, 방향 또한 업데이트 한다.(게이트 통과 포함)  I.delItem(S.head()); // Snake가 특정 아이템을 먹었다면 해당하는 아이템을 삭제한다.  SW.setValues(S.getLength(), S.growCount(), S.poisonCount(), S.warpCount()); // 스코어보드 갱신  MW.setValues(S.getLength(), S.growCount(), S.poisonCount(), S.warpCount()); // 미션보드 갱신  GW.ReadyToPrintMap(M.getTotalMap(), S.getBODY()); // 아이템과 게이트가 표시된 맵과 Snake를 합친다.  GW.Print(); // 위에서 합친 데이터를 출력  SW.Print(); // 스코어보드 출력  MW.Print(); // 미션보드 출력  if(S.Died()){ // 죽었다면  filename = "GameOver.txt"; // 게임 오버를 표시  M.LoadMap(filename);  S = Snake(M.getSnake());  GW.ReadyToPrintMap(M.getTotalMap(), S.getBODY());  GW.Print();  break; // 이후 탈출  }  if(MW.isClear()){ // 미션을 모두 클리어했다면  level++; // 다음 레벨로 증가  filename = "map" + to\_string(level) + ".txt";  if(access(filename.c\_str(), F\_OK) != 0){ // 찾고자 하는 맵이 없다면 == 마지막 단계의 맵까지 클리어 == 게임 클리어  filename = "Clear.txt"; // 게임 클리어를 표시  M.LoadMap(filename);  S = Snake(M.getSnake());  GW.ReadyToPrintMap(M.getTotalMap(), S.getBODY());  GW.Print();  break; // 이후 탈출  }  M.LoadMap(filename); // 찾는 맵이 있다면 해당 맵을 표시  S = Snake(M.getSnake());  SW = ScoreWindow(S.getLength(), M.getCol());  MW =MissionWindow(M.getCol());  MW.setNew(level);  I = Item(M.getRow(), M.getCol());  G = Gate(M.getRow(), M.getCol());  }  }  usleep(10000000); // 게임 종료 후 1초간 종료 화면 노출  endwin();  return 0;  } |

|  |
| --- |
| INIT.cpp |
| #include "INIT.h"  #include <ncurses.h>  void init(){  initscr();  noecho(); // 키 입력 보이지 않음  nodelay(stdscr, TRUE); // 키를 입력 대기한 상태로 다른 코드 실행 가능  keypad(stdscr, TRUE);  start\_color();  init\_pair(8, COLOR\_WHITE, COLOR\_WHITE); //MAP\_BLANK  init\_pair(1, COLOR\_WHITE, 8); //MAP\_WALL  init\_pair(2, COLOR\_WHITE, 16); //MAP\_IMMUNE\_WALL  init\_pair(3, COLOR\_WHITE, 11); //SNAKE\_HEAD  init\_pair(4, COLOR\_WHITE, COLOR\_YELLOW); //SNAKE\_BODY  init\_pair(5, COLOR\_WHITE, COLOR\_GREEN); //GROWTH\_ITEM  init\_pair(6, COLOR\_WHITE, COLOR\_RED); //POISON\_ITEM  init\_pair(7, COLOR\_WHITE, COLOR\_MAGENTA); //GATE  } |

|  |
| --- |
| GameWindow.cpp |
| #include "GameWindow.h"  #include <ncurses.h>  #include <unistd.h>  GameWindow::GameWindow(){}  void GameWindow::ReadyToPrintMap(vector<vector<int>> Map, vector<Pos> snake){  printMap = Map; // 아이템과 게이트가 표시된 맵 데이터  for(int i=1; i<snake.size(); i++)  printMap[snake[i].getX()][snake[i].getY()] = 4; // Snake의 몸통 표시  if(snake[0].getX() != -1 && snake[0].getY() != -1)  printMap[snake[0].getX()][snake[0].getY()] = 3; // Snake의 머리 표시  }  // 출력  void GameWindow::Print(){  clear();  for(int i=0; i<printMap.size(); i++){  for(int j=0; j<printMap[i].size(); j++){  if(printMap[i][j]==0){  attron(COLOR\_PAIR(8));  mvprintw(i, j\*2, " ");  attroff(COLOR\_PAIR(8));  }  else{  attron(COLOR\_PAIR(printMap[i][j]));  mvprintw(i, j\*2, " ");  attroff(COLOR\_PAIR(printMap[i][j]));  }  }  }  refresh();  } |

|  |
| --- |
| ScoreWindow.cpp |
| #include "ScoreWindow.h"  ScoreWindow::ScoreWindow(int stLen, int sty): currLen(stLen), maxLen(stLen){  ScoreBoard = newwin(9, 20, 1, (sty \* 2) + 5);  init\_pair(10, COLOR\_WHITE, COLOR\_BLACK);  wbkgd(ScoreBoard, COLOR\_PAIR(10));  wborder(ScoreBoard, '@', '@', '@', '@', '@', '@', '@', '@');  }  // 스코어보드 갱신  // 새 맵으로 넘어가면, 스코어보드 역시 초기화된다  void ScoreWindow::setValues(int length, int gt, int pt, int wt){  currLen = length;  if(length > maxLen)  maxLen = length;    growt = gt;  poisont = pt;  warpt = wt;  }  void ScoreWindow::Print(){  wbkgd(ScoreBoard, COLOR\_PAIR(10));  wborder(ScoreBoard, '@', '@', '@', '@', '@', '@', '@', '@');  wattron(ScoreBoard, COLOR\_PAIR(10));  mvwprintw(ScoreBoard, 2, 2, "Score Board");  mvwprintw(ScoreBoard, 3, 2, "B: %d / %d", currLen, maxLen); // 현재 길이 / 최대 길이  mvwprintw(ScoreBoard, 4, 2, "+: %d", growt); // Grow 아이템 획득 횟수  mvwprintw(ScoreBoard, 5, 2, "-: %d", poisont); // Posion 아이템 획득 횟수  mvwprintw(ScoreBoard, 6, 2, "G: %d", warpt); // Gate 진입 횟수  wattroff(ScoreBoard, COLOR\_PAIR(10));  wrefresh(ScoreBoard);  } |

|  |
| --- |
| MissionWindow.cpp |
| #include "MissionWindow.h"  MissionWindow::MissionWindow(int sty){  MissionBoard = newwin(9, 20, 12, (sty \* 2) + 5);  init\_pair(10, COLOR\_WHITE, COLOR\_BLACK);  wbkgd(MissionBoard, COLOR\_PAIR(10));  wborder(MissionBoard, '@', '@', '@', '@', '@', '@', '@', '@');  toB = 0;  toG = 0;  toP = 0;  toW = 0;  }  // 레벨에 따른 범위 내에서 랜덤하게 미션 부여  void MissionWindow::setNew(int level){  random\_device rd;  mt19937 gen(rd());  uniform\_int\_distribution<int> randB(9+level, 14+level);  uniform\_int\_distribution<int> randG(4+level, 8+level);  uniform\_int\_distribution<int> randP(1+level, 3+level);  uniform\_int\_distribution<int> randW(0+level, 1+level);  toB = randB(gen);  toG = randG(gen);  toP = randP(gen);  toW = randW(gen);  clearB = false;  clearG = false;  clearP = false;  clearW = false;  }  void MissionWindow::setValues(int length, int growt, int poisont, int warpt){  if(length >= toB)  clearB = true;  if(growt >= toG)  clearG = true;  if(poisont >= toP)  clearP = true;  if(warpt >= toW)  clearW = true;  }  bool MissionWindow::isClear(){  if(clearB && clearG && clearP && clearW)  return true;  return false;  }  void MissionWindow::Print(){  wbkgd(MissionBoard, COLOR\_PAIR(10));  wborder(MissionBoard, '@', '@', '@', '@', '@', '@', '@', '@'); // 테두리 설정  wattron(MissionBoard, COLOR\_PAIR(10));  mvwprintw(MissionBoard, 2, 2, "Mission Board");  mvwprintw(MissionBoard, 3, 2, "B: %d (%c)", toB, (clearB?'v':' ')); // 해당 미션 클리어 시 v자 표시  mvwprintw(MissionBoard, 4, 2, "+: %d (%c)", toG, (clearG?'v':' '));  mvwprintw(MissionBoard, 5, 2, "-: %d (%c)", toP, (clearP?'v':' '));  mvwprintw(MissionBoard, 6, 2, "G: %d (%c)", toW, (clearW?'v':' '));  wattroff(MissionBoard, COLOR\_PAIR(10));  wrefresh(MissionBoard);  usleep(500000);  } |

|  |
| --- |
| Map.cpp |
| #include "Map.h"  #include <iostream>  Map::Map(){}  void Map::LoadMap(string filename){  ifstream fin(filename);  string ins;  OnlyMap.clear();  totalMap.clear();  disnakeHead = {-1, -1};  disnakeBody.clear();  vector<int> tmpv; // 초기화  row = 0;  while(!fin.eof()){ // 파일 끝까지 입력받음  fin>>ins;  OnlyMap.push\_back(tmpv);  for(int i=0; i<ins.size(); i++){  // 맵의 해당 위치의 데이터가 Snake라면, 따로 추출하고 0을 저장함  if(ins[i] == '3'){  disnakeHead = {row, i};  OnlyMap[row].push\_back(0);  }  else if(ins[i] == '4'){  disnakeBody.push\_back({row, i});  OnlyMap[row].push\_back(0);  }  else  OnlyMap[row].push\_back(int(ins[i]-'0'));  }  row++;  }  col = ins.size();  totalMap = OnlyMap;  }  void Map::MakeMap(vector<item\_type> Items, vector<Pos> Gates){  totalMap = OnlyMap;  // 아이템 데이터 추가  for(int i=0; i<Items.size(); i++){  totalMap[Items[i].pos.getX()][Items[i].pos.getY()] = Items[i].type;  }  // 게이트 데이터 추가  for(int i=0; i<Gates.size(); i++){  totalMap[Gates[i].getX()][Gates[i].getY()] = 7;  }  }  vector<vector<int>> Map::getTotalMap(){  return totalMap;  }  vector<Pos> Map::getSnake(){  vector<Pos> CompleteSnake;  CompleteSnake.push\_back(disnakeHead);  for(int i=0; i<disnakeBody.size(); i++){  CompleteSnake.push\_back(disnakeBody[i]);  }  return CompleteSnake;  }  vector<vector<int>> Map::getMap(){  return OnlyMap;  }  int Map::getRow(){  return row;  }  int Map::getCol(){  return col;  } |

|  |
| --- |
| Snake.cpp |
| #include "Snake.h"  Snake::Snake(vector<Pos> LoadedBody){  dir = 'l';  BODY = LoadedBody;  isGrow = false;  isPoison = false;  isWarped = false;  isDie = false;  growscore = 0;  warpedLen = 0;  warptime = 0;  growtime = 0;  poisontime = 0;  }  Pos Snake::head(){  return BODY.front();  }  Pos Snake::nextHead(){  int key = getch(); // 키보드 입력을 통해 받아옴 // nodelay 적용  switch(key){  case KEY\_LEFT:  dir = 'l';  break;  case KEY\_RIGHT:  dir = 'r';  break;  case KEY\_UP:  dir = 'u';  break;  case KEY\_DOWN:  dir = 'd';  break;  }  if(dir=='l')  return {BODY[0].getX(), BODY[0].getY()-1};  else if(dir=='r')  return {BODY[0].getX(), BODY[0].getY()+1};  else if(dir=='u')  return {BODY[0].getX()-1, BODY[0].getY()};  else if(dir=='d')  return {BODY[0].getX()+1, BODY[0].getY()};  return BODY[0];  }  bool Snake::Died(){  return isDie;  }  bool Snake::Warp(){  return isWarped;  }  void Snake::collisonWith(vector<vector<int>> totalMap){  Pos tmp = head();  if(totalMap[tmp.getX()][tmp.getY()] == 5) // Grow 아이템과 충돌시  isGrow = true;  else if(totalMap[tmp.getX()][tmp.getY()] == 6) // Poison 아이템과 충돌시  isPoison = true;  else if(totalMap[tmp.getX()][tmp.getY()] != 0) // 벽과 충동시  isDie = true;  for(int i=1; i<BODY.size(); i++) // 몸과 충돌시  if(BODY[0].getX() == BODY[i].getX() && BODY[0].getY() == BODY[i].getY()){  isDie = true;  break;  }  }  void Snake::shrink(){  BODY.erase(BODY.begin() + BODY.size() - 1);  }  vector<Pos> Snake::getBODY(){  return BODY;  }  void Snake::update(vector<vector<int>> totalMap, Pos tHead, char nextDir){  // 머리의 다음 좌표가 게이트일 경우에 게이트 이동을 True로 변경  if(totalMap[nextHead().getX()][nextHead().getY()] == 7)  isWarped = true;  // 머리 이동과 진행방향 갱신  BODY.insert(BODY.begin(), tHead);  dir = nextDir;  collisonWith(totalMap); // 충돌 검사  if(!isGrow){ // Grow 아이템을 먹지 않았다면 꼬리를 1칸 삭제. 다음 머리 위치를 배열 맨 앞에 삽입했기 때문에 꼭 필요한 과정  shrink();  }  else{  growscore++;  growtime++;  isGrow = false;  }  if(isPoison){ // Poison 아이템을 획득시 꼬리를 1칸 추가로 감소시킴  shrink();  poisontime++;  isPoison = false;  }  if(isWarped){  if(warpedLen == 0)  warptime++;  warpedLen++;  growscore = 0;  if(warpedLen >= BODY.size()){  isWarped = false;  warpedLen = 0;  }  }  if(BODY.size()<3) // 길이가 3 미만이면 게임 오버  isDie = true;  }  int Snake::getGrowScore(){  return growscore;  }  int Snake::getWarpedLen(){  return warpedLen;  }  int Snake::getLength(){  return BODY.size();  }  int Snake::warpCount(){  return warptime;  }  int Snake::growCount(){  return growtime;  }  int Snake::poisonCount(){  return poisontime;  }  char Snake::getDirection(){  return dir;  } |

|  |
| --- |
| Item.cpp |
| #include "Item.h"  Item::Item(int R, int C): row(R), col(C){};  void Item::makeItem(vector<vector<int>> Map, vector<Pos> BODY, chrono::time\_point<chrono::system\_clock> gametime){  // 생성된 아이템이 없거나, 마지막 아이템이 생성된 후 5초가 지났다면 최대 3개까지 아이템 생성  // 4개까지 생성은 되나, 4개째는 생성 후 바로 다음 조건문에서 삭제됨  if(items.size() == 0 || (items.size() < 4 && chrono::duration\_cast<chrono::milliseconds>(gametime - items[items.size()-1].droptime).count() > 5000)){  random\_device rd; // 랜덤한 좌표로 생성  mt19937 gen(rd());  uniform\_int\_distribution<int> randrow(0, row-1);  uniform\_int\_distribution<int> randcol(0, col-1);  uniform\_int\_distribution<int> randitem(0, 4);  bool check = false;  int torow=0, tocol=0;  int item\_spec[5] = {5, 6, 5, 6, 5}; // Grow와 Poison의 생성 확률 비율을 3:2로 설정  do{  check = false;  torow = randrow(gen);  tocol = randcol(gen);  // 생성된 좌표가 Snake와 겹치거나, 타 아이템에 겹치거나, 벽과 겹치면 좌표를 다시 생성  for(int i=0; i<BODY.size(); i++){  if(BODY[i].getX() == torow && BODY[i].getY() == tocol){  check = true;  break;  }  if(BODY[i].getX()-2<torow&&BODY[i].getX()+2>torow&&BODY[i].getY()-2<tocol&&BODY[i].getY()+2>tocol){  check = true;  break;  }  if(Map[torow][tocol] != 0){  check = true;  break;  }  for(int i=0; i<items.size(); i++)  if(torow == items[i].pos.getX() && tocol == items[i].pos.getY()){  check = true;  break;  }  }  } while(check);  items.push\_back({{torow, tocol}, item\_spec[randitem(gen)], gametime});  }  // 아이템이 3개 전부 생성된 상태라면, 가장 오래된 아이템 삭제  if(items.size() > 0 && items.size() == 4 && chrono::duration\_cast<chrono::milliseconds>(gametime - items[0].droptime).count() > 5000)  items.erase(items.begin());  }  // Snake의 머리와 아이템이 겹친다면, 해당하는 아이템을 삭제함  void Item::delItem(Pos head){  for(int i=0; i<items.size(); i++)  if(head.getX() == items[i].pos.getX() && head.getY() == items[i].pos.getY()){  items.erase(items.begin() + i);  break;  }  }  vector<item\_type> Item::getItem(){  return items;  } |

|  |
| --- |
| Gate.cpp |
| #include "Gate.h"  Gate::Gate(int R, int C): row(R), col(C){};  bool Gate::makeGates(vector<vector<int>> map, int score, chrono::time\_point<chrono::system\_clock> nowtime, bool isWarped){  if(score >= 5 && gates.empty()){ // 게이트가 없는 상태에서 Grow 아이템을 5개 섭취 시 게이트 생성  random\_device rd; // 범위 내에 게이트 좌표 생성  mt19937 gen(rd());  uniform\_int\_distribution<int> randrow(0, row-1);  uniform\_int\_distribution<int> randcol(0, col-1);  bool check = false;  int torow = 0, tocol = 0;  do{  check = true;  torow = randrow(gen);  tocol = randcol(gen);  int d[4][2] = {{1, 0}, {0, -1}, {-1, 0}, {0, 1}};  for(int i=0; i<4; i++){  int nextx = torow + d[i][0];  int nexty = tocol + d[i][1];  if(nextx < 0 || row <= nextx || nexty < 0 || col <= nexty){  continue;  }  if(map[nextx][nexty] != 1 && map[nextx][nexty] != 2){  check = false;  }  }  } while(map[torow][tocol] != 1 || check); // 게이트가 벽 위에 생성되지 않았다면(Immune Wall 제외) 좌표 재생성, 게이트의 사방이 모두 막혀있다면 좌표 재생성  gates.push\_back({torow, tocol});  do{  check = false;  torow = randrow(gen);  tocol = randcol(gen);  int d[4][2] = {{1, 0}, {0, -1}, {-1, 0}, {0, 1}};  for(int i=0; i<4; i++){  int nextx = torow + d[i][0];  int nexty = tocol + d[i][1];  if(nextx < 0 || row <= nextx || nexty < 0 || col <= nexty){  continue;  }  if(map[nextx][nexty] != 1 && map[nextx][nexty] != 2){  check = false;  }  }  } while(map[torow][tocol] != 1 || (torow == gates[0].getX() && tocol == gates[0].getY()) || check); // 나머지 게이트쌍도 마찬가지로 생성. 다른 쌍과 좌표가 겹치지 않도록 한다.  gates.push\_back({torow, tocol});  madetime = nowtime; // 게이트가 생성된 시각 저장  }  if(chrono::duration\_cast<chrono::milliseconds>(nowtime - madetime).count() > 15000 && isWarped == false){ // 게이트가 생성된지 15초가 지났고, Snake가 게이트를 통과하고 있지 않다면 게이트 삭제  gates.clear();  return true;  }  return false;  }  // 게이트 통과 후 머리의 좌표  Pos Gate::headPath(vector<vector<int>> map, Pos t, char d){  if(!gates.empty() && t.getX() == gates[0].getX() && t.getY() == gates[0].getY()){  if(d == 'l'){  if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return {gates[1].getX(), gates[1].getY()-1};  else if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return {gates[1].getX()-1, gates[1].getY()};  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return {gates[1].getX()+1, gates[1].getY()};  else if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return {gates[1].getX(), gates[1].getY()+1};  }  else if(d == 'r'){  if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return {gates[1].getX(), gates[1].getY()+1};  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return {gates[1].getX()+1, gates[1].getY()};  else if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return {gates[1].getX()-1, gates[1].getY()};  else if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return {gates[1].getX(), gates[1].getY()-1};  }  else if(d == 'u'){  if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return {gates[1].getX()-1, gates[1].getY()};  else if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return {gates[1].getX(), gates[1].getY()+1};  else if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return {gates[1].getX(), gates[1].getY()-1};  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return {gates[1].getX()+1, gates[1].getY()};  }  else if(d == 'd'){  if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return {gates[1].getX(), gates[1].getY()-1};  else if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return {gates[1].getX()-1, gates[1].getY()};  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return {gates[1].getX()+1, gates[1].getY()};  else if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return {gates[1].getX(), gates[1].getY()+1};  }  }  if(!gates.empty() && t.getX() == gates[1].getX() && t.getY() == gates[1].getY()){  if(d == 'l'){  if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return {gates[0].getX(), gates[0].getY()-1};  else if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return {gates[0].getX()-1, gates[0].getY()};  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return {gates[0].getX()+1, gates[0].getY()};  else if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return {gates[0].getX(), gates[0].getY()+1};  }  else if(d == 'r'){  if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return {gates[0].getX(), gates[0].getY()+1};  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return {gates[0].getX()+1, gates[0].getY()};  else if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return {gates[0].getX()-1, gates[0].getY()};  else if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return {gates[0].getX(), gates[0].getY()-1};  }  else if(d == 'u'){  if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return {gates[0].getX()-1, gates[0].getY()};  else if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return {gates[0].getX(), gates[0].getY()+1};  else if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return {gates[0].getX(), gates[0].getY()-1};  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return {gates[0].getX()+1, gates[0].getY()};  }  else if(d == 'd'){  if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return {gates[0].getX(), gates[0].getY()-1};  else if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return {gates[0].getX()-1, gates[0].getY()};  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return {gates[0].getX()+1, gates[0].getY()};  else if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return {gates[0].getX(), gates[0].getY()+1};  }  }  return t; // 게이트를 통과하지 않는다면 기존 머리 좌표를 그대로 리턴  }  // 게이트 통과 후 진행 방향  char Gate::dirPath(vector<vector<int>> map, Pos t, char d){  if(!gates.empty() && t.getX() == gates[0].getX() && t.getY() == gates[0].getY()){  if(d == 'l'){  if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return 'l';  else if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return 'u';  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return 'd';  else if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return 'r';  }  else if(d == 'r'){  if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return 'r';  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return 'd';  else if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return 'u';  else if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return 'l';  }  else if(d == 'u'){  if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return 'u';  else if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return 'r';  else if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return 'l';  else if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return 'd';  }  else if(d == 'd'){  if(gates[1].getX()+1<row && (map[gates[1].getX()+1][gates[1].getY()] != 1 && map[gates[1].getX()+1][gates[1].getY()] != 2))  return 'd';  else if(gates[1].getY()-1>=0 && (map[gates[1].getX()][gates[1].getY()-1] != 1 && map[gates[1].getX()][gates[1].getY()-1] != 2))  return 'l';  else if(gates[1].getY()+1<col && (map[gates[1].getX()][gates[1].getY()+1] != 1 && map[gates[1].getX()][gates[1].getY()+1] != 2))  return 'r';  else if(gates[1].getX()-1>=0 && (map[gates[1].getX()-1][gates[1].getY()] != 1 && map[gates[1].getX()-1][gates[1].getY()] != 2))  return 'u';  }  }  if(!gates.empty() && t.getX() == gates[1].getX() && t.getY() == gates[1].getY()){  if(d == 'l'){  if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return 'l';  else if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return 'u';  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return 'd';  else if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return 'r';  }  else if(d == 'r'){  if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return 'r';  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return 'd';  else if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return 'u';  else if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return 'l';  }  else if(d == 'u'){  if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return 'u';  else if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return 'r';  else if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return 'l';  else if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return 'd';  }  else if(d == 'd'){  if(gates[0].getX()+1<row && (map[gates[0].getX()+1][gates[0].getY()] != 1 && map[gates[0].getX()+1][gates[0].getY()] != 2))  return 'd';  else if(gates[0].getY()-1>=0 && (map[gates[0].getX()][gates[0].getY()-1] != 1 && map[gates[0].getX()][gates[0].getY()-1] != 2))  return 'l';  else if(gates[0].getY()+1<col && (map[gates[0].getX()][gates[0].getY()+1] != 1 && map[gates[0].getX()][gates[0].getY()+1] != 2))  return 'r';  else if(gates[0].getX()-1>=0 && (map[gates[0].getX()-1][gates[0].getY()] != 1 && map[gates[0].getX()-1][gates[0].getY()] != 2))  return 'u';  }  }  return d; // 게이트를 통과하지 않는다면 기존의 진행 방향 그대로 리턴  }  vector<Pos> Gate::getGates(){  return gates;  } |

# 자기평가

|  |
| --- |
| **작성요령 (5점)**  **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.** |

프로젝트 시작 1일차에 1, 2단계를 구현, 2일차에 3, 4단계를 구현하였고, 3일차에 마지막 단계 구현과 보고서 작성을 하였다.

교수님께서 팀프로젝트로 진행해보는 것이 도움이 될 것이라고 하셔서, 미리 생각해둔 사람과 팀을 만들어 프로젝트를 진행하고자 하였지만, 이미 팀이 완성되었기에 혼자서 프로젝트를 진행하게 되었다.

C++이나 코드를 작성하는 데에는 익숙하기에 프로젝트를 구현하는 것 자체에는 큰 어려움이 없었으나, 앞으로 팀 단위로 일하는 경우가 많을텐데, 하나라도 더 팀 경험을 쌓지 못했다는 사실은 꽤나 아쉽게 다가왔다.

다음 학기에 이런 기회가 또 온다면 먼저 적극적으로 팀을 만들어볼 것이다.

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | pdf | 2021 C++ 프로젝트.pdf | 오픈소스디자인2021 | 2021 | 국민대학교 소프트웨어융합대학 |  |
| 2 | pdf | 90.ncurses\_tutorial.pdf | 오픈소스디자인2021 | 2021 | 김준호 |  |
| 3 | github | parkgeonhu/snake-game | github/parkgeonhu | 2020 | 박경후 | nodelay() 함수 사용을 통해 getch() 대기 무시 방법을 알아냄 |

# 부록

|  |
| --- |
| **작성요령 (15점)**  **프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.** |

## 사용자 매뉴얼

1. Project\_Snake.zip의 압축을 해제한다.
2. Project\_Snake/5/ 경로로 들어간다.
3. Makefile을 이용하여 make를 수행한다.
4. 터미널에 ./main을 입력하여 프로그램을 실행한다.
5. 게임 스타트 화면이 나타나면 ‘e’키를 입력하여 게임을 스타트한다.
6. 하얀색은 빈 공간, 회색은 Wall, 검은색은 Immune Wall을 의미한다.
7. 연노란색은 Snake Head, 주황색은 Snake Body를 의미한다.
8. 초록색은 Grow아이템, 빨간색은 Poison아이템, 연보라색은 Gate를 의미한다.
9. 키보드 방향키를 통해 방향을 지정할 수 있다.
10. 이외의 규칙은 제시된 바와 같다.
11. 미션을 전부 달성 시 다음 맵으로 넘어간다.

## 설치 방법

Makefile을 준비해두었으므로 터미널 경로 위에 ‘make’를 입력하여 컴파일을 완료할 수 있다. 이후 ‘./main’을 입력하여 프로그램을 실행한다.