**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | 슈슉 뱀 |
| 팀 명 | 아스키 |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.2 |
| **Date** | 2024-Jun-16 |

|  |  |
| --- | --- |
| **팀원** | 송예찬 (팀장) |
| 조민석 |
|  |
|  |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “ 슈슉 뱀 ”을 수행하는 팀 “아스키”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “아스키”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-프로젝트명.doc |
| **원안작성자** | 송예찬, 조민석 |
| **수정작업자** | 송예찬, 조민석 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2024-06-16 | 송예찬 | 1.0 | 최초 작성 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**본 양식은 소프트웨어학부 C++프로그래밍 과목의 프로젝트 결과보고서 작성을 위한 기본 양식입니다. 문서의 필수 항목을 제시하는 것이니 폰트, 문단 구조 등의 디자인 부분은 자유롭게 설정하기 바랍니다. 양식 내에 붉은 색으로 기술한 부분은 지우고 작성하기 바랍니다.**

**목 차**

[1 개요 4](#_Toc43103653)

[2 개발 내용 및 결과물 5](#_Toc43103654)

[2.1 목표 5](#_Toc43103655)

[2.2 개발 내용 및 결과물 6](#_Toc43103656)

[2.2.1 개발 내용 6](#_Toc43103657)

[2.2.2 시스템 구조 및 설계도 6](#_Toc43103658)

[2.2.3 활용/개발된 기술 6](#_Toc43103659)

[2.2.4 현실적 제한 요소 및 그 해결 방안 6](#_Toc43103660)

[2.2.5 결과물 목록 7](#_Toc43103661)

[3 자기평가 8](#_Toc43103662)

[4 참고 문헌 8](#_Toc43103663)

[5 부록 8](#_Toc43103664)

[5.1 사용자 매뉴얼 8](#_Toc43103665)

[5.2 설치 방법 8](#_Toc43103666)

# 개요

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

**프로젝트의 전체적인 구조 및 개발 내용을 명확하게 기술한다**

* **개발환경 (MAC + VsCode, Window + VsCode, Window + Visual Studio)**
* **ncurse, pdcurse 설치 방법**

**MAC : homebrew를 이용해 ncurse 설치. brew install ncurse**

**Window : pdcurse 설치 (**[**https://github.com/wmcbrine/PDCurses/releases/tag/3.9**](https://github.com/wmcbrine/PDCurses/releases/tag/3.9)**)**

**Or vcpkg를 이용해 pdcurse 설치 (https://github.com/microsoft/vcpkg)**

* **IDE 세팅**

**VsCode – Mac : task.json 의 args에 –** **lncurses 옵션 추가**

**VsCode – Window : 설치한 pdcurse를 readme에 따라 wincon경로에 cmd를 열어 각자 자신의 mingw 버전에 따라 make -f Makefile 을 실행. (이 팀의 경우 mingw32-make를 사용. 설치된 mingw의 bin에 가면 확인가능). 여러 파일들이 생성되는데 pdcurses.a를 libpdcurses.a로 변경. task.json 의 args에 -lpdcurses, -I 에 PDCurses 폴더 경로 추가, -L에 wincon 경로 추가.**

**Visual Studio : vcpkg 설치 후 pdcurse 설치 시 자동으로 Visual Studio 적용됨.**

**이 프로젝트는 크게 (1) Map.cpp, (2) snake.cpp, (3)Gamemanger.cpp, (4) main.cpp로 구성되어 있다. Main을 제외한 파일은 헤더파일도 같이 구성되어 있다.**

1. **Map.cpp**

**Map class와 Position 구조체, Gate\_Position 구조체로 구성됨.**

**Position 구조체 : x, y 좌표 변수**

**Gate\_Position 구조체 : Position 상속, gate가 들어오고 나갈 수 있는 방향 배열**

**Map class :**

* + **생성자에 map 이중 포인터와 new 키워드를 이용한 이중 배열 생성**
  + **소멸자에 delete 키워드를 이용해 삭제**
  + **기타 map관련 함수들이 존재**
  + **map을 초기화하는 함수**
  + **스네이크, 아이템, 게이트의 위치를 map에 반영**
  + **아이템, 게이트 위치 랜덤 생성**
  + **게이트 들어오고 나갈 수 있는 방향 설정 등**

1. **Snake.cpp**

**Snake를 class로 구현했다.**

**멤버변수로는 snake의 헤드와 몸통의 좌표를 저장하는 snake\_body, 방향을 저장하는 direction, 각 아이템들과 게이트 사용횟수를 저장하는 cnt\_growth, cnt\_poison, cnt\_gate, 그리고 snake가 게이트를 통과하는 중인지 확인하는 변수인 use\_gate, 마지막으로 snake의 꼬리가 이전에 있던 좌표를 저장하는 prev\_tail를 구현했다.**

**멤버함수로는 방향을 입력받아 방향을 바꾸는 set\_direction()함수, 매 틱마다 움직임을 구현하는 move(), 각 아이템과의 상호작용을 하는 eat\_growth()와 eat\_poison(), snake를 초기값으로 리셋하는 reset(), 게이트와 상호작용하는 go\_gate(), 마지막으로 각 멤버함수를 받아오는 함수들인 get\_direction(), get\_cnt\_growth(), get\_cnt\_poison(), get\_cnt\_gate()들로 구현했다.**

1. **Gamemanger.cpp**

**Gamemanger class와 Score\_Board구조체, Mission\_Board 구조체로 구성됨.**

**Score\_Board 구조체 : 현재, 최대 스네이크 길이, 아이템, 게이트 사용 횟수 변수**

**Mission\_Board 구조체 : 목표 스네이크 길이, 목표 아이템, 게이트 횟수 및 충족 여부 변수**

**Gamemanger class :**

* + **생성자에 curse 초기 세팅 및 스테이지 별 미션 모드 초기화**
  + **시간 측정을 위한 시작 시간과 중간 시간 체크 함수**
  + **스테이지 클리어, 게임 종료 조건 확인 함수**
  + **시간을 확인해 아이템과 게이트를 생성 및 삭제 판단 함수**
  + **미션 체크 및 디스플레이 함수**

1. **main.cpp**

**헤더파일에 선언된 클래스들을 선언. Initscr()으로 curse 사용. 아이템과 게이트 생성과 소멸을 위한 bool 변수 선언. 키 입력 시 게임 시작. 기 후 while 무한루프를 반복하며 게임 실행. 게임을 다 깨거나 중간에 죽으면 루프 탈출. Endwin()으로 curse 사용종료**

# 개발 내용 및 결과물

## 목표

|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.** |

|  |  |  |
| --- | --- | --- |
| 적용단계 | 내용 | 적용 여부 |
| 1단계 | Map의 구현 | 적용/미적용 |
| 2단계 | Snake 표현 및 조작 | 적용/미적용 |
| 3단계 | Item 요소의 구현 | 적용/미적용 |
| 4단계 | Gate 요소의 구현 | 적용/미적용 |
| 5단계 | 점수 요소의 구현 | 적용/미적용 |

**프로젝트의 각 적용 단계별 구현 목표를 명확하게 제시한다. 제시한 권고안의 내용을 포함하여, 변경된 부분 등을 구체적으로 단계별 구현 목표를 작성해야 한다.**

1단계 : pdcurse library 함수들을 사용하여 2차원 배열로 된 snake Map을 game 화면으로 표시하는 프로그램을 완성한다. 단, wall과 immune wall을 잘 구분한다.

2단계 : 맵 위에 snake를 표시하고, 방향키를 입력 받아 snake가 움직이도록 구현한다.

3단계 : 맵 위에 growth, poison, 그리고 special아이템과을 구현하고 snake와 상호작용 처리한다.

4단계 : 맵 위에 임의의 위치에 한 쌍의 gate가 출현하고 각 gate에 snake가 통과할 수 있도록 구현한다. 단, snake가 gate를 통과 중일 시 gate를 생성하지 않는다.

5단계 : 우측에 게임 점수를 표시하는 화면 구성과 stage를 4개로 설정하고 mission을 달성하면 다음 맵으로 진행하도록 구현한다.

## 개발 내용 및 결과물

### 개발 내용

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

**1단계 :**

* **map의 구성 요소**

**Map class :**

* + **생성자에 map 이중 포인터와 new 키워드를 이용한 이중 배열 생성**
  + **소멸자에 delete 키워드를 이용해 삭제**
  + **map을 초기화하는 함수**
* **map과 다른 객체와의 상호작용**
  + **스네이크 위치를 map에 반영**

**2단계 :**

* **snake 키 입력 처리 방법**

**chrono의 milliseconds()함수와 thread의 sleep\_for()함수를 이용하여 프로그램을 지연시키도록 구현하고 이후 getch()함수를 이용하여 키보드 입력을 읽고 입력 받은 키에 따라 snake의 진행방향을 바꾸도록 구현**

* **snake의 움직임 표현 방법**

**vector로 구현된 snake에서 몸통부분은 앞 인덱스의 좌표를 받아오고 헤더부분인 0번 인덱스의 좌표를 진행 방향에 따라 1움직이도록 구현한다.**

* **맵 위에 snake 표시 -> map과 다른 객체와의 상호작용**
* **Tick에 대한 변화 주기**

**스테이지 클리어시 Tick이 빨라지게 되도록 설정.**

**3단계 :**

* **Item 생성 방법 및 주기**

**Map class :**

* + **아이템, 게이트의 위치를 map에 반영**
  + **아이템, 게이트 위치 랜덤 생성 random으로 좌표를 생성하고 그 좌표가 적절한 좌표가 아니면 다음 인덱스의 좌표를 이용.**
  + **게이트 들어오고 나갈 수 있는 방향 설정 등**

**Gamemanger class :**

* + **시간을 확인해 아이템과 게이트를 생성 및 삭제 판단 함수**
  + **시간을 특정 주기로 나누고 주기의 일정한 구간 동안 아이템과 게이트가 유지됨.**
* **배열에 표현되는 값의 종류들(item + snake)**

**10 : grow\_item, 11 : poison\_item, 13 : special\_item**

**20: snake 머리, 21 : snake 몸**

* **Snake와 item의 상호작용 처리**

**Growth item을 먹을 시 push\_back()함수를 사용하여prev\_tail에 저장된 좌표를 snake\_body에 마지막에 넣어 꼬리부분 뒤로 몸 길이가 1길어지도록 구현하였고 poison item을 먹을 시 push\_pop()을 통해 꼬리부분이 없어지며 몸 길이가 1 줄어들도록 구현했다.**

* **새로운 item**

**Snake의 길이가 1 늘어나고 tick이 빨라지는 item 구현**

**4단계 :**

* **배열에 표현되는 값의 종류들(map +gate)**

**0 : 빈공간, 1 : 벽, 2 끼인 벽,**

**12 : gate**

* **Snake wall 상호작용**

**Gamemanger class에서 snake 머리와 벽이 만나면 게임 종료.**

* **Snake gate 상호작용**

**한 쌍의 gate들의 좌표와 진행가능한 방향을 받아 snake의 헤드부분이 gate좌표와 겹쳤을 때 snake의 헤드부분의 좌표를 다른 gate에서 가능한 진행방향을 계산한 후 진행방향에 따라 gate좌표에 1움직인 좌표를 받도록 하여 gate를 통과하도록 구현하였고 나머지 몸통부분은 움직일 때 앞 인덱스를 받아오도록 구현했기 때문에 gate를 잘 통과할 것이다. 또한 gate를 통과 중인지 알기 위해 use\_gate를 이용하여 헤드부분이 gate를 지날 때 use\_gate++를 해주고 이후 use\_gate의 값이 snake\_body의 길이와 같을 때까지 1증가 시키며 통과 중이라는 표시로 true값을 리턴하고 true 값일 때 gate가 생성되지 않도록 구현했다.**

* **Wall에 대한 변화**

**스테이지 변경 시 외곽에만 있던 벽이 빈공간 중간에도 생성됨. map이 논리적인 구조로 생성되기 때문에 map의 크기와 상관없이 벽의 비율이 유지됨.**

**5단계 :**

* **우측에 게임 점수 score board와 mission board**

**Score\_Board 구조체 : 현재, 최대 스네이크 길이, 아이템, 게이트 사용 횟수 변수**

**Gate\_Position 구조체 : 목표 스네이크 길이, 목표 아이템, 게이트 횟수 및 충족 여부 변수**

**Gamemanger class :**

* + **Score\_Board 와 Mission\_Board 을 비교해 미션완료 여부를 체크함**
  + **위의 구조체 정보를 바탕으로 디스플레이**
* **Score 보드에 시간과 속도 표시**

**틱에 대한 변화를 이용해 속도를 표시하고, 시간을 표시하여 gate와 item의 생성 주기를 이용하도록 함**

### 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.** |

**소스코드 나열 및 설명**

**<개발자 : 송예찬>**

static const auto tickDuration = std::chrono::milliseconds(tick\_time);

std::this\_thread::sleep\_for(tickDuration);

while (true) {

int ch = getch();

if (ch == ERR) break;

switch (ch) {

case 'q':

quit = true;

break;

case KEY\_LEFT:

direction = 'l';

break;

case KEY\_RIGHT:

direction = 'r';

break;

case KEY\_UP:

direction = 'u';

break;

case KEY\_DOWN:

direction = 'd';

break;

}

}

Tick time만큼의 tick를 만들고 이후 getch()함수로 입력을 받는다. 만약 입력이 없을 시 기존 방향을 유지하고 방향키 입력이 있으면 해당 방향으로 direction을 갱신해준다.

// save tail position

prev\_tail = snake\_body.back();

// move snake body

for (int i = snake\_body.size() - 1; i > 0; i--) snake\_body[i] = snake\_body[i - 1];

// move snake head

switch (direction) {

case 'l':

snake\_body[0].x--;

break;

case 'r':

snake\_body[0].x++;

break;

case 'u':

snake\_body[0].y--;

break;

case 'd':

snake\_body[0].y++;

break;

}

Prev\_tail에 꼬리부분의 값을 저장한 후 몸통부분을 앞 인덱스의 좌표값을 받도록 하고 헤드부분의 값은 현재 진행 방향에 따라 좌표값을 1조정하도록 하여 snake의 움직임을 구현했다.

if (snake\_body[0].x == item.x && snake\_body[0].y == item.y) {

snake\_body.push\_back(prev\_tail);

cnt\_growth++;

}

if (snake\_body[0].x == item.x && snake\_body[0].y == item.y) {

snake\_body.pop\_back();

cnt\_poison++;

}

Snake의 헤드부분이 아이템과 겹쳤을 때 growth item이면 prev\_tail을 snake\_body에 push\_back()하고 poison item이면 pop\_back()하고 각 아이템을 먹은 횟수를 저장하는 변수인 cnt\_growth와 cnt\_poison을 각각 1증가시킨다.

**<개발자 : 조민석>**

class Map

{

private:

random\_device rd;

/\* data \*/

public:

int\*\* map; // map 이중 포인터

int map\_size{};

Map(int map\_size=21); // 이중 포인터 new로 배열 할당

~Map();

void reset\_map(int stage=0); // 스테이지별 map 초기화

// 스네이크, 아이템, 게이트의 위치를 map에 반영

void snake\_to\_map(Snake &snake);

void grow\_item\_to\_map(Position grow);

void poison\_item\_to\_map(Position poison);

void special\_item\_to\_map(Position special);

void gate\_to\_map(Gate\_Position gate);

// 아이템, 게이트의 위치 생성

Position create\_item();

Gate\_Position create\_gate();

// map 범위 밖 검사

bool check\_map\_Pos(int x, int y);

// 게이트 방향 찾기

Gate\_Position reset\_gate\_direction(Gate\_Position gate);

Gate\_Position find\_gate\_direction(Gate\_Position gate);

};

class Gamemanger

{

private:

int grow\_create\_count{-1}, poison\_create\_count{-1}, special\_create\_count{-1};

public:

// 속도를 계산하기 위한 tick 시간

int tick\_time{300};

int stage=0;

// 미션과 스코어를 기록하기 위한 구조체 생성

Score\_Board score\_board;

Mission\_Board mission\_board;

// 시간 측정을 위한 변수 선언

chrono::system\_clock::time\_point start;

Gamemanger();

~Gamemanger();

// 시작 시각 기록

void get\_game\_start\_time();

// 함수 호출 시 중간 시각과 시작 시각 자로 진행 시간 파악

double get\_game\_progress\_time();

// 색깔, 입력 등 여러 초기 세팅

void initial\_setting();

// 스테이지 별 미션 초기화

void reset\_mission\_by\_stage(bool test=false);

// 스테이지 클리여 여부 확인

bool is\_stage\_clear();

// 게임 종료 확인

bool is\_gameover(Map &Map, Snake &snake);

// 진행시간에 맞춰 아이템을 생성 혹은 삭제해도 되는지 확인

bool can\_create\_grow\_item();

bool can\_create\_poison\_item();

bool can\_create\_special\_item();

bool can\_create\_gate();

bool can\_delete\_item();

bool can\_delete\_gate(Snake snake, Gate\_Position gate1, Gate\_Position gate2);

// 아이템 사용 여부에 따라 점수판에 반경

bool is\_eat\_item(Position item, Snake &snake);

void update\_score\_board(bool grow=false, bool poison=false, bool gate=false);

void reset\_score\_board();

void check\_misson();

// 화면 출력

void display(Map &Map);

void display\_debug(Map &Map, Snake &snake, bool check=false, string test\_string="test");

void display\_map(Map &Map);

void display\_score(Map &Map);

void display\_misssion(Map &Map);

};

int main(){

initscr(); // 시작

// 각종 클래스 선언

Map Map;

Gamemanger GM;

Snake snake;

Position grow, poison, special;

Gate\_Position gate1, gate2;

string temp\_debug;

bool existence\_grow\_item{0}, existence\_poison\_item{0}, existence\_special\_item{0}, existence\_gate{0};

bool use\_gate\_start{1};

Map.snake\_to\_map(snake);

GM.display(Map);

while (getch()==ERR) // 아무키 입력시 게임 시작

GM.get\_game\_start\_time(); // 시작 시각 기록

while(!snake.quit){

snake.set\_direction(); // 입력

snake.move(); // 입력받은 방향대로 이동

// 아이템 생성

if(GM.can\_create\_grow\_item()){

if(!existence\_grow\_item){

grow=Map.create\_item();

Map.grow\_item\_to\_map(grow);

existence\_grow\_item=true;

}

}

if(GM.can\_create\_poison\_item()){

if(!existence\_poison\_item){

poison=Map.create\_item();

Map.poison\_item\_to\_map(poison);

existence\_poison\_item=true;

}

}

if(GM.can\_create\_special\_item()){

if(!existence\_special\_item){

special=Map.create\_item();

Map.special\_item\_to\_map(special);

existence\_special\_item=true;

}

}

// gate 생성

if(GM.can\_create\_gate()){

if(!existence\_gate){

gate1=Map.create\_gate();

gate1=Map.find\_gate\_direction(gate1);

Map.gate\_to\_map(gate1);

gate2=Map.create\_gate();

gate2=Map.find\_gate\_direction(gate2);

Map.gate\_to\_map(gate2);

existence\_gate=true;

}

}

// 아이템 획득 시 action

if(GM.is\_eat\_item(grow,snake)){

if(GM.score\_board.Current\_len!=GM.score\_board.Max\_len) snake.eat\_growth(grow);

GM.update\_score\_board(true,false,false);

}

if(GM.is\_eat\_item(poison,snake)){

snake.eat\_poison(poison);

GM.update\_score\_board(false,true,false);

}

if(GM.is\_eat\_item(special,snake)){

if(GM.score\_board.Current\_len!=GM.score\_board.Max\_len) snake.eat\_growth(special);

GM.update\_score\_board(true,false,false);

if(snake.tick\_time>=60){

snake.tick\_time=snake.tick\_time-30;

GM.tick\_time=snake.tick\_time;

}

}

// gate 통과 시 action

if(existence\_gate){

if(!snake.go\_gate(gate1,gate2)) use\_gate\_start=true;

else if(snake.go\_gate(gate1,gate2) && use\_gate\_start){

GM.update\_score\_board(false,false,true);

use\_gate\_start=false;

}

}

// snake head가 벽, snake 몸체을 만나거나 길이가 2아래면 종료

if(GM.is\_gameover(Map,snake)) break;

// 스테이지 클리어시 다음 스테이지로

if(GM.is\_stage\_clear()){

GM.stage++;

snake.tick\_time=300-(40\*GM.stage);

GM.tick\_time=snake.tick\_time;

if(GM.stage==5) break;

Map.reset\_map(GM.stage);

GM.reset\_score\_board();

GM.reset\_mission\_by\_stage();

snake.reset();

Map.snake\_to\_map(snake);

GM.display(Map);

while (getch()==ERR)

existence\_grow\_item=false;

existence\_poison\_item=false;

existence\_special\_item=false;

existence\_gate=false;

use\_gate\_start=true;

GM.get\_game\_start\_time();

continue;

}

// Map relaod

Map.reset\_map(GM.stage);

Map.snake\_to\_map(snake);

if(GM.is\_eat\_item(grow,snake) || GM.can\_delete\_item()) existence\_grow\_item=false;

if(existence\_grow\_item) Map.grow\_item\_to\_map(grow);

if(GM.is\_eat\_item(poison,snake) || GM.can\_delete\_item()) existence\_poison\_item=false;

if(existence\_poison\_item) Map.poison\_item\_to\_map(poison);

if(GM.is\_eat\_item(special,snake) || GM.can\_delete\_item()) existence\_special\_item=false;

if(existence\_special\_item) Map.special\_item\_to\_map(special);

if(GM.can\_delete\_gate(snake,gate1,gate2)){

use\_gate\_start=true;

existence\_gate=false;

}

if(existence\_gate){

Map.gate\_to\_map(gate1);

Map.gate\_to\_map(gate2);

}

// 출력하기

GM.display(Map);

}

endwin(); // 끝

}

### 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.** |

**\*Ncurse, pdcurse**

**Initscr()와 endwin() 사이에 curse 함수 사용.**

**raw(), keypad(stdscr, bool), noecho(), nodelay(stdscr, bool), curs\_set(int)를 통해 여러 설정 가능**

**init\_color()를 통해 rgb 색을 저장. init\_pair로 지정된 색으로 배경, 글자색을 조합하여 저장**

**newwin으로 포인터 변수를 선언하고 mvwprintw로 특정 좌표에 print 가능**

**\*Vector**

**stl의 컨테이너인 vector를 사용하여 snake의 헤드와 몸통의 좌표를 저장하고 push\_back()과 pop\_back()함수를 이용하여 growth와 poison아이템과 상호작용을 구현**

**\*Chrono, Thread**

**Chrono의 milliseconds()함수와 Thread의 sleep\_for()함수를 이용하여 프로그램을 일정시간을 지연시킬 수 있도록 구현**

### 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.** |

각자의 개발환경이 윈도우와 mac으로 달랐고 이로 인해 프로그램 지연을 위해 사용한 window.h 라이브러리의 sleep()함수가 윈도우에서 잘 작동했지만 mac환경에서는 작동되지 않는 제한 요소가 발생했다. 이를 해결하기 위해서 지연을 위한 다른 방법들을 찾아보았고 chrono라이브러리의 milliseconds()함수와 thread의 sleep\_for()함수를 같이 사용하여 mac과 윈도우 두 환경에서 모두 프로그램을 일정시간 지연시킬 수 있도록 구현했다.

### 결과물 목록

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

1. **Snake.cpp**

**Snake를 class로 만들어 현재 진행 방향과 몸통의 좌표, 그리고 게이트와 아이템 사용 횟수를 저장한다. 그리고 snake의 방향전환, 아이템과 게이트의 상호작용을 구현**

**멤버변수**

**cnt\_growth : growth아이템 먹은 횟수**

**cnt\_poison : poison아이템 먹은 횟수**

**cnt\_gate : gate 통과 횟수**

**use\_gate : snake가 gate를 통과하는 중인지 확인용 변수**

**prev\_tail : snake 꼬리 부분이 전에 위치한 좌표저장**

**snake\_body : snake의 머리와 몸통 부분의 좌표값들**

**멤버함수**

**Set\_direction() : 키보드로 방향을 입력 받아 방향을 바꿈**

**Move() : snake의 1틱에 움직임을 구현**

**Eat\_growth(const Position item) : growth 아이템과의 상호작용**

**Eat\_posion(const Position item) : poison 아이템과의 상호작용**

**Reset() : snake의 모든 값을 초기값으로 초기화**

**Go\_gate() : gate와의 상호작용**

**Get\_direction() : 멤버변수 direction을 받는다.**

**Get\_cnt\_growth() const : 멤버변수 cnt\_growth를 받는다.**

**Get\_cnt\_poison() const : 멤버변수 cnt\_poison를 받는다.**

**Get\_cnt\_gate() const : 멤버변수 cnt\_gate를 받는다.**

1. **Map.cpp**

**Map을 관리하는 파일이다.**

**아이템, 게이트의 좌표를 생성한다**

**아이템, 게이트, 스네이크를 map에 반영한다.**

1. **Gamemanger.cpp**

**게임의 보조적인 기능을 담당하는 파일이다.**

**기본적으로 다른 파일들의 정보를 score, mission board를 관리하고 화면을 출력한다.**

**Curse의 초기 설정, 시간관리, 시간을 이용한 아이템, 게이트 생성 및 소멸 관리를 한다.**

1. **Main.cpp**

**각 파일들의 기능을 바탕으로 게임을 진행한다.**

**While 문을 통해 게임을 진행하고 클리어 혹은 종료조건이 나오면 프로그램을 종료한다.**

1. **영상 링크 :** <https://youtu.be/f8KlR6Fnb4w>

# 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.** |

**<송예찬>**

**저의 역할은 snake를 class로 snake가 방향을 입력 받는 것 그리고 기본적인 움직임, 아이템과 gate와의 snake의 상호작용 시 작동을 구현하는 것을 담당했습니다.**

**먼저 go\_gate()함수의 코드 간결화가 가장 아쉽습니다. Gate의 진행방향을 계산하여 결정하는 코드를 각 방향, 각 gate마다 계산하여 결정하는 것이 아닌 진출할 게이트의 진출가능 방향을 받아 계산하여 공통된 부분을 줄일 수 있었을 것입니다. 또한 구조체를 담은 vector로 snake의 좌표를 표현했는데 아이템을 먹어 snake\_body에 구조체를 push하거나 pop할 때 메모리 측면에서 하나도 생각하지 않았다는 것이 개인적으로 아쉬운 부분입니다.**

**이 프로젝트를 수행할 때 팀원과 서로 구현이 필요한 부분을 소통을 하며 맞춰가는 것과 외부 라이브러리 설치와 사용하는 것이 어려웠습니다. 하지만 계속해서 팀원과 소통하며 필요한 구현을 서로 맞춰가며 구현을 했고 외부 라이브러리도 인터넷 검색을 통해 문제를 해결했습니다.**

**이 프로젝트로 다양한 라이브러리의 사용, class구현, 헤더파일로 분리하는 등 수업에서 배운 내용을 활용하며 조금 더 c++라는 언어를 다루는 것에 도움이 되었습니다.**

**이 프로젝트 운영에 있어 stl강의가 학기 마지막에 위치해 c++언어에 익숙한 사람이 아니면 stl을 충분히 활용하기 힘들다고 생각된다. 이러한 부분에서 stl를 쉽게 사용할 수 있도록 개선이 필요하다고 생각된다.**

**<조민석>**

**처음 curse 세팅을 하는 것이 힘들었다. 서로 다른 환경을 사용했지만 결국은 잘 해결했다. 코드의 양으로 보면 많은 것을 구현한 것 같다. 하지만 main을 포함한 대부분을 구현하다 보니 class에서 private const를 제대로 활용하지 못한 것 같다. 다음에는 인원이 많을 때 좀 더 활용할 수 있는 기회가 있었으면 좋겠다.**

# 참고 문헌

**참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:**

웹페이지

* <https://magmatart.dev/development/2017/06/15/ncurses4.html>
* <https://docs.python.org/ko/3.14/howto/curses.html>
* https://jabdon4ny.tistory.com/76

# 부록

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

Github(https://github.com/kevincms/snake)에 있는 main.exe 실행

## 사용자 매뉴얼

**프로젝트 실행 후 사용안내, 따라하기 등 포함**

* 프로그램 실행 후 아무 키나 입력하면 게임 시작
* 중간에 나가고 싶으면 q 입력
* 죽거나 스테이지 모두 클리어 시 자동 프로그램 종료

## 설치 방법

**프로그램 컴파일 및 실행 방법 포함**

* 기본적으로 main.exe을 실행해도 되지만 맵의 사이즈를 변경하고 싶다면 map size를 변경후 새로 컴파일해 main.cpp를 만들 수 있다. vscode에서 task.json을 개요에서 설명한 것처럼 세팅한 뒤 컴파일을 해주면 된다.