**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Snake Game* |
| 팀 명 | *F* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.3 |
| **Date** | 2021-Jun-18 |

|  |  |
| --- | --- |
| **팀원** | 김재민 |
| 육종호 |
|  |
|  |
|  |

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

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-Snake\_Game.doc |
| **원안작성자** | 김재민, 육종호 |
| **수정작업자** | 김재민, 육종호 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2021-5-31 | 육종호 | 1.0 | 최초 작성 |  |
| 2021-06-02 | 김재민 | 1.1 | 내용 수정 | 수정된 연구내용 추가 |
| 2021-06-03 | 육종호 | 1.2 | 내용 수정 | 수정된 연구내용 추가 |
| 2021-06-18 | 김재민 | 1.3 | 최종 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.**

개발하고자 하는 프로그램은 <Snake Game> 으로, C++ 프로그래밍 언어를 기반으로 Ncurses 라이브러리를 사용하여 구현한다.

Ncurses 라이브러리 설치는 Ubuntu 기준, terminal에서 아래와 같이 수행한다.

1. 아래의 명령문을 사용하여 다운로드를 수행한다.

* wget https://ftp.gnu.org/pub/gnu/ncurses/ncurses-6.2.tar.gz -P ~/Downloads

2. 다운로드 파일 “ncurses-6.2.tar.gz” 압축을 해제한다.

* tar xzfv ncurses-6.2.tar.gz

3. 디렉터리 “ncurses-6.2” 로 이동 후 환경을 설정한다. (설치 시 make 가 필요, 설치가 안되어 있는 경우 “sudo apt install make” 사용하여 설치)

* ./configure

4. make를 사용하여 빌드하고 make install 을 사용하여 인스톨을 수행한다.

* make
* sudo make install

5. 라이브러리를 설치한다.

* sudo pat-get update
* sudo apt-get install libcurses5-dev libcursesw5-dev

설치 후 컴파일 방법은 아래와 같다.

* g++ -o snake 파일 -lncurses
* ./snake

<Snake Game> 의 전체적인 구조는 2차원 배열로 된 Snake Map, Map 위를 움직이는 Snake, Map 위에 출현하는 Growth Item 과 Poison Item, Map의 Immune Wall 과 Wall, Map 위에 출현하는 Gate, Game Score을 표시하는 화면으로 구성되어 있다.

프로그램을 제작하는 과정은 아래와 같은 순서로 진행한다.

* 1단계는 Ncurses Library 함수들을 사용하여 2차원 배열로 된 Snake Map을 Game 화면으로 표시하는 프로그램을 완성한다.
* 2단계는 1단계 맵 위에 Snake를 표시하고, 화살표를 입력 받아 Snake가 움직이도록 프로그램을 완성한다.
* 3단계는 2단계 프로그램에서 Map 위에 Growth Item과 Poison Item을 출현하도록 수정한다.
* 4단계는 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 수정한다.
* 5단계는 4단계 프로그램에서, 우측에 게임 점수를 표시하는 화면을 구성한다.
* 앞의 단계 외의 기능을 추가할 수 있다.

위 단계들을 통해 만들어진 <Snake Game> 은 2-1에서 설명할 규칙들을 준수하며 작동하고, 위 게임의 전반적인 과정은 방향기 조작을 통해 Snake을 움직이며 Growth Item을 획득하면 길어지고 Poison Item을 획득하면 몸이 짧아지며 Gate을 통과하며 이동할 수도 있고 이러한 과정들이 모두 Game Score에 기록된다. 그리고 Mission을 통해 Game Score의 각 항목들이 목표치를 도달하도록 값을 설정하여 게임을 진행한다.

프로그램의 최종 실행은 Snake\_game-main 폴더에 main.cpp, map,cpp 등 <Snake Game> 동작에 필요한 모든 파일들을 저장하고 g++ -o snake \*.cpp -lncurses 명령어를 통해 컴파일하고 ./snake 명령어를 통해 <Snake Game> 프로그램을 실행시키는 것이다.

# 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 – Map의 구현**

* Ncurses Library 함수들을 사용하여 2차원 배열로 된 Snake Map을 Game 화면으로 표시하는 프로그램을 완성한다. Map의 크기는 25\*25로 하였다. 주의할 점은 Wall과 Immune Wall을 잘 구분해야 한다. (ex -> Wall은 1로 표현, Immune Wall은 2로 표현) Wall은 Snake가 닿을 수 있는 벽을 뜻하고, Immune Wall은 Snake가 닿을 수 없는 각 모서리 벽을 뜻한다.

**2단계 – Snake 구현**

* 1단계 맵 위에 Snake를 표시하고, 화살표를 입력 받아 Snake가 움직이도록 프로그램을 완성한다. Snake의 Head와 Tail을 위의 Wall, Immune Wall과 같이 구분해야 한다. 그리고 Game Rule #1을 준수해야 한다.
* Game Rule #1
* Snake는 진행 방향의 반대방향으로 이동할 수 없다. Head 방향이 진행 방향이며 단 한번의 방향키 입력만으로 반대 방향인 Tail 방향으로 이동할 수 없다. 이동 시 실패(Game Over)로 간주한다.
* Snake는 자신의 Body를 통과할 수 없고 벽을 통과할 수 없다. Body 또는 Wall을 통과 시 실패(Game Over)로 간주한다.
* Head 방향 이동은 일정시간(틱)에 의해 이동한다.
* 진행방향과 같은 방향기 입력은 무시하고 방향키는 프로그램 사용자 키보드의 방향키 입력키를 통해 직접 입력하며, 방향키 입력 시 방향키의 방향으로 다음 틱에 Snake가 이동한다.

**3단계 - Growth Item과 Poison Item 구현**

* Growth Item과 Poison Item을 Map 배열에 표현할 때 값을 정한다. 화면상에 표현 시, 색이나 기호를 달리하여 구분할 수 있도록 한다. Growth Item과 Poison Item을 명확히 구분해 주도록 한다. 그리고 Game Rule #2를 준수해야 한다.
* Game Rule #2
* Snake의 이동방향에 Item이 놓여 있는 경우 Snake가 Item을 획득한다.
* Growth Item의 경우 몸의 길이(Tail)가 1 증가한다.
* Poison Item의 경우 몸의 길이(Tail)가 1 감소하며 몸의 길이가 3보다 작아지면 실패(Game Over)로 간주한다.
* Item은 Snake Body가 있지 않은 임의의 위치에 출현한다.
* Item은 출현 후 일정시간이 지나면 사라지고 다른 위치에 나타나야 한다.
* 동시에 출현할 수 있는 Item의 수는 2개로 제한한다.

**4단계 – Gate 구현**

* 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 수정한다. Wall(Immune Wall 포함)과 Gate를 Map배열에 표현할 때 값을 결정한다. 화면상에 표현 시, Gate는 Wall과 구분할 수 있도록 하고 다른 요소와 역시 구분할 수 있도록 한다. 그리고 Game Rule #3, #4, #5를 준수해야 한다.
* Game Rule #3
* Gate는 두 개가 한쌍이다.
* Gate는 겹치지 않는다.
* Gate는 임의의 위치에 있는 벽에서 나타난다.
* Gate에 Snake가 진입하면, 다른 Gate로 진출한다.
* Gate에 Snake가 진입중인 경우, Gate는 사라지지 않으며 다른 위치에서 Gate가 나타나지 않는다.
* Gate는 한 번에 한 쌍만 나타난다.
* Gate가 나타나는 벽이 가장자리에 있을 때 항상 Map의 안쪽 방향으로 진출한다. (상단 벽 -> 아래 방향, 하단 벽 -> 위 방향, 좌측 벽 -> 오른쪽 방향, 우측 벽-> 왼쪽 방향)
* Game Rule #4
* Gate가 나타나는 벽이 Map의 가운데 있을 때 다음의 순서로 진출한다.
  + - * + 진입 방향과 일치하는 방향이 우선
        + 진입 방향의 시계방향으로 회전하는 방향
        + 진입 방향의 역시계방향으로 회전하는 방향
        + 진입 방향과 반대방향
* Game Rule #5
* Wall은 Gate로 변할 수 있다.
* Immune Wall은 Gate로 변할 수 없다.
* 모든 Wall은 Snake가 통과할 수 없고 Snake Head와 충돌 시 실패(Game Over)로 간주한다.
* Gate는 게임 시작 후 일정 시간 후 출현하도록 한다.

**5단계 – Game Score 구현**

* 4단계 프로그램에서, 우측에 게임 점수를 표시하는 화면을 구성한다. 그리고 Mission을 설정하여 매 게임마다 임의의 값을 주는 방식으로 처리한다. Mission을 달성하면 다음 Map으로 진행하도록 프로그램을 완성한다. Stage는 최소 4개로 구성하고, 각 Stage의 Map은 서로 달라야 한다. 그리고 Game Rule #6를 준수해야 한다.
* Game Rule #6
* 다음과 같은 메뉴들로 Game Score을 구성한다.
* 게임 중 몸의 길이 계산
* 게임 중 획득한 Growth Item의 수
* 게임 중 획득한 Poison Item의 수
* 게임 중 Gate 사용 횟수
* 게임 시간(Seconds로 계산)

* 미션은 각 항목별로 목표치 도달 시 게임을 종료하도록 설정한다.

## 개발 내용 및 결과물

### 개발 내용

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

**1단계 – Map의 구현**map.txt 파일에 머리가 닿을 수 있는 벽(Wall)은 1, 닿지 못하는 벽(Immune Wall)은 2, 그 외에 뱀이 이동할 수 있는 공간은 0으로 설정하여 작성한다. C++의 fstream 라이브러리를 통해 0, 1, 2로 구성된 텍스트 파일을 불러와서 읽은 다음 2차원배열에 저장을 한다. 2는 \*, 1은 +, 0은 공백 ‘ ‘ 으로 지정하여 char 형 배열에 저장한다. 이 때 char 배열의 크기는 25\*25로 설정한다.

**2단계 – Snake 구현**

위와 같이 만들어진 맵 위에 표현될 Snake의 머리(head)와 꼬리(tail)를 각각 ‘@’, ‘0’으로 설정하고 deque를 통해 머리와 꼬리 좌표를 저장하여 Snake를 구현한다. 이 때 추후 Item 획득 여부에 따라 deque의 길이가 변경됨에 따라 Snake의 길이가 유동적으로 변하게 한다. 그리고 Game Rule #1을 준수하기 위해 Snake의 머리 방향을 기본적인 진행 방향으로 설정하고 방향키 입력을 받아 그 방향을 표현하기 위해서 char형 변수에 저장한다, 현재 진행 방향과 키입력을 받은 후의 char형 변수와 비교하여 방향이 다를 시, 새로 입력 받은 방향으로 전환하도록 한다. 이 때, 현재 진행 방향과 반대 방향의 키가 새로 입력되었을 시, 실패(Game over)로 간주되도록 구현하고. 현재 진행 방향과 같은 방향의 키가 새로 입력되었을 시, 입력을 무시하도록 구현한다. 또한 Snake의 이동은 0.1초마다 이루어지도록 구현한다. 마지막으로 Snake의 머리 좌표 값이 Body를 표현하는 deque 안의 좌표 값들이나 Wall의 좌표 값과 일치할 경우 실패(Game over)로 간주하도록 구현하였다.

**3단계 - Growth Item과 Poison Item 구현**

Growth Item은 ‘G’, Poison Item은 ‘S’로 설정하여 Map 배열에 표현한다. 이때 같은 자리에 계속해서 Item이 나오는 것을 방지하기 위해, ctime, cstdlib의 라이브러리를 이용하여 난수 생성하여 Map에 무작위로 출현하도록 구현하였다. 또한 Item은 5초마다 새로 갱신되도록 구현하였으며, Item의 최대 출현 개수는 2개로 제한하였다. 이렇게 출현한 Item이 뱀의 머리 좌표 값과 일치할 경우 Item 획득으로 간주하여 ‘G’일 때는 Snake의 길이를 1 증가시키고, ‘S’일 때는 Snake의 길이를 1 감소시킨다. 이 때 Snake의 길이가 2보다 작거나 같을 시 실패(Game Over)로 간주한다.

**4단계 – Gate 구현**

3단계에서의 Item과 마찬가지로 ctime, cstdlib의 라이브러리를 이용하여 난수 생성하되 Item과는 다르게 Map의 Wall(Immune Wall 제외)에 무작위로 한 쌍이 출현하도록 구현하였다. 이 때 Gate는 ‘#’으로 표현하였고 게임 시작 후 6초 이후에 출현하도록 하였다.

**5단계 – Game Score 구현**

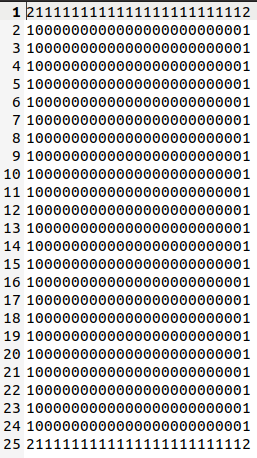
4단계 프로그램에서, 우측에 게임 점수를 표시하는 화면을 구성한다. 화면에는 Item을 획득했을 때의 누적 점수, 현재 Snake의 길이와 최대 Snake의 길이, 게임 중 획득한 Growth Item, Poison Item의 각각의 수, 마지막으로 게임 중 Gate 사용 횟수를 표현한다. 그리고 Game Score 밑에 Mission을 설정하였다.

### 시스템 구조 및 설계도

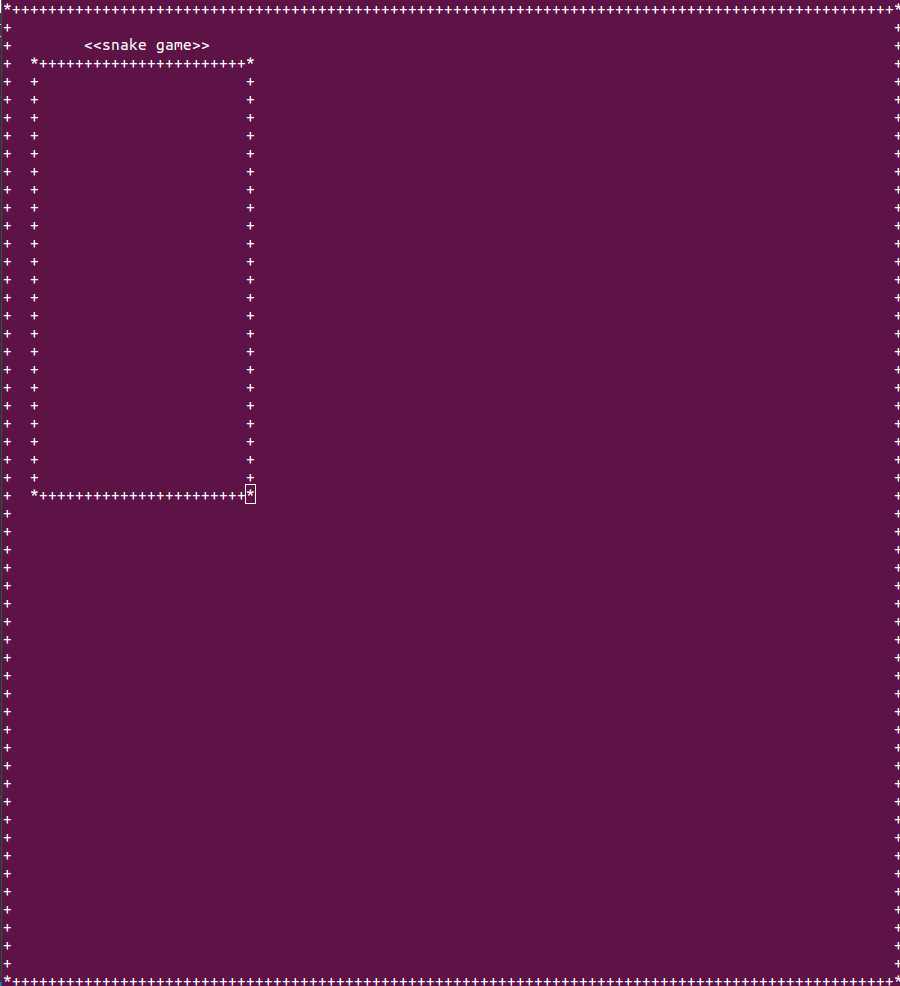
|  |
| --- |
| **작성요령 (30점)**  **프로젝트의 각 세부 목표의 주요 기능(알고리즘 등)에 대해서 기술한다. 세부 목표별로 수정한 프로그램 소스 파일을 나열하고, 해당 파일에서 세부 목표를 달성하기 위해 작성한 클래스/함수에 대해 나열하고, 각 요소에 대해 간략한 설명을 작성한다. 또한 각 요소의 개발자를 명시한다.** |

**1단계 – Map의 구현**

* map.h 파일 제작
* 내부에 class Map을 구현하고 Map을 표현하기 위해 크기가 25\*25인 char형 2차원 배열 v와 txt 파일을 불러와 줄 getMap() 함수, 화면에 그려줄 render() 함수를 선언한다. 이후 용이하게 접근할 수 있도록 접근 지정자는 public으로 지정한다.
* map.cpp 파일 제작
* “map.h” 헤더 파일을 불러오고 각 함수안의 내부 로직을 구현하였다.
* getMap() 함수
* getMap()은 fstream 라이브러리를 통해 .txt파일을 불러온다. 크기가 30인 char형 배열 buffer를 선언하여 한 라인 씩 받아와 v에 저장한다. if문으로 2는 \*, 1은 +, 0은 ' '(공백)으로 바꾸어 저장하였다.
* render() 함수
* 화면에 그리기 위해서 ncurses.h 라이브러리를 불러온다. 초기 화면의 크기는 60\*100으로 지정하였다. 외곽 테두리는 \*과 +로 두었고, 이 안에, 뱀이 그려질 새로운 화면(게임화면)을 넣어준다.
* 게임화면은 초기 창의 3,3 위치, 크기 25 \* 25 로 설정하였다. for 문을 통해 v에 저장된 값들을 화면에 출력해준다.
* 이후에 바뀐 것이 있다면, refresh를 통하여 바뀌는 부분을 다시 나타낸다. 그 이유는 추후 score Box, Mission Box는 이동 후에도 점수 또는 미션을 확인하여 변화가 있다면 보여주어야 하기 때문이다.
* Main.cpp파일 제작
* main() 함수
* map.h 헤더 파일을 불러와 Map Class의 객체 m을 선언한다.
* while 문에 무한 루프 조건을 넣고, render함수를 불러온다. unistd.h 헤더를 이용하여 usleep을 통해 딜레이를 넣어준다. 0.1초의 딜레이를 주기 위해 100000값을 넣었다.
* while 문 밖에 endwin을 선언해주면, 화면이 계속해서 표현된다.



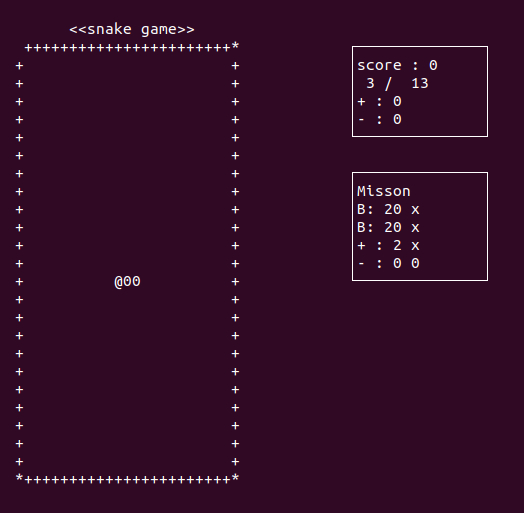
**<0, 1, 2로 구성된 txt 파일>**



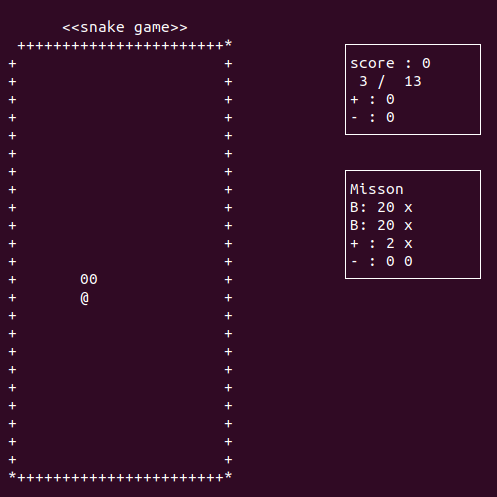
**<초기 Map 실행화면>**

**2단계 – Snake 구현**

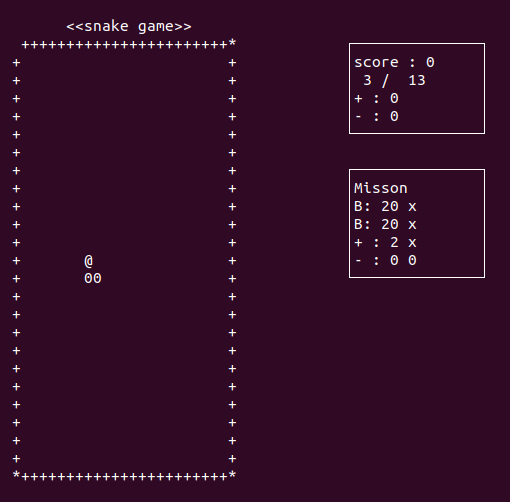
* snake.h 파일 제작
* snake의 좌표값을 파악하기 위해 deque를 사용하였다. snake의 x좌표와 y좌표를 이용하기 위해 각각 deque로 선언하였고 snake의 머리 좌표는 head\_x와 head\_y로 따로 변수로 선언하여 초기값을 각각 모두 13으로 부여하였다. 이는 뱀의 첫 시작위치를 의미한다.
* 따로 변수로 선언한 이유는 추후 아이템 획득 판별이나 벽에 닿았을 때 이벤트 처리에 용이하기 때문이다.
* tail\_x와 tail\_y 또한 꼬리의 좌표로, 뱀의 머리가 꼬리와 닿았을 때의 이벤트를 처리하기 위해 선언하였다.
* length는 뱀 길이로, 뱀 길이에서 -1 만큼 머리 뒤에 '0'이 달려야 한다. 추후 뱀의 길이에 따라 개수가 달라지므로 사용하기 용이하게끔 선언하였다.
* direction은 초기값 l로 두었는데, 현재 진행 방향을 의미하고 있는 변수이다.
* l은 진행방향이 오른쪽에서 왼쪽으로 향하고 있음을 의미한다. 그 외에 위에서 아래로는 d, 아래에서 위는 u, 왼쪽에서 오른쪽으로는 r로 설정하였다.
* Draw() 함수
* Draw 함수는 맵(2차원 배열), 뱀의 머리좌표 값, 길이 이렇게 3가지의 매개변수를 가지고 있으며 받아온 좌표 값에 뱀의 머리부터 새로 배열에 저장해준다.
* Move() 함수
* Move함수는 현재 진행방향과 입력된 키의 방향을 비교하여 주는 함수이며, 이를 통해서 방향을 꺾는 이벤트 처리를 담당한다.
* snake.cpp 파일 제작
* snake.h 파일을 불러온다. Move 함수는 먼저 인자 값으로 char 형 변수 dir을 받아온다. 만약 진행방향이 l이면 direction도 l, r이면 r 값을 설정해준다.그리고 각 진행 방향에 대하여 deque의 좌표 값도 알맞게 조정을 해준다.
* 하나의 예시를 들어 설명하면, 만약 진행방향이 l일때, direction은 l이 된다.
* 그리고 body\_x.좌표 값에 push\_front로 head\_x(뱀의 머리 x좌표)를 넣어주는데, 이때 후위 연산자 --을 이용한다. 그리고 마지막 좌표 값을 pop해준다. 그 다음 body\_y의 맨 앞에는 변화를 줄 필요가 없으므로 그대로 head\_y의 좌표를 넣고 마지막 좌표 값을 뺀다.
* 이렇게 되면 body\_x 처음은 이후에 한 칸 앞당겨진 좌표가 들어갈 것이고 맨 끝좌표가 없어진다. 또한 head\_x--를 통해 head\_x 값도 1이 낮아졌을 것이다.
* 계속해서 l방향으로 전진하게 된다면, 계속해서 head\_x의 좌표가 하나씩 줄어들며 왼쪽으로 이동하는 원리이다. r방향이면 head\_x++가 들어갈 것이고, 만약 위 또는 아래 방향이라면, head\_y--, head\_y++ 가 들어갈 것이다.
* Draw 함수는 Move함수 호출 이후에 호출해주어야 하는데, 방향을 확인한 후에 그려야 하기 때문이다. 배열에 뱀의 위치를 Move함수에서 방향에 따라 재조정을 하고, 그에 맞게 재조정된 좌표를 받아서 그려주는 역할이다.
* 꼬리는 for문이 끝난 이후에 x와 y값이 각각 tail\_x와 tail\_y가 된다.
* control.h 파일 제작
* Control class를 구현한다. 안에는 총 4가지의 메소드를 가지고 있다.
* Control() 함수는 현재 진행방향을 갱신하는 역할이다.
* check\_dir() 함수는 기존 진행방향과 현재 진행방향이 정반대일 경우를
* 확인하기 위한 메소드이다. 이 경우는 실패(Game Over)에 해당되기 때문이다.
* check\_wall() 함수는 벽에 부딪힐 경우를 확인하기 위한 메소드이다. 이 경우 또한 실패(Game Over)에 해당되기 때문이다.
* 마지막으로 check\_item() 함수는 뱀이 아이템을 획득할 때, 이벤트를 처리하기 위한 함수이다.
* control.cpp
* control 함수는 char형 변수 dir을 인자 값으로 갖는데, 이는 현재 진행 방향 값이 들어온다. 사용자가 누른 키 값에 따라 현재 진행방향이 달라지도록 구현하였다.
* 사용자가 입력한 키가 왼쪽이면 l 오른쪽이면 r 위는 u 아래는 d로dir 값을 함수 내부에서 바꾸어 주고, 그 값을 반환한다.
* 그리고 바로 check\_dir 함수를 호출하여 방향키 입력 전 direction과 이후의 direction을 비교한다. 정반대 방향일 경우 게임이 끝나야 하므로 총 4가지의 상황이 나온다.아래의 4가지 경우, false를 반환하여 실패(Gama Over)하도록 구현하였다.
* 현재 진행 방향 : u / 키 입력 후 진행 방향 : d
* 현재 진행 방향 : d / 키 입력 후 진행 방향 : u
* 현재 진행 방향 : l / 키 입력 후 진행 방향 : r
* 현재 진행 방향 : r / 키 입력 후 진행 방향 : l
* check\_wall() 함수의 경우도 진행할 때마다 확인해주어야 하는데, 벰의 머리 좌표 값이 만약 외곽 벽의 좌표 값과 같을 시에 false를 반환하여 실패(Game Over)하도록 구현하였다. 이 때 외곽 벽의 좌표는 x 또는 y 둘 중 하나가 0 또는 24일 경우밖에 없다.
* check\_item() 함수도 진행하면서 확인해주어야 하는데 원리는 위 로직들과 같다. 뱀 머리 좌표 값과 아이템의 좌표 값이 같으면true를 반환하여 아이템 습득으로 간주하고, 그 외에는 false를 반환하여 아이템을 획득하지 않았음으로 간주한다.



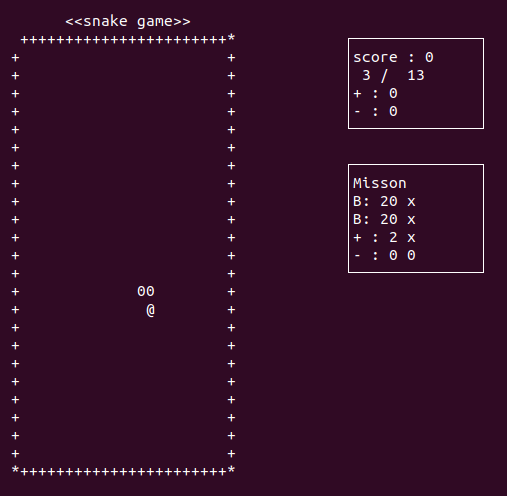
**<게임 시작과 함께 Snake 등장>**



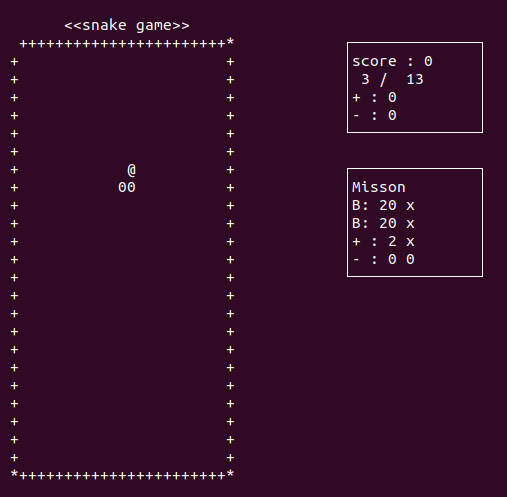
**<진행 방향 left -> down>**



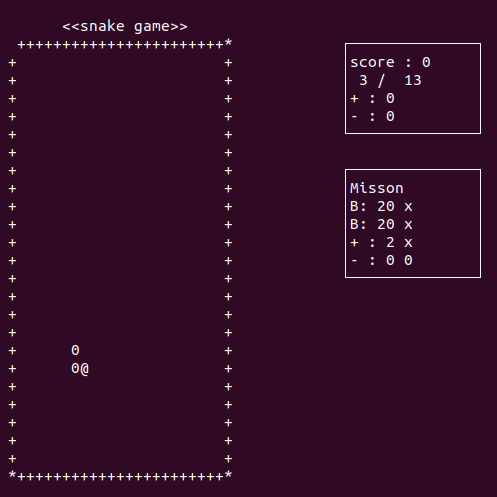
**<진행 방향 left -> up>**



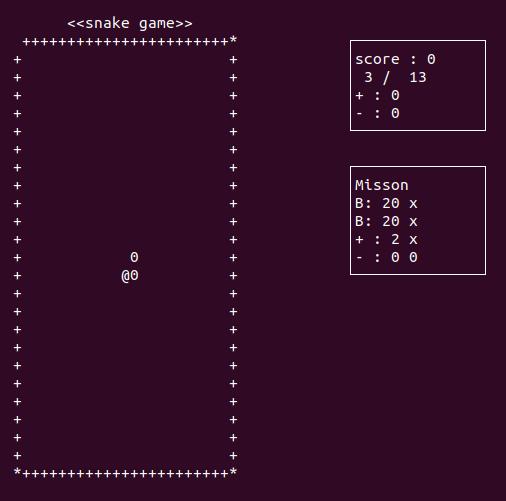
**<진행 방향 right -> down>**

****

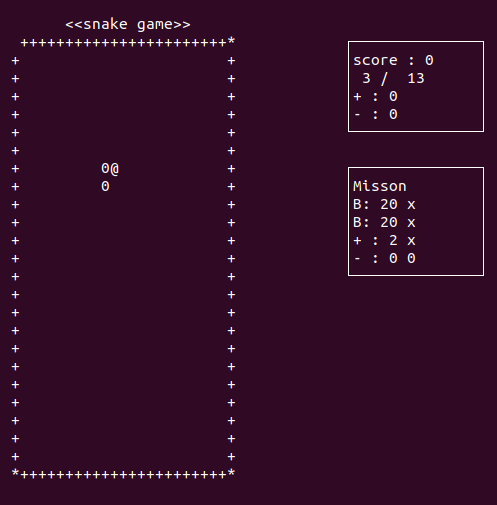
**<진행 방향 right -> up>**



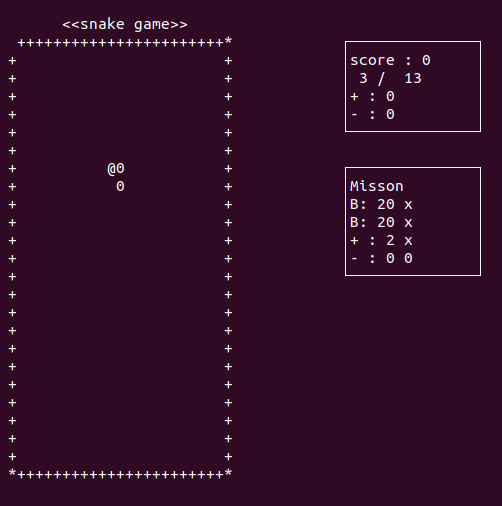
**<진행 방향 down -> right>**

****

**<진행 방향 down -> left>**

****

**<진행 방향 up -> right>**



**<진행 방향 up -> left>**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<반대 방향 키 입력 시 Game Over>**

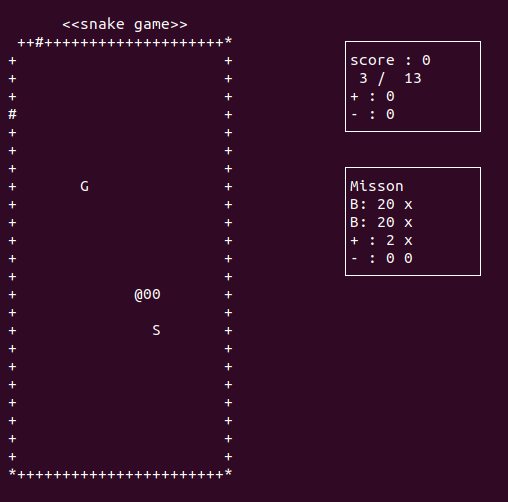
**텍스트이(가) 표시된 사진

자동 생성된 설명**

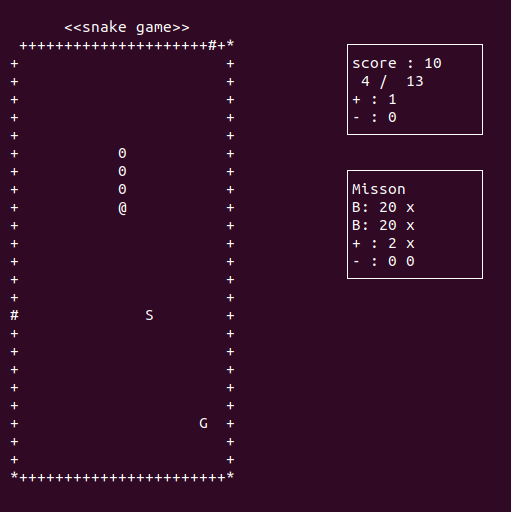
**<벽에 부딪혔을 시 Game Over>**

**3단계 - Growth Item과 Poison Item 구현**

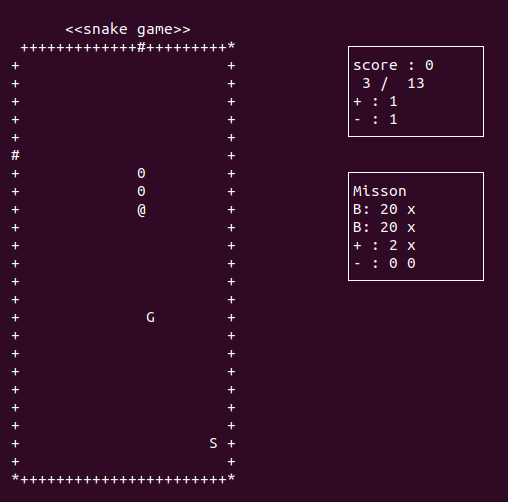
* Grow\_item() 함수와 Small\_item() 함수를 map class에 추가한다. 로직은 둘 다 같다.
* 이전에 아이템을 획득하지 못했다면, 이전 아이템을 사라지게 하고 새로운 위치에 아이템을 생성한다.
* 새로운 좌표를 생성하기 전에 먼저, 이전 아이템 좌표 값에 ' '으로 자리매김을 처리해준 뒤에, ctime, cstdlib 라이브러리를 이용하여 새로운 난수를 생성하였고, 아이템 좌표 값을 갱신해준다.
* 그리고 그 좌표 값에 해당되는 곳을map에도 표시해준다.
* 그리고 함수가 50번째 호출 때만 실행하도록 구현하였는데, main에서 0.1초의 딜레이를 가지기 때문에 5초마다 새로 갱신된다.
* 아이템은 외곽 벽에 나와선 안되므로 범위를 1~23 사이의 난수가 나오도록 지정하였다.

****

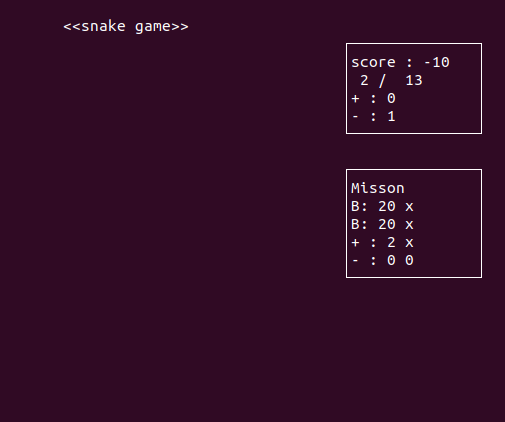
**<Growth Item(‘G’)과 Poison Item(‘S’)>**



**<Growth Item(‘G’) 획득 후 Body 1 증가>**



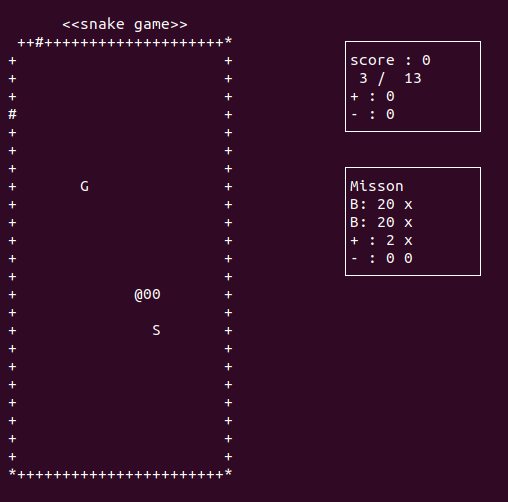
**<Growth Item(‘G’) 획득 후 Body 1 증가 -> Poison Item 획득 후 Body 1 감소>**

****

**<Poison Item 획득 후 Body 1 감소 -> Body 2 -> Game Over>**

**4단계 – Gate 구현**

* potal() 함수
* ctime, cstdlib 라이브러리를 이용하여 새로운 난수를 생성하였고, 그 위치에 Gate가 생기도록 구현하였다.
* Item과 다른 점은 벽에서만 생성되야 한다는 점이다.
* 따라서 임의의 변수를 난수로 생성하여 2로 나누었을 때 짝 홀을 판별하여 x와 y 둘 중 누가 0 또는 24의 값을 가질지 정하도록 구현하였다.
* 또한 Gate는 한 쌍으로 이루어지기 때문에, 총 4개의 난수를 생성해야 했고, 이를 한 쌍으로 보기 위해 2차원 배열에 넣었다.



**<Gate(‘#’)가 Wall에서 한 쌍으로 등장>**

### 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
* Curses 는 "cursor optimization“ 을 재미스럽게 발음한 것
* Curses 라이브러리는 상당히 유연하고 효율적인 API(Application Programing Interface) 를 제공한다.
* 커서를 움직이거나, 윈도우를 생성하고, 색깔을 만들고, 마우스 관련 함수를 제공한다.
* NCURSES 기능
* Ncurses 는 터미널 기능의 wrapper 생성이 가능
* 텍스트 모드에서 UI를 구현할 수 있는 프레임워크를 제공한다.
* 윈도우 등을 만들 수 있는 함수를 제공하며, 보다 확장된 라이브러리는 패널, 메뉴, 폼 등 기본적인 curses 라이브러리로부터 확장성을 제공한다.
* Snake Game의 UI 구현을 위해 사용하였다.
* STL 라이브러리 – Deque(Double Ended Queue)
* Queue와 비슷하지만 queue는 front에서만 삭제하고, end에서 삽입하는데, deque는 front와 end에서 삭제와 삽입이 모두 가능하다.
* 연속적인 메모리를 기반으로 하는 순차 컨테이너
* 벡터와 함수가 거의 비슷하지만, push\_front(), pop\_front()가 가능하고, capacity()는 안됨.
* 임의의 원소에 접근이 가능하다. 인덱스가 존재.
* resize를 통해 크기 변경이 가능하고, 크기가 가변적이기 때문에 삽입할 때 크기 고려하지 않아도 됨.
* Snake 표현 및 움직임 구현에 사용하였다.

### 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.** |

* Snake의 관절 부분 구현에 있어서 어려움을 겪었다.
* STL 라이브러리의 deque를 사용하여 관절이 꺾이는 부분을 구현하였다.
* Gate 구현에는 성공하였으나, Gate 작동은 실패하였다.
* 생각했던 해결방안으로는 Gate의 위치를 배열에 저장해 배열의 Gate 원소값과 Snake의 머리의 좌표 값이 일치할 때, 남은 하나의 Gate의 좌표 값으로 이동하도록 하는 방법이 있었으나, 결국은 실패하였다.
* Map 내부에 생성되도록 하는 Gate는 구현과 작동 모두 실패하였다.
* Score Board에서 Mission을 받고 수행 여부에 따라 Score Board 와 Mission 진행 상황이 즉각적으로 수정되는 것은 구현하였지만, Mission을 clear하고 다음 Map으로 넘어가는 부분을 구현하지 못했다.
* 생각한 해결방안으로는 main 함수에서 Mission 성공여부를 확인 후 다음 Map으로 넘어가도록 하는 방안이 있었지만 실패하였다.

### 결과물 목록

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

* Controll.h, Controll.cpp
* 키보드를 통해 Snake 조작
* manager.h, manager.cpp
* Snake 길이와 Game Score 관리
* map.h, map.cpp
* Map 파일을 받아와 저장 후, Game 화면에 수시로 정보를 받아와 Map을 그려주는 역할
* snake.h, snake.cpp
* Snake 구현 및 Map에서의 Snake의 위치 정보 관리
* main.cpp
* 모든 헤더 파일을 불러와 무한 루프를 통해 게임 실행

# 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.** |

**기본적으로 육종호 조원이 전체적인 틀을 잡아주었고 저는 그 안에서 부수적인 요소들과 디버깅을 담당하였습니다. 프로젝트의 코드 제작을 전반적으로 스스로 하지 못했다는 점에서 제 자신의 부족함을 느꼈고 앞으로 더 열심히 노력해야 함을 깨달았습니다. 프로젝트 수행 시 처음에 어떻게 해야할 지가 가장 막막했는데 육종호 조원이 틀을 잡아주어 도움이 되었고, 교수님께서 올려놓은 여러 파일들을 계속 읽어보고 이해했던 게 도움이 되었던 것 같습니다. 프로젝트 운영에 있어서는 아직 상당히 미흡합니다. 일단 위에서 언급했듯이, Wall 위의 Gate의 작동 여부, Map 내부 Gate의 구현과 작동 여부, 그리고 Mission에 따른 Map의 변화에 있어서 현 프로그램은 기능을 하지 못하므로 개선이 필요합니다. 또한 임의적으로 프로그램을 실행할 때 끊기는 느낌이 들며 진행이 매끄럽지 않을 때가 간혹 있는데 이 역시 개선해야할 점이라고 생각합니다.**

# 참고 문헌

**참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 |
| 1 | pdf | NCURSES | e-campus | 2021 | 국민대학교 소프트웨어 융합 대학 |

# 부록

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

## 사용자 매뉴얼

**프로젝트 실행 후 사용안내, 따라하기 등 포함**

* Game Rule #1
* Snake는 진행 방향의 반대방향으로 이동할 수 없다. Head 방향이 진행 방향이며 단 한번의 방향키 입력만으로 반대 방향인 Tail 방향으로 이동할 수 없다. 이동 시 실패(Game Over)로 간주한다.
* Snake는 자신의 Body를 통과할 수 없고 벽을 통과할 수 없다. Body 또는 Wall을 통과 시 실패(Game Over)로 간주한다.
* Head 방향 이동은 일정시간(틱)에 의해 이동한다.
* 진행방향과 같은 방향기 입력은 무시하고 방향키는 프로그램 사용자 키보드의 방향키 입력키를 통해 직접 입력하며, 방향키 입력 시 방향키의 방향으로 다음 틱에 Snake가 이동한다.
* Game Rule #2
* Snake의 이동방향에 Item이 놓여 있는 경우 Snake가 Item을 획득한다.
* Growth Item의 경우 몸의 길이(Tail)가 1 증가한다.
* Poison Item의 경우 몸의 길이(Tail)가 1 감소하며 몸의 길이가 3보다 작아지면 실패(Game Over)로 간주한다.
* Item은 Snake Body가 있지 않은 임의의 위치에 출현한다.
* Item은 출현 후 일정시간이 지나면 사라지고 다른 위치에 나타나야 한다.
* 동시에 출현할 수 있는 Item의 수는 2개로 제한한다.
* Game Rule #3
* 다음과 같은 메뉴들로 Game Score을 구성한다.
* 게임 중 몸의 길이 계산
* 게임 중 획득한 Growth Item의 수
* 게임 중 획득한 Poison Item의 수
* 게임 중 Gate 사용 횟수
* 게임 시간(Seconds로 계산)

## 설치 방법

**프로그램 컴파일 및 실행 방법 포함**

* Ubuntu 접속 후 Terminal 창을 연다.
* Snake 파일이 있는 경로로 들어간다.
* g++ -o snake \*.cpp -lncurses를 입력한다.
* ./snake를 입력한다.