**C++ 프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *SNAKE GAME* |
| 팀 명 | *2-4* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.1 |
| **Date** | 2020-Jun-26 |

|  |  |
| --- | --- |
| **팀원** | 배정민 |
| 서동하 |
|  |
|  |
|  |

|  |
| --- |
| **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 “2-4”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “2-4”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-Snake\_Game.doc |
| **원안작성자** | 배정민, 서동하 |
| **수정작업자** | 배정민, 서동하 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2020-06-20 | 서동하 | 1.0 | 최초 작성 |  |
| 2020-06-26 | 배정민 | 1.1 | 추가 작성 | 보고서 내용 추가 및 수정 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_Toc43103653)

[2 개발 내용 및 결과물 4](#_Toc43103654)

[2.1 목표 4](#_Toc43103655)

[2.2 개발 내용 및 결과물 5](#_Toc43103656)

[2.2.1 개발 내용 6](#_Toc43103657)

[2.2.2 시스템 구조 및 설계도 7](#_Toc43103658)

[2.2.3 활용/개발된 기술 9](#_Toc43103659)

[2.2.4 현실적 제한 요소 및 그 해결 방안 10](#_Toc43103660)

[2.2.5 결과물 목록 11](#_Toc43103661)

[3 자기평가 11](#_Toc43103662)

[4 참고 문헌 12](#_Toc43103663)

[5 부록 12](#_Toc43103664)

[5.1 사용자 매뉴얼 12](#_Toc43103665)

[5.2 설치 방법 13](#_Toc43103666)

# 개요

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

기본적으로 ncurses를 활용하여 snake 게임을 개발하였다. 게임을 초기화 하기 위해서 stages 폴더에 있는 stage 파일로부터 게임 정보를 읽어올 수 있도록 fstream을 사용하였다. 외부 파일에서 읽어온 게임 정보를 숫자로 바꾸기 위해 string으로 읽어오고 c\_str()을 사용하여 char\*로 변환하였고, 다시 int값으로 바꾸어주기 위해 cstdlib의 atoi()를 사용하였다. 게임의 시작과 종료를 알리는 문장 출력, 스코어 보드와 미션 보드에서의 문자열 출력을 위하여 string 라이브러리를 포함시켰다. growth, poison item, gate를 임의의 위치에 배치할 때는 cstdlib의 srand()와 rand()를 사용하였다. 마지막으로 게임의 진행 속도를 조정하는 sleep 함수를 구현하기 위해 unistd.h에 있는 get\_delay()를 활용하여 게임을 완성시켰다.

* 외부 라이브러리 설치 방법

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

# 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 |

1단계: 맵 데이터를 텍스트 파일에 저장, 게임 시작 후 텍스트 파일을 로드하는 형식으로 게임을 진행한다. map을 자료구조로 표현하고 Ncurses 상에서 그래픽으로 표현하기 위한 기반을 완성한다.

2단계: Snake 클래스를 추가하고 방향키에 맞추어 snake가 이동하며 아무 입력이 없을 시 이전 방향을 유지하도록 한다. 자신의 몸통과 부딪치거나 벽에 충돌 시 게임오버가 된다.

3단계: growth item, poison item에 규칙에 따라 몸통 길이를 증감하고 획득 시에 snake의 출력이 매끄럽게 보이도록 한다. 몸통의 길이가 3보다 짧아지면 게임오버가 된다. 각 item은 map에 한 개씩만 배치하도록 한다.

4단계: snake game 규칙에 맞추어 gate의 출입시 반대편 gate에 snake를 올바르게 배치할 수 있어야하며 gate 이용 중에는 gate 재배치를 허용하지 않는다.

5단계: snake의 정보, 구체적으로 현재 몸의 길이, 게임 내에서 기록한 최대 길이, 획득한 item의 개수, gate 이용 횟수, 게임을 진행한 시간(초)을 화면 오른쪽에 board 형태로 표시한다. 또한 각 stage 마다 미션을 설정하여 board로 알려주고, 미션의 진행상황을 알려준다. 모든 미션을 완료했을 경우 다음 stage로 넘어가며, 게임오버가 됐을 경우 해당 stage를 재시작한다.

## 개발 내용 및 결과물

### 개발 내용

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

**1단계**

게임을 터미널에서 표현하기 위해서 ncurses의 mvaddch를 활용하여 2차원 배열 map에 저장된 숫자를 보고 각각에 대응되는 그래픽 요소를 출력하는 것을 구현하였다. 게임에 필요한 정보들을 효과적으로 관리하고 변경할 수 있도록 txt파일에 map과 게임에 필요한 정보들을 저장해두고, ifstream으로 읽어와서 게임 데이터를 초기화한다.

**2단계**

게임 시작 후 Snake 클래스인 snake를 생성한다.

클래스 생성자로 몸통배열(snake가 가질 수 있는 최대 메모리를 확보) 입력값, flag를 초기화 시킨다.

방향키 입력 시 기본값으로 설정된 방향과 입력된 방향이 반대일 시 시작하자마자 죽는 현상이 발생하였다. 그래서 게임 시작 전 입력을 받아 입력 방향의 반대방향으로 snake 몸통 배열을 위치시켰다.

그 후 입력에 맞추어 head의 좌표 값을 이동시키며 몸통 배열의 앞에 배열의 좌표를 전달받는다. 이후 head가 벽의 좌표 또는 몸통에 좌표이면 게임 실패 flag를 on 아니면 게임을 진행시킨다.

**3단계**

srand 통한 랜덤 값으로 벽이 아닌 곳 중에서 item들을 배치한다. Snake가 이동하면서 grow item 좌표에 도달 시 head의 좌표를 item이 있던 좌표로 변경 후 몸길이 추가한다. 이후 grow item 재생성한다. Poison item 좌표에 도달 시 몸통 길이의 제일 끝 좌표의 데이터를 0으로(배경으로) 함으로써 제거하고 몸길이 감소한다. 이후 Poison item 재생성한다.

**4단계**

srand 통한 랜덤 값으로 벽 에서 GATE\_A 와 GATE\_B를 배치한다.

SNAKE가 이동하면서 GATE\_A의 좌표에 도달 시 GATE\_B를 GATE\_OUT으로 설정한다. 만약 GATE\_OUT이 가장자리 있다면 SNAKE의 방향은 항상 맵 안쪽으로 설정한다. GATE\_OUT이 맵 안쪽에 있다면 SNAKE의 방향부터 시작하여 시계방향으로 나갈 수 있는 지 확인한다.

**5단계**

게임에 필요한 점수들을 관리하는 Board 클래스를 정의했다. Board 클래스에서는 현재 게임의 진행 상황 및 달성해야하는 미션, 미션의 클리어 여부를 처리하고 화면의 별도의 공간에 표시해준다. 총 4개의 stage를 구성하였고, 미션을 클리어 했을 때에만 다음 stage로 넘어가고, 게임오버가 된 경우 현재 stage를 다시 반복하도록 게임 구조를 수정하였다. 각 게임의 목표 달성치는 stage txt 파일에 같이 기록해두었다.

### 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 (맵 구현) – main.cpp**

1. ncurses 사용을 위해 초기화작업 및 그래픽 요소를 표시할 window를 정의해준다.
2. 게임 요소들에 배경-0, 벽-1, … , 게이트-7 처럼 숫자를 대응시키고 enum으로 정의한다.
3. snake의 모든 그래픽 요소의 색상을 지정하기 위해 init\_color라는 함수를 정의하고 init\_pair로 텍스트 색상과 배경 색상을 지정해준다. immune wall을 제외한 모든 요소들은 오로지 배경 색상만으로 표현하였다.
4. 현재 게임의 map을 저장할 수 있도록 2차원 int 배열 MAP을 만들고, stages 폴더에서 원하는 stage txt파일을 불러와 MAP을 초기화하는 load\_map 함수를 정의한다. (load\_map 함수는 2, 3단계 구현 이후에 snake의 초기 위치를 txt파일에서 읽어온다.)
5. 현재 MAP을 Ncurses window에 표현하는 draw 함수를 정의한다. 이때, immune wall과 wall을 구분하기 위해 immune wall인 경우는 검은색 wall에 흰색으로 ‘X’ 문자를 표시하도록 하였다. 사용된 모든 문자가 반각문자이기 때문에 map의 한 요소를 표현하기 위해 두 개의 mvaddch 함수를 사용하였다.

(enum, init\_color, MAP, load\_map, draw 는 팀원 서동하에 의해 snake.cpp로 이동되었다)

* 사용 함수 및 클래스

1. int main() – 게임의 시작 및 진행, 종료, 게임 구동을 위한 초기화 작업 담당
2. void init\_color() – 그래픽 처리 목적, enum값을 팔레트 번호로 갖는 색상 쌍을 선언
3. void load\_map() – 외부에서 stage 파일을 읽어와서 각 stage마다 고유한 map이나 게임에 필요한 정보를 초기화
4. void draw() – window에 현재 map의 상태를 시각화한다. 이전에 정의한 init\_color의 정보를 활용

**2단계 (snake 이동),**

1. Snake 클래스의 생성자를 통해 각종 플래그, 몸통 배열 초기화, 아이템 세팅을 해준다.
2. 아이템 세팅은 랜덤으로 위치하도록 하며 map배열을 통해 벽이나 다른 아이템 위치 시 재설정을 해준다.
3. 게임 시작 화면에서 입력 받은 방향키를 direction으로 설정 후(방향키가 아닌 키 입력 시 default로 오른쪽) direction 방향을 snake의 머리로 설정하고 몸통을 머리와 direction의 맞추어 반대방향으로 세팅한다.
4. Input\_key에 맞추어 현재 커서를 이동시키며 아무런 입력이 없을 시 이전 direction값을 input\_key에 대입한다. 그 후 몸통 배열에서 배열의 앞의 값들을 하나씩 뒤로 이동시킨다.
5. 이동시킨 배열이 게임 종료 조건에 해당되는 지(벽에 충돌, 몸통 길이 조건) 확인하여 flag를 설정한다. 그 후 현재 커서를 snake의 머리로 설정한다.
6. 이동시킨 배열을 window 화면에 출력시키고 direction 변수에 input\_key를 대입한다.

* 사용 함수 및 클래스

1. class Snake – snake 클래스
2. Snake(WINDOW\* win); - snake의 변수 초기화 생성자
3. void snake\_move(); - 방향키 입력 및 현재 커서 이동 함수
4. void move\_snake\_arr(); - 현재 커서의 위치에 맞는 아이템 함수 실행 및 몸통 배열 이동 함수
5. void die\_check(); 게임 종료 확인 함수

**3단계 (item 구현)**

1. map배열을 통해 현재 커서 위치가 아이템 위치에 도착 시 grow item을 획득 시에는 꼬리 배열을 삭제시키지 않고 몸통 배열에 추가시키고 몸통 길이를 1추가해준다.
2. Poison item 획득 시 꼬리 배열과 그 앞 배열까지 삭제 시키고 몸통 길이를 1감소시킨다.
3. Item 획득 시에는 함수를 통해 아이템을 다시 재배치 시켜준다.
4. main함수에서 move의 카운트를 통해 일정 시간 동안 item 미획득시 item을 재배치시킨다.

* 사용 함수 및 클래스

1. void grow\_item\_setting(); - grow item 세팅 함수
2. void poison\_item\_setting(); - poison itme 세팅 함수

**4단계 (gate 구현)**

1. 벽으로 설정된 것 중에서 랜덤으로 gate를 2개 설정 시켜준다. 이때 두 개의 위치가 겹칠 시 다시 설정한다.
2. 두 개의 gate를 gate\_a, gate\_b로 설정하고 gate\_a에 snake로 들어갈 시 gate\_b가 gate\_out이 된다. Gate\_b에 snake가 들어갈 시 gate\_a가 gate\_out이 된다.
3. 게이트에 도달 시 FLAG\_GATE\_USE를 true로 설정해 snake가 gate를 통과 중에 gate가 재배치되는 것을 방지한다.
4. Gate\_out의 위치가 게임 판의 가장자리일 시 snake의 통과 위치를 항상 맵 안쪽으로 설정 시킨다. Gate\_out의 위치가 게임 판 중간에 위치 시 gate\_out의 위치에 direction방향부터 시작한 offset 배열을 더해 시계방향으로 snake가 위치할 수 있는 지 check하고 가능할 시 snake의 커서 위치를 이동시킨다.
5. main함수에서 move의 카운트를 통해 일정 시간 동안 gate 미사용 시 gate을 재배치시킨다.

* 사용 함수 및 클래스

1. void gate\_item\_setting(); -gate 아이템 세팅
2. void gate\_move\_arr(Position gate\_out); -gate 통과시 배열 이동함수
3. void input\_key\_set(int n); - input\_key을 int로 치환

**5단계 (미션 구현) – main.cpp board.h, board.cpp**

1. 스코어 보드와 미션 보드를 표시하기 위한 window를 main.cpp에서 두 개 더 정의한다.
2. 각 stage txt파일의 가장 마지막에 목표 달성치를 위한 네 개의 숫자를 지정해준다. 각각의 숫자는 ‘최대 달성 길이’, ‘growth 획득 수’, ‘poison 획득 수’, ‘gate 이용 수’를 의미한다.
3. 생성자에서 두 window와 현재 stage number를 받는 class Board를 정의하고 생성한다. Board에서는 현재 snake의 기록과 목표 달성치, 목표 달성 여부, 게임 플레이 시간 등 게임 플레이와 관련된 정보들을 모두 가지고 있다.
4. load\_map함수를 stage number와 함께 Board를 받아서, stage txt의 마지막에 있는 숫자들을 가지고 Board의 목표 달성치를 초기화하도록 수정한다.
5. 게임의 구조를 게임이 끝났을 때 (snake.FLAG\_GAME\_END가 true일 때) 미션을 모두 달성하였으면 다음 stage를 진행하도록, 그렇지 않으면 현재 stage를 다시 반복하도록 수정한다)
6. 게임 실행 중에는 매 step마다 (while(!snake.FLAG\_GAME\_END){…} 내부에서) board의 게임 상태 정보들을 snake에서 받아온다. 그리고 현재 score 정보와 mission 클리어 여부를 보여주는 board의 멤버함수 printScoreBoard와 printMissionBoard를 호출한다.
7. 매 step이 끝나기 전에 모든 미션을 클리어했는지 체크하여, 미션을 클리어한 경우 board의 미션 정보를 초기화, stage 값을 1 늘리고, 게임을 끝낸다.

* 사용 함수 및 클래스.

1. class Board – 게임의 진행 상황, 미션 클리어 여부 판단 및 시각화
2. void Board::printScoreBoard() – 게임 상황을 string으로 표현하고 mvwprintw를 통해서 window에 출력
3. void Board::printMissionBoard() – 미션 클리어 상황을 string으로 표현하고 mvwprintw를 통해서 window에 출력
4. void Board::checkSubMissionClear() – 각각 4가지 미션을 클리어했는지 판단하고, 클리어 했을 경우 mission window에 표시해주는 함수

### 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.** |

1. ncurses.h : 터미널에서 게임을 표현하기 위해 ncurses를 사용하였다. 문장이나 문자열을 특정 window에 표현하기 위해서 mvprintw()를 사용하였고, Map의 화면 표시나 미션의 클리어 여부 등 개별적인 문자를 특정 window에 표현하기 위해서 mvwaddch() 를 활용하였다. 게임을 터미널에서 표현하기 위해서 색상을 적극적으로 활용하였고 이를 위해서 init\_pair()를 사용하였다. 게임과 스코어 보드, 미션 보드를 개별적으로 표현하기 위해서 window 개념을 활용하였고, wrefresh(), wclear() 등을 통해서 원하는 window의 화면만 변경하였다.
2. string : 스코어 보드와 미션 보드를 표현하기 위해서 string을 활용하였다. 현재 플레이타임이나 snake의 최대 길이 같은 정보는 변수이지만 두 자리수를 넘어갈 수도 있는데 이 역시 to\_string() 및 c\_str()을 활용하여 변수가 fomatting 된 string으로 변환하고 ncurses의 mvwprintw 함수로 간단하게 window에 표현하였다. stage txt파일에서 각 줄을 읽어올 때 ifstream과 함께 getline()을 활용하여 string 형태로 읽어와서 각종 변수 및 MAP 배열을 초기화하는데 활용하였다.
3. Unistd.h : 리눅스의 sleep함수인 usleep함수와 딜레이를 구현하기 위한 get\_delay()함수를 이용하여 게임 속도를 조정할 수 있도록 하였다.
4. Cstdlib : grow, poison item와 gate를 배치하기 위해 랜덤으로 배치하기 위한 srand함수를 구현하기 위해 활용하였다. atoi함수를 이용하여 char형태로 저장된 stage.txt의 데이터들을 int로 변환하여 변수에 값을 입력해 주는데 활용하였다.
5. Fstream : 맵 데이터를 stage.txt 파일로부터 읽기 위하여 지정한 경로로부터 데이터를 읽어와 reammap 변수에 넣었으며 is\_open함수를 활용하여 제대로 파일을 열람했는지 확인하였다.

### 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.** |

매 스테이지마다 고유한 맵 정보나 snake 위치 정보, 미션 데이터를 소스 코드에 상수로 저장할 경우 비효율적이고 데이터를 수정하기 어려웠다. 이를 효과적으로 관리하기 위하여 stages 폴더 내의 stage<스테이지번호>.txt 파일들에 게임에 필요한 정보들을 숫자 형태로 기록하고, 정해진 규칙에 맞춰 불러와 초기화하는 식으로 해결하였다.

윈도우에서 키보드입력을 확인하는 kbhit() 함수가 리눅수에서 구현되지 않아 getch, ungetch 를 사용하여 버퍼에 재입력하는 방법으로 해결하였다.

키보드 입력으로 방향키가 아닌 값들이 버퍼에 남아서 snake의 컨트롤이 비정상적으로 동작하는 것을 방지하기 위하여 snake가 이동하는 매 step마다 버퍼를 비워주는 작업을 하였다.

리눅스에서 빠르게 디버깅을 하기 위해 check\_body\_arr(), check\_map\_arr() 함수를 만들어 실시간으로 배열의 변화를 보아 진행 상황을 확인하였다.

### 결과물 목록

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

1. Makefile – 프로젝트 컴파일을 위한 makefile

2. README.txt – 코드 및 파일 별 기여자 작성

3. main.cpp – 게임의 진행을 위한 main함수

4. snake.h – snake 게임 진행, 헤더파일

5. snake.cpp –snake 게임 진행 cpp파일

6. board.h – 미션 및 점수 출력 헤더파일

7. board.cpp - 미션 및 점수 출력 cpp파일

8. /stages – 각 stage와 관련된 파일이 들어 있는 폴더

# 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.** |

**배정민 – 크게는 프로젝트에서 기본적인 게임 구조와 그래픽 처리, 맵 구성 미션 구성 등을 맡았고 snake와 관련된 몇 가지 오류를 개선하였다. 프로젝트를 구현하기 위해서 좀 더 구조적으로, 기능적으로 소스코드나 클래스들을 구분하였으면 좋았을텐데 그부분이 다소 아쉬웠다. 미숙했던 점으로 서동하 팀원의 코드를 받아서 리눅스 환경에서 적용시키려 하였으나 실패하였고, 각 단계에 남아 있는 오류를 제거하는 과정에서 코드가 약간 지저분해지거나 불필요한 코드가 들어가기도 하였다. 팀 프로젝트를 통해서 원활한 개발을 위한 개발자간 소통과 계획의 중요성을 깨달을 수 있었다. 게임에 스레드를 활용하지는 않았는데 이와 관련하여 게임에 쉽게 적용할 수 있도록 간단하고 직관적인 예시가 추가되어도 좋을 것 같다.**

**서동하 – 프로젝트에서 snake의 움직임 및 아이템 구현을 맡았는데 처음에 윈도우 환경인visual studio에서 코드를 작성하였다. 그러나 리눅스에서 실행을 시켜보니 윈도우에서 작동하던 함수가 리눅스에서 호환이 안되는 것들이 매우 많았다. 때문에 리눅스에서 처음부터 다시 작성하여 처음에 시간을 많이 소비하였다. 대신 덕분에 리눅스 환경에서의 코드 작성에 익숙해질 수 있었다. 코드를 작성하면서 클래스에 대해 알 수 있었고 ncurses라이브러리를 다룰 수 있게 되었다. 아쉬웠던 점은 프로젝트의 오버스펙을 구현하기에 snake 게임은 개발할 수 있는 부분이 한정적이였다고 생각한다.**

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
|  | 서적 |  |  |  |  |  |
|  | 기사 |  |  |  |  |  |

# 부록

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

## 사용자 매뉴얼

1. 프로젝트 실행 후 게임 실행 화면이 출력된다.
2. 방향키를 누르면 해당 방향부터 snake가 이동되기 시작한다. 방향키가 아닌 키를 입력 시 snake는 오른쪽으로 이동한다.
3. Grow item 획득 시 몸의 길이를 하나 증가시키고 poison item 획득 시 몸의 길이를 하나 감소시킨다.
4. Gate에 입장 시 다른 쪽 gate로 snake가 한 칸씩 이동된다.
5. 화면에 우측에 현재 진행상황과 미션이 주어진다. 미션을 클리어해야 다음 stage로 넘어갈 수 있다. 총 stage는 4단계로 이루어져있다.
6. 게임 클리어시 게임 종료 화면이 출력된다.
7. 게임 종료 조건은 몸의 길이가 3보다 작아지거나 snake가 위치할 수 없는(wall,iwall)로 이동 시 종료된다.

## 설치 방법

1. 압축 파일을 적당한 위치에서 해제한다.
2. snakegame 폴더 내에 board.cpp, board.h, main.cpp, snake.cpp, snake.h, Makefile이 있는지 확인한다.
3. 터미널에서 snakegame 폴더로 작업 디렉토리를 변경한 후, make main 명령어를 수행한다. (오브젝트 파일을 제거하기 위해서는 make clean 명령어를 수행한다.)
4. main 실행 파일이 생성되면 ./main 명령어로 게임을 실행한다. (snakegame 폴더 내에 stages 하위 폴더와 stage1.txt 부터 stage4.txt 파일까지 없으면 게임이 정상적으로 실행되지 않는다.)