**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| **프로젝트 명** | *스네이크 프로젝트* |
| **팀 명** | *전과1범* |
| **문서 제목** | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.6 |
| **Date** | 2022-6-17 |

|  |  |
| --- | --- |
| **팀원** | 황찬우 (팀장) |
| 이정현 |
| 임상학 |
|  |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “스네이크 게임”를 수행하는 팀 “전과1범”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “전과1범”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서\_스네이크\_프로젝트.doc |
| **원안작성자** | 황찬우, 이정현, 임상학 |
| **수정작업자** | 황찬우, 이정현, 임상학 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2022-05-21 | 황찬우 | 1.0 | 최초 작성 | ncurse 라이브러리 실습 |
| 2022-05-22 | 황찬우 | 1.1 | 기능 추가 | 1단계, 2단계 구현 |
| 2022-05-23 | 황찬우 | 1.2 | 디버깅 | 버그 수정 및 규칙 #1 준수 구현 |
| 2022-06-14 | 황찬우 | 1.3 | 기능 추가 | 3단계 아이템 구현 |
| 2022-06-15 | 황찬우 | 1.4 | 기능 추가/디버깅 | 버그 수정 및 규칙 #2 준수 구현 |
| 2022-06-16 | 황찬우 | 1.5 | 기능 추가 | 4단계 게이트 구현 |
| 2022-06-17 | 황찬우 | 1.6 | 그래픽 추가 | 버그 수정 및 그래픽(색상) 구현 |

**목 차**

[**개요**](#_e7q5ou2tl3qj) **4**

[**개발 내용 및 결과물**](#_6vco40g6eepe) **5**

[목표](#_x04g8kgzdgy8) 5

[개발 내용 및 결과물](#_3znysh7) 7

[개발 내용](#_2et92p0) 7

[시스템 구조 및 설계도](#_tyjcwt) 7

[활용/개발된 기술](#_3dy6vkm) 10

[현실적 제한 요소 및 그 해결 방안](#_1t3h5sf) 10

[결과물 목록](#_2s8eyo1) 11

[자기평가](#_17dp8vu) **11**

[참고 문헌](#_3rdcrjn) **13**

[부록](#_26in1rg) **13**

[사용자 매뉴얼](#_lnxbz9) 13

[설치 방법(with CMake)](#_35nkun2) 14

# 개요

본 프로젝트는 C++를 이용하여 snake game을 만드는 것이 목표이다.

우리는 원활한 개발을 진행하기 위해서 기능들을 크게 map 클래스와 snake클래스로 나누어서 개발을 진행했다.

snake클래스의 멤버 함수로는 뱀과 관련된 부분에 대해서만 다루었고(움직이는 방향 입력 받기, 벽과 충돌했는지 여부 판단, 뱀 움직이기)

map 클래스의 멤버 함수로는 snake의 좌표를 받아 map에 삽입하고, 출력하는 기능(get\_set\_snake), gate의 생성(makeGate), snake가 gate 안에 있는지 판단하는 기능(is\_snake\_in\_gate), gate로 인한 워프 후 움직이는 방향을 정해주는 기능(warp\_direction)과 실제로 snake의 head를 워프시켜주는 기능(warp\_by\_gate), item의 생성(make\_item), item과 snake의 상호작용(change\_snake)과 이러한 기능들의 작동으로 인해 map(2차원 배열)에 그려진 결과물을 출력해주는 기능(draw)등을 구현하였다.

또한 우리는 사용자들에게 TUI를 제공하기 위해서 ncurses 라이브러리를 사용했다. ncurses를 설치하기 위해서 ubuntu 환경 기준 터미널에 sudo apt-get install libncurses5-dev libncursesw5-dev를 입력하면 설치가 진행되고, g++ -std=c++11 -o (원하는 이름) (해당 파일 이름).cpp -lncursesw 를 통해 유니코드를 인식하는 컴파일을 진행할 수 있다.

그 이외에도 Growth/Poison item의 위치 / gate의 위치를 랜덤하게 업데이트 하기 위해서 item/gate를 만드는 함수에 cmath의 rand()를 사용했다.

또한, 1단위시간 당 1번씩 움직이게 만들기 위해서 unistd.h의 usleep()를 main함수의 while 루프 부분에 삽입했다.

ncurses의 윈도우는 \* char 형식의 문자열만 인자로 받을 수 있었다. 이 때문에 int 변수를 출력하기 어려웠다. 이를 해결하기 위해 string의 to\_string함수를 이용해서 int형 변수를 string형으로 변환 하고, cstring의 strcpy()함수를 사용하여 \*char 변수에 복사 해서 출력했다.

# 

# 개발 내용 및 결과물

## 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. **Map의 구현**

* **stage별 Map 구현**
* **Map을 중심으로 프로그램 설계**

**Ncurse를 이용, gameWin 윈도우를 만들고 Map배열을 사용하여 그 위에 맵의 초기 상태를 출력한다.**

**매 단위시간마다 snake, item, gate의 좌표 변화를 map에 업데이트 후 refresh 한다.**

**Map으로 모든 데이터를 가져와서 처리하는 중앙처리구조.**

1. **Snake 표현 및 조작**

**- 사용자의 조작에 따라 방향을 바꾸며 이동하는 Snake 구현**

**사용자의 key input에 따라 snake 객체의 방향을 바꿔주는 get\_dirt 함수 구현**

**매 단위시간마다 snake 객체의 좌표를 변경해주는 move\_snk 함수 구현**

**벽과 충돌을 감지하는 is\_crashing\_wall 함수 구현**

1. **Item 요소의 구현**

* **먹으면 snake의 길이가 1칸 길어지는 Growth 아이템 구현**
* **먹으면 snake의 길이가 1칸 줄어드는 Poison 아이템 구현**

**두 아이템 모두 같은 item 요소이며, 구분은 Growth 는 5 , Poison은 6으로 맵에 표현.**

**아이템은 일정 시간이 지나면 사라지고, 다른 위치에 재 생성 (벽이나 스네이크가 있는 위치엔 생성 X)**

**snake가 아이템을 먹은 상태 , 먹은 아이템의 종류를 처리해서 snake의 길이를 늘리거나 줄이는 함수구현**

1. **Gate 요소의 구현**

* **통과하면 반대쪽 Gate로 Warp 하는 Gate 구현**

**immune Wall을 제외한 Wall에 Gate를 생성하는 함수 구현**

**Snake가 Gate를 지나고 있는지 판단하는 함수를 통해 Snake가 지날 동안 Gate가 닫히기 않게 구현**

**Gate를 지날 경우 출구에서 나올 방향을 계산 하는 함수를 구현**

**Gate를 지나면 다른 Gate에서 나오는 함수를 구현**

1. **점수 요소의 구현**

* **플레이어의 행동에 따른 점수 구현 (조건 충족시 stage 이동)**

**현재점수:**

**snake의 멤버변수로 counter 변수들을 추가.(score, growth, poison, gate 체크)**

**미션:**

**단계별 목표, 현재 스테이지 정보를 담을 mission class 구현.**

**mission 객체를 생성하여 snake 객체의 정보와 비교하며 미션 수행 상황 업데이트.**

**모든 미션 수행시, 다음 stage로 이동**

# 

## 

## 개발 내용 및 결과물

## 개발 내용

|  |
| --- |
| **스네이크 프로젝트를 시작하기 전, nucurse의 사용 방법을 익히기 위해 라이브러리 실습을 개별적으로 진행함. 배열을 이용한 맵(1단계)을 구현하기 앞서 스네이크의 움직임을 구현함. 이후 다른 상호작용을 구현하던 중 맵이 배열로 구현되지 않는다면 많은 부분에서 어려움이 있을 것이라고 현재 구현하던 부분은 팀원 중 한명이 계속 이어 나가고 나머지는 맵을 구현하며 팀원이 만들던 부분을 고쳐 맵에 옮겨 넣기로 함. (버전 1.0 - 1.1)**  **버전 1.1 에서 부족했던 부분들을 보완하고(버전 1.2), 3단계를 위한 아이템을 생성하기 위해, 아이템이 생성되는 랜덤 위치와 배열에 표현할 값을 구현함(버전 1.3). 스네이크가 아이템을 획득하더라도 늘거나 줄지 않는 버그가 생겨 이를 디버깅하며 규칙 #2도 함께 구현함(버전 1.4).**  **4단계 게이트 생성을 위해 규칙 #3을 우선적으로 구현함. 스네이크가 게이트에 진입 중일 때를 확인하기 위해 int형의 스네이크의 길이와 동일한 값을 갖는 변수를 생성하여 진입 중 지속적으로 값이 0에 수렴하게 설정함. 이 값이 0일 때와 0이 아닐 때를 구분하여 진입 하지 않는 상황, 진입 중인 상황을 구분함. 이후 규칙 #4를 구현하기 위하여 게이트에 진입 했을 시 출구(다른 게이트)의 위치와 출구에서 나올 방향을 정해주고 스네이크의 머리 부분이 출구 방향을 따를 수 있도록 변경함(버전 1.5).**  **4단계 구현 이후 스코어 보드나 미션 보드와 같은 추가 윈도우, 각 요소들에게 색상을 넣어 구분을 명확하게 함(버전 1.6).** |

### 시스템 구조 및 설계도

|  |
| --- |
| **1단계 Map**   * **class Map**   SnakeGame의 중심이 되는 Class이다. 멤버변수로 현재 stage를 저장하는 변수,  현재 게임이 진행중인 상태인지, 종료된 상태인지를 저장하는 bool gameover 변수,  Map 정보를 저장하는 2차원 배열 map이 있다.  생성자로 stage정보를 받아 stage에 맞는 맵을 구성한다.  이후에 snake, item, gate 의 좌표를 얻어 이들의 생성, 소멸, 움직임을 관리하는 역할을 하는 class이다.  (개발자 : 이정현)   * **void get\_set\_snake(snake& s)**   snake 객체로부터 snake의 헤더,바디의 좌표를 받아와서 실제 map의 좌표로 업데이트 하는 함수. (개발자 : 이정현)   * **void draw(int \*\*map ,WINDOW\* a)**   업데이트 된 snake, gate , item 요소들을 실제로 window에 그려주는 역할을 한다. map 전체를 for문으로 돌며 map에 표현된 숫자들을 매핑된 실제요소들로 출력해준다.  **(0: space , 1: immune wall , 2: nomal wall 3: snake head , 4: snake tail , 5: growth item , 6: poison , 7: gate)**  (개발자 이정현)  **2단계 Snake**   * **class Snake**   내부적으로 snk라는 2차원 배열을 가지고 있다.(snk[0][0]=head의 y좌표,snk[0][1]=head의 x좌표) 또한, dirt\_x, dirt\_y라는 int 변수를 통해서 현재 진행방향을 표현한다.(왼쪽=> dirt\_y = 0, dirt\_x = -1, 오른쪽=> dirt\_y = 0, dirt\_x = 1) (개발자:황찬우) score를 기록하기 위해, cnt\_Growth, cnt\_Poison, cnt\_Gate가 선언되어있다. (개발자:이정현)   * **void move\_snk()**   Snake 클래스 내부에 dirt\_x, dirt\_y이라는 변수를 통해 진행 방향을 나타내는데(위로 움직인다면 dirt\_y=-1,dirt\_x=0), 이때 snake의 head에 dirt\_x와 dirt\_y를 각각 더해주어 head를 움직이고, head 뒤 꼬리 부분은, for loop를 활용해 맨 뒤칸 부터 한칸씩 앞으로 옮기는 연산을 수행한다.(개발자:황찬우)   * **void get\_dirt()**   getch()함수를 통해 사용자의 입력을 받아온다. nodelay()함수 덕분에, 사용자가 아무것도 입력하지 않았다면 가장 최근의 dirt\_x와 dirt\_y가 그대로 유지되며, dirt\_x와 dirt\_y는 초기값이 있기 때문에 프로그램 실행 후 아무런 입력이 없어도 혼자 움직인다. 만약 사용자가 입력을 한다면, switch-case문을 통해 w,a,s,d에 따른 각각의 방향을 dirt\_x와 dirt\_y 변수에 저장한다.(개발자:황찬우)   * **void is\_crashed(int\*\* map)**   snake가 wall에 머리를 박았는지를 검사하는 함수이다. snake 클래스 밖에서 선언되었으며, map을 입력받아 wall이 있어야 할 위치에 snake의 head가 있는지를 감지해내는 함수이다.(개발자:황찬우,임상학)  **3단계 Item**   * **void make\_item(int times)**   시간을 재는 times 변수를 입력받아서 만약 times를 90으로 나누었을 때의 나머지가 0인 경우에(즉, 매 9초마다) 맵에 있는 5와 6(Growth/Poison item)을 지우고, map 위에 있으면서 wall이 아닌 새로운 서로 다른 2개의 점을 골라내어 하나에는 5를(Growth) 나머지 하나에는 6을(Poison)을 넣는다.  (개발자:황찬우,이정현,임상학)   * **void change\_snake(snake &s)**   아이템을 먹게된 상황을 처리하는 함수이다. snake의 head부분의 좌표가 아이템의 좌표와 일치하는 경우를 아이템을 먹은 상황으로 인지하는데, 이 때 5번에 해당하는 growth 아이템을 먹은 경우에는 snake의 꼬리를 한 칸 늘려주고 6번에 해당하는 poison 아이템을 먹은 경우에는 snake의 꼬리는 한 칸 줄어들게 된다.  동시에 점수에 관여하는 cnt\_growth , cnt\_poison의 값도 1씩 증가한다.  (개발자 : 황찬우, 이정현 , 임상학)  **4단계 Gate**   * **bool is\_snake\_in\_gate(snake& s, int& gateCode)**   Snake가 Gate안에 있는지 확인해주는 함수. Snake는 Head를 제외하고 나머지는 방향을 갖지 않고 앞의 Body를 그대로 따라가는 특성을 활용하여 Head가 Gate와 만나게 되면 gateCode에 Snake.size를 대입하고 매 틱이 지날때마다 1씩 줄어들게 하여 Body가 모두 Gate를 통과함과 동시에 gateCode가 0이 되게 하는 역할을 한다. (개발자: 임상학)   * **void makeGate(int time, int gateCode)**   gateCode가 0일때(Snake가 통과 중이지 않을 때) 새로운 Gate 쌍을 생성하도록 하는 함수. 일정한 생성 주기를 갖고 있으며, 조건에 맞는 위치에 Gate 쌍이 생성될 때 까지 반복하는 while문을 실행한다. (개발자: 황찬우, 임상학)   * **void warp\_direction(int y, int x, int& y\_dir, int& x\_dir)**   Snake가 Gate를 통과할 때 나와야 할 방향을 지정 해주는 함수. (개발자: 임상학)   * **void warp\_by\_gate(snake& s, int gateCode)**   Snake가 Gate를 통과할 때 나올 위치를 지정 해주는 함수. Snake의 Head의 위치가 Gate의 위치와 동일 할 때(= gateCode == Snake.size 일 때) Head의 위치를 받아오고 받아온 위치와 다른 위치에 있는 Gate의 위치를 찾는다. 다른 위치의 Gate를 찾은 후, warp\_direction()함수를 통해 나올 방향을 찾고 그 Gate의 x, y 좌표에 각각 나올 방향의 수치 x\_dir, y\_dir을 더한 위치를 Head에 대입해주면 Snake의 Body가 Head를 따라 Gate를 빠져나온다. (개발자: 임상학)  **5단계 Score & Mission**   * **class mission**   mission class는 멤버변수로 현재 stage를 표현하는 변수int stage 와 단계별로 달성해야 할 미션을 담은 배열 int mission\_list[3][4] 을 갖는다. 이후에 snake 객체의 값을 score window에 업데이트 하거나 Mission\_inst 의 값과 비교하며 Mission window를 업데이트 하는 역할을 한다. (개발자 : 이정현)   * **void updateScore(snake &s, WINDOW \*a)**   snake객체의 몸통의 길이를 나타내는 size 변수를 가져와 score 구성, item을 먹을 때 마다 증가하는 cnt\_Growth , cnt\_Poison 변수를 이용해 Score를 표시하는 Window를 업데이트 한다. window 에 업데이트 할때 mvwprintw 함수는 는 \*char형 변수만을 인자로 받기 때문에 별도로 데이터형을 변환하는 과정을 거친다. (개발자 : 이정현)   * **void updateMission(snake &s, WINDOW \*w, mission &m)**   updateScore 함수와 매커니즘은 같다. 다른점은 mission 객체의 멤버변수인 Mission\_list 의 지정된 값과 snake 객체의 값을 비교하여 미션을 완료했는지 비교하는 과정이 들어간다. 마찬가지로 별도 데이터형을 변환하는 과정을 거친 후, Mission Window에 업데이트 한다.(개발자 : 이정현) |

### 활용/개발된 기술

|  |
| --- |
| **1.cmath**  **rand() -** rand()함수를 사용하여 map의 멤버 함수인 make\_item함수에서 Growth/Poison item의 위치를, makeGate함수에서 gate의 위치를 랜덤하게 만들 수 있었다.  **2.unistd.h**  **usleep() -** usleep()함수를 사용하여 map.cpp의 main함수의 while loop에서, 한 틱이 끝난 후 다음 틱 까지의 시간 차이를 구현할 수 있었다.  **3.string**  **to\_string() -** ncurses의 윈도우는 \* char 형식의 문자열만 인자로 받을 수 있기 때문에 map의 int요소를 가져와 출력하기엔 부적합 하였다. 이러한 문제를 해결하기 위해 string의 to\_string함수를 이용해서 int형 변수를 string형으로 변환 한 후, cstring의 strcpy()함수를 사용하여 \*char 변수에 복사 해주었다.  **4.ncurses**  **nodelay() -** nodelay()를 사용하여 getch()로 입력키를 받는 과정에 만약 입력을 받지 못했다면, 진행 방향을 그대로 유지하게 만들었다.  **wborder() -** wborder()를 사용하여 생성한 window의 윤곽을 손쉽게 그릴 수 있었다.  **mvprintw(),mvwprintw() -** mvprintw(),mvwprintw()를 통해 특정 window의 원하는 좌표에 \* char 형식의 문자열을 출력할 수 있었다. |

### 현실적 제한 요소 및 그 해결 방안

|  |
| --- |
| score window와 mission 윈도우에는 점수를 입력하게 된다. 그러나 ncurses의 윈도우는 char\* 형식의 문자열만 인자로 받을 수 있기 때문에 int형 변수들(map의 snake의 멤버 변수)을 출력하기엔 부적합 하였다. 이러한 문제를 해결하기 위해 string의 to\_string함수를 이용해서 int형 변수를 string형으로 변환 한 후, cstring의 strcpy()함수를 사용하여 char\* 변수에 복사 해주어서 제한 상황을 해결했다. |

### 결과물 목록

|  |
| --- |
| **snake.cpp - Snake의 움직임을 관리하는 클래스**   1. **snk[][] 배열의 형태로 각 Body(Head 포함)의 위치를 저장** 2. **get\_dirt() 사용자에게 방향을 입력 받음** 3. **move\_snk() Snake를 움직임**   **map.cpp - 게임의 전체적인 부분을 관리하는 클래스**   1. **get\_set\_snake() Snake 클래스로부터 Body 의 위치들을 받아옴** 2. **make\_item() 다른 것과 겹치지 않는 랜덤한 위치에 아이템을 생성함** 3. **change\_snake() Snake가 아이템을 먹을 경우 크기를 늘리거나 줄임** 4. **makeGate() 벽 위치 중 두 곳에 랜덤하게 Gate를 생성함** 5. **is\_snake\_in\_gate() Snake가 Gate를 통과 중인지 확인함** 6. **warp\_direction() Gate를 통과할 경우 나아갈 방향을 전달함** 7. **warp\_by\_gate() Snake가 Gate를 통과할 시 다른 Gate로 이동**   **mission.cpp - 각 단계별 미션의 목표치를 저장한 클래스**  **main 함수 내부(map.cpp 안에 함께 구현됨)**   1. **각 window 초기화 및 초기 변수 설정** 2. **try - catch 문을 이용한 게임 종료 조건 처리** 3. **get\_time, sleep\_time 을 이용하여 일정 주기 반복 처리** 4. **draw() 변경된 맵을 출력** 5. **updateScore() 획득한 점수를 win\_scoreboard 에 반영** 6. **updateMission() 달성한 미션을 win\_missionboard 에 반영** 7. **is\_crashed() Snake가 충돌했는지 판정** |

# 자기평가

|  |
| --- |
| **황찬우 - 프로젝트 초반에 snake클래스를 구현하여 Growth/Poison item을 생성하고, 이들을 통해 뱀의 사이즈가 늘거나 줄어드는 로직 까지 구현했었다. 다만 이러한 구현이 gate를 생성하는 것에 불리하다는 것을 알게되었고, 팀원들의 도움 덕에 map클래스를 새로 만들어서 snake클래스에 구현되어있던 여러 함수들을 map클래스의 멤버함수로 추가할 수 있었다. 초반에 snake클래스를 만들 때 까지는 기여도가 높았으나, gate 구현 이후로는 아무래도 팀원의 수정을 거친 코드를 사용해야해서 ‘나의 것’으로 만드는 데 까지 꽤나 오랜 시간이 걸리게 되었고, 그 때문에 큰 성과를 내지 못한점이 매우 아쉬웠다.**  **프로젝트 수행에 있어서 어려운 점은 여태 한번도 써보지 못한 ncurses를 써야한다는 점이었다. 물론 구글링을 통해서 많은 정보들을 획득할 수 있었으나, 결국 완벽히 이해하지는 못했다. 따라서 ncurses에 관한 조금 더 구체적인 교안이나 강의가 있었으면 조금 더 프로젝트 진행이 수월했을 것 같다.**  **이정현 - 세부적인 게임 로직을 개발하기 보다는 map 클래스를 구성하며 전체적인 게임의 진행방향을 제시했고, 후에는 윈도우 구성, 점수 , 미션 관리등을 하며 내가 할 수 있는 일을 진행 했다. 팀원중에서 상대적으로 개발역량이 부족했기에 snake의 움직임, gate 로직같은 중요한 게임 요소를 함께 개발하기엔 효율적이지 않았다. 그래서 프로젝트 진행에 막힘이 없도록 내 능력내에서 수행 할 수 있는 일을 했다. 훌륭한 팀원들이 있었기에 성공적으로 프로젝트를 마무리 할 수 있었다. 혼자였다면 절대 완성하지 못했을거다. 많이 배웠고 힘들었지만 즐거운 시간이었다.**  **프로젝트 운영에 아쉬웠던 점은 snake게임에 대한 설명이 조금 부족했다고 느꼈다. 대부분 룰은 이해가 갔지만 일부 gate 룰에서 헷갈리는 부분이 있었다. 조금 설명이 보강되면 좋겠다는 생각이 든다.**  **임상학 - 프로젝트 진행 초반 단계에서 라이브러리에 대한 이해도 부족으로 기초가 되는 클래스들의 설계에 어려움이 있었으나 팀원들의 도움으로 어렵지 않게 시작할 수 있었다. 팀원들이 만든 클래스에 기반하여 뱀의 길이 변경, 아이템 생성, 게이트 생성 등 게임 내 요소끼리의 상호작용을 위한 함수를 담당하였다. 게이트를 생성하고 뱀을 이동시키는 함수를 작성하면서 처음에 접근을 어떻게 해야할 지 막막했으나 팀원들과 토론하며 각자 해결책을 내놓음으로써 만족스러운 결론에 도달 할 수 있었다고 생각한다. 서로간에 프로젝트 이해도가 다르더라도 부족한 부분을 보충 설명해주며 이해도를 높일 수 있었기에 원활한 팀프로젝트가 될 수 있었다. ncurses 라이브러리가 학부생이 C++을 실습하는데 크게 부족함이 없다고 생각하지만 나온지 오래된 라이브러리면서 최근 업데이트가 크게 이루어지지 않아 일부 요소가 부족하거나 활용하기 어려운 면이 있다. 다른 라이브러리를 사용하여 이러한 점이 개선된다면 좋겠다.** |

# 참고 문헌

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **번호** | **종류** | **제목** | **출처** | **발행년도** | **저자** |
| 1 | 웹페이지 | nodelay(3) - Linux man page | https://linux.die.net/man/3/nodelay | 미상 | die.net |
| 2 | 웹페이지 | NCURSES PROGRAMMING HOW TO | https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/ | 05.06.20. | Pradeep Padala |
| 3 | PDF | 90. ncurses\_tutorial | Kookmin Univ. | 미상 | 김준호 |

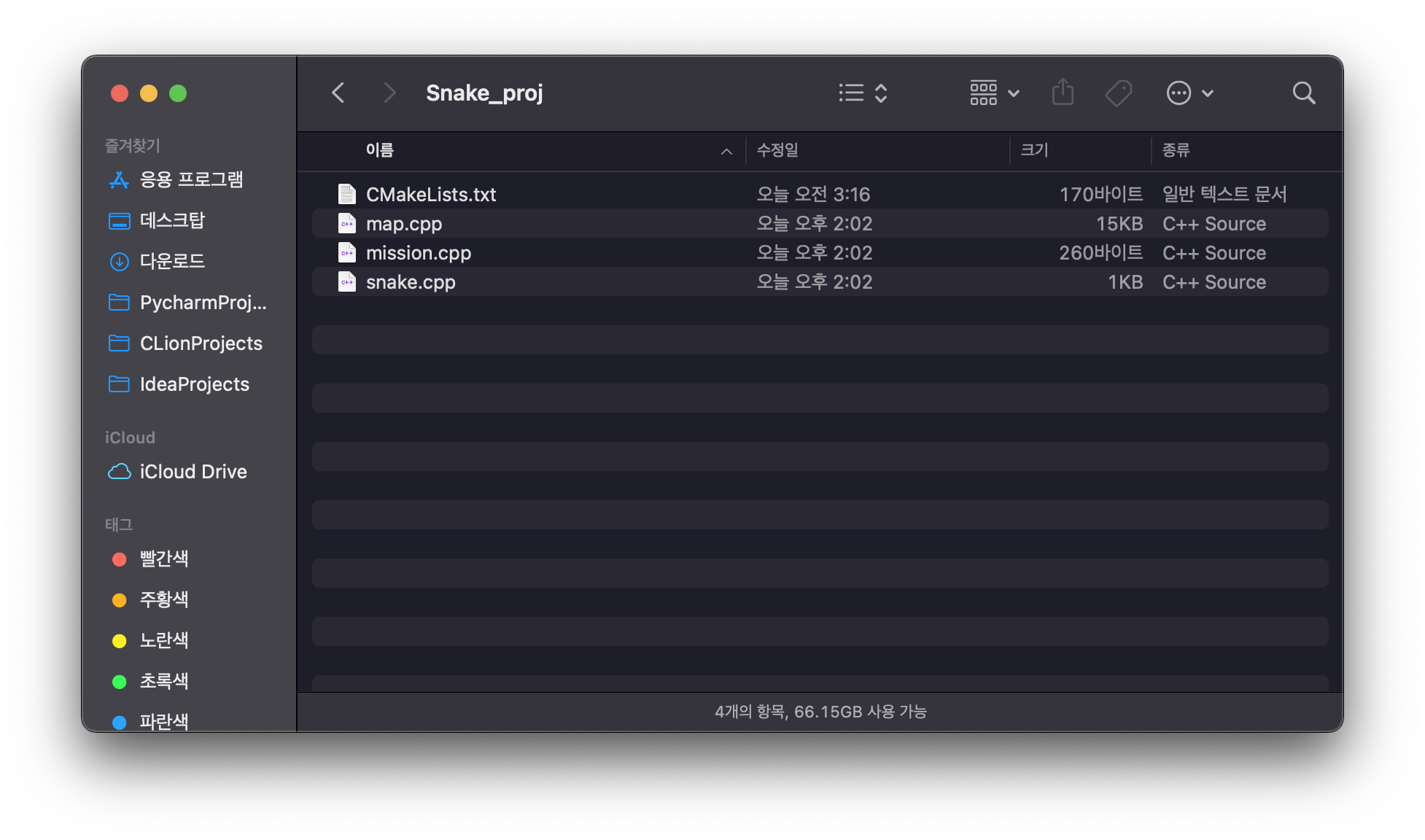
# 부록

## 사용자 매뉴얼

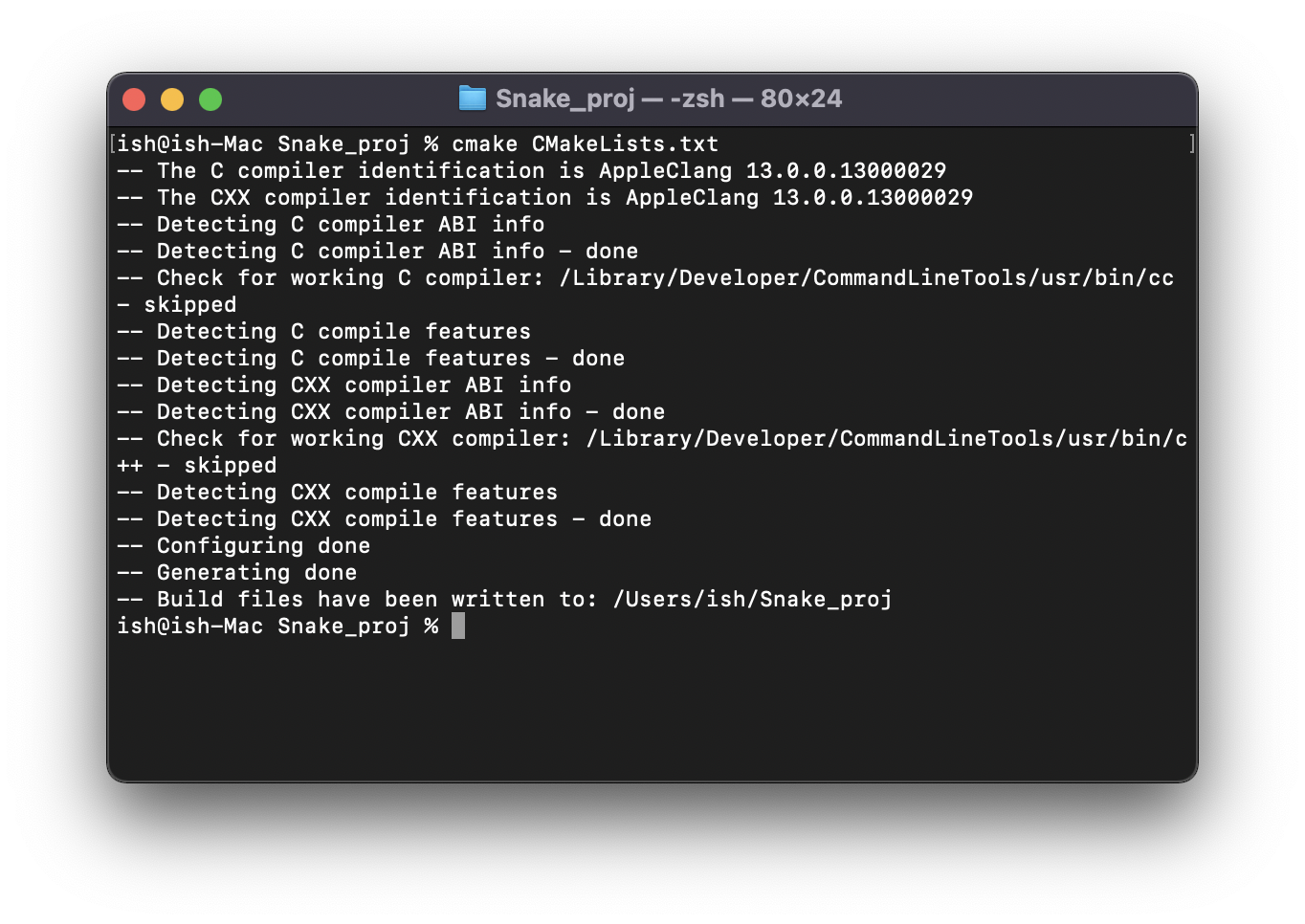
1. **w,a,s,d를 통해 각각 상,좌,하,우 로 움직이는 것이 가능하다.**
   1. **만약 프로그램이 시작되어도 아무런 행동을 취하지 않는다면, 프로그래밍 과정에서 제작자가 임의로 지정한 방향으로 움직인다.**
2. **@는 게이트 표시이고, 흰색 1,2 은 벽을 의미한다. 벽에 충돌하면 패배 메시지와 함께 게임이 종료된다.**
3. **초록색 G 물약을 획득시 뱀의 길이가 증가한다.**
4. **붉은색 P 물약을 획득시 뱀의 길이가 감소한다.**
5. **자주색 @ 게이트를 통과하면 다른 게이트로 이동한다.**
6. **사이즈가 3 미만이 되면 패배 메시지와 함께 게임이 종료된다.**
7. **뱀의 머리가 몸통을 통과하려고 하면 패배 메시지와 함께 게임이 종료된다.**
8. **mission에 정의된 도전과제를 완료하면 (v)체크 표시가 생기게 된다.**
9. **mission을 모두 완료하면 승리 메시지와 함께 게임이 종료된다.**

## 설치 방법(with CMake)

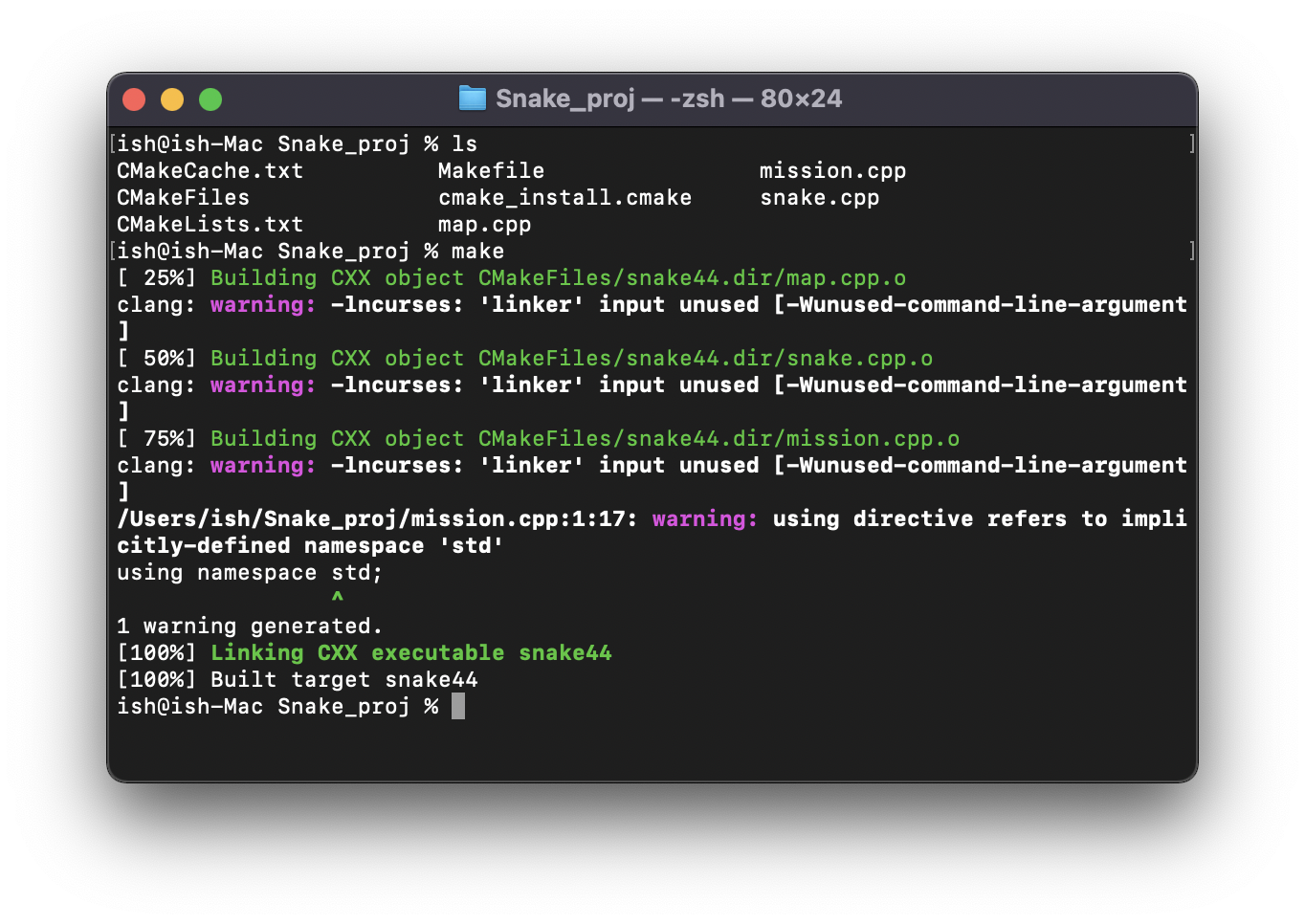
1. **CmakeLists.txt, map.cpp, mission.cpp, snake.cpp 을 동일한 폴더 안에 설치한다.**

****

1. **터미널에 “cmake CMakeLists.txt” 를 입력한다. (cmake 설치 필요)**

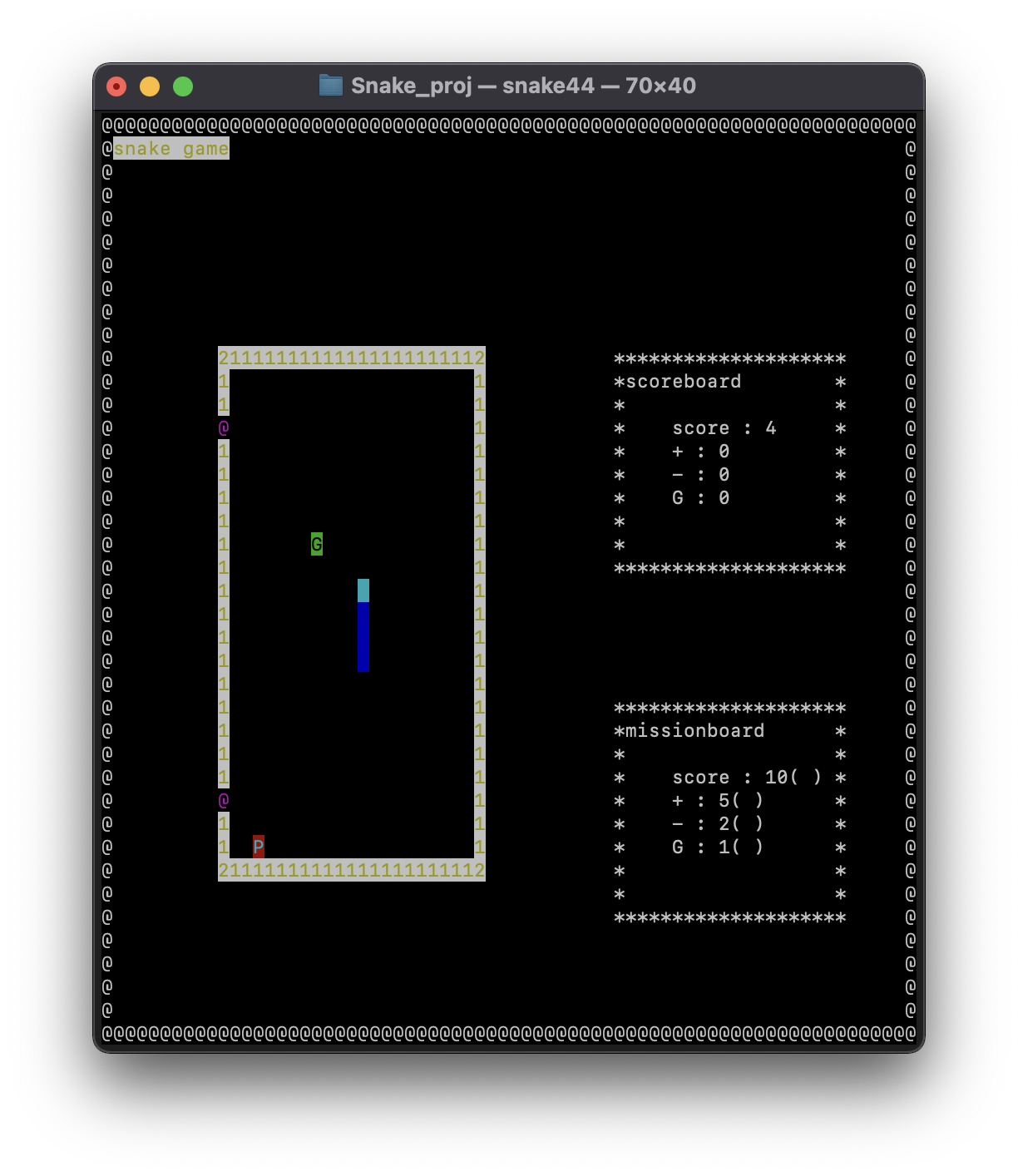
****

1. **CMakeCache.txt, Makefile, CMakeFiles, cmake\_install.cmake 이 생성되었다면 터미널에 “make”를 입력한다.**

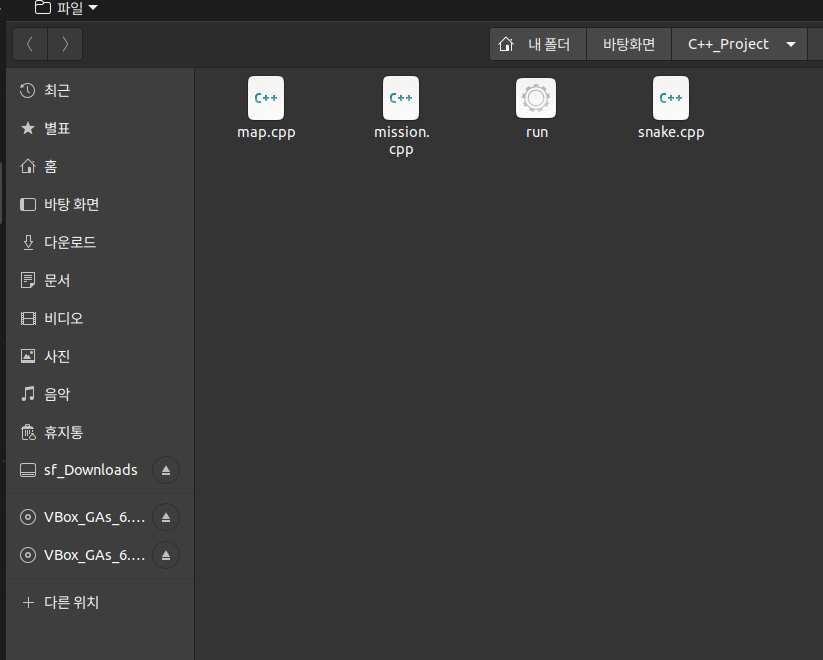
****

1. **터미널에 “./snake44”를 입력하면 파일이 실행된다.**

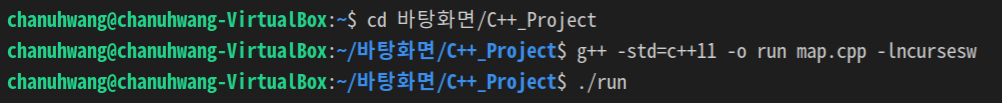
**(원활한 실행을 위하여 터미널의 크기는 70x40 이상을 권장)**

****

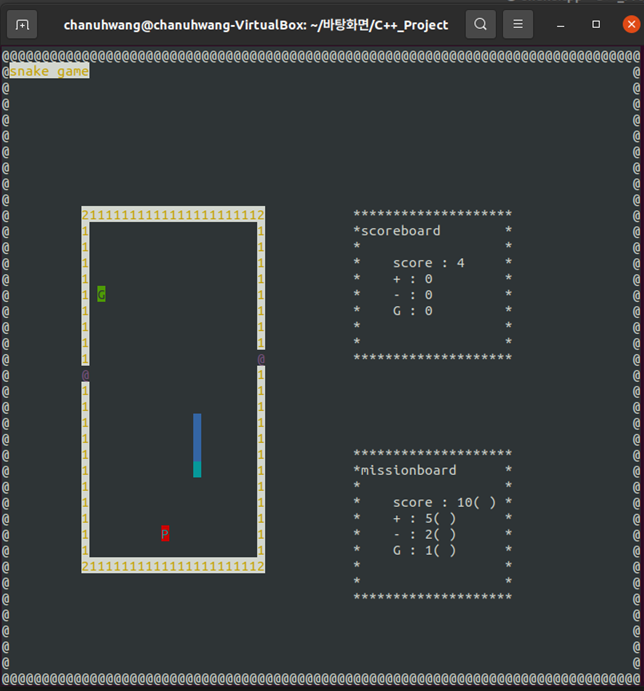
**5.3 설치방법(without CMake)**

****

1. **Snake.cpp, map.cpp, mission.cpp를 다운받는다. 되도록이면 같은 폴더 안에 넣는다.**

****

1. **g++ -std=c++11 -o (원하는 이름) map.cpp -lncursesw를 터미널에 입력한다. (단, 이때 cd 명령어를 이용해 Snake.cpp, map.cpp, mission.cpp가 있는 폴더 까지 이동하는 과정이 필요하다.)**
   1. **cd (주소)를 입력하면 바라보는 경로를 옮길 수 있다.**
   2. **만약 ncurses가 설치되어있지 않다면 터미널에** (Ubuntu**:**sudo apt-get install libncurses5-dev libncursesw5-dev / CentOS7:sudo yum install ncurses-devel)를 입력하여 **ncurses를 설치한 후 위의 과정을 다시한번 시도한다.**

****

1. **./(원하는 이름)을 입력해 프로그램을 실행한다.**