**C++프로그래밍**

**프로젝트**

| **프로젝트 명** | C++ 프로젝트 Snake Game |
| --- | --- |
| **팀 명** | 묻고 더블로가 |
| **문서 제목** | 결과보고서 |

| **Version** | 1.2 |
| --- | --- |
| **Date** | 2022-6-15 |

| **팀원** | 20191658 장정안 |
| --- | --- |
| 20191660 전병우 |
| 20191681 최정훈 |
|  |
|  |

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

**문서 정보 / 수정 내역**

| **Filename** | 최종보고서-C++ 프로젝트 Snake Game.doc |
| --- | --- |
| **원안작성자** | 장정안, 전병우, 최정훈 |
| **수정작업자** | 장정안, 전병우, 최정훈 |

| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| --- | --- | --- | --- | --- |
| 2022-05-30 | 장정안 | 1.0 | 최초 작성 | 프로젝트 내용 및 초안내용 작성 |
| 2021-06-03 | 장정안 | 1.1 | 내용 수정 | 추가된 기능 수정 |
| 2021-06-15 | 장정안 | 1.2 | 내용 수정 | 자기평가 작성 및 오탈자 검수 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_gjdgxs)

[2 개발 내용 및 결과물 5](#_30j0zll)

[2.1 목표 5](#_1fob9te)

[2.2 개발 내용 및 결과물 6](#_3znysh7)

[2.2.1 개발 내용 6](#_2et92p0)

[2.2.2 시스템 구조 및 설계도 6](#_tyjcwt)

[2.2.3 활용/개발된 기술 6](#_3dy6vkm)

[2.2.4 현실적 제한 요소 및 그 해결 방안 6](#_1t3h5sf)

[2.2.5 결과물 목록 7](#_2s8eyo1)

[3 자기평가 8](#_17dp8vu)

[4 참고 문헌 8](#_3rdcrjn)

[5 부록 8](#_lnxbz9)

[5.1 사용자 매뉴얼 8](#_35nkun2)

[5.2 설치 방법](#_1ksv4uv) 8

# 개요

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

프로젝트는 총 8개의 파일로 구성되어 있다. 파일을 실행하는 방법이 적혀져 있는 파일인 README.md, 게임을 첫 실행하면 나오는 메인 페이지 구현을 위한 파일인 Main.cpp와 게임을 구성하고, snake 생성 및 맵 구현 알고리즘이 들어가 있는 파일인 Stage.cpp, About\_game.cpp, About\_snake.cpp와 Stage.cpp, About\_game.cpp, About\_snake.cpp의 알고리즘 구현을 위한 헤더파일인 Stage.h. 그리고 게임 구성에 필요한 여러가지 물체들의 위치와 설정을 위한 Something.h 마지막으로, 게임 실행을 위한 컴파일을 하는 파일인 makeFile로 구성되어 있다.

개발 내용은 map크기를 최소 21x21로 하며 최소 4개 이상의 map 구현과 snake의 구현, 그리고 구현한 snake가 입력받는 방향으로 이동하게끔 하는것과, 만약 snake가 벽에 부딪히거나 길이가 3보다 작아지게 되면 게임이 종료되게 하는 것이다. 또한 게임을 진행하면서 stage 성공의 조건은 Snake의 길이, 각 아이템의 획득 개수, Gate통과 횟수로 판별한다. HELP 메뉴는 메인 메뉴에서 조작키 설명을 하며 OPTION은 플레이어가 Snake의 이동속도를 조절한다. 프로젝트를 위해 ncurses, unistd, vector, fstream 총 4개의 외부 라이브러리를 사용하였다. 프로젝트의 UI구현을 위하여 ncurses를 사용하였으며, 게임 진행 중 Satge를 완료하였을 때 Pause를 하도록 하였다. 그리고 게임 종료(Game Over)후 대기 시간을 처리하기 위해 unistd를 사용하였다. vector를 통해 manualTxt, itemPos, GatePos를 사용하기 편하도록 하였다. 마지막으로 파일 입출력을 위해서는 fstream를 사용하였으며, 위에 나열한 외부 라이브러리는 stage.h에 #include를 통해 사용하였다.

# 개발 내용 및 결과물

## 목표

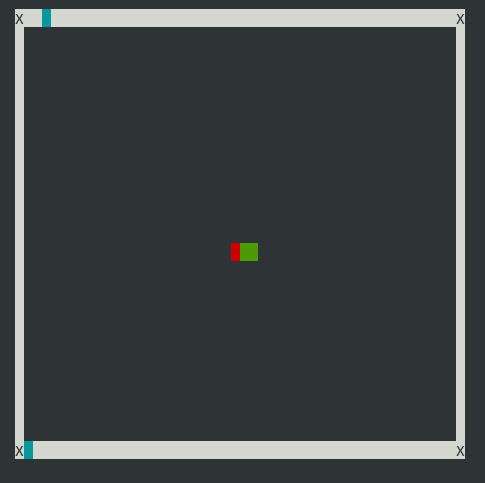
| **작성요령 (10점)**  **프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.** |
| --- |

| 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 |

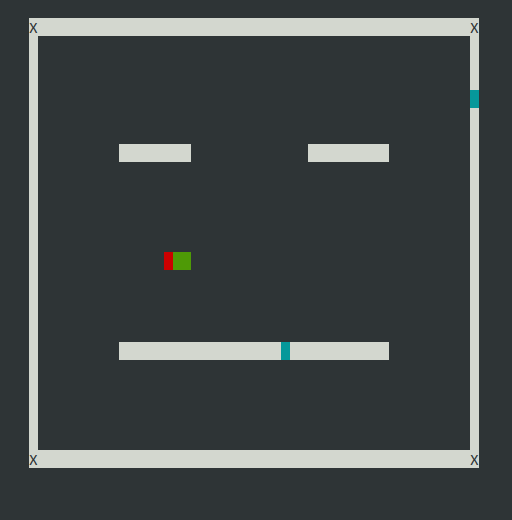
**1. Map의 구현**

총 5단계로 구성되어 있으며, 단계가 높아질 수록 더 장애물들의 난이도에 차이를 두었다.

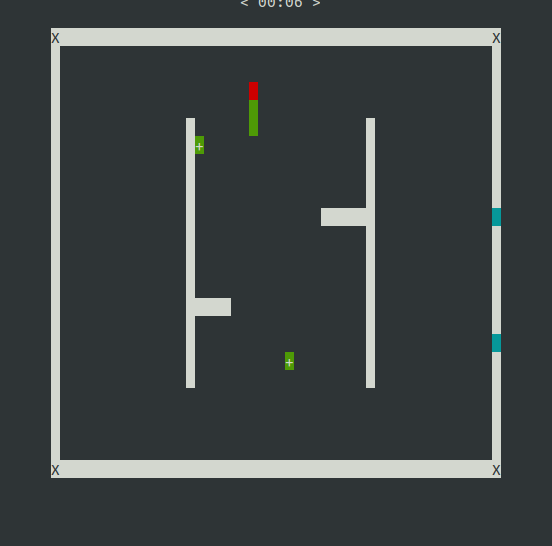
STAGE 1 map) 25x50의 크기의 아무런 장애물도 없는 직사각형의 map

****

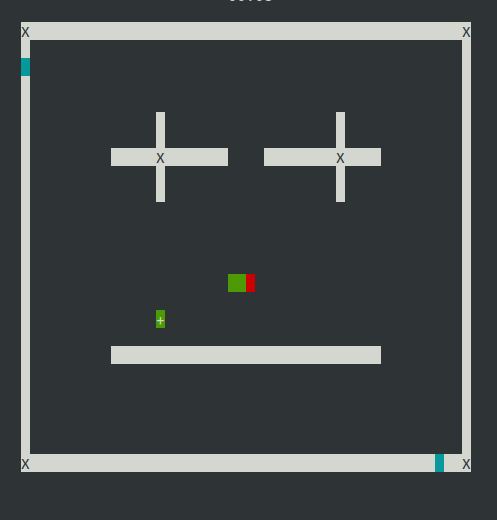
STAGE 2 map ) 25x50크기지만 크기 15인 긴 벽 한개와 크기가 5인 작은 벽 두개를 추가하였다.

****

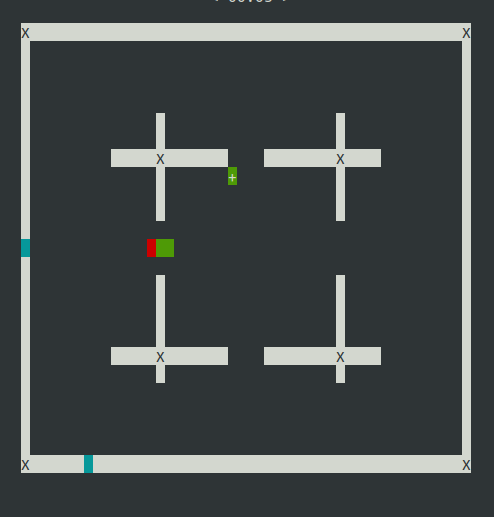
STAGE 3 map ) ㅏ 자형 벽과ㅓ자형 벽이 2개 있는 map



STAGE 4 map ) 상단에 십자가 모양의 벽이 2개 있고 아래 긴 벽이 가로로 하나 있는 map



STAGE 5 map ) 상, 하단에 십자가 모양의 벽이 총 4개 있는 map



**2. Snake 표현 및 조작**

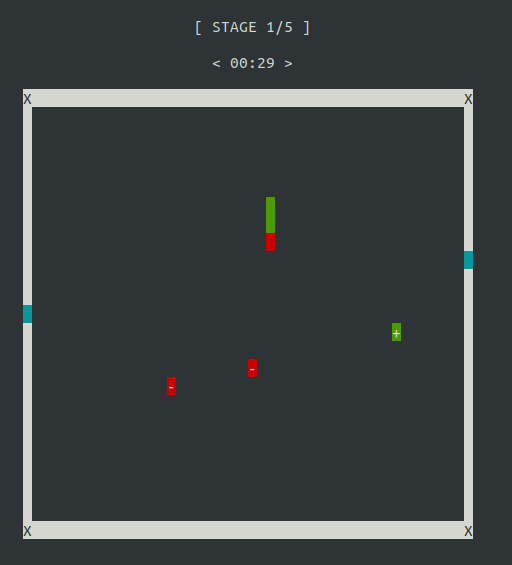
 - Snake의 모습

snake의 헤드는 숫자 0 으로 표시하고, 몸통과 꼬리 부분은 영어 O로 표현하였다. 하지만 게임 실행시 색이 뚜렷하게 보이지 않아서 색을 채워 넣었다.(head는 빨간색, body는 초록색). init\_pair(SNAKE\_HEAD, a, b)에서 a와b를 같은 색으로 지정하면 해당 색의 직사각형이 되는 원리를 활용하였다.

게임 시작시 Snake의 길이 3으로 시작한다. 만약 총 길이가 3보다 작아지게 되면 게임 종료(GAME OVER)를 한다.

조작은 방향키 UP, DOWN, LEFT, RIGHT로 구현하였고, 게임 도중 head의 진행방향이 반대인 키를 누르는 순간 게임 종료(GAME OVER)가 되도록 구현했다.

**3. Item 요소의 구현**

****

****

**(좌) Poison의 모습, (우) Growth의 모습**

아이템은 총 2가지가 있다.

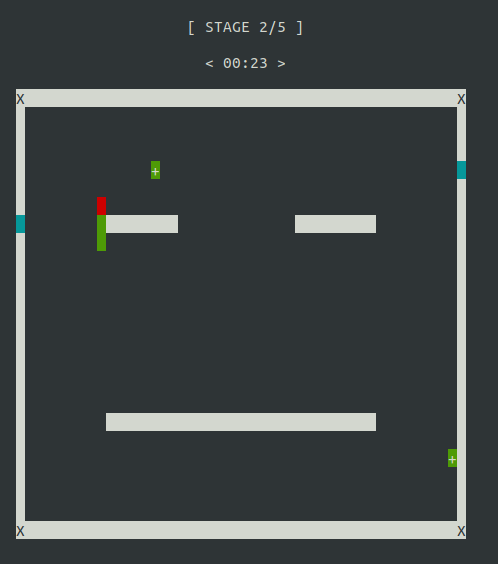
Growth와 Poison이 있는데, Snake가 Growth를 획득하게 되면 Snake의 꼬리가 한 칸 더 길어지게 구현하였고, Poison은 꼬리가 한칸 적어지게 구현하였다.

Growth를 먹어서 Snake의 길이는 최대 10칸까지 늘릴 수 있으며, 만약 Snake의 길이인 MAX\_LENGTH가 10이 되면 Item생성시 Growth는 생성하지 않고, Poison만 생성시킨다.

Poison은 Snake의 꼬리부분을 자르도록 한다. 만약 Snake의 몸 전체 길이가 3일때, Poison을 획득하게 되면 몸 길이가 3 보다 작아지게 되어 바로 게임 종료(GAME OVER)가 된다. 또한, Poison을 획득하는 미션을 모두 완료한 후, Poison은 더 이상 생성되지 않고, Growth만 생성된다.

Item은 스네이크가 존재하는 곳을 제외한 곳에서 무작위로 5초 간격을 두며 없어지고 새로운 item들이 한번에 최대 3개가 생성된다.

**4. Gate 요소의 구현**

←**Gate의 모습**

Gate는 두개가 한 쌍으로 구성되어 있으며, 두개의 Gate가 한 군데에 겹쳐서 나타나지 않도록 구현하였다. Gate는 게임 시작과 동시에 생성되며, Gate가 나타나는 위치는 임의의 위치이지만, 사방이 막혀있는 Immune Wall에는 나타나지 않는다. 또한, 나타날때 Gate는 한번에 한 쌍만 나타난다. Snake는 두 개의 Gate중 어느 방향으로나 들어가고 나갈 수 있으며, 한번 Snake가 지나간 Gate는 사라지고 새로운 Gate가 열리게 생성하였다. Snake가 지나가지 않은 게이트는 사라지지 않도록 하였다.

Snake가 Gate에 들어갈때 벽이 뚫려 있는 경우에는 어느 방향으로든 들어갈 수 있다.

그렇지만, 만약 Gate를 통해 나오게 되는 벽이 Map을 둘러싸는 가장자리 벽인 경우에는 항상 Map의 안쪽으로 진출하게끔 하였다.

예시)

좌측 벽 -> 오른쪽 방향

우측 벽 -> 왼쪽 방향

상단 벽 -> 아래 방향

하단 벽 -> 윗 방향

또는 Gate를 통해 나오게 되는 벽이 Map의 가운데 있을 경우에는 다음의 순서로 진출하게끔 구현하였다.

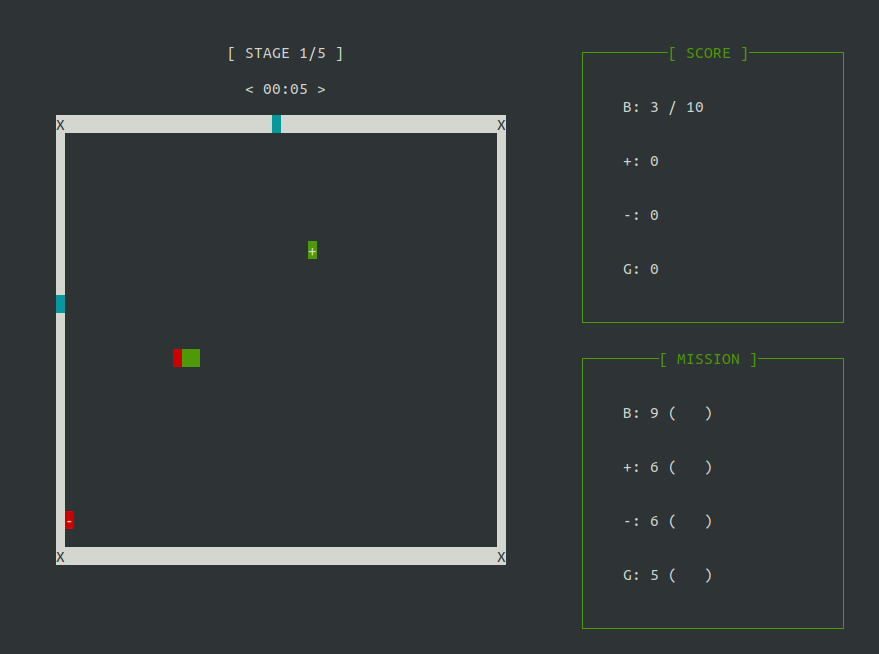
1. 진입 방향과 일치하는 방향

2. 진입 방향의 시계방향으로 회전하는 방향

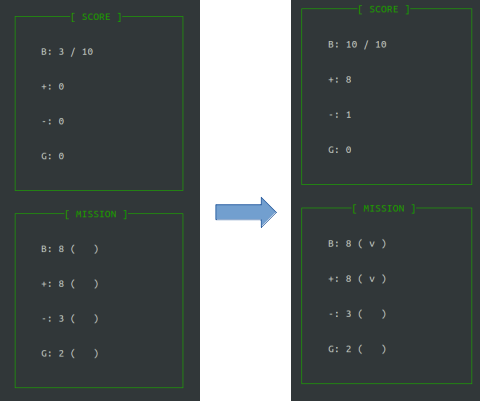
3. 진입 방향의 역시계방향으로 회전하는 방향

4. 진입 방향과 반대방향

**5. 점수 요소의 구현**

****

플레이 화면 우측상단에는 현재 점수(Score) 달성 상황을 보여주고, 우측 하단에는 달성해야하는 미션(MISSION)들을 보여준다.

****

위 그림의 아래 MISSION을 달성하게 되면 옆 괄호에 V자가 나타나게 되어 달성 여부를 표시해 주게 된다. 위 왼쪽 그림은 게임 시작과 동시에 몸 길이 8, Growth item 8개 획득, Poison Item 3개 획득, Gate 두번 통과를 미션으로 준 상황이다. 오른쪽 그림은 몸 길이 8과 Growth Item 미션이 완료되어 V가 되어 있는 상태이다.

MISSION은 매 판 마다 랜덤으로 부여된다.

B는 게임 중 몸의 최대 길이를 계산한다. (B : 현재길이 / 최대길이 로 표시)

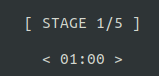
+는 게임 중 획득한 Growth Item의 수를 나타낸다.

-는 게임 중 획득한 Poison Item의 수를 나타낸다.

G는 게임 중 Gate를 지나간 횟수를 나타낸다.

또한 Game Play Time은 게임 상단에 나타내는데, mm:ss 형식으로 나타낸다.

Pause하면 시간은 멈추고, Resume하면 시간이 다시 작동하는 형식이다.



## 개발 내용 및 결과물

### 개발 내용

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

snake game을 만들면서 설정한 목표는 크게 두가지 이다. 첫번째 목표는 맵을 총 5단계로 구현하는 것이었으며, 두번째 목표는 snake 조작의 구현이었다.

첫번째 목표를 달성 시에 난이도의 차이를 어떻게 설정 할지에 대한 고민이 있었다. 고민 해결의 첫번째 방안은 맵의 크기를 줄이는 것이었고 두번째의 방안은 맵의 총 크기는 그대로 두고 , 맵 안에 벽으로 처리되는 장애물들을 넣는 것이었다. 팀원과의 논의 끝에 사용자가 게임을 실행 했을 시에 장애물들을 넣어서 난이도를 두는 것이 게임의 흥미에 더 도움이 된다고 판단하여 맵의 총 크기는 그대로 두고, 맵 안에 장애물들을 넣어서 난이도의 차이를 두기로 하였다.

두번째의 목표 달성시에 두가지 고민이 있었다. 첫번째는 Snake의 조작에 관한 것이었고 두번째는 Head와 Tail의 구분을 어떻게 하면 직관적으로 할 수 있을지에 관한 고민 이었다. 첫번째 고민의 해결 방안으로는 snake 조작을 wasd로 할 것인지, 키패드의 UP,DOWN,LEFT, RIGHT로 할 것인지가 있었다. 둘 중에서 고민한 결과 사용자 입장에서 더욱 직관적인 키패드(UP,DOWN,LEFT, RIGHT)로 조작을 하기로 결정하였다. 그리고 두번째 고민은 HEAD 와 TAIL의 명확한 구분을 어떻게 할 것인가 였다. 색을 사용하여 구분을 할 것인지 아니면 다른 방법으로 할 지 고민한 결과 숫자 0과 영어 O의 크기와 모양이 비슷하다는 점을 발견하여, Snake의 HEAD는 숫자 0으로, Body와 tail부분은 영어 O로 구분하였다. 하지만, 가독성의 문제로 Head부분과 Tail부분의 색을 다르게 하여 직사각형으로 표현하였다.

그 다음 Gate와 Item은 플레이어가 게임 플레이에 혼동을 하지 않도록 확실하게 구분이 가능한 색깔로 결정하였다. Gate와 Item의 구현 과정은 아래에 더 자세하게 설명하겠다..

다음으로, Gate가 나타날 수 있는 벽과, 나타날 수 없는 벽의 구분을 위해 문자 X를 사용하였다.

그러고, 플레이어가 입력하는 키들은 DEFINE으로 선언하였다.

### 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.** |
| --- |

Snake의 구현을 위해 작성한 코드 부분은 About\_snake.cpp부분에서 makeSnake(), moveSnake(), Bam을 사용하였다. 처음 게임 시작 시, snake를 생성하는 함수는 makeSnake()이다. makeSnake에서는 처음 snake의 크기를 3으로 설정하고, 처음 나타낼 위치와, HEAD는 0으로, BODY는 O으로 생성시킨다. moveSnake()에서는 Snake의 움직임을 구현하였다. 뱀이 지나가고 난 자리는 다시 빈 자리로 처리하고, 키패드 LEFT를 누르면 뱀이 왼쪽으로 진행하도록 하였다. 또한, 뱀이 움직이는데 만약 HEAD가 벽을 지나가게 되거나, 또는 뱀의 몸이 벽을 만나게 되면 GAMEOVER가 되도록 구현하였다. 또한 뱀의 머리(head)가 GATE를 지나가면 GATE로 들어가게끔 구현하고, ITEM을 지나가면 ITEM을 획득하도록 하였다. 뱀의 진행 방향은 dir라는 변수를 사용하여 나타내었다. 이 기능은 장정안이 구현하였다.

GATE의 구현은 appearGate(), disappearGate()를 사용하여 구현하였다. appearGate()는 맵에 나타난 벽 들 중 랜덤으로 한 군데를 골라서 나타내게끔 하였다. 두 쌍이 한군데에 나타나지 않도록 하였고, 생성하고 난 뒤에는 두 게이트를 gatePos에 한 쌍으로 묶어주었다. disappearGate()에서는 생성된 게이트들을 쌍으로 묶은 gatePos로 받아 스네이크가 지나가고 난 뒤엔 다시 벽으로 바꿔주게끔 구현하였다. 벽으로 바꾼 후에는 다시 gatePost를 초기화해 새롭게 생성되는 gate를 받을 수 있게끔 하였다. 이것은 최정훈이 구현하였다.

나머지 snake와 관련된 GATE의 함수가 있는데 enterGate(Something \*head)와 findRoot(Something \*gate)가 있다. enterGate에서는 snake의 head가 게이트로 들어온 뒤에 일어나는 일들을 구현하였다. 먼저 chkEnter를 통해 현재 스네이크가 들어왔음을 나타내고, snake가 gate1 로 들어왔는지, gate2로 들어왔는지를 판단한 후에, findRoot()를 통해 snake가 나가는 위치와 방향을 반환해 준다. findRoot는 snake가 현재 들어온 방향과 나갈 gate의 위치에 따라 snake가 나가서 진행할 방향을 반환해 주는 함수이다. 이 기능은 장정안이 구현하였다.

Map의 구현은 setMap, copyMap, drawMap을 통해 구현하였다. 먼저 Stage의 총 맵 수는 5개이고, 25x50의 고정크기이다. setMap에서는 25x50의 맵을 만들고, 가장자리는 벽으로, 중간부분은 빈 부분으로 처리한다. 그러고, 맵의 모서리 부분은 게이트가 열리지 않는 IMMUNE\_WALL로 설정하였다. 또한, stage에 따라 각각 wall의 위치와 중간 장애물의 위치를 구현하였다. copyMap을 통해 게임 플레이 시 해당하는 스테이지의 Map을 불러오게 된다. 그리고 drawMap()을 통해 불러온 맵들에 ncurses를 이용해 벽과 빈 공간, Immune wall등에 UI를 적용시키고, Score와 Mission, Stage 현황등을 전체적으로 구현하게 된다. 이 기능은 최정훈이 구현하였다.

Item의 구현은 appearItem(), disappearItem()을 통해 구현하였다. appearItem은 한번에 최소 1개 ~ 최대 3개 까지 나타나게 하였고 나타나는 item의 타입은 growthItem과 poison중 랜덤으로 나타나게 하였으나, snake의 길이가 이미 최대 길이면 poison만 나타나고, poison의 미션을 완료하면 poison만 나타나게끔 하였다. item이 나타나는 위치는 빈 공간에만 나타나도록 구현하였다. disappearItem에서는 아이템이 있었던 곳을 다시 빈공간으로 바꿔주게끔 구현하였다. 이 기능은 최정훈이 구현하였다.

snake가 item을 먹기때문에 그것을 위한 함수인 eatItem(int item)를 구현하였다. eatItem에서는 스네이크가 Growth를 지나가면 snake의 길이를 1만큼 늘려주고, Poison을 지나가면 꼬리의 길이가 1만큼 줄어들게끔 구현하였다. 또한 snake의 길이가 이미 max\_length일때에는 Growth를 먹어도 길이가 늘어나지 않도록 예외처리를 해줬다.. 이 기능은 장정안이 구현하였다.

Mission은 setMission, isMissionClear을 통해 구현하였는데, setMission에서는 총 4개의 미션을 설정하였는데, 첫번째로 스네이크의 길이 미션은 6~10 사이로 나타나게끔 설정하고, growthItem을 획득하는 미션은 4~8 사이, poisonItem의 미션은 3~6사이, 그리고 gate를 지나는 미션은 1~5 사이로 설정할 수 있게끔 rand 를 사용하여 설정해 주었다. 이렇게 설정한 미션은 statMission이라는 배열에 선언해 준다. isMissionClear를 통해 미션 수행 여부를 판단할 수 있는데, stat이라는 현재 상황을 저장한 배열과 statMission이라는 배열을 비교해서 만약 미션을 달성했다면, ‘v’라는 글자를 나타내고 성공한 만큼 count를 하게 된다. 만약 모든 미션이 성공하여 count=4가 되면 true를 리턴해, 성공했음을 반환한다. 이 기능은 전병우가 구현하였다.

이러한 코드들을 Play에서 사용하여 게임을 구현하였는데, play함수에서는 위에서 설명한 함수들을 불러오고, 키 입력을 받아 게임을 진행할 수 있게끔 구현하였다. 누르는 방향키에 따라 dir이 바뀌고, 만약 현재 dir과 눌리는 방향키가 반대일 경우 게임오버가 되게끔 구현하였고, pause 단축키인 ‘P’를 입력시 게임이 pause되게 끔 구현하였다. pause에 걸리면 ‘R키를 누를 때 까지 대기하게끔 구현하였다. 또한, Stage를 스킵할 수 있게끔 하는 커스텀 단축키도 구현하였다. 또한 해당 스테이지의 모든 미션을 완료하면 다음 스테이지로 넘어 갈 수 있게끔 하는 기능과 뱀의 길이가 3미만이 되면 바로 게임오버가 되게끔 하는 기능도 구현이 되어 있다. 이 기능은 전병우가 구현하였다.

게임 play 이전에 나오는 메인 메뉴는 main.cpp와 menu()으로 구성되어 있다. main.cpp을 통해 어떤 메뉴를 실행 시킬지 결정하는 것을 구현하였고, menu()를 통해 메인 메뉴의 UI를 구현하였다. 메인 메뉴에서의 HELP메뉴는 help()함수를 통해 구현되었다. stage에서 선언한 menuTxt와 shorTxt를 모니터를 통해 노출시킨다. OPTION메뉴는 option()함수를 통해 구현되었는데, 이 메뉴에서는 snake의 진행 속도를 설정할 수 있다. 기본적으로는 4레벨인 0.1 초에 한칸씩 움직이게 되어 있지만 1로 설정하면 0.5초에 한칸, 2는 0.25초에 한칸, 3은 0.2초에 한칸, 4는 0.1초에 한칸, 5는 0.05초에 한칸씩 움직이게끔 설정할 수 있게끔 구현하였다. 이 기능은 모두 다같이 구현하였다.

### 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.** |
| --- |

프로젝트를 위해 사용한 외부 라이브러리는 ncurses, unistd, vector, fstream을 사용하였다.

ncurses를 사용해서 프로젝트의 UI를 구현하였다. 또한 unistd은 Pause시키거나, Stage Clear, Game Over후 약간의 대기 시간을 주기 위해 usleep()을 통해 사용하였다. 또한 vertor는 manualTxt, itemPos, GatePos를 활용하기 편하게 만들기 위해 쌍을 만들거나 push\_back과 clear를 위해 사용하였다. fstream은 파일 입출력을 위해 사용하였다.

이 프로젝트를 수행하면서, 기존 새롭게 고안한 알고리즘은 메인 화면의 option부분이다. 이를 통해 snake의 진행 속도를 플레이어가 총 5단계 안에서 설정할 수 있도록 하였다.

### 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.** |
| --- |

게임을 만들면서, 단계 별로 맵이 의도한 대로 생성되는지, 아이템과 게이트는 정상적으로 작동하는지 확인하는데 시간이 많이 들었다. 이를 위해 아래와 같은 개발자 이스터 에그를 추가했다.

맵 구현과 테스트를 위해 미션을 완료하지 않아도 다음 스테이지로 넘어 갈 수 있는 단축키를 개발하였다. 게임 플레이 중 ‘s’를 누르면 미션성공 여부와 관계없이 이번 스테이지를 클리어하고 다음 스테이지로 넘어 갈 수 있다.

### 결과물 목록

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

**결과물은 총 8개이다.**

1. **README.md**

**: 파일을 실행하는 방법이 적혀져 있는 파일이다.**

1. **Main.cpp**

**: 게임을 첫 실행하면 나오는 메인 페이지 구현을 위한 파일**

1. **Stage.cpp**

**: 게임의 총 알고리즘이 들어가 있는 파일이다.**

1. **Stage.h**

**: 게임의 알고리즘 구현을 위한 헤더파일이다.**

1. **Something.h**

**: 게임 구성에 여러가지 물체들의 위치와 설정을 위한 파일**

1. **makeFile**

**: 게임 실행을 위한 컴파일을 간단하게 할 수 있도록 도와주는 파일이다.**

1. **About\_snake.cpp**

**: Snake의 행동과 관련된 기능(생성, 움직임, 게이트 통과, 아이템 획득)이 구현된 파일이다.**

1. **About\_game.cpp**

**: 게임진행과 관련된 기능(미션설정, 미션성공여부, 게임종료 여부)이 구현된 파일이다.**

# 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.** |
| --- |

장정안 - 이런 게임을 만드는 프로젝트를 처음으로 해보았고 심지어 nucurses라는 library를 처음 써보아서 그런지 처음에는 프로젝트 시작부터가 굉장히 두렵고 막막했습니다. 아직까지 부족한 부분도 많고, c++에 있어서도 잘 모르는 부분이 있는 상태로 프로젝트를 진행하다보니 하지 않아도 될 고생을 했던 부분도 많이 있었던 것 같습니다. 각자 게임을 만들면서 맡은 부분이 있었는데 저는 snake와 관련된 함수 구현을 맡았습니다. 이론에만 국한되어 있다가 팀을 이루어서 이렇게 맡은 부분을 수행하여 함께 결과물을 갖는 프로젝트를 한 덕분에 부족한 점도 많이 깨달았습니다. 그리고 앞으로 이 부족한 부분들을 채워가야 겠다는 다짐도 할 수 있었습니다. 무엇보다 이 프로젝트를 통해서 가장 성장한 부분은 모르는 부분이 있거나 막히는 부분에서 팀원들과의 소통이 굉장히 중요하다는 것을 깨달았다는 점입니다. 팀원들과 함께 고민하고 서로가 맡은 부분을 피드백해준 덕분에 이번 프로젝트를 잘 마무리 할 수 있었다고 생각합니다. 이 프로젝트 운영에 있어서 개선이 필요하다고 생각하는 부분은 저희 팀 경우에는 의사소통이 잘 되어 비교적 원할하게 진행되었지만 그렇지 못한 팀들도 보였습니다. 팀원들간에 소통이 핵심이 되는 팀 프로젝트에서 원할한 소통이 이루어질 수 있는 시간을 따로 갖도록 수업시간이 끝나고 강의실에서 팀별로 모여 10분이라도 논의를 할 수 있는 시간을 주면 개선을 할 수 있지 않을까 생각합니다.

전병우 - 처음엔 어떻게 구현을 하여야 할지 막막 하였지만 팀원들과 천천히 전체적인 틀을 잡고 각각의 부분에서 세부적인 틀을 잡으니 할 수 있을 것 같다는 자신감이 생겼습니다. c++ 언어에 대한 이해도 깊지 않은 상태에서 게임 구현을 하려고 하니 막히는 부분이 많았지만 그럴때마다 팀원들과 고민거리에 대한 해결방안을 모색해낸 결과 혼자였으면 불가능 하였을 것도 해내었습니다. 이러한 과정 속에서 평소에 혼자서만 고민하는 것보다 고민을 공유하며 같이 생각 하는 것에 대한 의미를 깨닳게 된 것 같습니다. 또한 직접 부딪혀 가면서 배우는 것이 많다는 것을 배웠으며 앞으로는 되든 안되는 한 번 해보는 것이 나 자신에게 도움이 됨을 깨닫게 됐습니다. 미션 플레이 시 단축키 구현과 커스텀 단축키 구현 및 게임 기능 메뉴 구현을 위주로 역할을 맡았는데 구현시 어려움을 겪을 때 개인이 맡은 부분이 아니더라도 나머지 팀원들이 자기 역할 인 것 처럼 도움을 주었기 때문에 프로젝트 구현을 할 수 있었습니다.

최정훈 - 미래에 게임개발을 꿈꾸고 있는데, 이번 프로젝트가 좋은 기회가 되었던 것 같다. 시험기간과 프로젝트 진행기간이 어느정도 겹치는 부분이 있어서 적절히 시간을 배분하는것이 가장 어려웠던 것 같다. 처음에는 과연 지금까지 배운것만으로 구현할 수 있을까 라는 생각도 들었지만, 한번에 한 기능씩 차근차근히 구현하다 보니까 어느새 완성되있어서 놀라우면서도 같이 고생한 팀원들에게 감사했다. 본인은 맵 설계와 구현부분을 위주로 맡았는데, 5개의 맵에 대해서 점점 난이도를 올리면서 벽을 배치하려다보니, 창작의 고통을 느꼈었다. 아쉬웠던 점으로는 개발을 하면서 다른 팀들과 구별되는 점으로 게임이 종료되거나 아이템을 먹을 때 효과음을 추가하고 싶었지만, 소리추가하는 방법을 몰라서 구현하지 못했다. 추후에 시간이 된다면 효과음 추가까지 구현해보면 좋을것 같다.

# 참고 문헌

| **번호** | **종류** | **제목** | **출처** | **발행년도** | **저자** | **기타** |
| --- | --- | --- | --- | --- | --- | --- |
| 01 | 웹페이지 | https://blackinkgj.github.io/ncurses/ | 인터넷 | 2019.04.11 |  |  |
| 02 | 웹페이지 | https://psman2.tistory.com/entry/ncurses-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D | 인터넷 | 2011. 5. 2. 11:37 | 윤상배 |  |
| 03 | 웹페이지 | https://game-happy-world.tistory.com/47 | 인터넷 | 2021. 4. 23 |  |  |
| 04 | 웹페이지 | https://invisible-island.net/ncurses/announce.html | 인터넷 |  |  |  |

# 부록

## 사용자 매뉴얼

**프로젝트 실행 후 메인 화면에는 ‘PLAY’, ‘HELP’, ‘OPTION’, ‘EXIT’이 나온다.**

**각각의 메뉴에 대해 설명하겠다.**

**‘PLAY’는 게임을 실행하는 메뉴이다.**

**‘HELP’는 조작 키 설명과 단축키 설명이 나온다.**

**‘OPTION’은 Snake의 속도를 조정할 수 있다.**

**‘EXIT’은 프로그램이 종료되게 된다.**

**메뉴를 선택할 수 있는 키는 방향키 UP, DOWN으로 선택 할 수 있고, 선택을 결정하기 위해서는 ‘ENTER’키를 누르면 해당 메뉴로 들어갈 수 있다. ‘ESC’를 누르면 메인 화면으로 다시 돌아오게 된다.**

**다음으로 게임 실행중 단축키들에 대해 설명하겠다.**

**총 ‘P’, ’R’, ’ESC’, ‘S’ 4개의 단축키가 있다.**

**‘P’는 일시정지 단축키이다. Pause의 앞글자로 표현하였다.**

**‘R’은 게임 제개 단축키이다. Resume의 앞글자로 표현하였다. ‘P’를 눌러 게임을 일시정지한 후, 다시 진행할 때 누른다.**

**‘ESC’는 하던 게임을 중지시키고 메인 메뉴로 돌아가는 단축키이다.**

**< \*개발자 Easter Egg\* >**

**‘S’키를 누르면 현재 Stage를 그 즉시 clear하고 다음 Stage로 넘어간다. 구현 목적은 개발당시 각종 버그해결과 에코 체킹 및 원활한 보고서 작성에 필요한 사진을 찍기위함이다. 게임을 플레이하는 플레이어 입장에서는 일종의 ‘치트키’이므로 Help메뉴에서는 이와 관련된 내용을 서술하지 않았다.**

## 설치 방법

**main.cpp, Makefile, readme.md, stage.h, stage.cpp, About\_game.cpp, About\_snake.cpp 파일이 포함된 압축 폴더를 다운받는다.**

**압축을 푼 다음에 해당 폴더에서 새로운 터미널을 연다.**

**그다음 make를 실행시켜 컴파일을 한 뒤에 ‘./run’을 통해 프로그램을 실행할 수 있다.**