**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | Snake-Game 제작 |
| 팀 명 | 3조 |
| 문서 제목 | sNake-game 제작보고서 |

|  |  |
| --- | --- |
| **Version** | 1.4 |
| **Date** | 2020.06.19 |

|  |  |
| --- | --- |
| **팀원** | 김학균(20171607) |
| 김민준(20171591) |
| 노성환(20171615) |
| 문성찬(20171620) |

|  |
| --- |
| **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 “3조”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “3조”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-Snake-Game.doc |
| **원안작성자** | 김학균, 김민준, 노성환, 문성찬 |
| **수정작업자** | 김학균, 김민준, 노성환, 문성찬 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2020-06-16 | 김학균 | 1.0 | 최초 작성 | 전체적인 목차에 따라 내용 구성(초안) |
| 2020-06-18 | 김학균 | 1.1 | 개발내용 작성및 동영상제작 | 1단계~3단계 개발내용 작성 및 목표 작성 및 결과물 제출목록 동영상 제작 |
| 2020-06-19 | 노성환 | 1.2 | 개발내용 작성 | 4단계~5단계 개발내용 작성 |
| 2020-06-19 | 김민준 | 1.3 | 개발내용 작성 | 2.2.1개발내용(전체적인 틀)작성 |
| 2020-06-19 | 문성찬 | 1.4 | 확인 및 수정 | 보고서 내용 검토 및 수정 |
|  |  |  |  |  |
|  |  |  |  |  |

**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

**사용한 언어는 C++ 언어와 Ncurses를 사용하고, 이를 이용한 코드 파일은 4개로 구성한다.**

* SnakeGame의 헤더파일을 만들어서 SnakeGame에 필요할 변수와 메소드를 선언했다. 그리고 snakeGame.cpp에서 snakeGame 헤더파일에서 선언한 변수와 메소드를 구현해주었다. main.cpp에서는 전체적으로 snakeGame.cpp에서 구현된 메소드를 작동시키기 위해 구현했다**.**

**#include “snakeGame.h”**

**#include <unistd.h>**

**main.cpp**

**#include “snakeGame.h”**

**#include <unistd.h>**

**snakeGame.cpp**

**snakeGame.h**

**#include <iostream>**

**#include <vector>**

**#include <ncurses.h>**

**#inlcude <cstdlib>**

**#include <ctime>**

# 개발 내용 및 결과물

## 목표

**아래의 목표를 기준으로 필요한 변수와 함수들을 헤더파일을 작성하였다.(아래의 사진 참고)**

|  |  |  |
| --- | --- | --- |
| 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 Map의 구현 목표**

* 사용자의 터미널 크기만큼 Map 크기를 구성하도록 한다.
* Height와 Width를 이용해 외벽에 White 색상으로 벽을 세운다.
* Mission을 Success하면 Map의 크기가 작아지게 한다.
* 단계마다 내부의 Wall을 다르게 배치해 Map의 모습을 달리 해준다.

**2단계 Snake 표면 및 조작**

* Snake의 진행방향은 Head쪽이다.
* Tail 방향으로 이동할 수 없다.
* Tail 방향으로 이동시 실패 (Game Over)
* Snake는 자신의 Body(몸)을 통과할 수 없다.
* Snake는 벽(Wall)을 통과할 수 없다.
* 몸의 길이가 3보다 작아지면 Game Over이다.
* Map을 다양한 구조로 구현함에 따라, Wall에 Snake의 Head가 닿으면   
  충돌하여 게임을 중지하도록 한다.

**3단계 Item 요소의 구현**

* GrowthItem을 +로 표시하여, Snake의 Head가 GrowItem과 (x, y) 좌표가 같으면   
  몸의 길이가 1 증가한다.
* PoisonItem을 -로 표시하여, Snake의 Head가 PoisonItem과 (x, y) 좌표가 같으면   
  몸의 길이가 1 감소한다.
* Snake Body가 있지 않은 임의의 위치에 출현시키도록 한다.
* 출현 후 일정시간이 지나면 사라지고 다른 위치에 나타나도록 한다.
* 동시에 출현할 수 있는 Item의 수는 3개로 제한한다.
* 추가적으로SpeedItem을 구현한다. \*로 표시하여 Snake의 Head가 PositionSpeed와   
  (x, y) 좌표가 같으면 Snake의 속도가 증가한다.

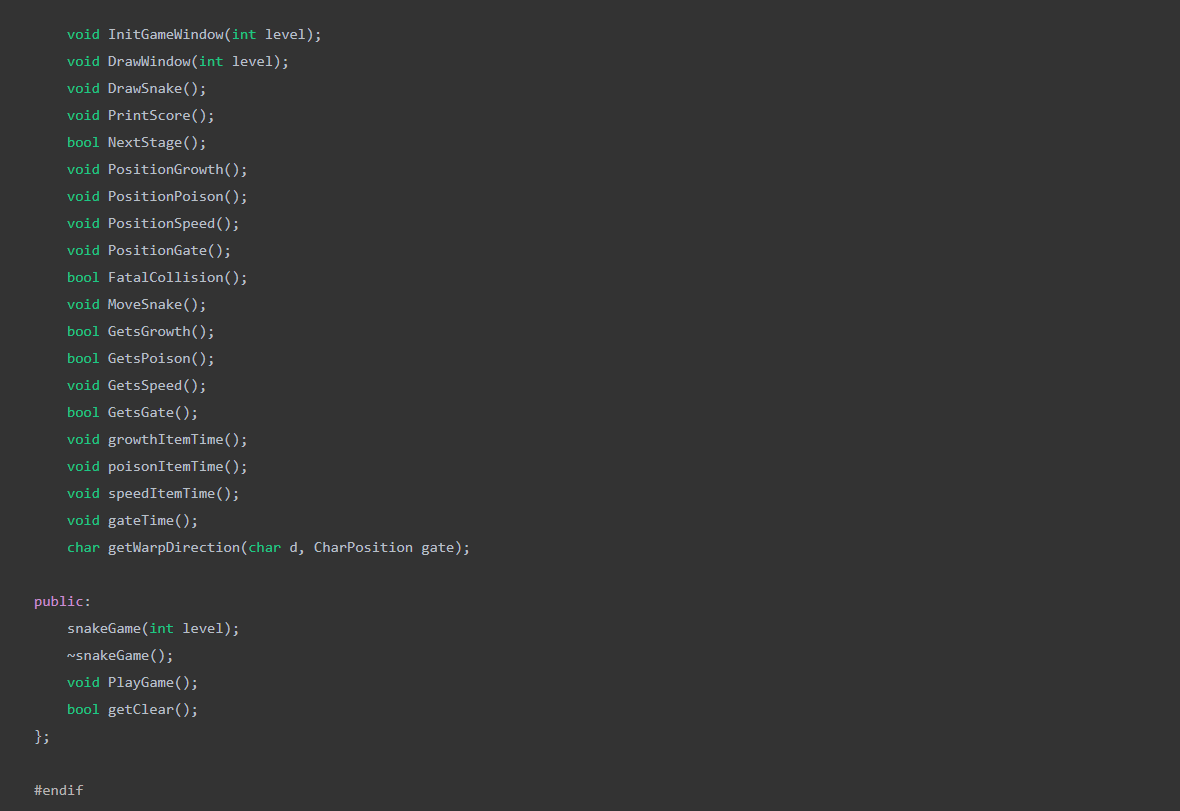
**4 단계 Gate 요소의 구현**

* 임의의 위치(Wall)에 한 쌍의 Gate가 출현하며 Snake의 Head가 Gate에 닿으면   
  다른 한쪽의 Gate로 이동한다.
* Gate의 진입하는 Snake의 진행방향에 따라서 다른 한쪽의 Gate에서 나오는  
  Snake의 진행방향을 설정한다
* 이때, 진행방향의 우선순위를 시계방향의 순으로 설정해준다.

**5단계 점수 요소의 구현**

* 프로그램 실행 후 우측에 ScoreBoard와 Mission을 나타내는 화면을 구성하도록 한다.
* Stage안의 게임 요소를 획득하게 되어 해당 요소의 충족요건을 달성하게 되면   
  점수판에 체크 표시(v)를 한다.
* 모든 미션을 성공하면 “Mission Complete”라는 화면을 띄우고 다음 Stage로   
  이동할 수 있도록 한다.

**위의 목표를 기준으로 필요한 변수와 함수들을 헤더파일을 작성하였다.**



## 개발 내용 및 결과물

### 개발 내용

**1단계 Map의 구현 목표**

* Map을 만들기 위해서 총 3가지의 파트로 구분했습니다.
* 첫번째로는 init함수를 생성해서 맵의 크기를 설정해주었습니다.   
  이때 맵의 크기는 터미널의 크기에 따라 결정하기 위해 ncurses의 내장 함수인 initsrc()를 사용했습니다.
* 또한 게임내에서는 마우스 커서가 존재하지 않기 때문에 ncurses의   
  내장함수인 curs\_set값을 0으로 주어 Map위에서 커서를 나타내지 않았습니다.
* 두번째로는 Snake의 머리가 닿으면 게임에서 패배하게 되며 맵의 큰 틀이 되는   
  외벽을 생성하였습니다.
* 두번째로는 Snake의 머리가 닿으면 게임에서 패배하게 되며 맵의 큰 틀이 되는   
  외벽을 생성하였습니다.
* 가장자리 외벽을 생성할 때 move로 좌표를 이동시키기 위해 maxwidth 와 maxheight를 사용했습니다.   
  이유로는 벽을 그리기 위해서는 charposition에 해당하는 x,y값 즉 좌표 값이 필요하기 때문입니다.
* 또한 Addch()를 함수를 사용했습니다.   
  추후에 벽을 사용하기위해 wall이라는 동적 배열을 사용하여 좌표 값에   
  해당하는 부분을 하나씩 추가해주었습니다.
* 세번째로는 stage마다의 다양한 Map을 구현했습니다.   
  저희는 stage마다 level이라는 변수를 switch문에 사용해서 level별maxheight, maxwidth를 줄여나가는 방식과 다양한 중간 벽을 만들어 다채로운 stage를 구현했습니다.
* Stage 2와 Stage 4에서 k,q변수를 사용해서 내부 Wall 효율적으로 표시해 중간벽을   
  보다 쉽게 생성했습니다.

**2단계 Snake 표면 및 조작**

* 기본적으로 Snake의 진행방향은 head로 설정하였으며 moveSnake()라는 함수를 만들어 계속해서 사용자의 키 값을 받아들이는 getch()함수를 사용해 switch문으로 head방향과 반대되는 방향(진행방향이 “동”이였으면 “서”쪽으로 누를 경우 “남”이었으면 “북”방향으로 누르는 방향 등등)이 들어오면 게임에서 패배하게끔 구현했고 Backspace가   
  들어올 경우 게임이 종료하게끔 구현했습니다.
* 게임의 패배조건 중 하나로 Snake과 벽이 충돌할 경우는 Snake의 머리의 좌표 값이 벽의 좌표 값과 일치할 경우로 조건을 세웠으며 이는 FatalCollision()의 함수를 통해 실행되게 만들었습니다.
* 두번째로 Snake의 머리가 Snake의 몸에 닿을 경우 또한 좌표 값을 사용해서   
  일치할 경우 패배하도록 만들었습니다.
* 세번째로 Snake의 길이가 3보다 작을 경우 역시 if문으로 처리해 게임에서 패배하게끔 구현했습니다.
* 네번째로 Snake의 Head가 Gate의 좌표 값과 일치할 경우 Gate에 들어오는 Snake의 진입방향에 따라 다른 한쪽의 Gate에서 나오는 좌표를 지정하였습니다.

**3단계 Item 요소의 구현**

* 저희는 총 3가지의 item을 구현했습니다.  
  먼저 Snake의 길이가 성장하는 아이템 “+” : growthItemchar  
  Snake의 길이가 감소되는 아이템 “-” : poisonItemchar  
  Snake의 속도가 증진되는 아이템 “\*” : speedItemchar
* 각각의 item은 position의 함수를 가졌으며 random(좌표 값)으로 생성하게 만들고  
  함수가 독립적이기때문에 아이템은 각각의 시간을 가졌습니다. (위치 또한 독립적)  
  따라서 동시에 바뀌지 않고 아이템 각자마다의 시간이 지나면 위치가 벽을 제외하고 랜덤으로 바뀝니다.
* 또한 생성된 위치가 Snake의 몸체 부분이거나 벽일 경우 다시 생성하게끔 구현하였으며  
  while문을 이용해 구현했습니다.

**4 단계 Gate 요소의 구현**

* gate는 벽 위에 생성되는 워프 장치라고 보시면 됩니다.  
  이를 위해 저희는 2개의 변수 gate\_idx1, gate\_idx2를 선언하고 Wall Vector로부터   
  중복되지 않게 인덱스를 무작위로 받아와서 외벽과 중간벽 중 랜덤하게 생성되게 하였습니다.
* 벽과 다른 색으로 나타내서 구별할 수 있도록 init\_pair, attron함수를 사용했습니다.
* Gate 또한 3단계에서 구현한 Item과 동일하게 시간이 지나면 다른 곳에 생기도록  
  gateTimer라는 변수를 이용하였습니다.
* Gate에 들어오는 Snake의 진입방향을 따져서 들어오는 방향의 매개변수와 다른 한쪽의 Gate의 매개변수를 받아서 해당 Gate의 동서남북을 방향을 나타내는 Charpoistion객체를 나타내주고 각각의 Block들이 Wall인지 아닌지 판단하는 Bool 변수들을 선언해주었습니다.
* 그 이후에 조건문들을 이용해 해당 Bool 변수의 알맞은 값을 할당하고 아까 받아온 Snake의 진입방향에 따라서 우선순위를 시계방향으로 한 방향을 return 해주었습니다.

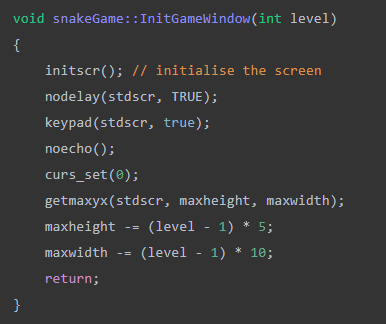
**5단계 점수 요소의 구현**

* 점수요소를 저희는 점수판과 미션판으로 구현하였습니다.
* 점수판과 미션판의 좌표의 경우 게임이 진행되는 window 오른쪽에 형성되고 출력하게끔  
  move함수와 printw 함수를 사용했습니다.
* 점수판의 경우 게임을 진행 하는 도중 획득한 아이템의 개수, 게이트의 통과 횟수 등을 보여주기 위해 변수 scoreGrowthItem, scorePoisonItem, scoreSpeedItem ,scoreGate를 선언하였으며 획득할 때마다 +=1 연산을 처리했습니다.
* 또한 Snake의 현재 길이 상태를 보여주기 위해 currentLength를 선언해 역시  
  실시간으로 입력을 받아왔습니다.
* 미션판의 경우 미션에 부합하는 item의 개수와 게이트 통과횟수를 생성하기 위해서  
  required로 구성된 변수를 생성한 후 각각의 item을 먹을 경우  
  required item의 개수를 줄여서 미션 현황 목록을 갱신시켜 주었습니다.
* 또한 미션을 충족했을 경우를 조건문을 이용해서 미션판에 ‘v’표시로 바꾸어 표시했습니다.

### 시스템 구조 및 설계도

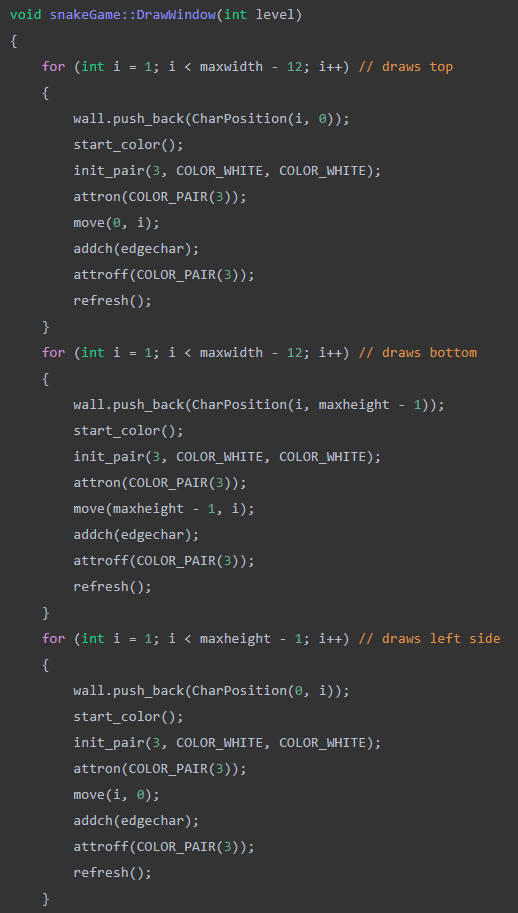
**1단계 Map의 구현 목표**

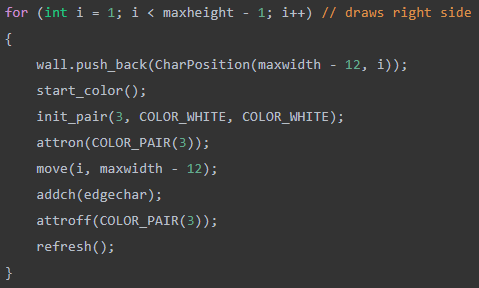
**1.1. 사용자의 터미널 크기만큼 Map 크기를 구성하도록 한다.**



* curses모드를 시작하여 단말기와 키패드를 허용하도록 설정한다.   
  그리고 사용자에 의해 타이핑 입력을 echo하기 위해 noecho()를 지정한다.   
  게임할 때 불편한 커서를 보이지 않기 위해 curs\_set(0)을 선언해주었다.

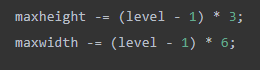
**1.2. maxHeight와 maxWidth를 받아와 외벽에 white 색상으로 벽을 세운다**.



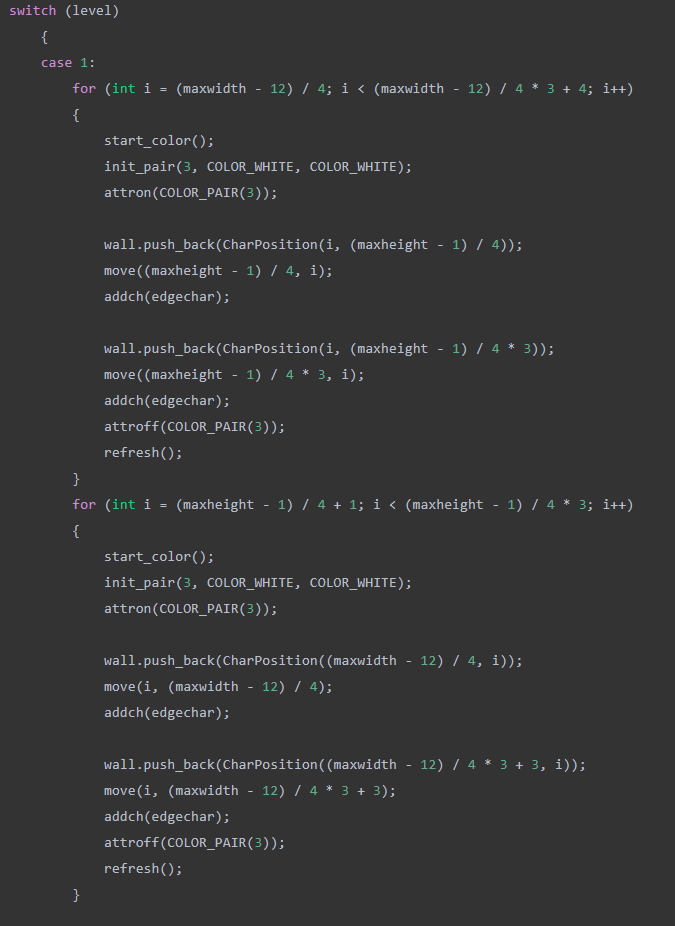


* 실행화면 크기 maxheight, maxwidth에 따라 가장자리에 벽을 생성해준다.
* 하지만 전체화면을 벽을 생성하면 ScoreBoard와 Misson을 보여줄 화면이 없어   
  기존의 maxwidth값에서 -12만큼의 값을 빼주도록 했다.
* 벽은 하얀색으로 나타내도록 하고 벽을 따라 나타내도록 한다.

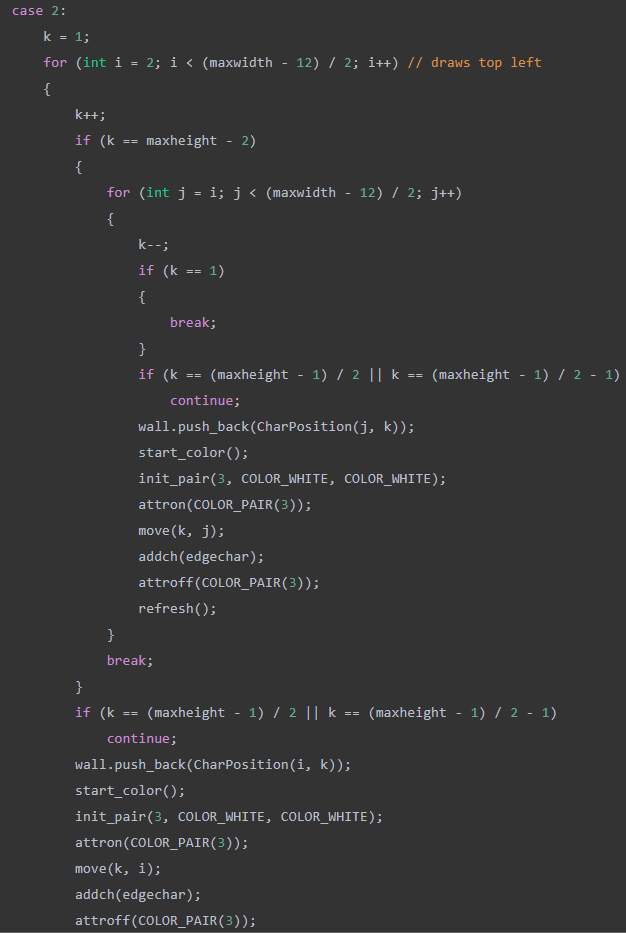
**1.3. 미션성공을 하고 다음 Stage에 넘어갈 때 maxHeight와 maxWidth의   
크기를 줄여준다**.

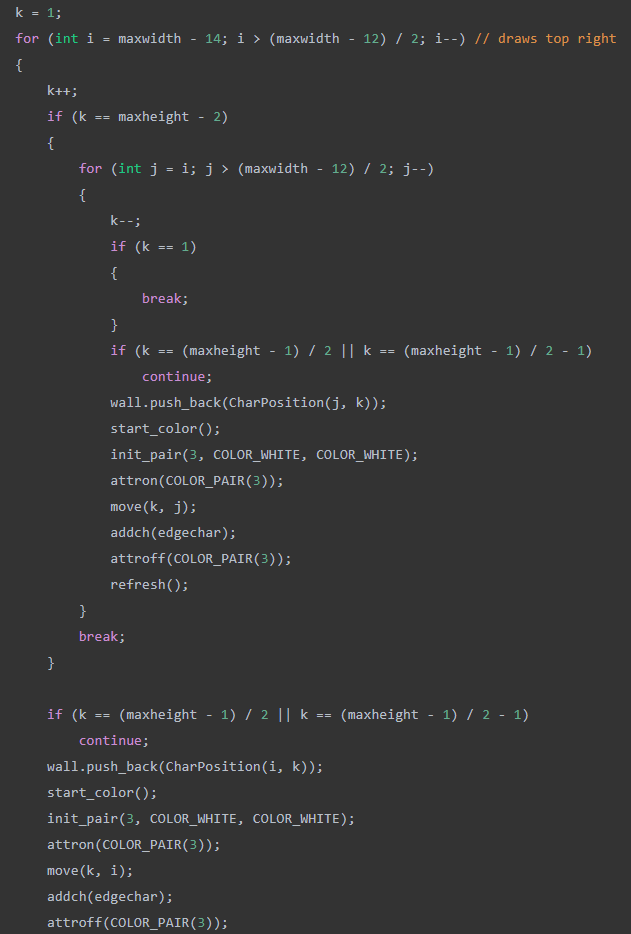
* 
* Mission을 Success하면 level이 증가하게 되는데 그에 따라 기존의Map의   
  크기에서 maxheight과 maxwidth를 작아지게 한다.

**1.4. 추가적으로 구현한 부분 (Stage별로 맵 구현)**

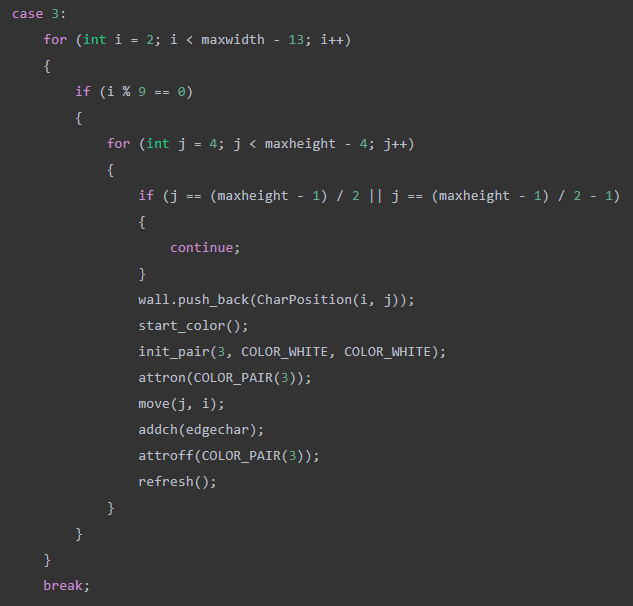


* Stage 1의 Map에서의 내부 Wall은 가운데에 큰 직사각형이 있는 구조로 구현해준다.

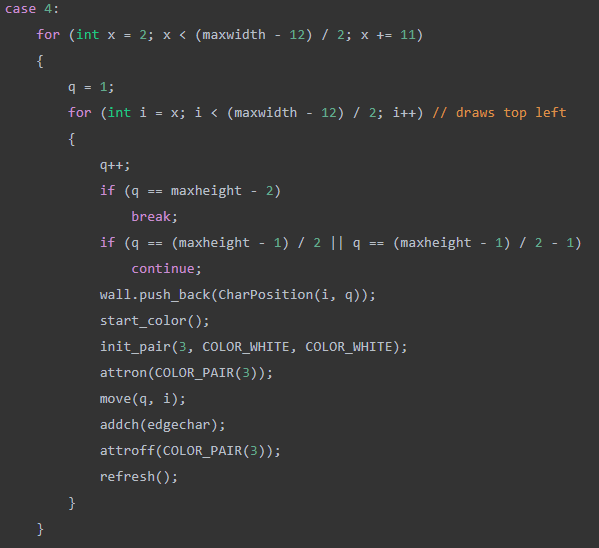


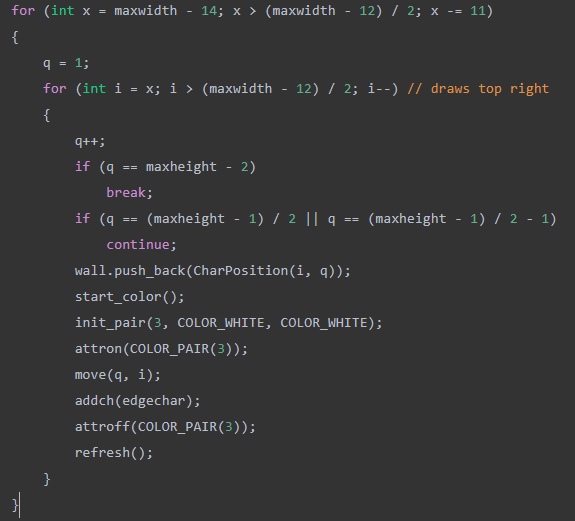


* Stage 2의 Map에서의 내부 Wall은 알파벳 ‘W’나 지그재그를 표현한 듯한 구조로 구현해준다



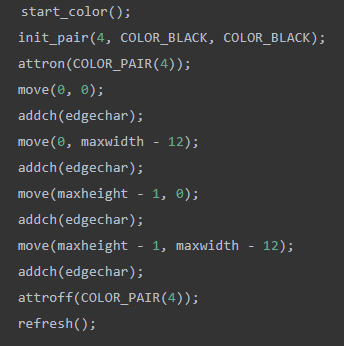
* Stage 3의 Map에서의 내부 Wall은 세로로 된 줄무늬를 표현한 듯한 구조로 구현해준다.





* Stage 4의 Map에서의 내부 Wall은 Stage 2의 Map 구조를 반영해 더욱 복잡하고 어려운

구조로 구현해준다.



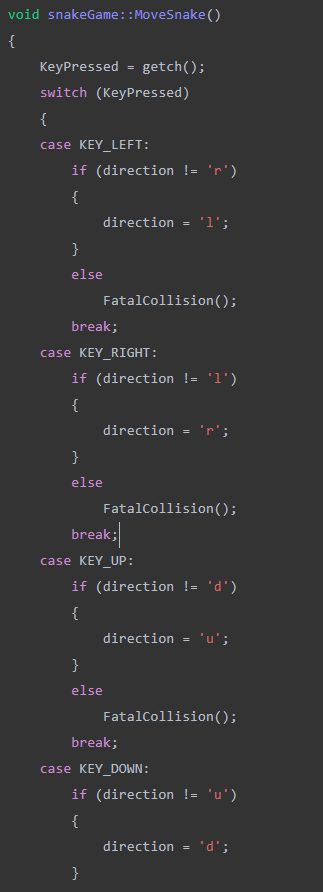
* 가장자리 벽에서 모서리에 위치한 Immune Wall을 생성해준다.
* 기존 Wall과 차이를 두기 위해 색깔을 검정색으로 설정해준다.

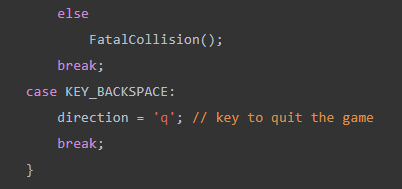
**1.5. 기타 사항**

* 후에 Wall들을 이용한 각종 기능 구현과 동작을 위해 Wall을 생성함과 동시에   
  미리 선언해둔 vector 객체의 해당 Wall의 좌표를 담은 객체를 저장한다**.**

**2단계 Snake 표면 및 조작**

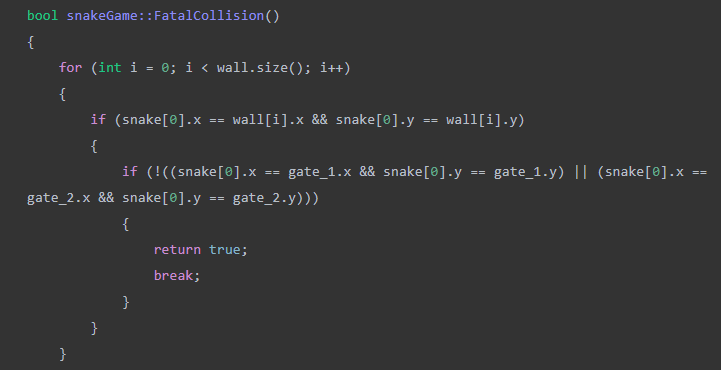
**2.1. Snake의 진행방향은 Head쪽이다.**



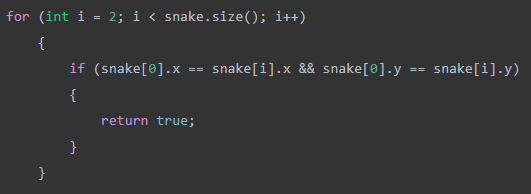


* 다음은 Snake의 Move에 대한 코드이다. 방향키에 입력에 대해 Snake의 방향을 움직인다.  
  BACKSPACE키를 누르게 되면 게임이 종료되게 한다.

**2.2. Snake는 벽(Wall)을 통과할 수 없고, 자신의 몸을 통과할 수 없다.**

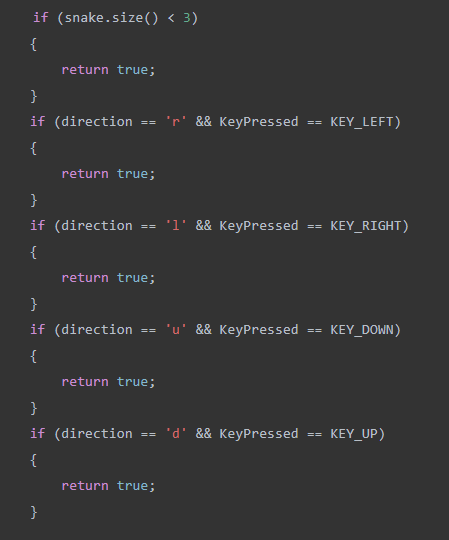


* 다음의 코드는 Snake의 Head가 되는 (x, y) 좌표가 벽과 일치하는 경우이다.   
  그럴 때 충돌을 일으키기 때문에 게임을 종료하도록 한다.
* 반면 Gate를 만났을 때는 충돌하는 것이 아니므로 해당 사항에 넣지 않는다.



* Snake의 Head가 자신의 Body 닿게 되면 게임을 중단하도록 한다.

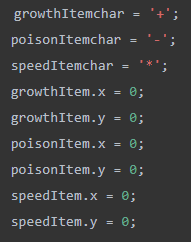
**2.3. Snake의 길이가 3보다 짧아진 경우, 또는 진행방향의 반대키를 누른 경우   
종료하게 된다.**



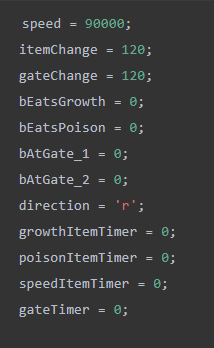
* 다음의 코드는 Snake의 길이가 3보다 짧아진 경우와 Snake의 진행방향의 반대키를   
  누른 경우이다.
* Direction은 현재 Snake의 진행방향을 나타내고 있고 KeyPressed는 그 다음 누른 키를   
  나타내고 있다. 만약 서로가 반대이면 true를 return하여 게임을 종료하도록 한다.

**3단계 Item 요소의 구현**

**3.1. Item을 표현하기 위한 변수들 초기화**



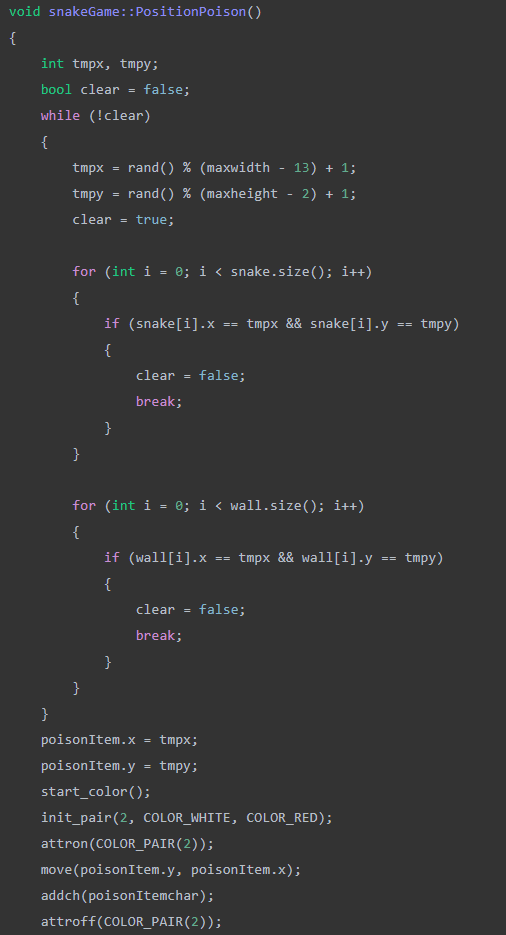
* 위 코드는 Item들을 화면에 표시하기 위함, 그리고 위치를 초기화하는 코드입니다.



* Snake Head가 Item과 Gate에 닿을 경우에 반환할 bool타입의 변수를 0으로   
  초기화해준다.
* Item와 Gate가 생성된 후에 어느 정도 시간이 지났는지 check를 위한 각각의 변수,   
  Snake의 speed를 초기화 하는 변수들을 구현해준다.
* 또한 Item과 Gate에 대한 지속 시간을 나타내주는 itemChange, gateChange 변수도  
  초기화해준다.

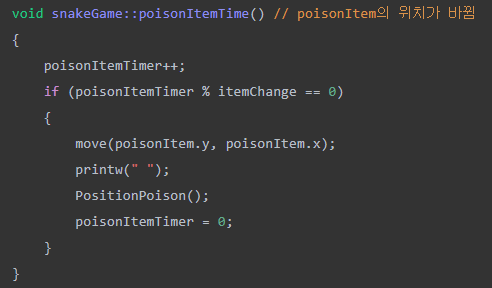
**3.2. Poisen Item을 화면 표현 및 구성**

**3.2.1. Poison Item 위치 선정**



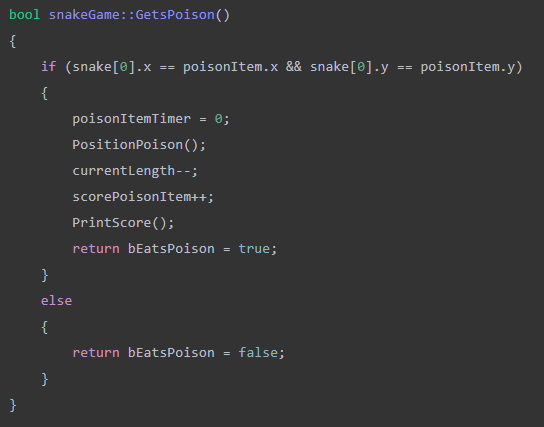
* 위 코드는 Poison Item의 위치를 나타낸다.   
  그리고 그 위치가 벽이나 Snake의 몸통에 해당되면 다시 위치를 탐색하도록 한다.

**3.2.2. Poison Item 시간 설정**

****

* 위 코드는 Posion Item의 시간 설정을 하는 메소드이다.
* 일정 시간이 경과되면 다시 Poison Item의 위치를 변경시키도록 한다.

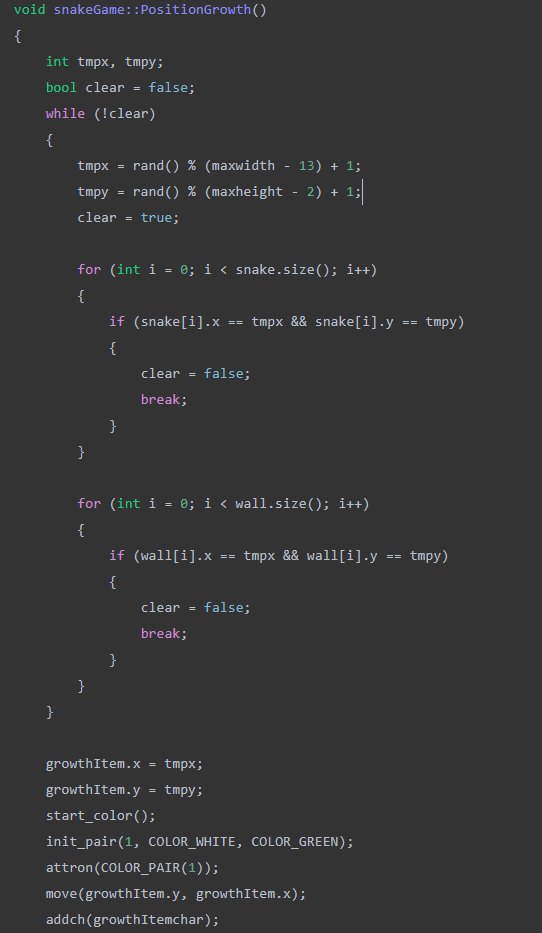
**3.2.3. Get Poison Item**



* 위의 코드는 Snake의 Head가 Poison Item의 위치와 같을 때, 다시 Poison의   
  위치를 설정해주고 Misson에서 요구하는 Poison Item의 개수에 1을 더해준 다음   
  몸의 길이가 -1 감소하도록 구현해준다.

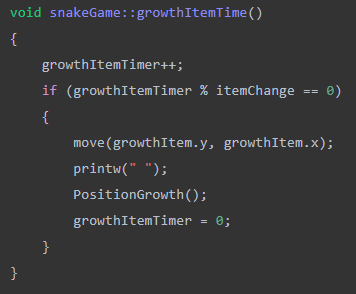
**3.3. Growth Item을 화면 표현 및 구성**

**3.3.1. Growth Item 위치 선정**

****

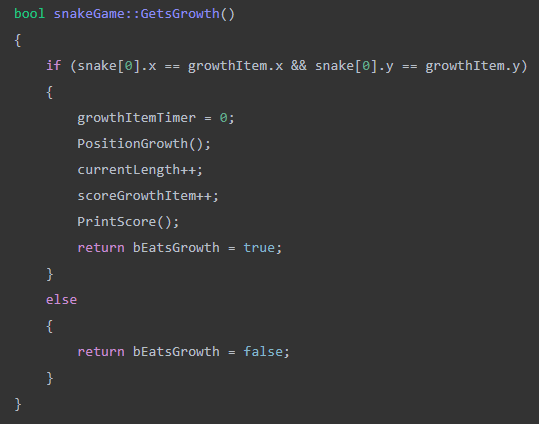
* 위 코드는 Growth Item의 위치를 나타낸다.   
  그리고 그 위치가 벽이나 Snake의 몸통에 해당되면 다시 위치를 탐색하도록 한다.

**3.3.2. Growth Item 시간 설정**

****

* 위 코드는 Growth Item의 시간 설정을 하는 메소드이다.
* 일정 시간의 경과되면 다시 Growth Item의 위치를 변경시키도록 한다.

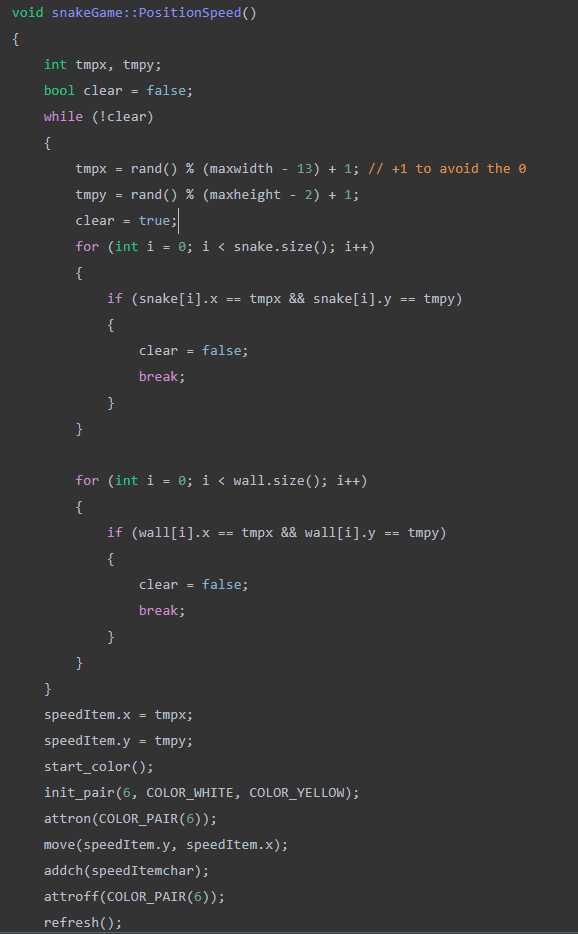
**3.3.3. Get Growth Item**

****

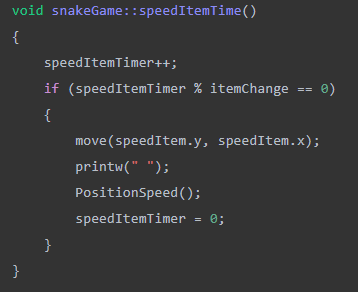
* 위의 코드는 Snake의 Head가 Growth Item의 위치와 같을 때 다시 Poison의 위치를   
  설정해주고 Misson에서 요구하는 Poison Item의 개수에 1을 더해준 다음   
  몸의 길이가 +1 증가하도록 구현해준다.

**3.4. Speed Item을 화면 표현 및 구성**

**3.4.1. Speed Item 위치 선정**

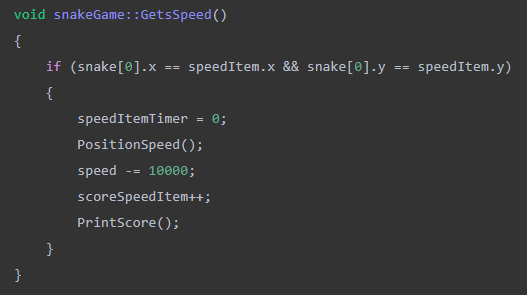
* ****
* 위 코드는 Speed Item의 위치를 나타낸다.   
  그리고 그 위치가 벽이나 Snake의 몸통에 해당되면 다시 위치를 탐색하도록 한다.

**3.4.2. Speed Item 시간 설정**



* 위 코드는 Speed Item의 시간 설정을 하는 메소드이다.
* 일정 시간이 경과되면 다시 Speed Item의 위치를 변경시키도록 한다.

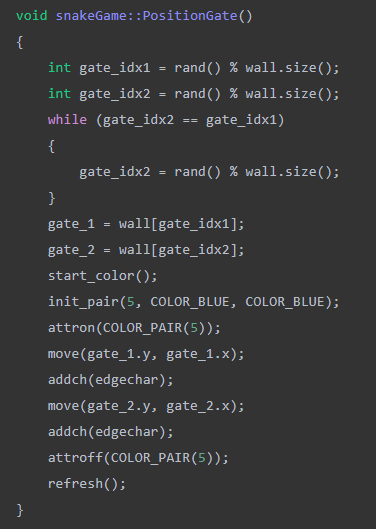
**3.4.3. Get Speed Item**

****

* 위의 코드는 Snake의 Head가 Speed Item의 위치와 같을 때 다시 Speed Item의 위치를   
  설정하고 Misson에서 요구하는 Speed Item의 개수에 1을 더해준 다음   
  Snake의 속도 빨라지도록 구현해준다.

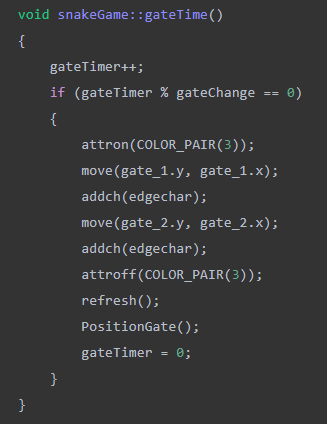
**4단계 Gate 요소 구현**

**4.1. Gate의 위치 구현**

****

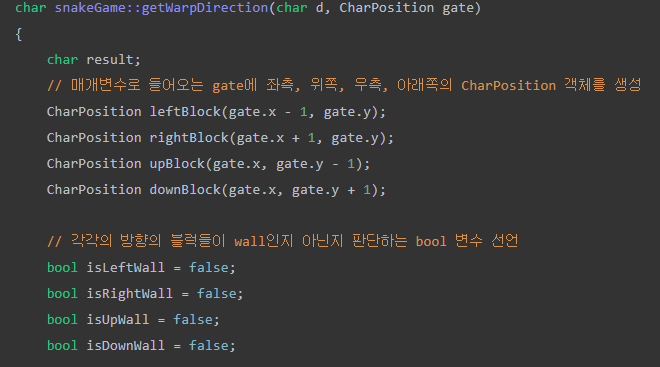
* gate\_idx 1, 2는 중복을 피하면서 Wall의 좌표를 저장하는 vector의 인덱스를 무작위로   
  받아와 gate\_1, \_2에 넣어주고, 해당 인덱스의 좌표에 색을 넣어주어 벽과 구별해준다.

**4.2. Gate 시간 설정**

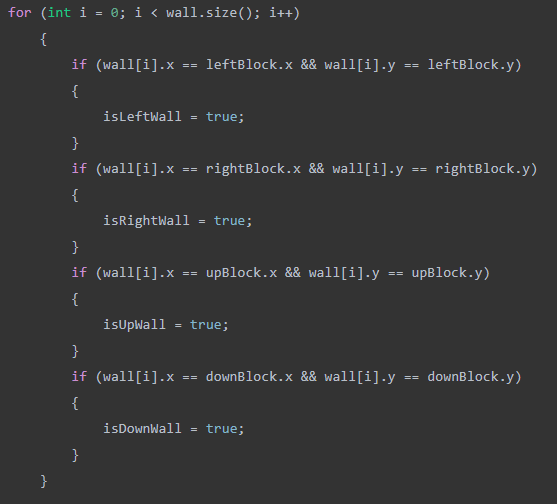


* gateTimer를 이용해 일정 시간이 지나면 원래 게이트는 다시 벽으로 변하고,   
  다른 새로운 게이트 1쌍이 생성되도록 구현해준다.
* 그 후 다시 타이머를 0으로 초기화 한다.

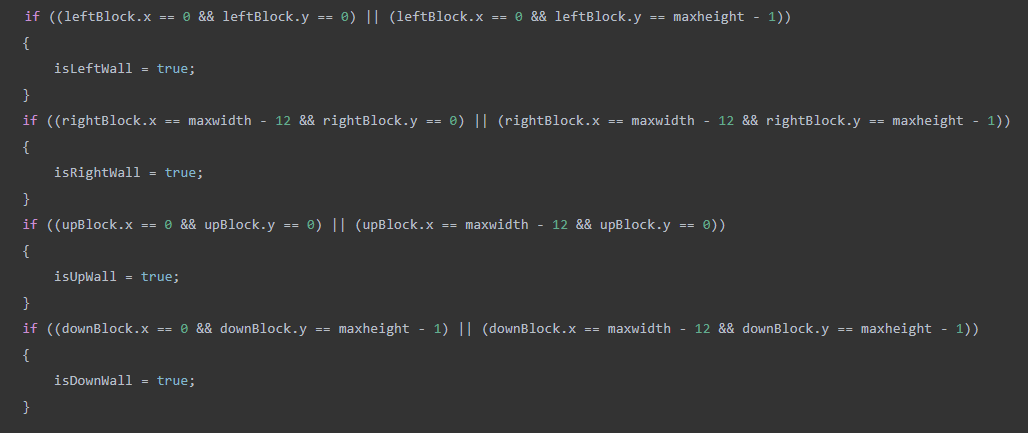
**4.3. Gate 방향 설정**

****

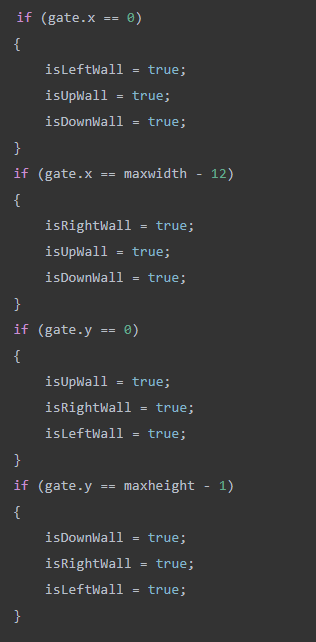
* Gate을 통과할 때 Snake의 진입 방향을 따지면서,   
  게이트 주변에 벽을 인식하기 위해 게이트의 위, 아래, 오른쪽, 왼쪽에 벽이 있는지   
  확인해야 하므로 각 위치의 좌표를 담은 객체와 각 위치가 Wall인지를 판단해주는   
  변수를 생성해준다.



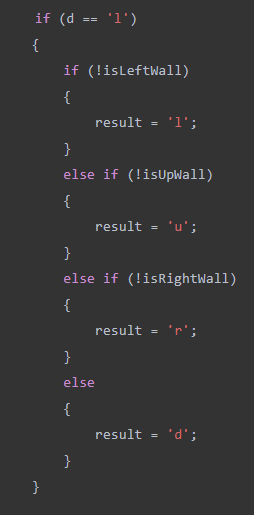
* 4방향을 나타내는 객체의 좌표가 map에 있는 Wall과 동일하다면   
  각각에 해당하는 bool타입의 변수를 true로 바꿔준다.

****

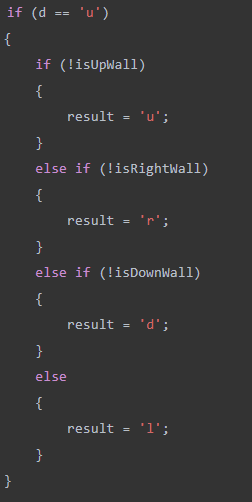
* Gate의 생성지점이 Immune Wall과 근접해 있을 때를 고려해 추가적으로  
  조건문을 작성해준다.
* 벽이 아닌 모서리(immune wall)의 왼쪽, 오른쪽, 위, 아래의 좌표가 Gate인 경우에도   
  각각에 해당하는 bool타입의 변수를 true로 바꿔준다.

****

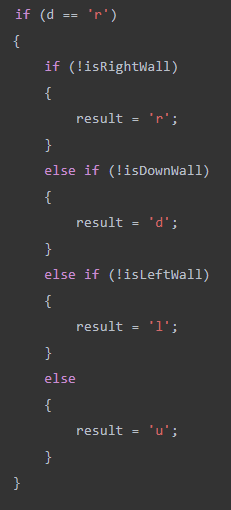
* 화면 가장자리에 Gate가 생성되는 것을 고려하여,  
  왼쪽, 오른쪽, 위, 아래의 좌표의 경우도 생각하여 각 좌표에 해당하는 bool타입의   
  변수를 true로 변경한다.



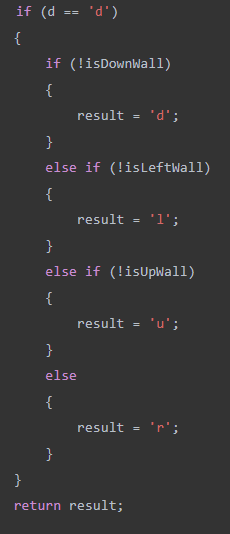
* Snake의 Gate 진입 방향이 왼쪽인 경우에,   
  다른 Gate에서의 나가는 방향을 해당 Gate주위의 벽의 유무를 판단하여   
  시계방향 순으로 결정한다.

****

* Snake의 Gate 진입 방향이 위쪽인 경우에,   
  다른 Gate에서의 나가는 방향을 해당 Gate주위의 벽의 유무를 판단하여   
  시계방향 순으로 결정한다.



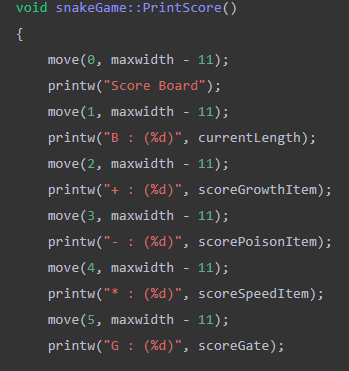
* Snake의 Gate 진입 방향이 오른쪽인 경우에,   
  다른 Gate에서의 나가는 방향을 해당 Gate주위의 벽의 유무를 판단하여   
  시계방향 순으로 결정한다.



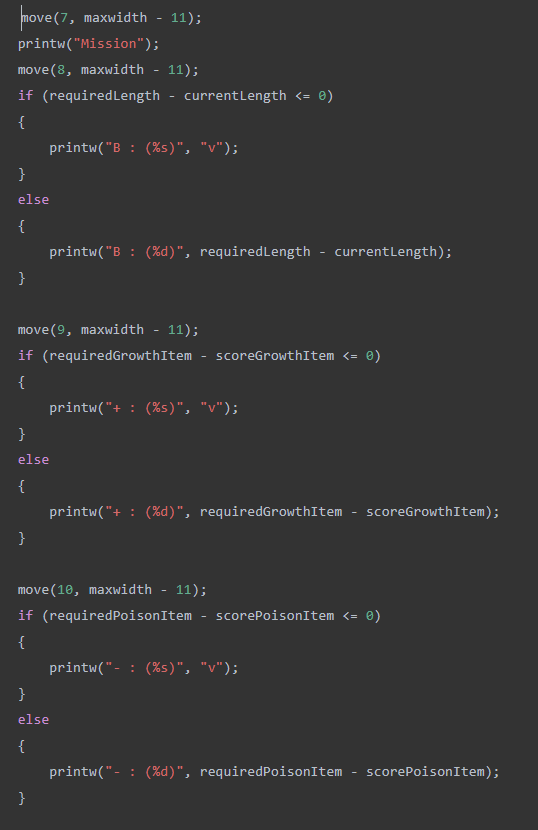
* Snake의 Gate 진입 방향이 아래쪽인 경우에,   
  다른 Gate에서의 나가는 방향을 해당 Gate주위의 벽의 유무를 판단하여   
  시계방향 순으로 결정한다.

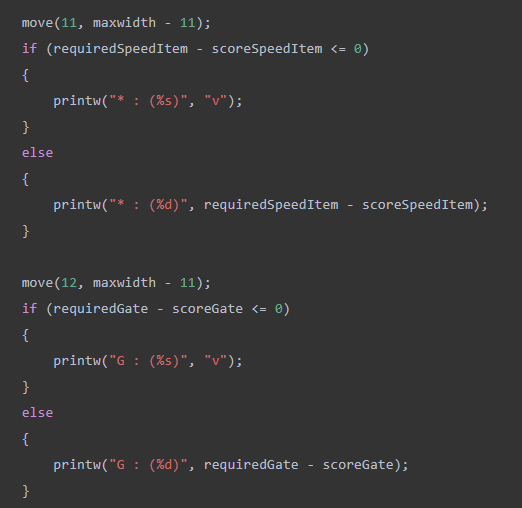
**5단계 점수 구현**

**5.1. 우측에 표시할 점수판과 Misson Board 구현**

****

* 게임 중 몸의 길이, 획득한 성장 아이템, 감소 아이템, 스피드 아이템 게이트 통과 횟수   
  등을 게임 화면 우측에 표시하기 위해,   
  실행 화면 크기의 maxwidth의 바로 옆에 게임 중 각각의 정보들을   
  실시간으로 입력을 받아와 출력한다.

****

****

* 점수를 표시해 주는 곳 아래에는 각 스테이지 별 미션을 표시해준다.
* 현재 길이나, 아이템 횟수에서 미션을 충족시킬 Snake의 길이나,   
  아이템 횟수까지 남은 수를 표시해준다.
* 미션을 충족한 항목은 ‘v’ 표시로 바꿔준다.

### 활용/개발된 기술

1. **SpeedUpItem 추가**

* 기존의 GrowthItem 및 PoisonItem을 활용하여 Snake의 길이를 조절해주는 Item 구현
* 이에 추가로 Snake의 이동 속도를 증가시키는 SpeedUpItem을 추가 구현
* PlayGame() Method에 모든 Method를 실행시키다 보니 Speed가 빨라지면   
  PlayGame() 메소드 또한 빨라져 모든 Method가 빨리 진행하게 됨.
* 오히려 이런 부분이 속도가 증가함에 따라 Play 난이도가 어렵게 구현됨.

1. **다양한 Stage Map 구현**

* 좀 더 Game을 지루하지 않게 만들기 위해서 다양한 Map을 구현하도록 했다.
* Stage가 진행될 수록 난이도 증가에 대한 부분을 Map을 더 복잡하고   
  어렵게 만들어줌으로써 해결했다.
* 또한 Gate의 통과 부분을 더욱 흥미롭게 만들기 위해 이색적인 내부 Wall들을 구현했다.

### 현실적 제한 요소 및 그 해결 방안

**1단계**

* 초반부터 Map의 구현에 어려움이 있어 Google의 자료를 통해 Ncurses 중에 화면   
  입력 및 출력의 지식을 얻어 구현을 하였음

**2단계**

* Snake에 지켜야 할 부분이 많아서 움직임을 많이 검사하다 보니 어려웠다.

**3단계**

* 없음

**4단계**

* Gate의 동서방향의 Block이 벽인지 아닌지 또는 절대벽에 해당하는 지 등등 고려할 부분이 많았다. 그래서 구현은 했는데, 코드가 반복된 부분에서 해결해 볼 수 있었을 것 같았는데 많이 아쉽다.

**5단계**

* 없음

### 결과물 목록

**Main.cpp**

* snakeGame.cpp에 정의된 메소드를 실행하기 위한 파일

**snakeGame.cpp**

* snakeGame에 필요한 메소드를 구현한 파일

**snakeGame.h**

* snakeGame.cpp에 사용될 변수들과 method 및 필요한 자료를 담는 좌표 구조체를 선언한 파일.

**Makefile**

* SnakeGame을 컴파일하고 실행하기 위해 생성한 파일
* 동영상 제출본 : <https://www.youtube.com/watch?v=pW56Sfm-k4k&t=129s>

# 자기평가

|  |  |
| --- | --- |
| **팀** | **평가** |
| **김학균** | 안녕하십니까 이번 Snake-Game 프로젝트의 팀장을 맡은 김학균입니다.  저는 이 프로젝트를 해결하기 위해서 Ncurses의 어떠한 인터페이스들이  있는지 확인하고 팀원들에게 전달해주었습니다.  이를 토대로 Map의 기본 틀 구현 및 Stage마다 Map의 크기 감소,  Snake의 Collison 로직, Gate의 위치 구현을 맡아서 구현했습니다.  하지만 초반에 어떻게 해야 할지 몰랐던 팀원들과의 역할을 좀 더  정확하게 구분하기 위해 Github를 통해 branch마다 자기들이 맡은 부분을 Commit하도록 하였고, 이를 통해 좀 더 구조적으로 효율있는  프로젝트 구조를 구성할 수 있었습니다.  또한 코로나 사태로 팀원들과 함께 모이지 못해 매주 금요일날 정기적으로 Discord를 통해 자신들이 구현했던 Part를 Live를 통해 소통하였습니다.  아쉬웠던 부분은 기본적인 Ncurses 라이브러리를 파악하고 공부하는데  시간이 많이 들였던 점입니다.  제 스스로 프로젝트에 시간을 투자했더라면 Game을 더 재밌게  구성할 수 있었을 것 같습니다. |
| **김민준** | 이번 C++ project Snake Game에서 제가 맡은 부분은 item의 생성되는  부분과 stage마다 맵 구현이었습니다. 대학에 입학하고 나서의  첫 프로젝트였기에 다소 낯선 탓인지 몰라도 팀원들에게 큰 도움을 주지 못하는 느낌이 있었습니다.  그래서인지 몰라도 더욱 열심히 노력해야 하겠다 피해를 끼치면 안 되겠다는 생각을 하고 있었고 다행히 팀원들이 잘 이끌어주어 주었고 쌍방향적인 소통(디스코드 간의 코드리뷰, 카카오톡과 같은 채팅방에서의 다양한 의견 공유, 대면으로 만나 협업 코딩)이 있었기에 프로젝트를 성공적으로  마무리할 수 있었습니다. |
| **노성환** | 제가 이번 프로젝트에서 수행한 일은 아이템들이 올바르게 생성될 수 있도록 구현하고, Snake이 해당 아이템을 획득하였을 때의 움직임을 구현하는데 이바지 하였습니다. 이번 팀 프로젝트가 난이도가 있었던 만큼 팀원들간 협력과 소통이 중요하다는 것을 알게 되었고, 서로 구현한 것을 리뷰해주며 처음 접한 ncurses와 c++공부에도 도움이 되어서 보람찬 프로젝트였습니다. |
| **문성찬** | 이번 프로젝트에서의 저의 역할은 Snake Game의 필수 코드의 구현을  맡았으며 그중에서도 필수 적용 단계인 Map구현, Item구현, Gate와 워프를  주로 구현했습니다.  프로젝트를 수행하면서 일단 초반 팀원들과 기본적인 코드의 틀을  중요하게 생각하여 해당 틀을 구상하고 잡는데 많은 시간과  노력을 들였습니다.  코드의 틀을 팀원들과 함께 구상함으로써 멤버 변수와 함수의 이름 형식을  통일할 수 있어서 많은 도움이 되었고, 각자의 역할을 분배해 협업하면서  코드를 구현해 생각보다 이른 시간에 프로젝트를 완성할 수 있었습니다.  또한 GITHUB를 이용해 팀원들의 코드들을 코드 리뷰하고 코드를 분할  작업함으로써 유지보수와 구현의 효율성을 높일 수 있었습니다.  다만, 처음 접하는 ncurses 라이브러리를 이용하면서 초반엔 해당 코드들을  이해하고 구현하는데 다소 어려움이 따랐습니다. |

# 참고 문헌

**참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | Webpage | Interface to the ncurses Library | <http://www.math.rwth-aachen.de/~Browse/doc/chap2.html> |  |  |  |
| 2 | Webpage | Ncurses Programming Guide | <http://www.cs.ukzn.ac.za/~hughm/os/notes/ncurses.html> |  |  |  |

# 부록

## 사용자 메뉴얼

1. zip파일을 Download 받는다.
2. Terminal에 make run을 입력한다.
3. 게임화면을 나오면 y를 눌러 시작한다.
4. 게임을 진행하면서 4개의 Stage의 Mission을 Clear하도록 한다.
5. (유의) 충돌하거나 Snake의 길이가 3미만이면 게임이 종료된다!
6. Stage 4까지 모든 Mission Clear 한다면 성공

## 설치 방법

1. zip파일을 Download 받는다.
2. 압축을 풀어 원하는 위치에 저장한다.