**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Snake Game* |
| 팀 명 | *1-3조* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 24 |

|  |  |
| --- | --- |
| **팀원** | 양진우(팀장) |
| 권순호 |
| 박승현 |
|  |
|  |

|  |
| --- |
| **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 “1-3”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “1-3”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-Snake Game.doc |
| **원안작성자** | 양진우, 권순호, 박승현 |
| **수정작업자** | 양진우, 권순호, 박승현 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2020-06-25 | 양진우 | 1.0 | 최초 작성 |  |
| 2009-65-26 | 권순호 | 1.1 | 내용 수정 | 목표, 사용자 메뉴얼 추가 |
| 2009-06-27 | 박승현 | 1.2 | 내용 수정 | 개발 내용, 설치 방법 추가 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**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 7](#_Toc43103660)

[2.2.5 결과물 목록 8](#_Toc43103661)

[3 자기평가 9](#_Toc43103662)

[4 참고 문헌 9](#_Toc43103663)

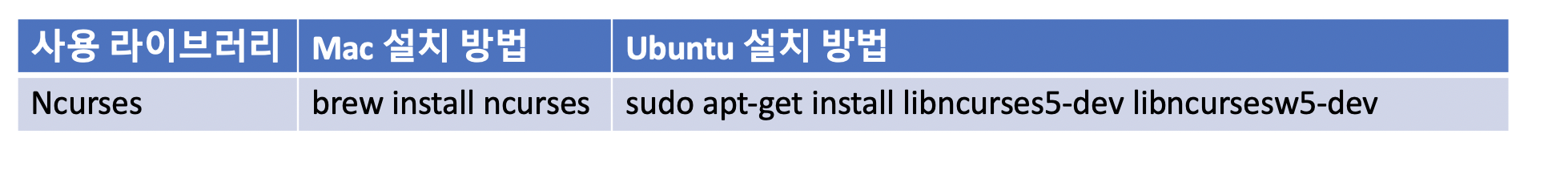
[5 부록 9](#_Toc43103664)

[5.1 사용자 매뉴얼 9](#_Toc43103665)

[5.2 설치 방법 10](#_Toc43103666)

# 개요

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

****

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

# 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 |

**프로젝트 목표 : 요구한 모든 단계 수행 완료**

**1단계 : Ncurses 라이브러리 함수들을 사용하여 2차원 배열로 된 Snake Map을 게임 화면으로 표시하는 프로그램을 완성한다.** ✓

**2단계 : 1단계의 맵 위에 Snake를 표시하고 화살표를 입력 받아 Snake가 움직이도록 프로그램을 완성한다**✓

**3단계 : 2단계 프로그램에서 Map위에 Growth Item와 Poison Item을 출현하도록 수정한다.** ✓

**4단계 : 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과 할 수 있도록 수정한다.** ✓

**5단계 : 4단계 프로그램에서, 우측에 게임 점수와 게임 상황을 표시하는 화면을 구성한다.** ✓

## 개발 내용 및 결과물

### 개발 내용

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

**개발에 앞서 각 단계별로 중간점검 때 간단하게 구상한 것을 바탕으로 코드를 짰습니다.**

**일차적으로 스네이크 게임의 5단계를 모두 구현하는 것을 목표로 삼았습니다.**

1. **초기 설정과 맵 화면에 표시**

**2차원 배열 형태의 맵을 화면에 표시하고 색을 입힙니다. 프로젝트에서 사용할 요소들을 enum 선언해주고 색을 입히는 함수를 만들어 사용합니다.**

**`**

**2. Snake의 생성과 이동**

**초기 설정에서는 배열을 만드는 방법을 생각했습니다.**

**배열 2개를 만들어 하나는 현재 위치를 저장하고 다른 하나에 키보드 입력에 따른**

**다음 값을 저장하고 다시 처음 배열에 복사하는 빙법을 생각하였습니다.**

**실제 개발 단계에서는 class를 활용하여 한 배열 안에서 각각의 인덱스에 x, y값**

**각각 저장하고 바꾸어 주는 식으로 구현하였습니다.**

**이동할 때 이동 방향 4개를 구분하여 각각의 경우에 x, y값을 업데이트 하는 방식**

**으로 구현하였습니다. snake class에 선언한 snake의 길이를 가져와**

**for문을 사용해 v[i].x = v[i-1].x의 형식으로 0번 인덱스까지 앞에 있는 값을 바로**

**뒤 칸에 대입해주고 마지막에 가장 앞의 위치를 나타내는 인덱스 0번의 값을**

**방향에 따라 바꾸어 줍니다. 가령 이동방향이 오른쪽 일 때 v[0].x = v[1].x + 1;**

**v[0].y = v[1].y의 방식입니다. 이때 이동방향은 변수 선언을 통해 저장하고 키보드**

**입력에 따라 바꾸어 줍니다.**

**3. 아이템들의 출현**

**1. Growth, Poison를 랜덤함수를 활용하여 생성합니다. 단 이 둘의 위치가 snake나**

**게이트와 겹치면 다시 랜덤한 위치를 받아옵니다. 겹치는 지의 판별은 for문을**

**사용하여 snake혹은 gate의 모든 좌표와 검사합니다.**

**2. 플레이어가 아이템을 먹었을 경우 각각의 경우에 함수를 만들고 길이를 늘리거나**

**줄여줍니다.**

**4. 게이트 생성**

**아이템과 마찬가지로 랜덤한 값을 통해 게이트의 위치를 설정합니다.**

**이때 snake, 위의 두 아이템과의 겹치는지 여부를 판별하고 이번에는 또 wall안에 있는지 없는지 판단합니다. 게이트에 진입 할 때 게이트의 위치를 기준으로 나오는 방**

**향을 판단해줍니다.**

**5. 디스플레이 화면**

**화면에 표시할 정보들을 따로 저장할 score파일을 만들어 class를 선언해주고**

**이를 통해 상황에 따라 값들을 업데이트 해주면서 그 값들을 화면에 표시해줍니다.**

### 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 : Ncurses 라이브러리 함수들을 사용하여 2차원 배열로 된 Snake Map을 게임 화면으로 표시하는 프로그램을 완성한다. (개발자 : 양진우,권순호)**

1. **각각의 특징 (Gate, Poison, Growth, Immune\_Wall 등)에 따른 색들을 매칭해준다.**

enum BLOCKS{

NONE,

EMPTY,

WALL,

IMMUNE\_WALL,

HEAD,

BODY,

GATE,

GROWTH,

POISON,

};

1. **Ncurses의 init\_pair를 사용하여 각각의 색들을 입혀준다.**

void Snake::colorInit(){

start\_color();

init\_pair(EMPTY,COLOR\_BLACK,COLOR\_BLACK);

init\_pair(WALL, COLOR\_RED, COLOR\_RED);

init\_pair(IMMUNE\_WALL, COLOR\_CYAN, COLOR\_CYAN);

init\_pair(HEAD,COLOR\_BLUE,COLOR\_BLUE);

init\_pair(BODY,COLOR\_GREEN,COLOR\_GREEN);

init\_pair(GATE,COLOR\_YELLOW,COLOR\_YELLOW);

init\_pair(GROWTH,COLOR\_GREEN,COLOR\_GREEN);

init\_pair(POISON,COLOR\_MAGENTA,COLOR\_MAGENTA);

}

1. **이에 따라 색을 출력해준다.**

**2단계 : 1단계의 맵 위에 Snake를 표시하고 화살표를 입력 받아 Snake가 움직이도록 프로그램을 완성한다. (개발자 : 양진우)**

1. **Key를 입력받을 변수를 설정하고 계속 key를 입력받는다.**
2. **Timeout()함수를 이용하여 만약 0.2초 이상 key를 받지 않는다면 무시하고 계속 지속하도록 만든다.**
3. **화살표(w,a,s,d)에 따른 snake의 방향을 바꿔준다. 만약 그 이외의 key를 눌렀으면 무시한다.**

void Snake::play(){

timeout(200);

int key;

while((key=getch())!='p'){

timer();

if(key == 'w' || key == 'a' || key == 's' || key == 'd'){

moveSnake(key);

}

else

moveSnake();

}

}

1. **Wall, Immune Wall을 원하는 대로 자유자재로 구열하기 위해 2차원 배열을 만들고 언제든지 변경핧 수 있도록 하였다.**

**const int NCOL = 35, NLINE = 21; // map\_win의 가로, 세로**

**const int MAP[NLINE][NCOL] = {**

**{2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2}**

**};**

**3단계 : 2단계 프로그램에서 Map위에 Growth Item와 Poison Item을 출현하도록 수정한다. (개발자 : 양진우)**

1. **지역함수 x, y에 임의의 값을 Random함수를 이용해서 임의로 설정한다. 단, 벽과 닿아서는 안되므로 다음과 같은 식을 통해서 예외처리를 사전에 해준다.**

x = rand()%33 + 1;

y = rand()%19 + 1;

1. **만약 x,y값이 snake와 같거나, gate랑 서로 다른 아이템들의 위치 겹치면 다시 Random 함수에서 x,y값을 꺼내온다.**

void Snake::generatePoison(){

int x,y;

bool conti = true;

while(conti){

conti = false;

//gate의 경우 양 끝점에 나와도 되지만, item들과 같은 경우 경계선에 있으면 안된다.

x = rand()%33 + 1;

y = rand()%19 + 1;

//snake과 겹치는 지 확인

for(int i = 0; i < len; i++){

if(v[i].x == x && v[i].y == y){

conti = true;

break;

}

}

//growth 겹치는 지 확인

if(growth.x == x && growth.y == y)

conti = true;

//gate과 겹치는 지 확인

for(int i = 0; i < 2; i++){

if(gate[i].x == x && gate[i].y == y){

conti = true;

break;

}

}

}

poison.x = x; poison.y = y;

isPoison = true;

wattron(map\_win,COLOR\_PAIR(POISON));

mvwprintw(map\_win,y,x,"#");

wattroff(map\_win,COLOR\_PAIR(POISON));

wrefresh(map\_win);

}

void Snake::generateGrowth(){

int x,y;

bool conti = true;

growthTime = time(NULL);

mvwprintw(map\_win,growth.y,growth.x," ");

while(conti){

conti = false;

x = rand()%33 + 1;

y = rand()%19 + 1;

//snake과 겹치는 지 확인

for(int i = 0; i < len; i++){

if(v[i].x == x && v[i].y == y){

conti = true;

break;

}

}

//poison과 겹치는 지 확인

if(poison.x == x && poison.y == y)

conti = true;

//gate과 겹치는 지 확인

for(int i = 0; i < 2; i++){

if(gate[i].x == x && gate[i].y == y){

conti = true;

break;

}

}

}

growth.x = x; growth.y = y;

isGrowth = true;

wattron(map\_win,COLOR\_PAIR(GROWTH));

mvwprintw(map\_win,growth.y,growth.x,"#");

wattroff(map\_win,COLOR\_PAIR(GROWTH));

wrefresh(map\_win);

}

* 1. **만약 snake가 Poison아이템을 먹으면 가장 뒤에 있는 부분(꼬리 부분을) 줄인다. temp라는 동적배열을 임의로 만들고, 하나 줄어든 snake의 값을 저장 그리고 snake를 재할당한다.**

void Snake::eatPoison(){

cordinate\* temp = new cordinate[len];

for(int i = 0; i < len; i++){

temp[i].x = v[i].x; temp[i].y = v[i].y;

}

len--;

// if(len-1 < 3){

// exit(1);

// }

score.poison++;

mvwprintw(score\_win,2,6,"%d",len-1 );

mvwprintw(score\_win,6,6,"%d",score.poison );

if(score.poison >= score.poisonMi){

mvwprintw(Mission\_win,6,10,"V");

MiPoisonItem = true;

}

wrefresh(score\_win);

wrefresh(Mission\_win);

delete[] v;

v = new cordinate[len];

v[0].x = poison.x; v[0].y = poison.y;

for(int i = 1; i < len; i++){

v[i].x = temp[i-1].x; v[i].y = temp[i-1].y;

}

score.poison++;

isPoison = false;

}

* 1. **만약 snake가 Growth아이템을 먹으면 가장 뒤에 있는 부분(꼬리 부분을) 하나를 추가한다. temp라는 동적배열을 임의로 만들고, 하나 추가된 snake의 값을 저장 그리고 snake를 재할당한다.**

void Snake::eatGrowth(){

cordinate\* temp = new cordinate[len];

for(int i = 0; i < len; i++){

temp[i].x = v[i].x; temp[i].y = v[i].y;

}

len++;

score.growth++;

if(score.Best <= len-1)

score.Best++;

mvwprintw(score\_win,2,6,"%d",len-1);

mvwprintw(score\_win,2,8,"/");

mvwprintw(score\_win,2,10,"%d",score.Best );

mvwprintw(score\_win,4,6,"%d",score.growth );

if(score.Best >= score.BestMi){ // 숫자 !!

mvwprintw(Mission\_win,2,10,"V");

MiBest = true;

}

if(score.growth >= score.growthMi) { // ,,

mvwprintw(Mission\_win,4,10,"V");

MiGrowItem = true;

}

wrefresh(score\_win);

wrefresh(Mission\_win);

delete[] v;

v = new cordinate[len];

for(int i = 1; i < len; i++){

v[i].x = temp[i-1].x; v[i].y = temp[i-1].y;

}

v[0].x = growth.x; v[0].y = growth.y;

score.growth++;

isGrowth = false;

}

**4단계 : 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과 할 수 있도록 수정한다. (개발자 : 권순호)**

1. **스네이크, Wall, Poision Item, Growth Item과 겹치지 않게 Gate를 생성한다. Gate 서로간의 간격도 무조건 뛰도록 한다.**

**void Snake::generateGate() {**

**int x, y;**

**bool conti = true;**

**while(conti){**

**conti = false;**

**x = rand()%35;**

**y = rand()%21;**

**// snake와 겹치는지 확인**

**for(int i = 0; i < len; i++){**

**if(v[i].x == x && v[i].y == y){**

**conti = true;**

**break;**

**}**

**}**

**//poison item과 겹치는지 확인**

**if(x == poison.x && y == poison.y)**

**conti = true;**

**//growth item과 겹치는지 확인**

**if(x == growth.x && y == growth.y)**

**conti = true;**

**//Wall 안에 없는지 확인**

**if(1 != MAP[y][x])**

**conti = true;**

**}**

**gate[0].x = x; gate[0].y = y;**

**conti = true;**

**while(conti){**

**conti = false;**

**x = rand()%35;**

**y = rand()%21;**

**// snake와 겹치는지 확인**

**for(int i = 0; i < len; i++){**

**if(v[i].x == x && v[i].y == y){**

**conti = true;**

**break;**

**}**

**}**

**//poison item과 겹치는지 확인**

**if(x == poison.x && y == poison.y)**

**conti = true;**

**//growth item과 겹치는지 확인**

**if(x == growth.x && y == growth.y)**

**conti = true;**

**//Wall 안에 없는지 확인**

**if(1 != MAP[y][x])**

**conti = true;**

**//gate1이 gate0 주변 범위 1 안에 있는지 확인 => 진입 방향 검출 때문**

**if(gate[0].x <= x+1 && gate[0].x >= x-1 && gate[0].y <= y+1 && gate[0].y >= y-1)**

**conti = true;**

**}**

**gate[1].x = x; gate[1].y = y;**

**isGate = true;**

**wattron(map\_win,COLOR\_PAIR(GATE));**

**mvwprintw(map\_win,gate[0].y,gate[0].x,"#");**

**mvwprintw(map\_win,gate[1].y,gate[1].x,"#");**

**wattroff(map\_win,COLOR\_PAIR(GATE));**

**wrefresh(map\_win);**

1. **게이트 진입 알고리즘 1 – Gate가 가장자리 Wall에 있을 경우**

**위,아래,왼쪽,오른쪽에 맞춰 상수 배열을 지정하여 그대로 사용할 수 있도록 하였다.**

**bool jumpGateNum = 1;**

**const int x\_seq[] ={0,1,0,-1};**

**const int y\_seq[] = {-1,0,1,0};**

**const char keyboard\_seq[] = {'w', 'd', 's', 'a'};**

**// gate[0] gate[1] 구분**

**if(gate[1].x <= v[0].x+1 && gate[1].x >= v[0].x-1 && gate[1].y <= v[0].y+1 && gate[1].y >= v[0].y-1)**

**jumpGateNum = 0;**

**// #1 먼저 가장자리일 경우**

**// 상단 벽**

**if(gate[jumpGateNum].y == 0)**

**{**

**v[0].x = gate[jumpGateNum].x+x\_seq[2];**

**v[0].y = gate[jumpGateNum].y+y\_seq[2];**

**dir = keyboard\_seq[2];**

**return;**

**}**

**// 하단 벽**

**if(gate[jumpGateNum].y == NLINE-1)**

**{**

**v[0].x = gate[jumpGateNum].x+x\_seq[0];**

**v[0].y = gate[jumpGateNum].y+y\_seq[0];**

**dir = keyboard\_seq[0];**

**return;**

**}**

**// 좌측 벽**

**if(gate[jumpGateNum].x == 0)**

**{**

**v[0].x = gate[jumpGateNum].x+x\_seq[1];**

**v[0].y = gate[jumpGateNum].y+y\_seq[1];**

**dir = keyboard\_seq[1];**

**return;**

**}**

**// 우측 벽**

**if(gate[jumpGateNum].x == NCOL-1)**

**{**

**v[0].x = gate[jumpGateNum].x+x\_seq[3];**

**v[0].y = gate[jumpGateNum].y+y\_seq[3];**

**dir = keyboard\_seq[3];**

**return;**

**}**

1. **게이트 진입 알고리즘 2 – Gate가 맵 중앙 Wall에 있을 경우**

**시계방향으로 계속 회전하며 길이 있는지 탐색하고, 있을 경우 진행 한다.**

**// #2 가장자리가 아닌 경우**

**// 시계 방향 회전**

**int cur\_seqnum;**

**for (int j = 0; j < 4; ++j)**

**{**

**if (dir == keyboard\_seq[j])**

**{**

**cur\_seqnum = j;**

**break;**

**}**

**}**

**while(1)**

**{**

**if(MAP[gate[jumpGateNum].y+y\_seq[cur\_seqnum]][gate[jumpGateNum].x+x\_seq[cur\_seqnum]] != 1)**

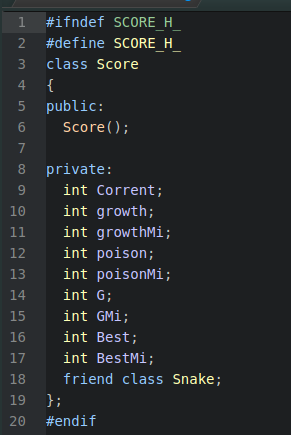
**break;**

**if(++cur\_seqnum == 4)**

**cur\_seqnum = 0;**

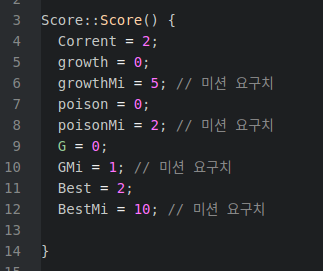
**}**

**5단계 : 4단계 프로그램에서, 우측에 게임 점수를 표시하는 화면을 구성한다.(개발자: 박승현)**

****

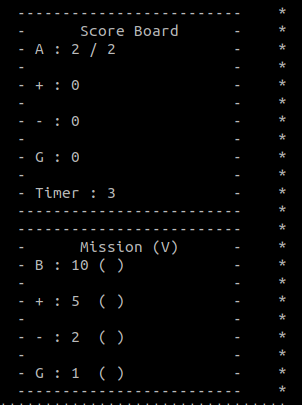
**Score.h 파일입니다.**

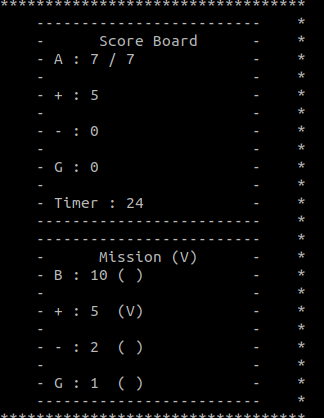
**미션에 필요한 각각의 수치와 현제 획득하거나 상태를 나타내는 변수, 현제 길이 등을 저장합니다.**



**Score.cpp 파일입니다. 초기 값 설정을 해줍니다 각각의 값들은 플레이 도중 상황에 맞게 업데이트 됩니다.**

**처음 디스플레이를 위한 코드입니다.**

**초기 설정대로 화면에 표시된 모습입니다.**

****

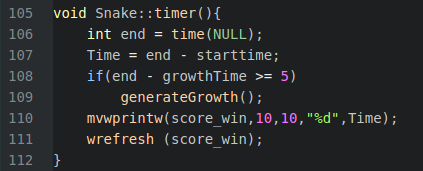
**위의 이미지는 상황에 따라 바뀐 모습을 화면에 업데이트하여 표시한 모습을 나타낸 것입니다. 밑의 미션의 v여부는 각 조건을 만족할 때 가장 빨리 파악할 수 있는 위치에 코드를 적어 바로 적용시켰습니다.**

****

**예를 들어 Grow아이템을 먹는 상황일 때 아이템을 먹은 수 하나 증가시키고**

**402번째 줄의 코드로 아이템을 먹은 수가 미션의 요구치를 넘어서면 V표시를 해주는 방식입니다.**

**마지막으로 timer 기능입니다.**

****

**Snake를 생성할 때 즉 게임이 시작될 때 시작 시간을 time(NULL)로 받아옵니다.**

**게임 플레이 이후 키를 받아올때마다 timer함수를 실행하여 현재 시간을 받아와**

**현재시간에서 시작시간을 뺀 값 게임 진행시간이 되므로 그 값을 화면에 표시해 줍니다. 아이템 생성 시간을 이 타이머 함수로 결정 하였습니다.**

### 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.** |

**기본 베이스 ncureses.h**

**타이머 설정을 위한 time.h**

### 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.** |

1. **키를 입력 받지 않았을 때에도 게임을 진행을 해야한다.**

**해결방법 : timout() function을 이용하여 일정시간이 지연되면 진행을 하는 방식을 사용**

1. **Memory Collision**

시계, 측정기이(가) 표시된 사진

자동 생성된 설명

**해결방법: memory를 할당을 더 하는것이 아니라 한번 delete[]를 하고 재할당을 통해 해결.**

### 결과물 목록

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

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

# 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.** |

**박승현 : 디스플레이 화면을 구현하는 역할을 하면서 사실 이번 c++과목은 알고랩, 실습, 시험준비 등 할 일이 많기도 해서 정말 힘들었던 과목이었습니다. 거기에 프로젝트와 보고서 까지 생기니 솔직히 많이 고통스러웠던 것이 사실이었지만 막상 진행하면서 내가 쓴 코드가 화면에 나타나는 것을 보면서 뿌듯함을 느꼈고 여러 과목 중 그래도 꽤 열심히 했구나 라는 생각이 들면서 꽤 많은 것을 배우고 간다 라는 생각이 들었습니다.**

**권순호 : 한학기동안 C++수업에서 배웠던 다양한 테크닉을 중규모 프로젝트에 참여하여 수행하니 매우 유익한 시간이었다. 회의를 통해 주어진 개발을 충실히 수행하였으며, Modern C++을 새로 배워 사용하지 못한 점은 아쉬웠다.**

**양진우 : 처음에 timeout()함수를 이용하지 않고, 게임을 구현을 할려고 했으나 못했습니다. 검색 중, timeout()함수를 쓰고 쉽게 게임을 구현을 했던게 뿌듯했습니다. 그리고 C++수업에서 배웠던 enum이나 클래스를 이용해서 게임을 구현했던 점이 재밌었습니다.**

# 참고 문헌

**참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 웹페이지 | 동기와 비동기적 실행 | 씹어먹는 C++ | 2019-05-18 | 씹어먹는 C++ |  |

# 부록

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

## 사용자 매뉴얼

**프로젝트 실행 후 사용안내, 따라하기 등 포함**

**5.1.1 실행**

**표지판, 그리기이(가) 표시된 사진

자동 생성된 설명**

**실행방법은 ./Snake를 입력하면 됩니다**

**5.1.2 게임플레이**

**개체, 시계, 모니터이(가) 표시된 사진

자동 생성된 설명**

**전체 화면의 구성입니다. 게임 맵, 점수판, 미션으로 구성되어 있습니다.**

**조작은 W,A,S,D로 하며 중지할 경우 P를 누르면 됩니다.**

**화면, 텔레비전, 모니터, 노트북이(가) 표시된 사진

자동 생성된 설명**

**스네이크의 모습입니다. 처음 시작할 때는 3칸으로 시작됩니다. 방향키로 잘 움직여서 미션을 깨도록 노력해야합니다.**

**개체, 시계, 화면이(가) 표시된 사진

자동 생성된 설명**

**Poison Item의 모습입니다. 스네이크가 먹을 경우 길이가 1 감소합니다.**

검은색, 화면, 남자, 빨간색이(가) 표시된 사진

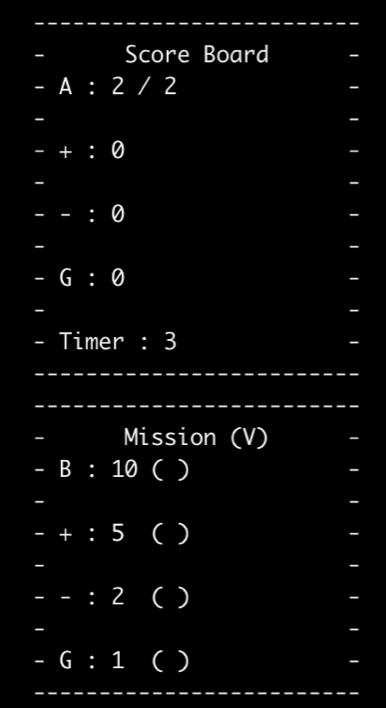
자동 생성된 설명

**Gate의 모습입니다. 진입 시 또 다른 게이트로 이동합니다.**

개체, 시계이(가) 표시된 사진

자동 생성된 설명

**Growth Item의 모습입니다. 스네이크가 먹을 경우 길이가 1증가합니다.**

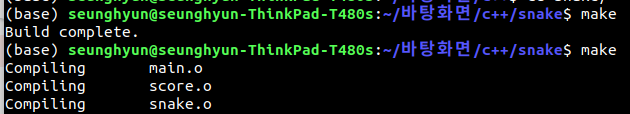


**점수판과 미션판입니다. 나와 있는 미션을 충족해 게임에서 승리하세요!**

## 설치 방법

**프로그램 컴파일 및 실행 방법 포함**

**먼저 한 폴더에 모든 파일을 저장합니다.**

****

****

**그후 그냥 makefile을 실행시키기 위한 make 명령어를 실행시키기만 하면됩니다..!**

****

**Linking 뒤의 이름을 잘 확인하고**

**./’’’ 으로 실행하면 게임이 시작됩니다!**

**저희는 유쾌한 분위기를 위해 팀장님의 별명을 차용하였습니다.**