



국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	<i>Snake Game</i>
팀 명	24조
문서 제목	결과보고서

Version	1.3
Date	2024-JUN-17

팀원	김 설
	정지원

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

CONFIDENTIALITY/SECURITY WARNING
<p>이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 "Snake Game"를 수행하는 팀 "24조"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "24조"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.</p>

문서 정보 / 수정 내역


Filename	최종보고서-SnakeGame.doc
원안작성자	김설, 정지원
수정작업자	김설, 정지원

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2024-06-13	김설	1.0	최초 작성	
2024-06-14	정지원	1.1	내용 추가	구조 및 설계 내용 추가
2024-06-15	김설	1.2	내용 추가	3,4,5 내용 완성
2024-06-16	정지원	1.3	내용 수정	개발 내용 세부사항 수정

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

목 차

1	개요	4
2	개발 내용 및 결과물	5
2.1	목표	5
2.2	개발 내용 및 결과물	7
2.2.1	개발 내용	7
2.2.2	시스템 구조 및 설계도	10
2.2.3	활용/개발된 기술	35
2.2.4	현실적 제한 요소 및 그 해결 방안	36
2.2.5	결과물 목록	38
3	자기평가	39
4	참고 문헌	39
5	부록	40
5.1	사용자 매뉴얼	40
5.2	설치 방법	42

 <div> 국민대학교 소프트웨어부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

1 개요

프로젝트를 개발하기 위해 사용한 언어는 C++입니다. 총 12개의 파일로 작성하였습니다. 아래는 각 파일에 대한 설명입니다.

1. Charposition.h & Charposition.cpp

- CharPosition 구조체를 정의하고 구현합니다. 이 구조체는 게임 내에서 위치 정보(x, y 좌표)를 관리하는 데 사용됩니다.
- 생성자를 통해 위치 정보를 초기화합니다.

2. Display.h & Display.cpp

- 게임 화면의 렌더링을 담당합니다. 게임 내의 벽, 뱀, 아이템 등의 그래픽 요소를 화면에 표시합니다.
- 게임 창의 경계와 장애물을 그리고, 뱀과 아이템의 상태를 업데이트하며 화면에 출력합니다.

3. Gameplay.h & Gameplay.cpp

- 뱀의 움직임과 게임 로직의 핵심 부분을 처리합니다. 사용자 입력을 받아 뱀을 조작하고, 게임의 진행 상황을 관리합니다.
- 사용자 입력에 따라 뱀의 방향을 변경하고, 뱀이 아이템을 먹거나 장애물에 부딪힐 경우를 처리합니다. 게임 오버와 스테이지 클리어 조건을 검사합니다.

4. Initialization.h & Initialization.cpp

- 게임의 초기 설정을 구성합니다. 게임 시작 시 필요한 설정을 초기화하고, 각 레벨에 맞는 환경을 설정합니다.
- 게임 환경(창 크기, 아이템의 타이머 등) 설정, 뱀의 초기 위치와 방향 설정, 초기 스코어 및 게임 상태 설정 등을 포함합니다.

5. Items.h & Items.cpp

- 게임 내의 아이템(성장, 독, 속도 아이템 등)을 관리합니다. 아이템의 배치, 시간 관리 및 효과를 처리합니다.
- 아이템의 무작위 배치, 아이템의 상태 갱신, 뱀이 아이템을 먹었을 때의 효과 처리 등을 수행합니다.

6. main.h & main.cpp

- 게임의 시작점을 제공합니다. main 함수를 통해 게임을 시작하고, 게임의 전체 흐름을 관리합니다.
- 게임 루프의 실행, 사용자의 게임 시작 및 재시작 의사 확인, 게임의 클리어 여부 확인 및 처리 등을 담당합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

2 개발 내용 및 결과물

2.1 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 단계] Map 의 구현

- ✓ 사용자의 터미널 크기에 맞게 화면을 조정하도록 한다.
- ✓ 외벽은 하얀색으로 설정한다.
- ✓ 각 level 마다 Wall 의 위치가 바뀌도록 구성한다.

[2 단계] Snake 표현 및 조작

- ✓ 방향키는 원하는 키로 직접 정한다.
- ✓ Head 방향 이동은 무시한다.
- ✓ Tail 방향으로 이동시 실패한다.
- ✓ Snake 는 자신의 Body 를 통과할 수 없다.
- ✓ Snake 는 벽을 통과할 수 없다.
- ✓ Head 방향 이동은 일정시간에 의해 이동한다.
- ✓ 몸의 길이가 3 보다 작아지면 실패한다.

[3 단계] Item 요소의 구현

- ✓ Snake 의 Head 위치와 Item 의 위치가 같은 경우 Item 을 획득한다.
- ✓ Growth Item 의 경우 몸의 길이가 1 증가한다.
- ✓ Poison Item 의 경우 몸의 길이가 1 감소한다.
- ✓ speed Item 의 경우 Snake 의 이동 속도가 증가한다.
- ✓ Item 의 경우 Snake Body 가 있지 않은 임의의 위치에 출현하도록 한다.
- ✓ 출현 후 일정시간이 지나면 사라지고 다른 위치에 나타나야 한다.
- ✓ 동시에 출현할 수 있는 Item 의 개수는 3 개로 제한한다.
- ✓ Growth Item 을 획득하면 몸의 길이는 진행방향으로 증가한다.
- ✓ Poison Item 을 획득하면 꼬리 부분이 1 감소한다.


 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

[4 단계] Gate 요소의 구현

- ✓ Gate 는 두 개가 한 쌍이다.
- ✓ Gate 는 겹치지 않는다.
- ✓ Gate 는 임의의 위치에 있는 벽에서 나타난다.
- ✓ Gate 에 Snake 가 진입하면, 다른 Gate 로 진출한다.
- ✓ Gate 에 Snake 가 진입중인 경우 Gate 는 사라지지 않고, 다른 위치에서 Gate 가 나타나지 않는다.
- ✓ Gate 는 한 번에 한 쌍만 나타난다.
- ✓ Gate 가 나타나는 벽이 가장자리에 있을 때 항상 Map 의 안쪽 방향으로 진출한다.
- ✓ Gate 가 나타나는 벽이 Map 의 가운데 있을 때 다음의 순서로 진출한다.
 - (1) 진입 방향과 일치하는 방향이 우선
 - (2) 진입 방향의 시계방향으로 회전하는 방향
 - (3) 진입 방향의 역시계방향으로 회전하는 방향
 - (4) 진입 방향과 반대방향

[5 단계] 점수 요소의 구현

- ✓ 먼저 Score Board 에는 다음의 내용을 포함하도록 한다.
 - (1) Snake 현재 길이
 - (2) 먹은 Growth Item 개수
 - (3) 먹은 Poison Item 개수
 - (4) 먹은 Speed Item 개수
 - (5) 통과한 Gate 개수
- ✓ 다음으로 Mission 에는 다음 스테이지로 넘어가기 위한 목표가 있고 그 목표를 달성하면 화면에 'v'로 체크 표시가 된다.
- ✓ Mission 을 모두 완료하면 다음 스테이지로 넘어간다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

2.2 개발 내용 및 결과물

2.2.1 개발 내용

1단계 : Map의 구현

- 맵은 2차원 배열로 구현되며, 각 셀은 벽(1), 면역 벽(2), 뱀의 머리(3), 뱀의 몸통(4)로 구분됩니다. 이를 통해 게임의 물리적 경계와 장애물을 설정합니다.
- `setWindow()` 함수를 사용하여 게임 윈도우를 초기화하고 설정합니다. `NCurses`의 `initscr()`, `noecho()`, `keypad()` 등의 함수를 활용하여 입력 처리 및 디스플레이 설정을 조정합니다.
- `renderWindow()` 함수는 맵을 시각적으로 표현하는 역할을 합니다. 이 함수는 각 게임 레벨에 맞게 맵을 다르게 구성하고, `ncurses`를 사용하여 다양한 색상과 속성을 벽에 적용합니다.
- 맵은 뱀, 아이템 및 기타 게임 객체와 동적으로 상호작용합니다. 예를 들어, 뱀이 벽에 부딪히면 게임 오버 로직이 처리됩니다.
- 맵의 크기는 사용자의 터미널 크기에 따라 동적으로 조정될 수 있으며, 이는 `getmaxyx()` 함수를 통해 구현됩니다. 이는 맵의 범위와 게임 플레이 영역을 사용자의 화면 크기에 맞게 최적화합니다.


2단계 : Snake의 표시 및 조작

- 키 입력 처리 방법:

Snake 게임의 핵심 기능 중 하나는 사용자 입력에 반응하여 뱀을 움직이게 하는 것입니다. 이를 위해, `ncurses` 라이브러리를 사용하여 키 입력을 수집합니다. `getKey()` 함수는 `nodelay()` 함수를 사용하여 키 입력이 없을 경우 바로 0을 반환하고, 입력이 있을 경우 해당 키 값을 반환합니다. 이를 통해 사용자는 실시간으로 방향키를 통해 뱀의 방향을 조정할 수 있습니다.

- Tick에 대한 변화 주기:

게임의 응답성을 조절하기 위해 'tick' 개념을 도입했습니다. Tick은 게임 내에서 시간의 흐름을 나타내며, 각 tick마다 뱀이 한 칸씩 이동합니다. 이동 속도는 사용자의 게임 레벨에 따라 조절될 수 있으며, 뱀이 아이템을 먹거나 특정 이벤트가 발생할 때마다 조정됩니다. `usleep()` 함수를 사용하여 tick 사이의 지연 시간을 설정하고, 뱀의 속도를 조절합니다. 이를 통해 게임의 난

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

이도를 조절하고 사용자 경험을 향상시킬 수 있습니다.

- Snake 초기화 및 표시: renderSnake() 함수에서는 뱀의 초기 위치와 길이를 설정합니다. 뱀은 게임 맵의 중앙에서 시작하며 길이는 설정에 따라 조정될 수 있습니다. 이후 뱀의 각 부분을 맵에 표시합니다.
- 움직임 및 충돌 검사: moveSnake() 함수는 현재 뱀의 머리 방향을 바탕으로 새로운 위치를 계산하고, 맵의 경계나 자신의 몸통과의 충돌을 검사합니다. 충돌이 감지되면 게임 오버 처리를 하고, 그렇지 않다면 뱀의 위치를 업데이트하고 맵에 다시 그립니다.
- 아이템 반응: 뱀이 성장 아이템이나 독 아이템을 먹었을 때의 처리도 moveSnake() 내에서 처리합니다. 아이템을 먹으면 뱀의 길이가 조정되고, 이는 게임의 진행 상황에 직접적인 영향을 미칩니다.

3 단계 : Item 요소의 구현

- Growth Item 과 Poison Item 은 맵 상의 랜덤 위치에 배치됩니다. 각 아이템은 고유한 색상과 기호로 표현됩니다 (예: Growth Item 은 녹색 '+' 기호, Poison Item 은 빨간 '-' 기호). 이 아이템들은 각각 Snake 의 길이를 늘리거나 줄이는 기능을 합니다. Snake 가 아이템을 먹으면 해당 아이템은 맵에서 사라지고, 새 위치에 새로운 아이템이 생성됩니다.
- 맵 데이터에서 아이템은 고유한 값을 갖습니다. 이 값들은 맵 배열 내에서 해당 아이템의 위치를 나타내는 데 사용됩니다. 아이템의 위치는 게임 루프의 각 반복마다 업데이트되어 동적인 게임 플레이를 제공합니다.
- Snake 가 아이템에 도달했을 때의 상호작용은 게임의 기본 메커니즘을 통해 처리됩니다. checkItemInteraction() 함수는 Snake 의 머리 위치와 맵 상의 아이템 위치를 체크하여, 먹힌 아이템에 따라 Snake 의 길이를 조정합니다. 이 로직은 또한 아이템을 새 위치에 재배치하는 역할도 수행합니다.
- 게임의 다양성을 더하기 위해 'Speed Boost Item'을 도입할 것을 제안합니다. 이 아이템은 Snake 의 이동 속도를 일시적으로 증가시켜 더 빠른 반응을 가능하게 하고, 게임의 긴장감을 높입니다. 이 아이템은 청색 '*' 기호로 표현되며, 맵 데이터에서는 7 로 구분됩니다.

4 단계 : Gate 요소의 구현

- Gate는 Wall 내의 임의의 위치에 쌍으로 생성됩니다. 이 Gate들은 화면상에서 Wall과 구분될 수 있도록 독특한 색상 또는 기호(예: 파란색 '#')로 표시됩니다. Gate는 맵 데이터에서


 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

고유한 값을 갖습니다 (예: 7). 이는 맵 배열에서 Gate의 위치를 나타내는 데 사용됩니다.

- Snake가 Gate에 도달하면, 다른 Gate로 순간 이동합니다. 이 동작은 teleportSnake() 함수를 통해 처리되며, 함수는 Snake의 현재 방향과 해당 Gate의 위치를 고려하여 새 위치와 방향을 결정합니다.
- Snake가 Wall 또는 Immune Wall에 부딪히면 게임이 종료됩니다. checkCollision() 함수는 Snake의 머리가 Wall의 좌표와 일치하는지 검사하고, 충돌이 발생했을 경우 게임 오버를 처리합니다.
- 'Dynamic Wall' 은 일정 시간마다 위치가 변경되거나, 일정 패턴으로 움직이는 벽을 말합니다. Dynamic Wall은 게임 맵에 새로운 장애물을 제공하고, Snake의 경로 설정에 고려 요소를 늘립니다. 이 벽은 다른 색상(예: 회색)으로 표시되며, 맵 데이터에서는 8로 구분됩니다.

5 단계 : 점수 요소의 구현

- 사용자는 자신의 뱀 길이와 각 미션의 완성 여부를 확인할 수 있습니다. 이는 displayScore() 함수에 의해 관리되며, 각 아이템의 획득 상태나 게이트 통과 횟수 등을 포함합니다.
- 점수판은 게임 중 발생하는 이벤트에 따라 동적으로 업데이트됩니다. 아이템을 획득하거나 게이트를 통과할 때마다 점수판의 정보가 즉시 갱신되어 게임의 진행 상태를 반영합니다.
- 각 스테이지는 특정 목표를 달성해야 합니다. 예를 들어, 특정 길이에 도달하거나 일정 수의 아이템을 수집해야 다음 스테이지로 넘어갈 수 있습니다.
- 최소 4개의 스테이지가 구성되며, 각 스테이지는 고유한 맵과 도전 과제를 제공합니다. 이는 setWindow() 함수에서 스테이지별로 다른 맵 디자인을 로드하여 구현합니다.

 <div> 국민대학교 소프트웨어부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

2.2.2 시스템 구조 및 설계도

1단계 : Map의 구현(김설 담당)

- set window

```
// window 초기화
void snakeGame::setWindow(int level)
{
    initscr(); // initialise the screen
    nodelay(stdscr, TRUE);
    keypad(stdscr, true);
    noecho(); // user input is not displayed on the screen
    curs_set(0); // cursor symbol is not displayed on the screen (Linux)
    getmaxyx(stdscr, maxHeight, maxWidth); // define dimensions of game window
    maxHeight -= (level - 1) * 3;
    maxWidth -= (level - 1) * 6;
    return;
}
```

- 기본적인 게임 윈도우 세팅을 설정해주었습니다. 단말기와 키패드를 허용하고, 사용자의 input 을 따로 보여주지 않기 위해 noecho()를 사용합니다. 또한 따로 커서를 보이지 않게 하기위해 curs_set(0)을 선언해주었습니다.
- 또한 스테이지가 올라갈수록 난이도를 높이기 위해 레벨에 따라 윈도우가 작아지도록 설정했습니다

- renderwindow

```
int k, q;
for (int i = 1; i < maxWidth - 12; i++) // 윗벽
{
    wall.push_back(CharPosition(i, 0));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(0, i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}

for (int i = 1; i < maxWidth - 12; i++) // 아래 벽
{
    wall.push_back(CharPosition(i, maxHeight - 1));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(maxHeight - 1, i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}
```



```
for (int i = 1; i < maxHeight - 1; i++) // 왼쪽 벽
{
    wall.push_back(CharPosition(0, i));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(i, 0);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}

for (int i = 1; i < maxHeight - 1; i++) // 오른쪽 벽
{
    wall.push_back(CharPosition(maxWidth - 12, i));
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(i, maxWidth - 12);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
    refresh();
}
```

- maxWidth, maxHeight 에 따라 가장자리 외벽을 생성해줍니다.
- score board 자리를 위해 오른쪽은 12만큼 빼 주었습니다.
- 하나씩 벽을 생성하며 미리 선언해둔 vector 에 wall의 좌표를 담은 객체를 저장합니다.
이를 이용하여 gate와 같은 wall을 이용한 각종 기능과 동작을 구현합니다.
- 벽은 하얀색으로 구현하였습니다.



```
// 레벨에 따라 맵 구성
switch (level)
{
case 1: // KMU 모양의 장애물
for (int i = 0; i < (maxWidth - 12) / 10; i++) {
start_color();
init_pair(3, COLOR_WHITE, COLOR_WHITE);
attron(COLOR_PAIR(3));

wall.push_back(CharPosition((maxWidth - 12) / 10 * 1, H));
move(H, (maxWidth - 12) / 10 * 1);
addch(edgechar);

wall.push_back(CharPosition((maxWidth - 12) / 10 * 2 - i, H));
move(H, (maxWidth - 12) / 10 * 2 - i);
addch(edgechar);
attroff(COLOR_PAIR(3));
refresh();
H++;
}
for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...

// M
H = (maxHeight - 1) / 5;
for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
H = (maxHeight - 1) / 5;

//U
for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
for (int i = 0; i < (maxWidth - 12) / 10; i++) { ...
break;
}
case 2: { // 3분할 장애물 ...
case 3: { // X모양의 장애물 ...
case 4: { // 9분할 장애물
{ ...
break;
}
}
```

- switch 문을 이용하여 레벨(스테이지)별로 다른 맵을 형성하도록 구현하였습니다.
- 이전의 외부벽을 세운 것과 같은 맥락으로 wall 을 추가하고, 색을 바꾸어 주었습니다.
- stage1 은 학교 이름인 'KMU'모양으로 장애물로 구현하였습니다
- stage2 는 화면을 3 분할 하는 장애물로 구현하였습니다.
- stage3 은 X 자모양으로 화면을 분할하는 장애물로 구현하였습니다.
- stage4 는 화면을 9 분할 하는 장애물로 구현하였습니다.



```
// 모서리 부분을 다른 색으로 표시
start_color();
init_pair(4, COLOR_BLACK, COLOR_BLACK);
attron(COLOR_PAIR(4));
move(0, 0);
addch(edgechar);
move(0, maxWidth - 12);
addch(edgechar);
move(maxHeight - 1, 0);
addch(edgechar);
move(maxHeight - 1, maxWidth - 12);
addch(edgechar);
attroff(COLOR_PAIR(4));
refresh();
```

- 벽의 가장자리에 위치한 immune wall 은 색을 검은색으로 구현하여 일반 벽과 차이를 두었습니다.


 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

2단계 Snake 표현 및 조작 (정지원 담당)

- moveSnake

```
// Snake 움직임 정의
void snakeGame::moveSnake()
{
    KeyPressed = getch();
    if (KeyPressed == leftKey) {
        if (direction != 'r')
        {
            direction = 'l';
        }
        else
        {
            checkCollision();
        }
    }
    else if (KeyPressed == rightKey) {
        if (direction != 'l')
        {
            direction = 'r';
        }
        else
        {
            checkCollision();
        }
    }
    else if (KeyPressed == upKey) {
        if (direction != 'd')
        {
            direction = 'u';
        }
        else
        {
            checkCollision();
        }
    }
    else if (KeyPressed == downKey) {
        if (direction != 'u')
        {
            direction = 'd';
        }
        else
        {
            checkCollision();
        }
    }
    else if (KeyPressed == KEY_BACKSPACE)
        direction = 'q';
}
```

- 방향키 입력에 대하여 snake 의 방향을 움직이도록 구현하였습니다. 백스페이스를 누르면 게임은 종료됩니다.

 <div> 국민대학교 소프트웨어부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

- checkCollision

```
// 게임 종료 조건
bool snakeGame::checkCollision()
{
    for (int i = 0; i < wall.size(); i++)
    {
        if (snake[0].x == wall[i].x && snake[0].y == wall[i].y)
        {
            if (!((snake[0].x == gate_1.x && snake[0].y == gate_1.y) || (snake[0].x == gate_2.x && snake[0].y == gate_2.y)))
            {
                return true;
                break;
            }
        }
    }
}
```

- Snake 는 벽(Wall)을 통과할 수 없고, 자신의 몸을 통과할 수 없습니다. 따라서 if 문을 이용하여 스네이크의 헤드부분이 벽과 부딪히게 된다면, 게이트인지 확인 후 게이트가 아니라면 게임이 종료되도록 구현하였습니다.

```
for (int i = 2; i < snake.size(); i++)
{
    if (snake[0].x == snake[i].x && snake[0].y == snake[i].y)
    {
        return true;
    }
}
```

- 또한 Snake 의 Head 가 자신의 Body 부딪치게 되면 게임을 중단하도록 합니다.




```
if (snake.size() < 3)
{
    return true;
}

if (direction == 'r' && KeyPressed == leftKey)
{
    return true;
}
if (direction == 'l' && KeyPressed == rightKey)
{
    return true;
}
if (direction == 'u' && KeyPressed == downKey)
{
    return true;
}
if (direction == 'd' && KeyPressed == upKey)
{
    return true;
}

return false;
```

- snake의 몸 길이가 3 미만인 경우에도 게임을 종료 시킵니다.
- 또한 if 문을 이용하여 snake의 진행방향과 역방향으로 가는 키가 눌린다면 게임은 종료되도록 구현하였습니다.

	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

3단계 : Item 요소의 구현 (정지원 담당)


(1) GrowthItem 구현

```
// 맵에 GrowthItem 표현
void snakeGame::placeGrowthItem()
{
    int tmpx, tmpy;
    bool clear = false;
    while (!clear)
    {
        tmpx = rand() % (maxWidth - 13) + 1;
        tmpy = rand() % (maxHeight - 2) + 1;
        clear = true;

        for (int i = 0; i < snake.size(); i++)
        {
            if (snake[i].x == tmpx && snake[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }

        for (int i = 0; i < wall.size(); i++)
        {
            if (wall[i].x == tmpx && wall[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }
    }
}
```

- 난수를 이용해서 임의의 위치를 받고 그 위치가 Snake, wall 과 겹치지 않도록 설정 합니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

```

growthItem.x = tmpx;
growthItem.y = tmpy;
start_color();
init_pair(1, COLOR_WHITE, COLOR_GREEN);
attron(COLOR_PAIR(1));
move(growthItem.y, growthItem.x);
addch(growthItemShape);
attroff(COLOR_PAIR(1));
refresh();
}

```

- 그 위치에 색상은 초록색으로 표현한 후 맵에 보이도록합니다.

```

// GrowthItem 위치 변경
void snakeGame::growthItemTime()
{
    growthItemTimer++;
    if (growthItemTimer % maxItemTime == 0)
    {
        move(growthItem.y, growthItem.x);
        printw(" ");
        placeGrowthItem();
        growthItemTimer = 0;
    }
}

```

- 일정시간이 지나면 GrowthItem 의 위치를 변경합니다.



```
// GrowthItem 먹었는지 확인
bool snakeGame::checkGrowth()
{
    if (snake[0].x == growthItem.x && snake[0].y == growthItem.y)
    {
        growthItemTimer = 0;
        placeGrowthItem();
        currentLength++;
        scoreGrowthItem++;
        displayScore();
        return eatGrowthItem = true;
    }
    else
    {
        return eatGrowthItem = false;
    }
}
```

- Snake 가 GrowthItem 을 먹었는지 체크합니다.
- Snake 의 Head 위치가 GrowthItem 의 위치와 일치해야 합니다.



(2) PoisonItem 구현

```
// 맵에 PoisonItem 표현
void snakeGame::placePoisonItem()
{
    int tmpx, tmpy;
    bool clear = false;
    while (!clear)
    {
        tmpx = rand() % (maxWidth - 13) + 1;
        tmpy = rand() % (maxHeight - 2) + 1;
        clear = true;

        for (int i = 0; i < snake.size(); i++)
        {
            if (snake[i].x == tmpx && snake[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }

        for (int i = 0; i < wall.size(); i++)
        {
            if (wall[i].x == tmpx && wall[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }
    }
}
```

- 난수를 이용해서 Snake의 위치와 Wall의 위치와 일치 하지 않는 곳을 찾습니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

```

poisonItem.x = tmpx;
poisonItem.y = tmpy;
start_color();
init_pair(2, COLOR_WHITE, COLOR_RED);
attron(COLOR_PAIR(2));
move(poisonItem.y, poisonItem.x);
addch(poisonItemShape);
attroff(COLOR_PAIR(2));
refresh();
}

```

- 해당 위치를 찾으면 색상은 빨간색으로 설정 한 후 맵에 보이도록 합니다.

```

// PoisonItem 위치 변경
void snakeGame::poisonItemTime()
{
    poisonItemTimer++;
    if (poisonItemTimer % maxItemTime == 0)
    {
        move(poisonItem.y, poisonItem.x);
        printw(" ");
        placePoisonItem();
        poisonItemTimer = 0;
    }
}

```

- 일정시간이 지나면 PoisonItem 의 위치가 바뀌도록 합니다.



```
// PoisonItem 먹었는지 확인
bool snakeGame::checkPoison()
{
    if (snake[0].x == poisonItem.x && snake[0].y == poisonItem.y)
    {
        poisonItemTimer = 0;
        placePoisonItem();
        currentLength--;
        scorePoisonItem++;
        displayScore();
        return eatPoisionItem = true;
    }
    else
    {
        return eatPoisionItem = false;
    }
}
```

- Snake 가 PoisonItem 을 먹었는지 확인합니다.
- 이 때 Snake 의 Head 부분이 PoisonItem 의 위치와 일치해야 합니다.




(3) SpeedItem 구현

```
// 맵에 SpeedItem 표현
void snakeGame::placeSpeedItem()
{
    int tmpx, tmpy;
    bool clear = false;
    while (!clear)
    {
        tmpx = rand() % (maxWidth - 13) + 1;
        tmpy = rand() % (maxHeight - 2) + 1;
        clear = true;

        for (int i = 0; i < snake.size(); i++)
        {
            if (snake[i].x == tmpx && snake[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }

        for (int i = 0; i < wall.size(); i++)
        {
            if (wall[i].x == tmpx && wall[i].y == tmpy)
            {
                clear = false;
                break;
            }
        }
    }
}
```

- 난수를 이용해서 임의의 위치를 찾습니다.
- 이 때 Snake의 위치와 Wall의 위치가 겹치지 않는 곳을 찾아야 합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

```

speedItem.x = tmpx;
speedItem.y = tmpy;
start_color();
init_pair(6, COLOR_WHITE, COLOR_YELLOW);
attron(COLOR_PAIR(6));
move(speedItem.y, speedItem.x);
addch(speedItemShape);
attroff(COLOR_PAIR(6));
refresh();
}

```

- 위치를 찾으면 그 위치를 노란색으로 설정한 후 맵에 보이도록 합니다.

```

// SpeedItem 위치 변경
void snakeGame::speedItemTime()
{
    speedItemTimer++;
    if (speedItemTimer % maxItemTime == 0)
    {
        move(speedItem.y, speedItem.x);
       printw(" ");
        placeSpeedItem();
        speedItemTimer = 0;
    }
}

```


- 일정 시간이 지나면 SpeedItem의 위치가 바뀌도록 합니다.

```

// SpeedItem 먹었는지 확인
void snakeGame::checkSpeed()
{
    if (snake[0].x == speedItem.x && snake[0].y == speedItem.y)
    {
        speedItemTimer = 0;
        placeSpeedItem();
        snakeSpeed -= 10000;
        scoreSpeedItem++;
        displayScore();
    }
}

```

- Snake가 SpeedItem을 먹었는지 확인합니다.
- 이 때 Snake의 Head 부분이 SpeedItem의 위치와 동일해야 합니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

4단계 : Gate 요소의 구현 (김설 담당)

```

//gate
void snakeGame::placeGate() //게이트 생성
{
    // 난수를 통해 idx를 받음
    int gate_idx1 = rand() % wall.size();
    int gate_idx2 = rand() % wall.size();
    while (gate_idx2 == gate_idx1) //중복되지않도록
    {
        gate_idx2 = rand() % wall.size();
    }
    gate_1 = wall[gate_idx1];
    gate_2 = wall[gate_idx2];
    start_color();
    init_pair(5, COLOR_BLUE, COLOR_BLUE);
    attron(COLOR_PAIR(5));
    move(gate_1.y, gate_1.x);
    addch(edgechar);
    move(gate_2.y, gate_2.x);
    addch(edgechar);
    attroff(COLOR_PAIR(5));
    refresh();
}

```

- Gate 를 표시할 때는 랜덤한 벽의 위치에 설정될 수 있도록 난수를 받았습니다.
- 위치를 받으면 그 위치에 게이트를 놓고 색상은 파란색으로 설정했습니다.



```
// Gate 위치 변경
void snakeGame::gateTime()
{
    gateTimer++;
    if (gateTimer % maxGateTime == 0)
    {
        attron(COLOR_PAIR(3));
        move(gate_1.y, gate_1.x);
        addch(edgechar);
        move(gate_2.y, gate_2.x);
        addch(edgechar);
        attroff(COLOR_PAIR(3));
        refresh();
        placeGate();
        gateTimer = 0;
    }
}
```

- 일정 시간이 지나면 기존 Gate 의 위치를 바꾸도록 했습니다.
- 원래 Gate 위치는 다시 하얀색 Wall 로 구현했습니다.

```
// Gate에 들어갔는지 확인
bool snakeGame::checkGate()
{
    if (snake[0].x == gate_1.x && snake[0].y == gate_1.y)
    {
        gateTimer = maxGateTime - snake.size() - 1;
        scoreGate++;
        displayScore();
        return atGate1 = true;
    }
    else if (snake[0].x == gate_2.x && snake[0].y == gate_2.y)
    {
        gateTimer = maxGateTime - snake.size() - 1;
        scoreGate++;
        displayScore();
        return atGate2 = true;
    }
}
```

- Snake 가 Gate 에 진입하였는지 확인합니다.
- 이 때 Snake 의 Head 부분이 Gate 의 위치와 겹쳐야 합니다.




```
// 게이트를 통과할 때
char snakeGame::getWarpDirection(char d, CharPosition gate)
{
    char result;
    CharPosition leftBlock(gate.x - 1, gate.y);
    CharPosition rightBlock(gate.x + 1, gate.y);
    CharPosition upBlock(gate.x, gate.y - 1);
    CharPosition downBlock(gate.x, gate.y + 1);

    bool isLeftWall = false;
    bool isRightWall = false;
    bool isUpWall = false;
    bool isDownWall = false;
}
```

- 게이트를 통과했을 때 다른 게이트의 어떤 곳으로 진출하게 될지 결정하는 함수입니다.
- 먼저 나오게 될 게이트 기준으로 상하좌우가 Wall 인지 확인하기 위해서 다음과 같은 구조체와 변수를 구현했습니다..

```
for (int i = 0; i < wall.size(); i++)
{
    if (wall[i].x == leftBlock.x && wall[i].y == leftBlock.y)
    {
        isLeftWall = true;
    }
    if (wall[i].x == rightBlock.x && wall[i].y == rightBlock.y)
    {
        isRightWall = true;
    }
    if (wall[i].x == upBlock.x && wall[i].y == upBlock.y)
    {
        isUpWall = true;
    }
    if (wall[i].x == downBlock.x && wall[i].y == downBlock.y)
    {
        isDownWall = true;
    }
}
```

- 게이트를 기준으로 왼쪽 블록이 벽인지 체크합니다. 만약 벽이라면 true 로 설정합니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

- 오른쪽, 위와 아래도 마찬가지로 벽이라면 true 로 설정합니다.

```

if ((leftBlock.x == 0 && leftBlock.y == 0) || (leftBlock.x == 0 && leftBlock.y == maxHeight - 1))
{
    isLeftWall = true;
}
if ((rightBlock.x == maxWidth - 12 && rightBlock.y == 0) || (rightBlock.x == maxWidth - 12 && rightBlock.y == maxHeight - 1))
{
    isRightWall = true;
}
if ((upBlock.x == 0 && upBlock.y == 0) || (upBlock.x == maxWidth - 12 && upBlock.y == 0))
{
    isUpWall = true;
}
if ((downBlock.x == 0 && downBlock.y == maxHeight - 1) || (downBlock.x == maxWidth - 12 && downBlock.y == maxHeight - 1))
{
    isDownWall = true;
}

```

- 게이트를 기준으로 상하좌우 블록이 Immune Wall 이라도 true 로 설정합니다.

```

if (gate.x == 0)
{
    isLeftWall = true;
    isUpWall = true;
    isDownWall = true;
}
if (gate.x == maxWidth - 12)
{
    isRightWall = true;
    isUpWall = true;
    isDownWall = true;
}
if (gate.y == 0)
{
    isUpWall = true;
    isRightWall = true;
    isLeftWall = true;
}
if (gate.y == maxHeight - 1)
{
    isDownWall = true;
    isRightWall = true;
    isLeftWall = true;
}

```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

- 게이트의 x 좌표가 0 으로 되어 있으면 왼쪽 벽입니다.
- 따라서 왼쪽, 위와 아래는 true 로 입니다.
- 마찬가지로 게이트가 오른쪽 벽, 위쪽 벽, 아래쪽 벽이면 해당하는 벽을 각각 true 로 설정합니다.



```
if (d == 'l')
{
    if (!isLeftWall)
    {
        result = 'l';
    }
    else if (!isUpWall)
    {
        result = 'u';
    }
    else if (!isRightWall)
    {
        result = 'r';
    }
    else
    {
        result = 'd';
    }
}


if (d == 'u')
{
    if (!isUpWall)
    {
        result = 'u';
    }
    else if (!isRightWall)
    {
        result = 'r';
    }
    else if (!isDownWall)
    {
        result = 'd';
    }
    else
    {
        result = 'l';
    }
}
```

```
if (d == 'r')
{
    if (!isRightWall)
    {
        result = 'r';
    }
    else if (!isDownWall)
    {
        result = 'd';
    }
    else if (!isLeftWall)
    {
        result = 'l';
    }
    else
    {
        result = 'u';
    }
}

if (d == 'd')
{
    if (!isDownWall)
    {
        result = 'd';
    }
    else if (!isLeftWall)
    {
        result = 'l';
    }
    else if (!isUpWall)
    {
        result = 'u';
    }
    else
    {
        result = 'r';
    }
}

return result;
}
```

- 들어온 방향에 따라 순서대로 처리를 하게 됩니다.
- 만약 기존 Gate 에서 왼쪽으로 들어왔다면 다른 Gate 에서 왼쪽이 벽인지 확인합니다.
- 벽이 아니라면 왼쪽 방향을 리턴하고 왼쪽이 벽이라면 위, 오른쪽 그리고 아래가 벽인지 차례대로 체크하게 됩니다..
- 방향이 위쪽, 오른쪽, 아래쪽일 때도 마찬가지로 Gate 를 진출하는 우선순위에 따라 확인하고 결과값을 return 합니다.

 <div> 국민대학교 소프트웨어학부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

5단계 : 점수요소의 구현(정지원,(김설) 담당)

```
// 점수 출력
void snakeGame::displayScore()
{
    //Score Board
    move(0, maxWidth - 11);
    printf("Score Board");

    move(1, maxWidth - 9);
    printf("B : (%d)", currentLength);

    move(2, maxWidth - 9);
    printf("+ : (%d)", scoreGrowthItem);

    move(3, maxWidth - 9);
    printf("- : (%d)", scorePoisonItem);

    move(4, maxWidth - 9);
    printf("* : (%d)", scoreSpeedItem);

    move(5, maxWidth - 9);
    printf("G : (%d)", scoreGate);
}
```

- Score Board 에서는 총 5 개를 표현했습니다.
- Snake 현재 길이, 각각 얻은 아이템 개수 그리고 통과한 게이트 개수 입니다.
- 각 점수는 해당하는 변수를 이용했습니다.



```
//Mission
move(7, maxWidth - 11);
printw("Mission");

move(8, maxWidth - 9);
if (requiredLength - currentLength <= 0)
{
    printw("B : (%s)", "v");
}
else
{
    printw("B : (%d)", requiredLength - currentLength);
}

move(9, maxWidth - 9);
if (requiredGrowthItem - scoreGrowthItem <= 0)
{
    printw("+ : (%s)", "v");
}
else
{
    printw("+ : (%d)", requiredGrowthItem - scoreGrowthItem);
}

move(10, maxWidth - 9);
if (requiredPoisonItem - scorePoisonItem <= 0)
{
    printw("- : (%s)", "v");
}
else
{
    printw("- : (%d)", requiredPoisonItem - scorePoisonItem);
}
```

- 다음 단계로 가기 위한 미션 성공 여부는 체크 표시로 표현했습니다.
- 각 조건 별로 요구되는 조건을 만족했다면 체크 표시를 하고 아직 만족을 못했다면 필요한 만큼 표시합니다.
- 이 코드에서는 Snake Length, GrowthItem, PoisonItem 의 조건을 만족해야 합니다.



```
move(11, maxWidth - 9);
if (requiredSpeedItem - scoreSpeedItem <= 0)
{
    printf("* : (%s)", "v");
}
else
{
    printf("* : (%d)", requiredSpeedItem - scoreSpeedItem);
}

move(12, maxWidth - 9);
if (requiredGate - scoreGate <= 0)
{
    printf("G : (%s)", "v");
}
else
{
    printf("G : (%d)", requiredGate - scoreGate);
}
```

- 마찬가지로 SpeedItem 과 Gate 에도 조건 충족 여부를 표시했습니다.



```
// 뱀 그래픽
for (int i = 1; i < 8; i++) {
    start_color();
    init_pair(3, COLOR_WHITE, COLOR_WHITE);
    attron(COLOR_PAIR(3));
    move(14, maxWidth - 9 + i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

for (int i = 1; i < 8; i++) {
    attron(COLOR_PAIR(3));
    move(18, maxWidth - 9 + i);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

for (int i = 1; i < 5; i++) {
    attron(COLOR_PAIR(3));
    move(14 + i, maxWidth - 9);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

for (int i = 1; i < 5; i++) {
    attron(COLOR_PAIR(3));
    move(14 + i, maxWidth - 1);
    addch(edgechar);
    attroff(COLOR_PAIR(3));
}

//뱀 혀
attron(COLOR_PAIR(3));
move(19, maxWidth - 6);
addch(edgechar);
attroff(COLOR_PAIR(3));

attron(COLOR_PAIR(3));
move(20, maxWidth - 7);
addch(edgechar);
attroff(COLOR_PAIR(3));

//뱀 눈
attron(COLOR_PAIR(3));
move(16, maxWidth - 7);
addch(edgechar);
attroff(COLOR_PAIR(3));

attron(COLOR_PAIR(3));
move(16, maxWidth - 5);
addch(edgechar);
attroff(COLOR_PAIR(3));
```

- 기존의 스코어 보드 외에 새로운 뱀 이미지를 추가했습니다. (김설)
- 모양은 뱀이고 색상은 White 를 이용했습니다. (김설)

 <div> 국민대학교 소프트웨어부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

```
//다음 스테이지로 넘어가는 기준 확인
bool snakeGame::nextStage()
{
    if (currentLength >= requiredLength &&
        scoreGrowthItem >= requiredGrowthItem &&
        scorePoisonItem >= requiredPoisonItem &&
        scoreGate >= requiredGate &&
        scoreSpeedItem >= requiredSpeedItem)
    {
        return true;
    }
    return false;
}
```

- 조건을 만족했다면 다음 스테이지로 넘어가도록 구현했습니다.(김설)

작성요령 (30 점)

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.

2.2.3 활용/개발된 기술

1. NCURSES 라이브러리

NCURSES 는 텍스트 기반의 유저 인터페이스를 생성하는 데 사용되는 프로그래밍 라이브러리입니다. 이 프로젝트에서는 NCURSES 를 활용하여 동적으로 변하는 게임 환경을 구현하고, 사용자 입력을 실시간으로 처리했습니다. 특히, `initscr()`, `noecho()`, `keypad()`, `printw()` 등의 함수들을 사용하여 게임 맵을 그릴수 있고, 키 입력에 따른 스네이크의 움직임을 제어했습니다.

2. STL (Standard Template Library)

C++에서 제공하는 표준 템플릿 라이브러리, STL 은 다양한 자료 구조와 알고리즘을 제공합니다. 이 프로젝트에서는 STL 의 `vector` 컨테이너를 사용하여 스네이크의 몸체를 나타내는 각 `segment` 의 위치를 저장하고 관리했습니다. `vector` 를 사용함으로써 동적으로 크기가 변하는 스네이크를 구현하였으며, 요소의 추가와 삭제가 자주 발생하는 게임 동작에 효율적으로 대응할 수 있었습니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

3. 추가 기능

3-1) snake 조작 키 설정

기존처럼 상하좌우 방향키를 사용한 이동이 아니라 사용자가 직접 키를 설정할 수 있게 했습니다. 게임이 시작되기 전 화면에 오른쪽, 왼쪽, 위, 아래의 키보드 입력을 받아 사용자로 하여금 방향키를 설정 할 수 있습니다..

3-2) speedItem(*)

Snake 의 이동 속도를 증가하게 해주는 speedItem(*) 추가적으로 구현하였습니다. 해당 아이템은 '*'로 표현하였습니다. level 이 증가할 때 마다 더 많은 아이템을 먹어야 클리어가 됩니다. 미션을 클리어하기 위해서는 더 많은 speedItem 이 필요하고, 스네이크의 이동속도가 빨라지면 세밀한 컨트롤을 하기 쉽지 않기 때문에 level 이 증가할 때 마다 난이도가 올라갑니다.

3-3) 스테이지 디자인

각 스테이지는 사용자 화면의 maxWidth, maxHeight 를 받아 제작하였습니다. 따라서 스테이지의 디자인은 사용자의 터미널 크기에 따라 변할 수 있습니다.

3-4) 스코어 보드 디자인

스코어 보드가 밋밋하다는 얘기를 들어 snake 모양의 그림을 추가하였습니다

2.2.4 현실적 제한 요소 및 그 해결 방안

1단계

복잡한 모양의 벽을 구현하는데 있어서 정확성과 대칭성을 유지하는 것이 어려웠습니다. 벽을 설계할 때, 루프와 조건문을 활용하여 벽의 패턴과 구조를 미리 계산하고 코드에 반복적으로 적용하였습니다. 또한 'ncurses' 라이브러리의 그래픽 기능을 이용하여 시각적으로도 정확하게 위치를 조정하였습니다.

2단계

사용자 입력을 받아 이를 바탕으로 움직임을 제어하는 것이 어려웠습니다. 'ncurses' 라이브러리의 'keypad' 함수를 사용하여 입력을 쉽게 받고, snake의 방향을 결정하는 로직을 클래스 내에 구현하여 반응성을 강화했습니다.

3단계

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

아이템을 먹었을 때 snake의 길이와 속성을 바로 업데이트하는 데 있어서 처리 속도와 정확성 문제가 발생했습니다. 아이템과 snake의 상호작용을 관리하기 위한 별도의 'vector'를 구현하여 아이템 획득 시 snake의 상태 변화를 효과적으로 관리할 수 있도록 설계하였습니다.

4단계

게이트를 통과한 snake가 다른 게이트에서 나올 때, 방향의 일관성을 유지하는 것이 어려웠습니다. 각 게이트 위치와 관련된 출현 방향을 계산하는 'getWarpDirection' 함수를 개발하여, 게이트를 통과하는 순간 snake의 위치와 방향을 동적으로 조정하였습니다.

5단계

다양한 게임 스테이지와 미션을 관리하며 사용자의 성과를 실시간으로 반영하는 점수 시스템 구축이 복잡했습니다. 각 레벨과 미션에 대한 요구사항을 'displayScore' 함수 내에 구현하여, 게임의 상태를 시각적으로 표현할 수 있는 점수판을 개발했습니다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

2.2.5 결과물 목록

파일	역할
main.cpp	게임의 진입점을 제공하며, 게임 루프와 사용자 입력 처리를 관리합니다.
main.h	main.cpp 파일에서 사용하는 함수들의 선언을 포함합니다.
Display.cpp	게임 맵과 스네이크, 아이템 등 게임 오브젝트의 시각적 표현을 담당합니다
Display.h	Display.cpp 의 함수들을 정의하는 헤더 파일입니다.
Gameplay.cpp	스네이크의 움직임, 충돌 검사, 게임 오버 조건 등 게임 로직을 처리합니다
Gameplay.h	Gameplay.cpp 의 함수들을 정의하는 헤더 파일입니다.
Initialization.cpp	게임 시작 시 초기화 과정을 담당하며, 스네이크와 게임 필드 설정을 초기화합니다.
Initialization.h	Initialization.cpp 의 초기화 함수들을 정의합니다.
Items.cpp	게임 내 아이템(성장 아이템, 독 아이템, 속도 아이템)과 게이트의 생성과 관리를 담당합니다.
Items.h	Items.cpp 에서 사용되는 아이템 생성과 관리 함수들의 정의를 포함합니다.
Charposition.cpp	스네이크와 아이템의 위치를 저장하는 데 사용되는 데이터 구조를 정의합니다.
Charposition.h	Charposition.cpp 에서 사용되는 구조체와 함수의 선언을 포함합니다.
Makefile	프로젝트의 빌드를 자동화하며, 소스 파일을 컴파일하여 실행 파일을 생성합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

3 자기평가

팀원	자기평가
김설	<p>프로젝트 전반에 걸친 모든 코딩과 디버깅을 담당했습니다. ncurses 라이브러리를 처음 사용하면서 터미널 기반의 그래픽 처리에 어려움을 겪었습니다. 특히, 입력과 화면 출력 사이의 동기화 문제가 가장 어려웠습니다. 복잡한 게임 로직, 특히 게이트와 아이템의 동적 배치 로직을 설계하고 구현하는 과정에서 많은 시행착오를 겪었습니다.</p> <p>문제 해결 과정에서 동료와의 토론과 온라인 리소스를 활용하며 프로그래밍 능력을 향상시킬 수 있었습니다.</p> <p>프로젝트 초기 단계에서 요구 사항 분석과 설계 단계를 보다 체계적으로 계획했다면, 개발 과정에서 겪은 어려움을 상당 부분 줄일 수 있었을 것이라는 생각이 듭니다. 후 프로젝트에서는 설계 단계의 중요성을 더욱 인식하고 시간을 할애할 계획입니다.</p> <p>처음 써보는 언어와 라이브러리였지만 다양한 실습과 도전을 통해 실력을 향상시킬 수 있는 기회였다고 생각합니다.</p>
정지원	<p>전반적인 개발과 정리를 담당했습니다. 오랜만의 조별과제라 어려움이 있었지만 다 해놓고 보니 뿌듯했습니다. 의미있는 경험이었던것 같습니다.</p>

4 참고 문헌

참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:

번호	종류	제목	출처	발행년도	저자	기타
1	웹페이지	Ncurses 라이브러리	IBM docs: https://www.ibm.com/docs/ko/aix/7.3?topic=concepts-ncurses-library	2023-03-24	IBM	

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

2	웹 페이지	Ncurses Programming HowTo	http://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO	2005-06	Praeep Padala	
3	웹 페이지	리눅스 프로그래머를 위한 가이드	http://www.xevious7.com/linux/lpg_8_3.html	1997-12-09	Euibeom. Hwang & SangEun. Oh	

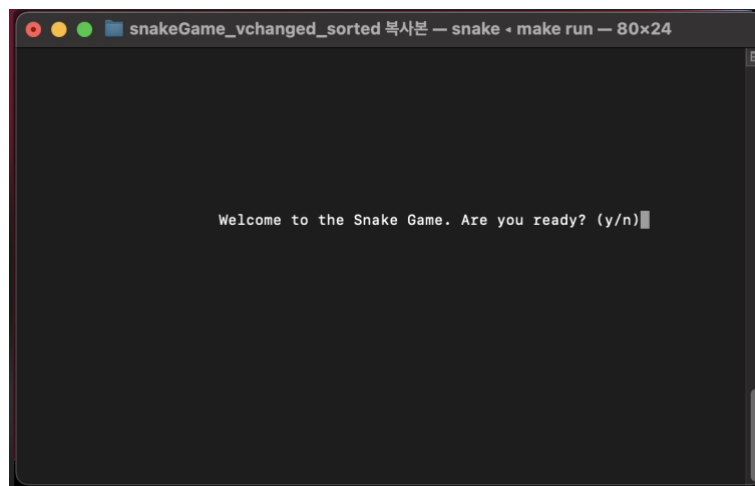
5 부록

작성요령 (15 점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

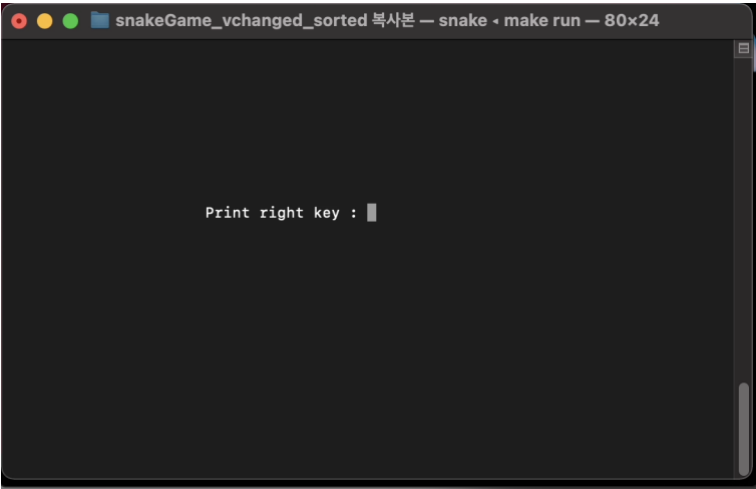
5.1 사용자 매뉴얼

1. 터미널에 `make run` 을 입력하면 다음과 같이 게임이 실행됩니다. y(yes)를 누르면 게임을 시작 할 수 있습니다.



2. 화면의 요구에 따라 상하좌우로 이용할 키를 지정합니다.

 <div> 국민대학교 소프트웨어부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17



 <div> 국민대학교 소프트웨어부 C++프로그래밍 </div>	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	24조	
	Confidential Restricted	Version 1.2	2024-JUN-17

3. 지정한 키를 이용해 게임을 진행합니다.



4. 아이템을 먹고 일정점수 이상을 획득하게 되면 다음 스테이지로 넘어가게 됩니다. 총 4 개의 스테이지가 있으며, 벽에 충돌하지 않고 모든 스테이지를 클리어하면 성공입니다.

5.2 설치 방법

1. 다음의 주소를 인터넷 브라우저 주소창에 입력후 다운로드 합니다.

<https://github.com/conney99/SnakeGame/tree/main>

2. 게임을 설치하고자 하는 디렉토리에 압축 해제합니다.

3.

4. 터미널에서 압축해제한 디렉토리에 접근 후 다음과 같이 입력합니다.

```
cd src
make
```

5. 게임이 실행됩니다.