국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	*스네이크 게임 구현*
팀 명	*C++ 15조*
문서 제목	결과보고서

Version	1.0.0
Date	2023.06.20

팀원	강희구
	손동석

문서 정보 / 수정 내역

Filename	15조_강희구, 손동석_최종보고서.docx
원안작성자	강희구
수정작업자	손동석

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2023-06-3	강희구	1.0	최초 작성	개요 작성
2023-06-15	손동석	1.0	내용 추가	목차 및 개요 2.2.1까지 작성
2023-06-17	강희구	1.0	내용 추가	목차 2.2.2
2023-06-20	손동석	1.0	내용 추가	보고서 최종 완성

목 차

1 개요

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

1.1 개요

2023학년도1학기 'C++프로그래밍' 과목 과제로, SnakeGame 을 구현하는 프로젝트이다.
C++언어를 사용하였고 ncurses 라이브러리를 이용하여 구현했다.

1.2 ncurses 라이브러리 설치방법

Mac 환경
brew install ncurses

Ubuntu 환경
sudo apt-get update
sudo apt-get install libncurses5-dev libncursesw5-dev

2 개발 내용 및 결과물

2.1 목표

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

적용단계	내용	적용 여부
1단계	Map 의 구현	적용
2단계	Snake 표현 및 조작	적용
3단계	Item 요소의 구현	적용
4단계	Gate 요소의 구현	적용
5단계	점수 요소의 구현	적용

<1단계 Map 의 구현 >

42x21의 크기의 맵을 제작한다.

<2단계 Snake 표현 및 조작 >

1단계에서 구현한 map 안에서 키보드의 화살표 방향 키를 입력 받아 snake 가 화살표 입력에 따라 움직이도록 한다.

<3,4단계 items, Gate 요소 구현 >

맵 위에 items, gates 가 출현할 수 있도록 한다.
items 요소는 맵 가장자리가 아닌 맵 내부에 무작위로 출현하고, gates 요소는 맵의 가장 바깥인 Wall 에 출력되도록 한다.(immune wall 은 제외)

<5단계 점수요소 구현>

점수 요소를 구현하고 맵 오른 쪽 화면에 출력되도록 한다.

2.2 개발 내용 및 결과물

2.2.1 개발 내용,

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

1. 초기 화면 설정

- ncurses 라이브러리를 사용하여 게임 창을 초기화한다. 이제 게임을 실행할 준비를 하고, 게임 루프를 시작하여 게임 로직을 구현하고 화면에 그래픽을 업데이트한다.

2. 맵의 구현

-"MakeMap()" 함수로 게임의 맵을 생성하고 그래픽으로 표시하는 역할을 한다. 맵 내부에는 "A"라는 문자로 구성되어 있으며, 맵의 가장자리에는 벽이 있다. 이 함수를 호출하면 게임 창에 맵이 그려진다.

3. 뱀의 초기위치 설정

-"LocateSnake()" 함수는 게임의 현재 스테이지에 따라 뱀의 초기 위치와 방향을 설정하고, 해당 위치에 뱀을 출력하여, 게임 시작 시 뱀을 준비 상태로 만든다.

4. 게임 오버 조건 설정

-"WrongDirection()을 구현히여 잘못된 방향으로 움직일 때 호출되어 게임 오버 상태로 전환하고 오류 메시지를 표시

-BumpedintoWall()함수를 구현하여. 뱀의 머리가 벽에 부딪혔는지 확인하고, 부딪혔을 경우 게임 오버 상태로 전환하고 오류메시지 표시.

-"SizeofSnake() 함수는 뱀의 길이가 짧아서 게임 오버 상태로 처리되고, 이를 사용자에게 알리는 메시지를 출력하는 역할을 수행한다.

-BumpedintoBody() 함수는 뱀이 자신의 몸통에 부딪혔을 때 게임 오버 상태를 처리하는 기능을 구현한 함수.

-"BumpedintoTrap()" 함수는 함정에 부딪혔을 때 게임 오버 처리를 수행하는 함수. 함정에 부딪혔을 때 화면에 게임 오버 메시지를 출력하고, 게임 오버 상태로 전환한다.

ㄴ

5. 스네이크 조작

사용자로부터 입력을 받아 뱀의 움직임을 제어하고, 게임 상태를 업데이트.

6. 게이트 생성

게임 내에 랜덤하게 게이트를 생성하는 기능.

7. 아이템 요소 생성

"RandGrowthItem()" 함수는 게임 내에 랜덤하게 성장 아이템을 생성하는 역할을 한다. 성장 아이템은 벽과 뱀의 위치에 대한 제약 조건을 만족해야 하며, 생성된 아이템은 게임 화면에 표시된다.

RandPoisonItem()" 함수는 게임 내에 랜덤하게 독 아이템을 생성하는 역할을 수행한다. 독 아이템은 벽과 뱀의 위치에 대한 제약 조건을 만족해야 하며, 생성된 아이템은 게임 화면에 표시된다.

8. 아이템 획득 구현

EatGrowthItem() 함수는 뱀의 머리 위치와 growth 아이템의 위치를 비교하여 아이템을 먹었는지 여부를 확인한다. 반환값을 통해 먹은 경우와 먹지 않은 경우를 구분할 수 있다.

EatPoisonItem()" 함수는 뱀의 머리 위치와 poison 아이템의 위치를 비교하여 아이템을 먹었는지 여부를 확인하고, 먹은 경우 뱀의 길이를 조정한다. 반환값을 통해 먹은 경우와 먹지 않은 경우를 구분할 수 있다.

9. 게이트 도착 구현

IsGate() 함수는 뱀의 머리가 게이트에 도달했는지를 확인하고, 게이트에 도달한 경우에는 게이트의 입구와 출구 좌표를 설정하고, 스코어보드를 업데이트한 후 true 를 반환한다. 도달하지 않은 경우에는 false 를 반환한다.

10. 스코어보드 구현

ScoreBoard() 함수는 게임의 점수판을 표시하는 기능을 구현한 함수로, 현재 게임 상황과 미션 목표를 확인하여 출력한다.

10. Trap 생성

RandTrap() 함수는 게임 내에서 무작위로 함정을 생성하는 기능을 구현한 함수이다. 현재 위치에 함정이 이미 존재하거나, 일정한 시간 간격마다 함정을 생성하는 조건을 확인하고 함정을 생성한다. 함정은 벽이 아닌 공간에 위치하며, '@'로 표시되어 게임 화면에 표시된다.

2.2.2 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.

소스파일 : SnakeGame.h , SnakeGame.cpp

1. 초기 화면 설정 startWindow()

1. initscr(): ncurses 모드를 초기화한다. 이 함수를 호출하면 터미널 창이 게임 창으로 설정되며, 텍스트 모드에서 ncurses 기능을 사용할 수 있다.

2. start_color(): 색상을 사용할 수 있도록 ncurses 색상 모드를 활성화한다. 이 함수를 호출하면 색상 기능을 사용할 수 있다.

3. init_pair(): 색상 쌍을 초기화합니다. 색상 쌍은 글자색과 배경색을 결합하여 사용된다. 각 init_pair() 호출은 색상 쌍 번호와 글자색, 배경색을 지정한다.

4. nodelay(stdscr, TRUE): stdscr 윈도우에서 논블로킹(non-blocking) 모드로 설정한다. 이 설정은 입력을 받을 때 프로그램이 멈추지 않고 바로 반환되도록 하여 게임이 지속적으로 실행될 수 있다.

5. noecho(): getch() 함수가 호출될 때 문자를 화면에 표시하지 않도록 설정. 사용자의 입력은 받지만 입력된 문자가 화면에 보이지 않는다

6. curs_set(0): 커서를 숨긴다.

7. cbreak(): 키 입력을 기다리지 않고 바로 처리할 수 있다.

8. keypad(stdscr, TRUE): stdscr 윈도우에서 특수 키(화살표 키, 함수 키 등)를 읽을 수 있도록 키패드 모드를 활성화.

9. getmaxyx(stdscr, vertical, horizontal): 현재 터미널 창의 크기를 가져와 vertical 과 horizontal 변수에 저장.

10. refresh(): 변경된 설정을 적용하여 게임 창을 화면에 업데이트. 이 함수를 호출하지 않으면 변경 사항이 실제 화면에 표시되지 않습니다.

2. 맵의 구현 makeMap()

1. height 와 width 초기화: 맵의 높이와 너비를 설정한다. (21x42)

2. start.y 와 start.x 시작 위치를 설정. (0으로 초기화)

3. 맵 그래픽 출력: 중첩 반복문을 사용하여 맵의 각 좌표에 대해 그래픽을 출력합니다. attron()과 attroff() 함수를 사용하여 색상 쌍을 설정하고, mvwprintw() 함수를 사용하여 문자를 출력. 맵 내부의 각 좌표에 "A"라는 문자를 출력한다.

4. 맵 경계 출력: attron()과 attroff() 함수를 사용하여 색상 쌍을 설정하고, mvwprintw() 함수를 사용하여 맵의 경계에 "x"라는 문자를 출력한다. 이를 통해 맵의 경계를 표시한다.

5. 벽 생성: 중첩반복문을 사용하여 맵의 가장자리에 벽을 생성한다. mvwprintw() 함수를 사용하여 벽을 나타내는 "." 문자를 출력하고, wall 배열과 wallCopy 배열을 사용하여 해당 위치에 벽이 있는지 여부를 저장하여. 뱀이나 item 요소가 벽을 통과하지 못하도록 한다.

6. attroff() 함수를 사용하여 색상 쌍 설정을 해제한다.

3. 뱀의 초기위치 설정 locateSanke()

srand((unsigned int)time(0)): 시간을 기준으로 난수 생성기를 초기화

Randwidth 와 Randheight 변수에 맵 내의 임의의 위치를 설정

int c = rand() % 4: 0부터 3까지의 난수를 생성하여 변수 c 에 저장.

direction 변수를 공백으로 초기화.

c == 0인 경우:

mvwprintw() 함수를 사용하여 뱀의 머리와 몸통을 해당 위치에 출력합니다.

xy 큐에 뱀의 좌표를 순서대로 추가합니다.

direction 변수를 'l'로 설정하여 뱀의 초기 이동 방향을 왼쪽으로 설정.

c == 1인 경우:

mvwprintw() 함수를 사용하여 뱀의 머리와 몸통을 해당 위치에 출력합니다.

xy 큐에 뱀의 좌표를 순서대로 추가합니다.

direction 변수를 'r'로 설정하여 뱀의 초기 이동 방향을 오른쪽으로 설정합니다.

c == 2인 경우:

mvwprintw() 함수를 사용하여 뱀의 머리와 몸통을 해당 위치에 출력합니다.

xy 큐에 뱀의 좌표를 순서대로 추가합니다.

direction 변수를 'u'로 설정하여 뱀의 초기 이동 방향을 위쪽으로 설정합니다.

c == 3인 경우:

mvwprintw() 함수를 사용하여 뱀의 머리와 몸통을 해당 위치에 출력합니다.

xy 큐에 뱀의 좌표를 순서대로 추가합니다.

direction 변수를 'd'로 설정하여 뱀의 초기 이동 방향을 아래쪽으로 설정합니다.

4. 잘못된 방향 설정 wrongDirection()

1. newwin(height, width, start.y, start.x): 새로운 윈도우를 생성하여 변수 win 에 할당. 이 윈도우는 게임 맵의 크기와 시작 위치에 대한 정보를 가지고 있다.
2. refresh(): 터미널 화면을 새로고침.
3. mvwprintw(win, 10, 5, "Game Over!!!\tWrong Direction!!!"): win 윈도우의 지정된 위치에 "Game Over!!! Wrong Direction!!!"라는 오류 메시지를 출력.
4. wrefresh(win): win 윈도우를 새로고침하여 메시지를 표시.
5. beep(): 비프음을 발생시켜 오류를 알린다.
6. usleep(1000000): 1초 동안 일시적으로 프로그램 실행을 멈춤.

7. death = true: death 변수를 true 로 설정하여 게임 오버 상태로 전환.

8. Clear() = 화면을 지운다.

6. 벽에 부딪힌 경우 BumpedintoWall()

1. "head" 변수에 뱀의 머리의 좌표를 설정 .

2. 만약 뱀의 머리가 벽에 부딪혔다면, (wall[head.y][head.x] == 1):

    - 화면을 지우고 초기화.
    - 오버메시지를 출력할 새로운 윈도우 객체 "win"을 생성.
    - "win" 윈도우의 지정된 위치에 "Game Over!!! Bumped into Wall!!" 메시지를 출력.
    - 비프음 출력.
    - "death" 변수를 TRUE 로 설정하여 게임 오버 상태임을 나타냅니다.

6. 스네이크 조작 controlSnake()

1. if(ch = getch()): getch() 함수를 호출하여 사용자의 입력을 가져온다. ch 변수에 입력된 키 코드가 저장.
2. switch(ch): ch 변수의 값을 검사하여 사용자의 입력에 따라 동작을 수행.
3. 각 case 문은 사용자의 입력에 따라 뱀의 방향을 설정하고, 해당 방향으로 뱀을 이동시키는 동작을 수행. 또한, 잘못된 방향으로 움직였을 경우에는 WrongDirection() 함수를 호출하여 게임 오버 상태로 전환.
4. 각 방향(l, r, u, d)에 따라 뱀의 이동을 처리. 해당 방향으로 뱀을 이동시키고, 이동 경로에 대한 조건들을 검사.
    - 머리와 꼬리의 위치를 업데이트하고, 새로운 머리와 꼬리에 대한 출력을 수행.
    - 머리와 꼬리 사이에 몸통이 있는지(IsBody()) 검사하여 충돌 여부를 확인. 충돌 시 BumpedintoBody() 함수를 호출하여 게임 오버 상태로 전환.
    - 성장 아이템을 먹었는지(EatGrowthItem()) 확인하고, growth 아이템을 먹은 경우 뱀의 길이와 점수를 업데이트.
    - IsTrap()여부를 검사하여 충돌 여부를 확인합니다. 충돌 시 BumpedintoTrap() 함수를 호출하여 게임 오버 상태로 전환.
    - poison 아이템을 먹었는지(EatPoisonItem()) 확인하고, poison 아이템을 먹은 경우 뱀의 길이와 점수를 업데이트.
5. 마지막으로 화면을 새로고침(refresh()).

7. 게이트 생성 RandGate()

- 게이트의 새로운 위치를 무한 루프를 통해 랜덤하게 선택.

- gate1과 gate2의 위치가 서로 다른 위치여야 한다.
- 선택된 위치에 뱀의 몸이 없어야 한다.
- 선택된 위치에 원래 있던 벽이 있어야 한다.
- 선택된 위치의 x, y 좌표가 0이 아니어야 한다.

- 선택된 위치에 게이트를 생성. wall 배열을 업데이트하여 해당 위치에 게이트가 있는 것을 표시한다. 게이트를 표시하기 위해 mvwprintw() 함수를 사용하여 화면에 문자를 출력한다. "G" 문자는 게이트를 나타낸다.

8. growth 아이템 생성 RandGrowthItem()

1. 아이템의 이전 위치에 원래 있던 벽이 없는 경우, 해당 위치에 "O" 문자를 출력하여 아이템을 표현. 그렇지 않은 경우, 해당 위치에는 아이템을 그리지 않는다.

2. 무한 루프를 통해 새로운 성장 아이템의 위치를 랜덤하게 선택한다.

    - 선택된 위치에 뱀의 몸이 없어야 한다.
    - 선택된 위치에 원래 있던 벽이 없어야 한다.

3. 선택된 위치에 성장 아이템을 설정한다. growth 구조체의 x 와 y 값을 업데이트하여 새로운 위치를 저장한다. 성장 아이템을 표시하기 위해 mvwprintw() 함수를 사용하여 화면에 문자를 출력한다. "5" 문자는 성장 아이템을 나타냅니다.

9. poison 아이템 생성 RandPoisonItem()

1. 독 아이템의 이전 위치에 원래 있던 벽이 없는 경우, 해당 위치에 "P" 문자를 출력하여 아이템을 표현. 그렇지 않은 경우, 해당 위치에는 아이템을 그리지 않는다.

2. 무한 루프를 통해 새로운 독 아이템의 위치를 랜덤하게 선택한다.

    - 선택된 위치에 뱀의 몸이 없어야 한다.
    - 선택된 위치에 원래 있던 벽이 없어야 한다.
3. 선택된 위치에 독 아이템을 설정한다. poison 구조체의 x 와 y 값을 업데이트하여 새로운 위치를 저장한다. 독 아이템을 표시하기 위해 mvwprintw() 함수를 사용하여 화면에 문자를 출력한다. "6" 문자는 독 아이템을 나타낸다.

10. growth 아이템 획득 EatGrowthItem()

1. 현재 뱀의 머리 위치와 성장 아이템의 위치를 비교한다. 머리 위치는 xy.back().second 와 xy.back().first 로 얻어오며, 성장 아이템의 위치는 growth.x 와 growth.y 로 저장되어 있다.

2. 머리 위치와 성장 아이템의 위치가 일치하는 경우, 뱀이 성장 아이템을 먹은 경우에는 TRUE 를 반환한다.

3. 뱀이 growth 아이템을 먹지 않은 경우에는 FALSE 를 반환한다.

11. poison 아이템 획득 EatPoisonItem()

1. 현재 뱀의 머리 위치와 독 아이템의 위치를 비교한다. 머리 위치는 xy.back().second 와 xy.back().first 로 얻어오며, 독 아이템의 위치는 poison.x 와 poison.y 로 저장되어 있다.

2. 머리 위치와 독 아이템의 위치가 일치하는 경우, 뱀이 독 아이템을 먹은 경우에는 추가적인 조건을 검사합니다.

3. 뱀의 현재 길이가 3보다 작은 경우, 즉 뱀이 최소 길이보다 짧을 때에는 SizeofSnake() 함수를 호출하여 뱀의 길이를 조정한다.

4. 조건에 따라 뱀의 길이가 조정된 경우 TRUE 를 반환한다.

5. 뱀이 독 아이템을 먹지 않은 경우에는 FALSE 를 반환한다.

12. 좌표의 뱀의 몸통 여부 확인 IsBody(rand1, rand2)

1. 주어진 좌표인 (rand1, rand2)와 뱀의 몸통의 각 부분을 비교한다. 뱀의 몸통은 xy 큐에 저장되어 있으며, xy.front().second 와 xy.front().first 로 각 몸통 부분의 좌표를 얻어온다.

2. 현재 몸통 부분의 좌표가 주어진 좌표와 일치하는 경우, 즉 뱀의 몸통 부분이 주어진 좌표와 겹치는 경우에는 TRUE 를 반환한다.

3. 그렇지 않은 경우, 현재 몸통 부분의 좌표가 주어진 좌표와 일치하지 않는 경우에는 xy 큐에서 해당 부분을 제거하고 다음 몸통 부분을 확인하기 위해 xy.pop()을 호출한다.

4. 뱀의 몸통과 주어진 좌표가 겹치지 않는 경우에는 FALSE 를 반환.

13. 뱀의 크기가 3보다 작은 경우 처리 SizeOfSnake()

1. 화면을 지우고, 새로운 윈도우 객체를 생성한다. 이 때, 윈도우의 크기는 게임 화면의 높이와 너비, 시작 위치를 기반으로 한다.

2. 생성한 윈도우 객체에 "Game Over!!! Too Short!!!"라는 메시지를 출력한다. 이 때, 메시지의 위치는 (10, 5)로 설정한다

3. 윈도우를 갱신하여 메시지가 화면에 표시되도록 한다.

4. 소리를 발생시키고, 1초 동안 대기한다.

5. "death" 변수를 TRUE 로 설정하여 게임 오버 상태로 표시한다.

14. Gate 도착 여부 확인 IsGate()

1. 만약 뱀의 머리가 "gate1"의 좌표와 일치하는 경우를 확인. 뱀의 머리의 x 좌표가 "gate1.x"와 같고, y 좌표가 "gate1.y"와 같은 경우.

    - "in" 변수의 x 좌표를 "gate1.x"로 설정.
    - "in" 변수의 y 좌표를 "gate1.y"로 설정.
    - "out" 변수의 x 좌표를 "gate2.x"로 설정.
    - "out" 변수의 y 좌표를 "gate2.y"로 설정.
    - "num_gate" 변수를 증가.
    - "ScoreBoard()" 함수를 호출하여 스코어보드를 업데이트.
    - true 를 반환.
2. 뱀의 머리가 "gate2"의 좌표와 일치하는 경우를 확인합니다. 즉, 뱀의 머리의 x 좌표가 "gate2.x"와 같고, y 좌표가 "gate2.y"와 같은 경우.

    - "in" 변수의 x 좌표를 "gate2.x"로 설정.
    - "in" 변수의 y 좌표를 "gate2.y"로 설정.
    - "out" 변수의 x 좌표를 "gate1.x"로 설정.
    - "out" 변수의 y 좌표를 "gate1.y"로 설정.
    - "num_gate" 변수를 증가.
    - "ScoreBoard()" 함수를 호출하여 스코어보드를 업데이트.
    - true 를 반환.
3. 위의 두 경우에 해당하지 않는 경우, false 를 반환.

15. Gate 입출구 설정, 뱀의 게이트 통과 ControlGate()

1. "IsGate()" 함수를 호출하여 뱀의 머리가 게이트에 도달했는지 확인 후. 도달한 경우에만 아래의 코드를 실행.

2. 뱀의 몸통 좌표를 순회하며 뱀의 꼬리 부분을 업데이트한다. 현재 뱀의 머리의 좌표를 가져온 후, "xy" 큐에서 해당 좌표를 제거한 다음, 다시 큐에 삽입한다.

3. "xy" 큐에서 뱀의 꼬리 부분을 한 번 더 제거합니다.

4. "in" 좌표에 대한 출력을 "G"로 갱신.

5. "IsEdgeGate()" 함수를 호출하여 게이트가 벽에 붙어 있는지 확인 후, 붙어 있는 경우 아래의 코드 블록을 실행한다.

    - 현재 방향에 따라 "out" 좌표를 업데이트.
    - 해당 방향으로 "out" 좌표를 한 칸 이동한 위치를 "xy" 큐에 추가.

16. 스네이크 몸에 충돌 BumpedintoBody()

1. 화면을 지우고 새로운 창을 생성. 창의 크기와 위치는 게임의 높이(height), 너비(width), 시작 위치(start.y, start.x)로 설정.

2. 생성한 창에 "Game Over!!! Bumped into Body!!!"라는 메시지를 출력. 메시지의 위치는 (10, 5)로 지정.

3. "beep()" 함수를 호출하여 비프음 출력.

4. "usleep()" 함수를 사용하여 1초 동안 프로그램 실행을 일시 중지.

5. "death" 변수를 TRUE 로 설정하여 게임 오버 상태로 표시.

17. 스코어 보드 ScoreBoard()

1. "Score Board"라는 제목과 점수 항목들을 출력. 각 항목은 라벨과 콜론(:)으로 구분한다.
2. "snake" 항목은 파란색으로 표시.
3. "gate" 항목은 검정색으로 표시.
4. "wall" 항목은 회색으로 표시.
5. "poison" 항목은 보라색으로 표시.
6. "growth" 항목은 초록색으로 표시.
7. "Trap" 항목은 노란색으로 표시.
8. 각 항목의 값을 문자열로 변환하여 출력한다.

18. Trap 생성 RandTrap()

1. 만약 현재 위치에 이미 함정이 존재하거나, 일정한 시간 간격 함정을 생성해야 하는 경우, 함정을 표시한다. 단, 함정이 위치할 수 있는 공간은 벽이 아니다. 함정은 'Q'로 표시됩니다.
2. 무작위로 함정의 위치를 생성한다. 생성된 위치가 뱀의 몸통이나 벽이 아니면서 함정이 위치할 수 있는 공간이어야 한다.
3. 생성된 함정의 위치에 '@'을 표시합니다. 함정은 빨간색으로 표시된다.

18. Trap 도착 여부 확인 IsTrap()

1. 뱀의 머리 좌표와 함정 좌표를 비교한다.
2. 뱀의 머리 좌표와 함정 좌표가 일치하면 TRUE 를 반환한다. 그렇지 않으면 FALSE 를 반환한다.

2.2.3 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.

<Ncurses> : GUI 기반 게임 구현에 이용

<queue> : 스네이크의 움직임을 표현할 때 queue.pop, push 를 사용하여 구현함.

<Ctime> : random seed 에 time(0)사용

<unistd.h> : Usleep()함수로 게임 오버시 딜레이 구현

2.2.4 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.

2.2.5 결과물 목록

작성요령 (5점)

결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.

Makefile : 컴파일 용도 make 파일

main.cpp : SnakeGame 실행파일

SnakeGame.h : Snakeame 클래스의 선언과 멤버변수와 함수 선언

SnakeGame.cpp : SnakeGame 클래스의 멤버함수 구현

3 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.

강희구 : C++을 통해 팀 프로젝트로 SnakeGame 을 구현해보며 Ncurses 라는 새로운 기능을 사용해본 의미있는 경험이었습니다. 한 학기 동안 배운STL, 배열 기능들을 실전해서 적용해보니 수업 시간 실습 때보다 사용하는데 어려움이 많았었는데, 이번 과제를 통해 사용법을 익힐 수 있었습니다. 개선점으로는 스코어 보드 업데이트가 생각처럼 되지 않아서 스코어보드 업데이트 부분에 개선이 필요합니다.

손동석 : 학기 중 수업에서 정말 많은 것들을 배울 수 있었는데, 직접 배운 것들을 적용해서 해보다 보니, 어려운 부분이 정말 많았습니다. 혼자 만들었다면 정말 버거웠을 만한 것들이었는데 팀원과 함께 개발을 하면서 실질적으로 많은 것을 배울 수 있었습니다. 특히 프론트엔드에 해당하는 UI 작성 부분에서 가장 많은 시간이 소요되었고 아직도 모르는 것이 더 많다고 생각합니다. 아직 수많은 버그가 산재해있고 기능들에 오류가 많으며, 구현하지 못한 부분들이 많아 시간이 좀 더 있었다면 더 나은 게임을 만들 수 있었을 것이라는 아쉬움이 남습니다.

4 참고 문헌

참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:

번호	종류	제목	출처	발행년도	저자	기타
1.	영상물	Making Snake in Ncurses – Tutorial 0 – What We will be Making (POC)	https://www.youtube.com/watch?v=MH6QIYJ2SwU&list=PL2U2TQ__OrQ_TV2-wuHqGaK8qlnxgKUvK	2021	Casual Coder	

5 부록

> 작성요령 (15점)
> 프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

사용자 매뉴얼

Snake : ■

Growth Item : ■

- 먹을 경우 뱀의 길이가 1 증가한다.

Poison Item : ■

- 먹을 경우 뱀의 길이가 1 감소한다.

Trap : ■

- 먹을 경우 게임이 실패한다.

Gate : G

- 진입할 경우 규칙에 따라 다음 게이트에서 빠져나온다.

Wall : ■

5.1 설치 방법

1. 하나의 같은 디렉토리에 makefile, main.cpp, SnakeGame.h, SnakeGame.cpp 를 저장하고 make 파일을 실행하여 컴파일하고 실행파일을 생성한다.

2. 터미널에서 make run 명령어를 실행하여 시작한다.

명령어 :

$ make

$ make run