C++ Programming Project

- Push Box Game -

소프트웨어학부
20181690 정유선
20181710 황예은

Index

1. 요구사항 분석

- 사용자 인터페이스 요구사항

{4, 4, 4, 4, 4, 4, 4}
{1, 1, 1, 1, 1, 4, 4}
{1, 0, 0, 0, 1, 4, 4}
{1, 3, 3, 3, 1, 4, 4}
{1, 2, 2, 2, 1, 1, 4}
{1, 0, 0, 0, 0, 1, 4}
{1, 0, 7, 0, 0, 1, 4}
{1, 1, 1, 1, 1, 1, 4}
{4, 4, 4, 4, 4, 4, 4}

왼쪽과 같은 2차원 배열이 주어졌을 때, game의 map을 구성한다.

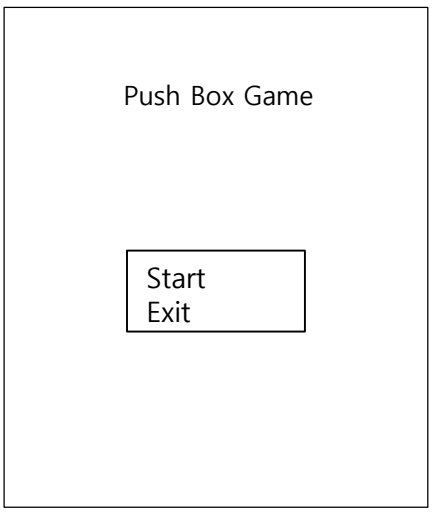
이때, 0은 빈 부분, 1은 벽, 2는 상자, 3은 목적지, 4는 바깥부분, 그리고 7은 캐릭터로 나타낼 것이다.

- 기능적 요구사항

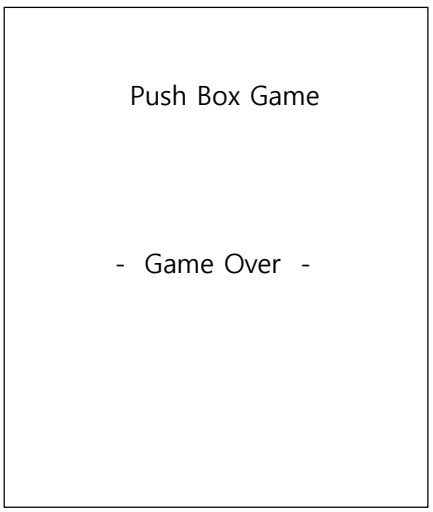
1) 방향키를 입력 받아 캐릭터가 움직일 수 있도록 한다.
2) 캐릭터의 이동방향에 벽이 있거나 상자가 두 개 이상 연속으로 있으면, 캐릭터가 못 움직이게 한다.
3) 이동방향이 빈 부분이거나, 상자가 한 개 있으면서 그 상자 반대편이 다른 상자나 벽에 막혀있지 않다면 캐릭터는 움직일 수 있다. 상자는 캐릭터와 함께 이동 방향으로 한 칸 이동해야 할 수 있다.
4) 캐릭터가 이동한 회수 step과 상자가 움직인 횟수 push를 구현한다.

- 추가 구현사항

- game 이 실행되어 map이 있는 화면이 나오기 전, 게임을 Start할지 Exit 정하는 시작화면과, game이 종료되었음을 알려주는 종료화면을 아래와 같이 추가할 것이다.
- Back Space 키를 누르면 게임 종료화면이 나타나게 할 것이다.



(그림1) 시작화면 구상도



(그림2) 종료화면 구상도

2. 구조설계서

- 파일 설계

project.h

기능
• 모든 헤더파일들을 include한 파일

setting.cpp / setting.h

method	type	parameter	기능
Setting	void	-	• ncurses의 팔레트(색 속성) 지정 • 시작, 게임, 종료 화면의 기본 틀 설정

window.cpp / window.h

Class	method	type	parameter	기능
Window	getwin * (getwin으로 시작하는 모든 메소드)	WINDOW*	-	• 지정된 window 리턴
	print _ level	void	WINDOW *win_level, const int level	• level 표시할 window 구성
	print _ push	void	WINDOW *win_push, const Int push	• push 표시할 window 구성
	print _ step	void	WINDOW *win_step, const int step	• step 표시할 window 구성
	print _ board	void	WINDOW *win_board, int (*pushbox)[9][7], int *r, int *c	• 2차원 배열의 숫자를 for문과 유니코드를 사용하여 벽, 상자, 목적지, 캐릭터, 장애물로 나타냄
	print _ end	void	WINDOW *win_end	• level 표시할 window 구성
	print _close	void	WINDOW *win	• widow 닫음

🔎 startend.cpp / startend.h

Class	method	type	parameter	return	기능
Start	StartHighlight	int	WINDOW *win_start	choice	• 시작화면에서 menu 선택할 때, key가 가리키고 있는 선택지 강조할 때 기능 구현
	print_menu	void	WINDOW *win_start, Int highlight	-	• 메뉴 선택할 때 강조되는 기능 구현

🔎 game.cpp / game.h

Class	method	type	parameter	return	기능
Game	moveCharacter	int	int (*pushbox)[9][7], int (*goal), int *r, int *c, int *push, int key	success (박스 목적지까지 도착 성공한 개수)	• 방향키를 입력 받고 기능적 요구사항에 맞게 character가 box를 움직이게 함.
	select_goal	Int*	int numOfGoal, int (*pushbox)[9][7]	Goal	• 목적지 위치 설정.

🔎 PushBoxGame.cpp

type	기능
int	• "project.h"를 include하여 Push Box Game 설계 • map을 구성할 배열 정의 • 목적지의 개수 numOfGoal 정의 • main함수 구현

- 주요 코드

- window.cpp의 Window::print_board메소드

```
49  void Window::print_board(WINDOW *win_board, int (*pushbox)[9][7], int *r, int *c)
50  {
51          for (int i = 0; i < 9; i++)
52          {
53                  for (int j = 0; j < 7; j++)
54                  {
55                          if ((*pushbox)[i][j] == 0)
56                          {
57                                  wattron(win_board, COLOR_PAIR(3));
58                                  mvwprintw(win_board, i, j, "u2B1A");
59                                  wattroff(win_board, COLOR_PAIR(3));
60                          }
61
62                          else if ((*pushbox)[i][j] == 1)
63                          {
64                                  wattron(win_board, COLOR_PAIR(4));
65                                  mvwprintw(win_board, i, j, "\u25A9");
66                                  wattroff(win_board, COLOR_PAIR(4));
67                          }
68
69                          else if ((*pushbox)[i][j] == 2)
70                                  mvwprintw(win_board, i, j, "\u2BBD");
71
72                          else if ((*pushbox)[i][j] == 3)
73                                  mvwprintw(win_board, i, j, "\u2B1A");
74
75                          else if ((*pushbox)[i][j] == 4)
76                          {
77                                  wattron(win_board, COLOR_PAIR(3));
78                                  mvwprintw(win_board, i, j, "\u2B1A");
79                                  wattroff(win_board, COLOR_PAIR(3));
80                          }
81
82                          else if ((*pushbox)[i][j] == 7)
83                          {
84                                  *r = i;
85                                  *c = j;
86                                  wattron(win_board, COLOR_PAIR(5));
87                                  mvwprintw(win_board, i, j, "\u2606");
88                                  wattroff(win_board, COLOR_PAIR(5));
89                          }
90                  }
91          }
92          wrefresh(win_board);
93  }
```

- 이중 for문으로 0: 빈 부분, 1 : 벽, 2 : 상자, 3 :목적지, 4 : 바깥부분, 7 : 캐릭터 를 유니코드로 구현

▩ Unicode number: U+25A9 벽(1)

☒ Unicode number: U+2BBD 상자(2)

⬚ Unicode number: U+2B1A 빈 공간(0) 목적지(3) 바깥부분(4)

☆ Unicode number: U+2606 캐릭터(7)

- game.cpp의 moveCharacter 메소드 중 case “KEY_BACKSPACE”와 “KEY_UP”

```
9   int Game::moveCharacter(int (*pushbox)[9][7], int *goal, int *r, int *c, int *push, int key, int num)
10  {
11          // 0: 빈부분, 1 : 벽, 2 : 상자, 3 :목적지, 4 : 바깥부분, 7 : 캐릭터
12          switch(key)
13          {
14                  // 게임 탈출(backspace)
15                  case KEY_BACKSPACE:
16                          return -2;
17                  case KEY_UP:
18                  // 이동방향에 벽이 있거나
19                          if (((*pushbox)[*r-1][*c] == 1) || ((*pushbox)[*r-1][*c] == 2 && (*pushbox)[*r-2][*c] == 1))
20                                  return -1;
21                          // 상자가 2개 이상
22                          if (((*pushbox)[*r-1][*c] == 2) && ((*pushbox)[*r-2][*c] == 2))
23                                  return -1;
24                          // 이동방향에 상자, 상자 앞이 빈부분이거나, 목적지
25                          if ((*pushbox)[*r-1][*c] == 2 && ((*pushbox)[*r-2][*c] == 0 || (*pushbox)[*r-2][*c] == 3))
26                          {
27                                  (*pushbox)[*r-2][*c] = 2;
28                                  (*push)++;
29                                  (*pushbox)[*r-1][*c] = 7;
30                                  (*pushbox)[*r][*c] = 0;
31                                  (*r)--;
32                          }
33                          // 이동방향이 빈부분이거나 목적지
34                          if(((*pushbox)[*r-1][*c] == 0) || ((*pushbox)[*r-1][*c] == 3))
35                          {
36                                  (*pushbox)[*r][*c] = 0;
37                                  (*pushbox)[*r-1][*c] = 7;
38                                  (*r)--;
39                          }
40                          break;
```

(Line 15) BACKSPACE키를 눌렀을 때의 case를 설정해 언제든 게임을 끝낼 수 있게 한다. 이때 -2를 리턴하는 이유는 후에 나온다.

(Line 17) 방향 key가 눌렸을 때, 요구사항에 맞추어 상자가 밀리는 경우와 그렇지 않은 경우를 나누었다. 이 작업을 key_down, key_left, key_right에 동일하게 적용한다.

- game.cpp의 select_goal메소드

```
133 // 목적지 위치 설정
134 int* Game::select_goal(int numOfGoal, int (*pushbox)[9][7], int *num)
135 {
136         int n = 0;
137         int *Goal;
138
139         // 동적할당 배열 초기화
140         Goal = new int[numOfGoal * 2];
141         for (int i = 0; i < 9; i++)
142         {       for (int j = 0; j < 7; j++)
143                 {
144                         // 목적지 위치 Goal 배열에 저장
145                         if ((*pushbox)[i][j] == 3)
146                         {
147                                 Goal[n] = i;
148                                 Goal[n+1] = j;
149                                 n += 2;
150                         }
151                 }
152         }
153         // Goal 크기
154         *num = n;
155
156         return Goal;
157 }
```

- startend.cpp 중 Start::print_start와 Start::print_menu메소드

```
Start::Start()
{
startx = 0;
starty = 0;
choices[0] = "Start";
choices[1] = "Exit";
n_choices = sizeof(choices) / sizeof(char *);
}

int Start::print_start(WINDOW *win_start)
{       int highlight = 1;
        int choice = 0;
        int key;
        print_menu(win_start, highlight);

        while(1)
        {       key = wgetch(win_start);
                switch(key)
                {       case KEY_UP:
                                if(highlight == 1)
                                        highlight = n_choices;
                                else
                                        --highlight;
                                break;
                        case KEY_DOWN:
                                if(highlight == n_choices)
                                        highlight = 1;
                                else
                                        ++highlight;
                                break;
                        case 10:
                                choice = highlight;
                                break;
                        default:
                                refresh();
                                break;
                }
                print_menu(win_start, highlight);
                if(choice != 0)
                        break;
        }
        clrtoeol();
        endwin();
        return choice;
}
```

```
void Start::print_menu(WINDOW *win_start, int highlight)
{
        int x, y, i;

        x = 2;
        y = 2;
        box(win_start, 0, 0);
        for(i = 0; i < n_choices; ++i)
        {
                if(highlight == i + 1) /* High light the present choice */
                {       wattron(win_start, A_REVERSE);
                        mvwprintw(win_start, y, x, "%s", choices[i]);
                        wattroff(win_start, A_REVERSE);
                }
                else
                        mvwprintw(win_start, y, x, "%s", choices[i]);
                ++y;
        }
        wrefresh(win_start);
}
```

- Start클래스의 두 메소드를 이용해, 시작화면의 menu에서 “UP key”와 “Down key”로 선택지를 고를 때, 선택된 부분에 highlight를 넣어준다.

- **PushBoxGame.cpp 의 main 중 while문**

```
94          while(1)
95          {
96                  // 화면 초기화
97                  w.print_board(win_board, pushbox, &r, &c);
98                  w.print_level(win_level, level);
99                  werase(win_push);
100                 w.print_push(win_push, push);
101                 werase(win_step);
102                 w.print_step(win_step, step);
103
104                 // 게임 진행
105                 key = wgetch(win_board);
106                 success = g.moveCharacter(pushbox, goal, &r, &c, &push, key, num);
107                 step++;
108
109                 // 게임 탈출
110                 if (success == -2) break;
111
112                 // 게임 성공
113                 if (success == numOfGoal[level])
114                 {
115                         level++;
116                         step = 1;
117                         push = 0;
118                         pushbox = &pushBox[level];
119                         goal = g.select_goal(numOfGoal[level], pushbox, &num);
120
121                         if (level == 3)
122                         {
123                                 endwin();
124                                 break;
125                         }
126                 }
127         }
```

- 게임화면을 구성해 주는 부분으로, success (박스가 목적지까지 도착 성공한 개수) 에 game.cpp의 moveCharacter함수를 이용해 게임의 상황을 저장한다.
- success 가 -2인 경우 loop문을 빠져나오는데, 이게 앞서 p.4에서 Backspace를 눌렀을 때 -2를 리턴하는 이유이다.

3. Make File

- Make File 코드

```
1   CC=g++
2   CPPFLAGS=-W -Wall
3
4   all:project
5
6   project:setting.o game.o window.o startend.o PushBoxGame.o
7           $(CC) $(CPPFLAGS) -o project setting.o game.o window.o startend.o PushBoxGame.o -lncursesw
8   setting.o:setting.cpp
9           $(CC) $(CPPFLAGS) -c -o setting.o setting.cpp
10  game.o:game.cpp
11          $(CC) $(CPPFLAGS) -c -o game.o game.cpp
12  window.o:window.cpp
13          $(CC) $(CPPFLAGS) -c -o window.o window.cpp
14  startend.o:startend.cpp
15          $(CC) $(CPPFLAGS) -c -o startend.o startend.cpp
16  PushBoxGame.o:PushBoxGame.cpp
17          $(CC) $(CPPFLAGS) -c -o PushBoxGame.o PushBoxGame.cpp
18
19  clean:
20          rm -rf *.o
```

- (Line1) g++을 CC 매크로로 정의
- (Line2) -W -Wall을 CPPFLAGS로 정의

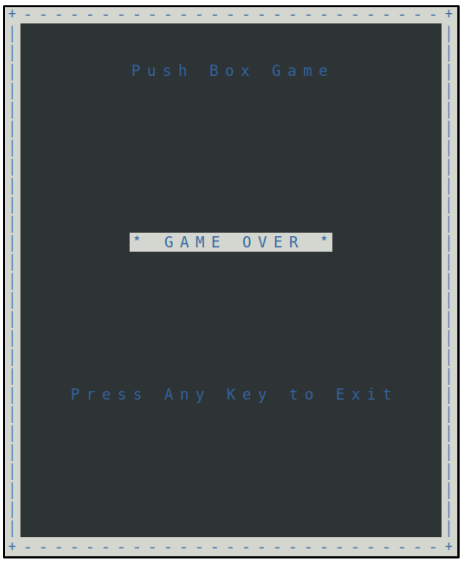
```
yoosun@yoosun-ThinkPad-T470:~/Desktop/newr$ make
g++ -W -Wall -c -o setting.o setting.cpp
g++ -W -Wall -c -o game.o game.cpp
g++ -W -Wall -c -o window.o window.cpp
window.cpp: In member function 'void Window::print_end(WINDOW*)':
window.cpp:95:32: warning: unused parameter 'win_end' [-Wunused-parameter]
 void Window::print_end(WINDOW *win_end)
                                ^~~~~~~
g++ -W -Wall -c -o startend.o startend.cpp
g++ -W -Wall -c -o PushBoxGame.o PushBoxGame.cpp
g++ -W -Wall -o project setting.o game.o window.o startend.o PushBoxGame.o -lncursesw
```

(그림3) Make File 실행 터미널 화면

4. 실행화면

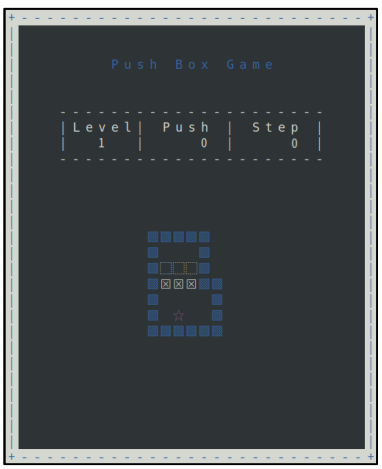
(그림4) 시작화면

- Make file 실행 시 뜨는 첫 화면이다.
- 키보드 Up, Down로 “Start”와 “Exit”를 선택할 수 있으며, 선택된 사항이 흰색으로 강조된다.

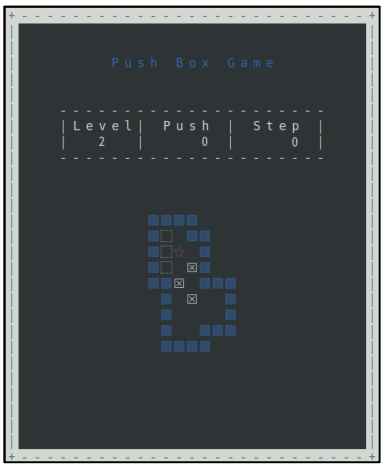


(그림5) 종료화면

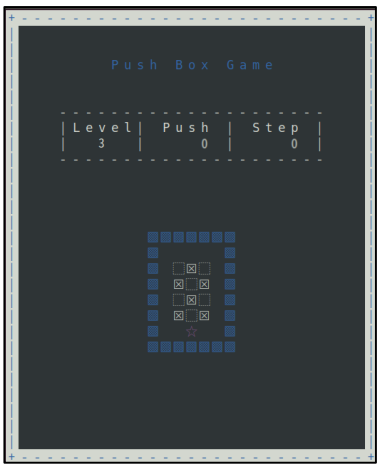
- Level 3까지 게임을 완료하거나, Backspace 키를 누르면 나오는 종료화면이다.



(그림6) level 1



(그림7) level 2



(그림8) level 3