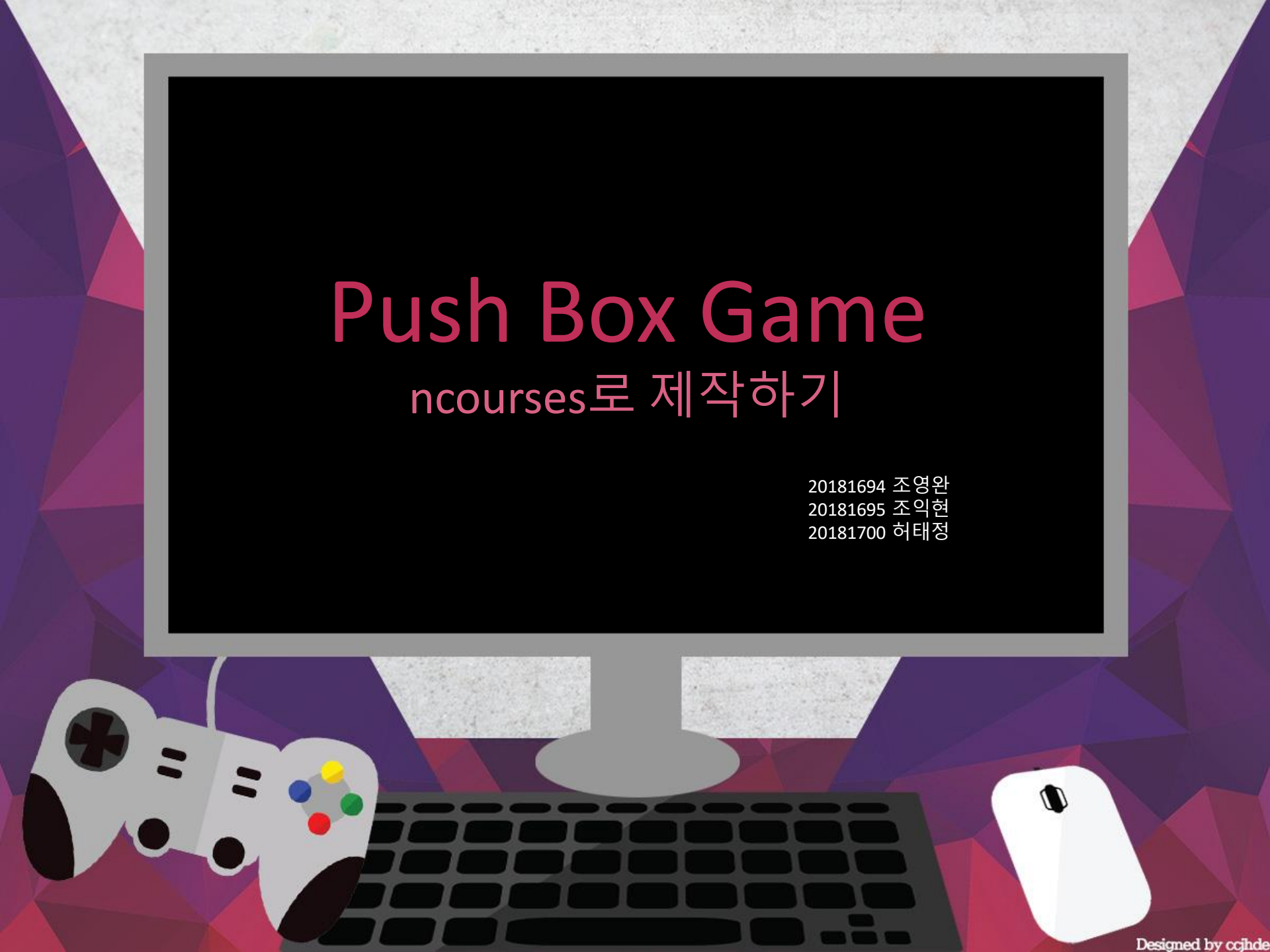
Push Box Game

ncourses로 제작하기

20181694 조영완
20181695 조익현
20181700 허태정



1 게임 기획

- 게임 설명
- 제작 기획

2 제작

- gameobject.cpp
- game.cpp
- main.cpp
- Makefile

3 완성

- 시작 페이지
- 게임 페이지
- 추가 기능 구현

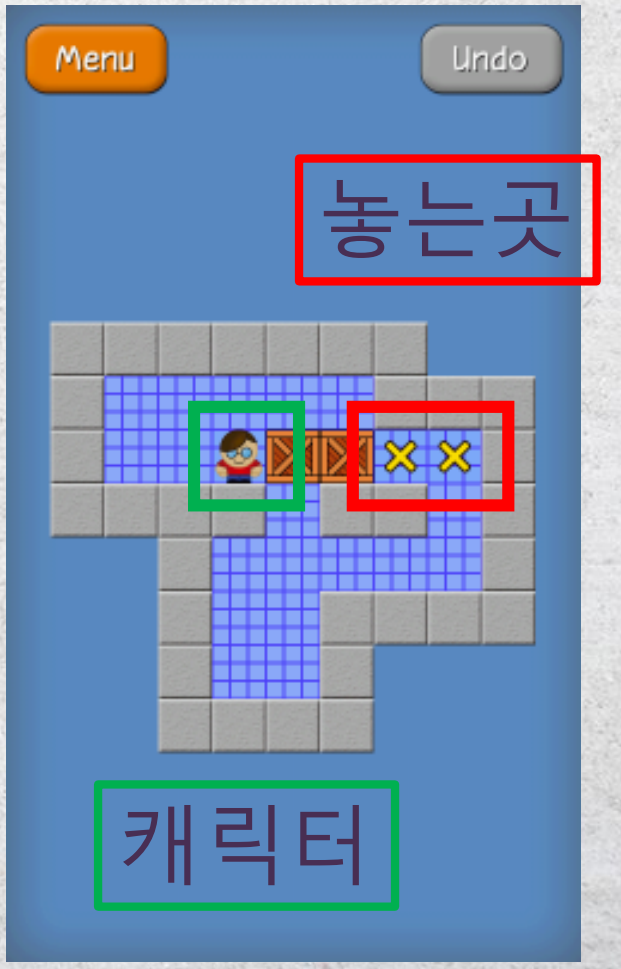
Designed by ccjhde

게임 설명

PushBoxGame이란?

게임 캐릭터가 상자를 밀어
모든 상자를 주어진 위치에
놓이게 만드는 게임!!

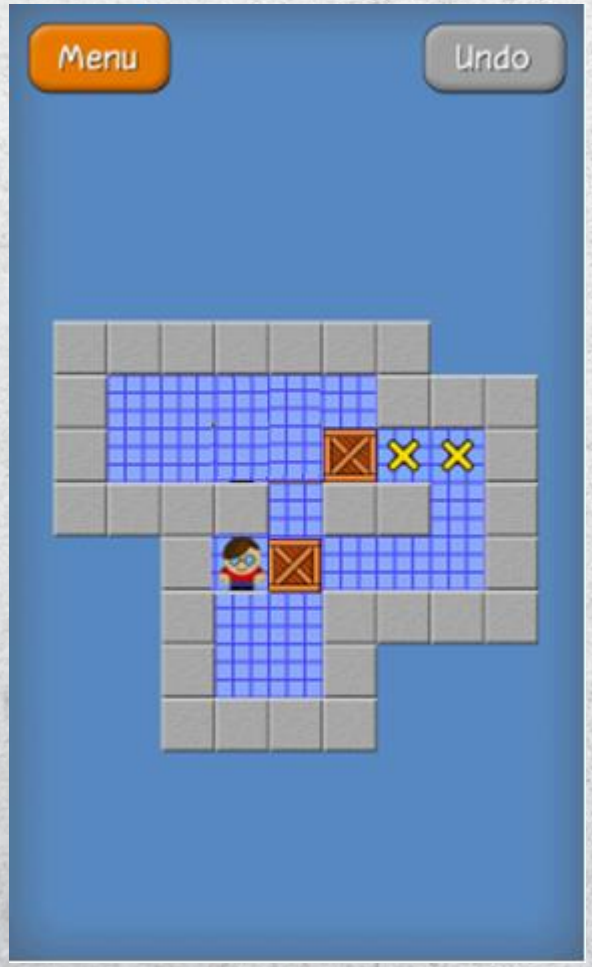
PushBoxGme – 게임 기획



게임 규칙

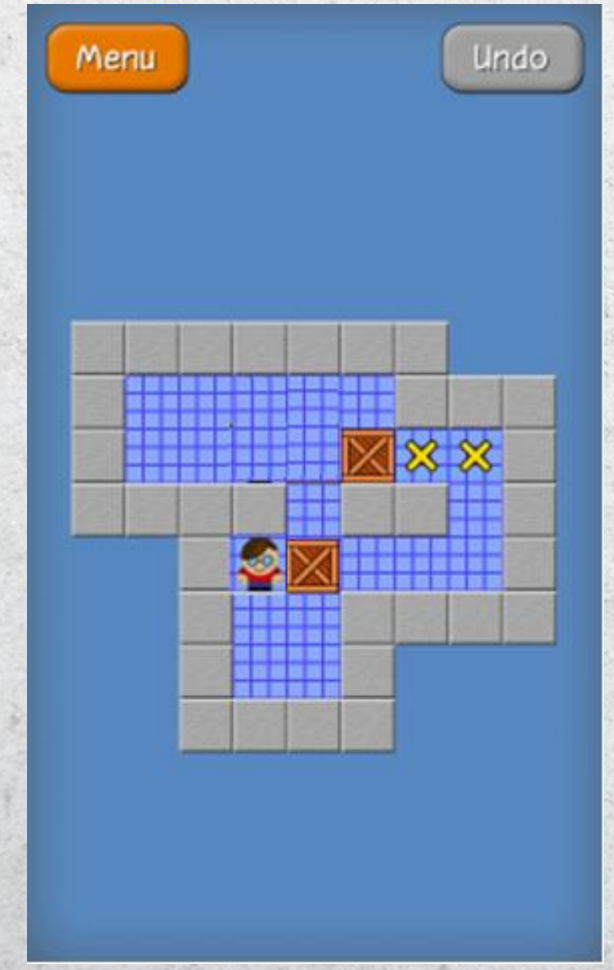
- 움직일 수 있는 경우
- 움직일 수 없는 경우
- 성공

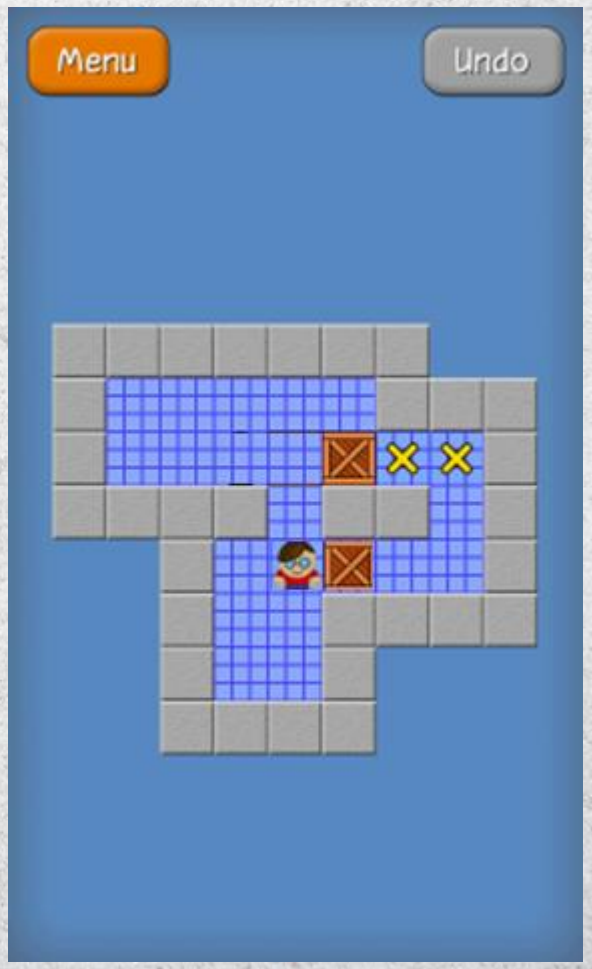
움직일 수 있는 경우



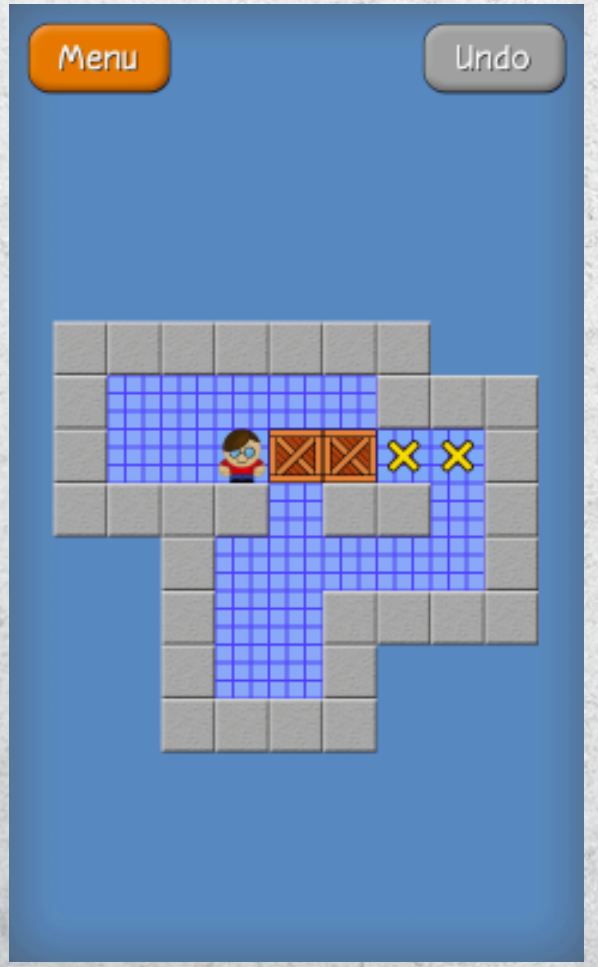
상자 하나만 이동 경로에 있는 경우
-> 상자도 같이 움직일 수 있다

움직임



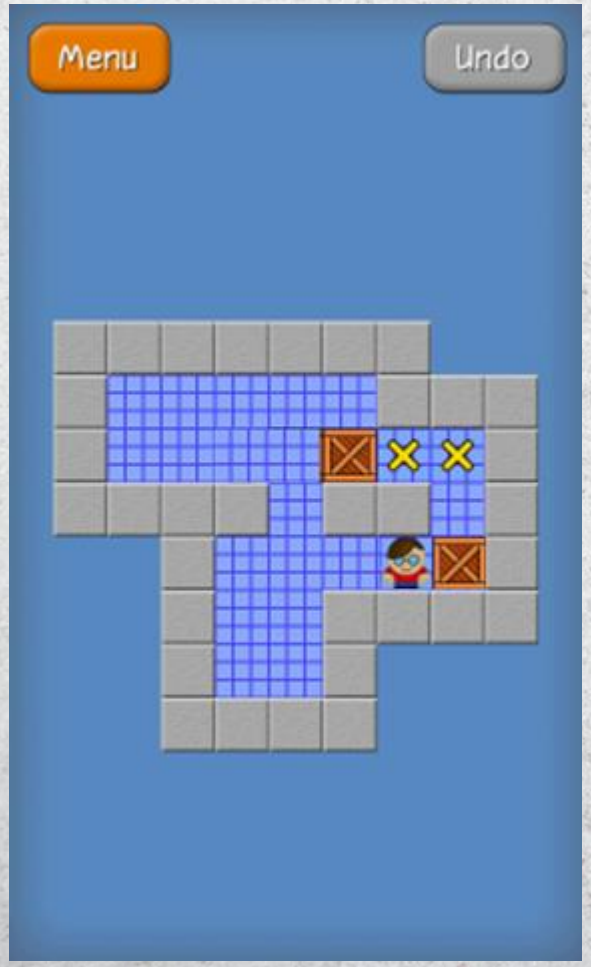


움직일 수 없는 경우 (1)



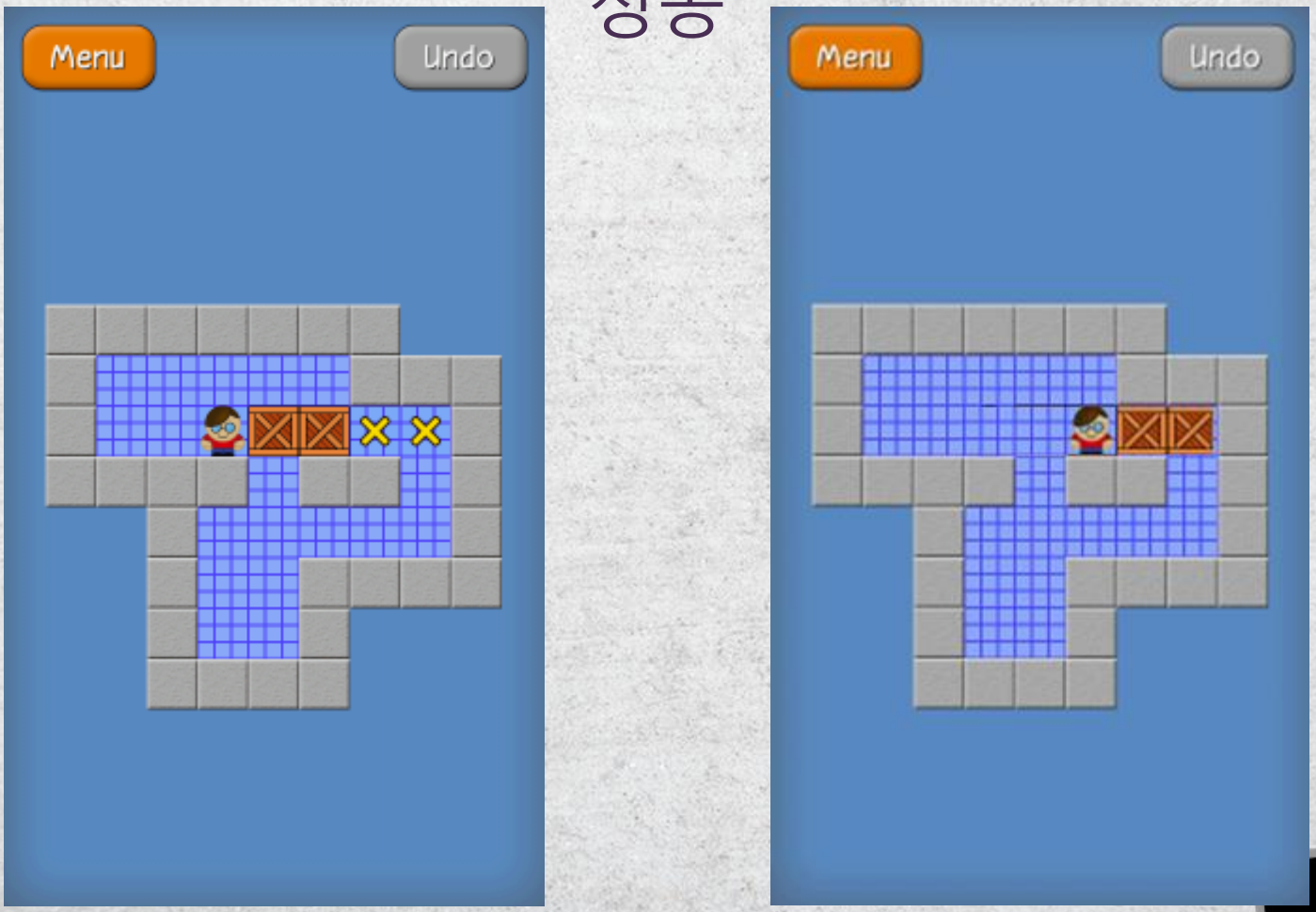
상자 뒤에
상자가 있는 경우

움직일 수 없는 경우 (2)



상자 뒤에
벽이 있는 경우

성공



모든 상자가 제자리에 가면 클리어!

제작 기획

- ncourses란?
- map 설정 방법
- 조작 방법
- 보여주어야 하는 내용
- 추가 구현 예상

ncurses란?

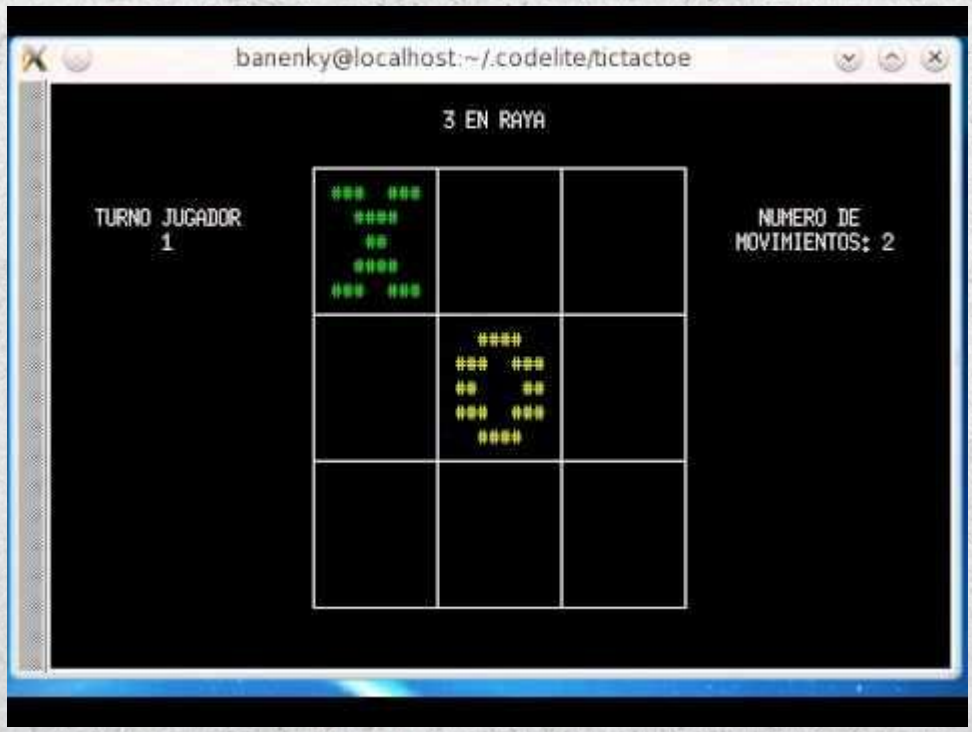
GNU에서 개발한 new curses 라이브러리로 텍스트 기반 UI이다.

장점

- 커서를 이동할 수 있어 키보드, 마우스로 쉽게 조작 가능
- 창의 크기, 색깔 등 제어 가능

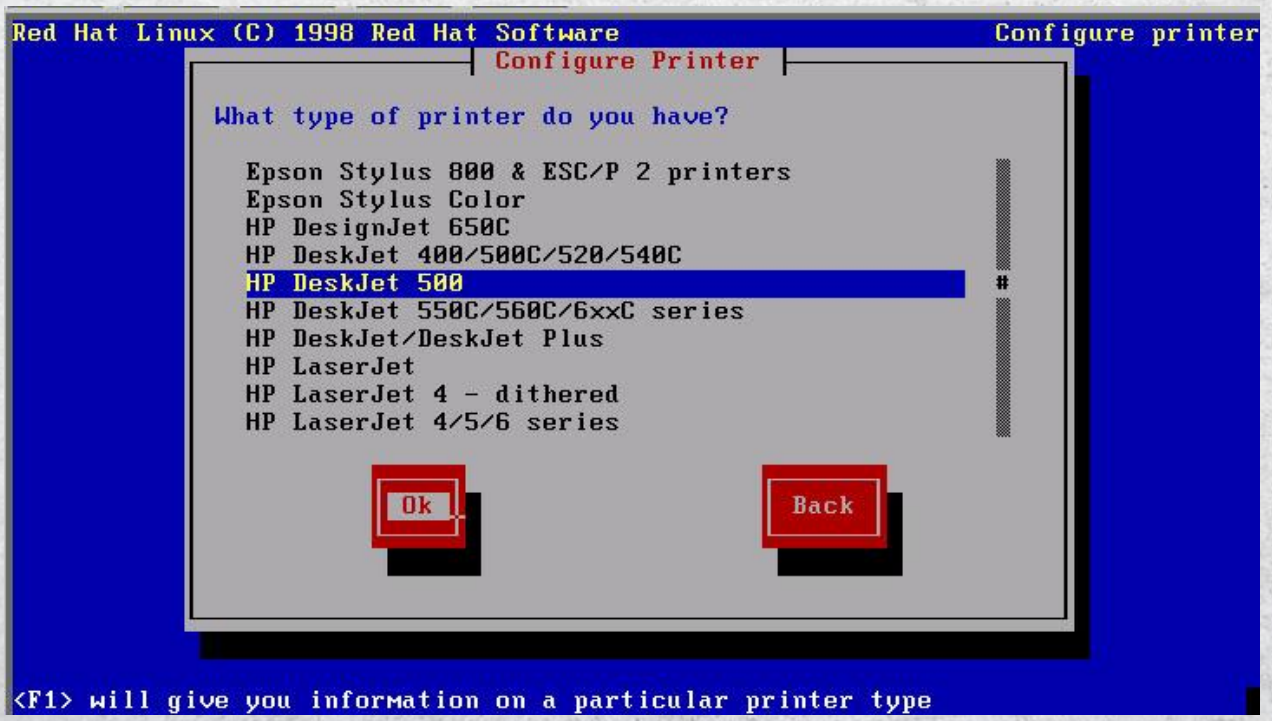
ncurses 이용한 예시

ncurses 이용한 빙고 게임

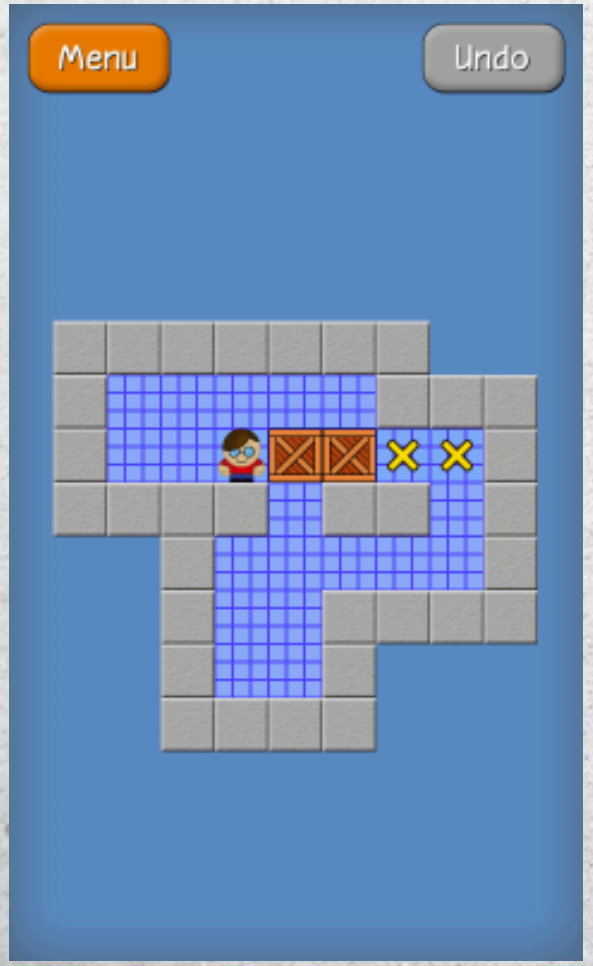


ncurses 이용한 예시

ncurses 이용한 설정 창



Map설정 방법



0번: map의 나머지
1번: 벽
2번: 상자
3번: 상자 도착 위치
4번: map이 아닌 부분

@: 캐릭터 위치

```
1 1 1 1 1 1 1 4 4
1 0 0 0 0 0 1 1 1
1 0 0@ 2 2 3 3 1
1 1 1 1 0 1 1 0 1
4 4 1 0 0 0 0 0 1
4 4 1 0 0 1 1 1 1
4 4 1 0 0 1 4 4 4
4 4 1 1 1 1 4 4 4
```

조작 방법

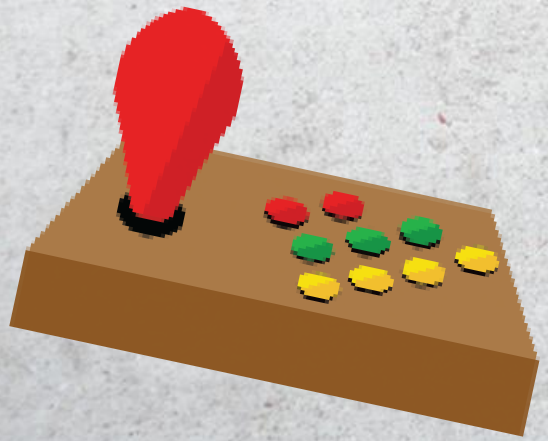
Enter키: 선택
P키: 게임 중지
↑키: 위로 이동
↓키: 아래로 이동
←키: 왼쪽으로 이동
→키: 오른쪽으로 이동

보여줄 내용

Step – 캐릭터가 움직인 횟수
Push – 상자가 움직인 횟수

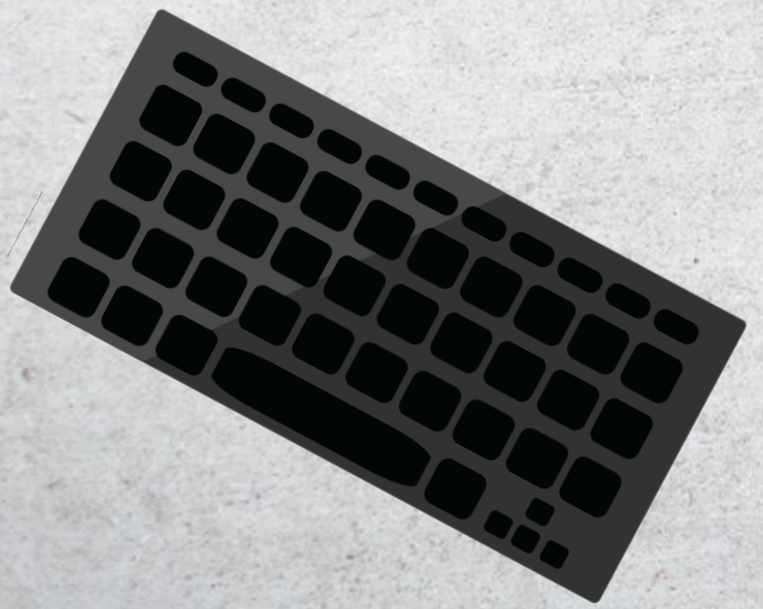
추가 구현 예상

1. 메뉴 만들기
2. P버튼으로 중단 기능
3. 레벨 선택 기능
4. 횟수 제한 기능

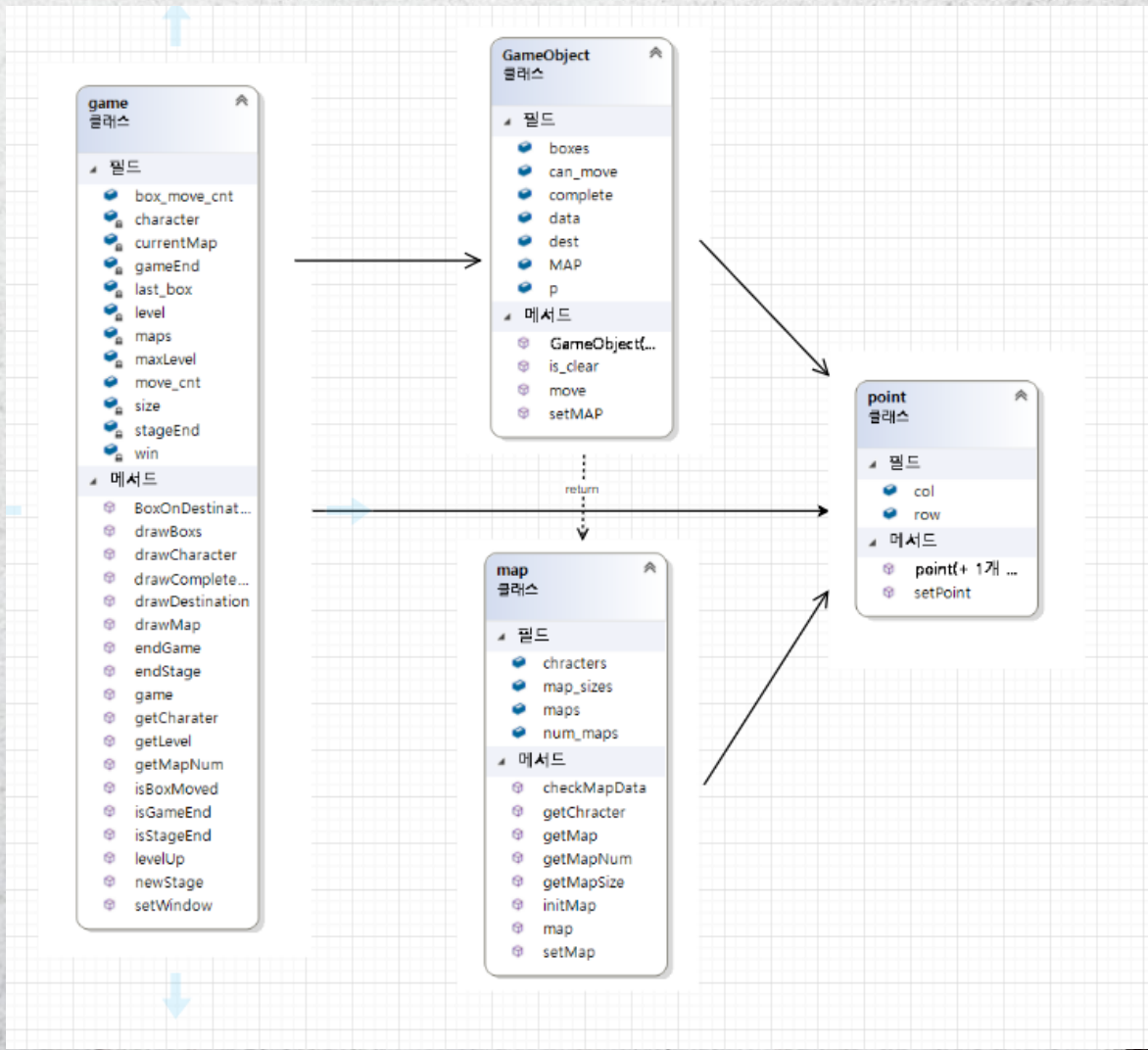


제작

- Source code
- Makefile



Class Diagram



PushBoxGme – 게임 제작

gameobject.h ▸ ...

```
1   #ifndef GAMEOBJECT_H
2   #define GAMEOBJECT_H
3   #include<iostream>
4   #include<vector>
5   #include<ncurses.h>
6   #include"point.h"
7   using namespace std;
8   class GameObject {
9   public:
10      static vector<vector<GameObject> > MAP;
11      static vector<GameObject> boxes;
12      static vector<point> dest;
13      static vector<point> complete;
14      point p;
15      bool can_move;
16      int data;
17      GameObject() {}
18      GameObject(int x, int y, int _data = 0, bool _can_move = false) :p(point(x, y)), data(_data), can_move(_can_move) {
19
20      }
21      GameObject(point p, int _data = 0, bool _can_move = false) :p(p), data(_data), can_move(_can_move) {
22
23      }
24      int move(int s,int c);
25      bool is_clear();
26      void setMAP(vector<vector<int> > currentMap);
27      friend ostream& operator<<(ostream& o, const GameObject &obj) {
28          return o << obj.data;
29      }
30  };
31  #endif
```

GameObject.h

Map,boxes,dest 는 static변수로 모든 gameobject의 객체들이 참조할 수 있도록 한다.

P는 현재 object의 좌표, data는 맵에서 받은 숫자를 의미한다.

PushBoxGme – 게임 제작

```cpp
#include"gameobject.h"
int GameObject::move(int s,int c) {
    if (!can_move)
        return -1;
    int _x = 0, _y = 0;
    switch (s) {
    case 1: _x = 1; break;
    case 2: _x = -1; break;
    case 3: _y = 1; break;
    case 4: _y = -1; break;
    default: cout << "1 : x+1 / 2 : x-1 / 3 : y+1 / 4 : y-1" << endl; return -2;
    }
    int x = p.col + _x;
    int y = p.row + _y;
    for (int i = 0; i < boxes.size(); i++) {
        if (boxes[i].p.col == x && boxes[i].p.row == y) {
            if (c == 2) {
                return -1;
            }
            if (boxes[i].move(s, 2) == 0) {
                p = point(x, y);
                return 0;
            }
            return -1;
        }
    }
    if (MAP[x][y].data == 0 || MAP[x][y].data == 3) {
        p = point(x, y);
        return 0;
    }
    else if (MAP[x][y].data == 1) {
        return -1;
    }
    return 0;
}
```

GameObject.cpp
Move 함수
인자값은 객체가 이동할 방향과 콜스택위치이다.

이동할 공간에 박스가 있다면 박스의 move함수를 실행하고 콜스택위치를 2로 만든다. 만약 이번 박스가 콜스택 2번째 move에서 찾은 박스라면 이동을 못하기 때문에 -1을 보낸다.

1인 벽을 제외한 나머지의 경우는 바로 좌표를 이동해 준다.

PushBoxGme – 게임 제작

```
void GameObject::setMAP(vector<vector<int> > currentMap) {
    GameObject::MAP = vector<vector<GameObject> >();
    GameObject::boxes = vector<GameObject>();
    GameObject::dest = vector<point>();
    for (int i = 0; i < currentMap.size(); i++) {
        vector<GameObject> tmp = vector<GameObject>();
        for (int j = 0; j < currentMap[i].size(); j++) {
            int n = currentMap[i][j];
            if (n == 1 || n == 4)
                tmp.push_back(GameObject(i, j, n));
            else if (n == 2) {
                tmp.push_back(GameObject(i, j, 0, true));
                GameObject::boxes.push_back(GameObject(i, j, 2, true));
                n = 0;
            }
            else {
                if (n == 3)
                    GameObject::dest.push_back(point(i, j));
                tmp.push_back(GameObject(i, j, n, true));
            }
        }
        GameObject::MAP.push_back(tmp);
    }
}
```

GameObject.cpp
setMAP 함수

Static변수들을 초기화 해준다.(setMAP은 스테이지가 새로 시작할때 수행됨을 가정)

현재 맵을 보고 게임에서 움직일 수 있는 요소(box)를 구분하고 게임에 영향을 주는 gameobject들을 만들어준다.

PushBoxGme – 게임 제작

```
bool GameObject::is_clear() {
    for (int i = 0; i < GameObject::boxes.size(); i++) {
        for (int j = 0; j < GameObject::dest.size(); j++) {
            if (GameObject::dest[j].col == GameObject::boxes[i].p.col &&
                GameObject::dest[j].row == GameObject::boxes[i].p.row) {
                break;
            }
            if (j == GameObject::dest.size() - 1)
                return false;
        }
    }
    return true;
}
```

GameObject.cpp
Is_clear 함수

클리어 요소를 확인하는 함수이다. 박스들의 위치와 박스의 도착지 위치들이 모두 같다면 true를, 하나라도 다르다면 false를 Return한다.

Game.h 헤더파일

```
class game{
private:
    WINDOW *win; // 게임 화면
    // character chrac; // chracter
    map maps; // 맵 데이터
    int maxLevel;
    int level = 0;
    bool gameEnd = false;
    bool stageEnd = false;

    point size;
    GameObject character;
    vector<vector<int> > currentMap;
    vector<GameObject> last_box;
public:
    int move_cnt = 0;
    int box_move_cnt = 0;
    game(WINDOW*&);
    void newStage(int level=0);
    void drawMap();
    void drawCharacter();
    void drawBoxs();
    void drawDestination();
    void drawCompleteBox();
    void levelUp();
    GameObject& getCharater();

    void endStage();
    void endGame();
    bool isStageEnd();
    bool isGameEnd();
    int getMapNum();
    int getLevel();
    bool isBoxMoved();
    bool BoxOnDestination();

    void setWindow(WINDOW *&another);
};
```

Game클래스

Game 클래스는 메인 윈도우 위에 Gamescreen을 그리고 그 위에 내용들을 적고 game안에서 사용하는 오브젝트들을 관리하는 game driver클래스이다.

Game클래스의 생성자와 New stage함수

```
game::game(WINDOW *&win) {
    this->win = win; // 윈도우 설정
    maps.initMap(); // 맵 세팅
    maxLevel = maps.getMapNum();
}
```

```
void game::newStage(int level){
    // 데이터 셋
    if (level >= maxLevel) {
        char msg[] = "Last Level";
        mvwprintw(win, IN_WINDOW_Y, IN_WINDOW_X+10, msg);
        gameEnd = true;
        return;
    }
    move_cnt = 0;
    this->move_cnt = 0;
    this->level = level;
    stageEnd = false;
    size = maps.getMapSize(level);
    character = maps.getChracter(level);
    currentMap = maps.getMap(level);
    character.setMAP(currentMap);
    GameObject::complete.clear();
    last_box = GameObject::boxes;
}
```

게임클래스가 처음 만들어
Map클래스를 통해서
Map을 생성하고

Map에 관련된 정보들을
받아오게 된다.

NewStage로 각 레벨
마다의 맵과 character,
Box등을 다시
로드하게 된다.

Draw 함수들

```
void game::drawMap(){
    // 맵상에 화면 그려줌
    // cout << "맵을 생성합니다." << endl;
    for(int row=0; row<size.row; row++){
        char ctemp[2];
        for(int col=0;col<size.col;col++){
        // cout << "row : " << size.row << " col : " << size.col << endl;
            sprintf(ctemp, "%d", GameObject::MAP[row][col].data);
            mvwprintw(win, row+IN_WINDOW_Y, col+IN_WINDOW_X, ctemp);
        }
    }

    //drawBoxs();
    //drawCharacter();
}
void game::drawBoxs() {
    for (int i = 0; i < GameObject::boxes.size(); i++) {
        mvwprintw(win, GameObject::boxes[i].p.col + IN_WINDOW_Y
            , GameObject::boxes[i].p.row + IN_WINDOW_X, "2");
    }
}
```

Draw 함수들

```
void game::drawDestination() {
    for (int i = 0; i < GameObject::dest.size(); i++) {
        mvwprintw(win, GameObject::dest[i].col + IN_WINDOW_Y
            , GameObject::dest[i].row + IN_WINDOW_X, "3");
    }
}
void game::drawCompleteBox() {
    for (int i = 0; i < GameObject::complete.size(); i++) {
        mvwprintw(win, GameObject::complete[i].col + IN_WINDOW_Y
            , GameObject::complete[i].row + IN_WINDOW_X, "2");
    }
}
void game::drawCharacter(){
    mvwprintw(win, character.p.col+IN_WINDOW_Y, character.p.row+IN_WINDOW_X
}
```

Draw함수들은 game클래스가 관리하고 있는 window에 각각의 객체들을 그려주는 역할을 한다.
Destination, Box, Character, Map 등을 각자의 위치에 맞게 그려준다.

Game Level에 관련되는 함수들

```
void game::levelUp(){
    level++;
    this->newStage(level);
}
void game::endStage(){ stageEnd = true; }
void game::endGame(){ gameEnd = true; }
bool game::isStageEnd(){
    stageEnd = character.is_clear();
    return stageEnd;
}
bool game::isGameEnd(){ return gameEnd; }
GameObject& game::getCharater(){
    return character;
}
```

이 함수들은 각각이 레벨을 올리거나 플레이하고 있는 레벨을 끝내던가

또는 모든 스테이지를 클리어해서 게임이 아예 종료될 때 신호를 보내고

그에 맞춰 동작을 하도록 만든다.

Box의 움직임에 관여하는 함수

```
bool game::isBoxMoved() {
    for (int i = 0; i < last_box.size(); i++) {
        if (last_box[i].p.col != GameObject::boxes[i].p.col ||
            last_box[i].p.row != GameObject::boxes[i].p.row) {
            last_box = GameObject::boxes;
            box_move_cnt++;
            BoxOnDestination();
            return true;
        }
    }
    return false;
}
bool game::BoxOnDestination(){
    GameObject::complete.clear();
    for (int i = 0; i < GameObject::boxes.size(); i++) {
        for (int j = 0; j < GameObject::dest.size(); j++) {
            if (GameObject::dest[j].col == GameObject::boxes[i].p.col &&
                GameObject::dest[j].row == GameObject::boxes[i].p.row) {
                GameObject::complete.push_back(last_box[i].p);
            }
        }
    }
}
```

Box가 움직일 때마다

Box가 목적지에 도착하는 것을 감지하야

서로 구분을 할 수 있도록 도와주는 함수

Main.cpp 파일

지금껏 구현한 클래스와 함수들을 모아서 실제로 화면을 구성하고 루프를 생성시키는 main파일

Main의 함수들과 전역변수

```
void init_colors();
void handle_menu_cursor(int key, int &y); // 메인 메뉴에서 커서 이동
void select_this_menu(int y); // Enter입력으로 메뉴에 따른 아이템 실행
void show_pause_screen(WINDOW *win, string message="  Press Any KEY To\n
void gameStart(int prefix=0); // Main game loop로 들어감
void gameLoop();
void selectLevel();
void draw(WINDOW *); // gameObjects 놈들 draw
```

Ncurses의 색을 지정하는 기능,
메뉴, 일시정지화면, 게임시작(세팅), 실제 게임, 레벨 선택 등을 함수로 구현하였다.

메인 메뉴창 핸들러

```
void handle_menu_cursor(int key, int &y){
    int max = 6, min = 4;
    switch(key){
        case KEY_UP:
            if(y>min)y--;
            break;
        case KEY_DOWN:
            if(y<max)y++;
            break;
        case 'p':
            show_pause_screen(stdscr);
            break;
        case 10:
            select_this_menu(y);
            break;
    }
}
```

키보드 위아래 방향키를 가지고 메뉴를 선택

Enter키를 누르면 select_this_menu 함수를 호출하여 선택한 메뉴에 맞는 동작을 실행한다.

레벨 선택

```
void selectLevel(){
    int key, level=0, maxLevel = g.getMapNum();

    g.setWindow(levelSelect);
    wbkgd(levelSelect,COLOR_PAIR(9));
    while(key != 'q' || key != 'Q' || key != 10){
        g.newStage(level);
```

```
    switch(key){
        case KEY_LEFT:
        case KEY_DOWN:
        if(level>0) level--;
        else mvwprintw(levelSelect, 2, 3, "This level is Min Level");
        break;
        case KEY_RIGHT:
        case KEY_UP:
        if(level<maxLevel-1) level++;
        else mvwprintw(levelSelect, 2, 3, "T his level is Max Level");
        break;
    }
    if(key == 10){
        break;
    }
}
wclear(levelSelect);
wrefresh(levelSelect);
gameStart(1);
```

방향키를 이용해서
레벨별 맵을 확인하고

Level을 선택해서
플레이를 할 수 있도록

gameStart(1)을
호출하여

선택한 레벨 정보를 가지고 게임을 시작한다.

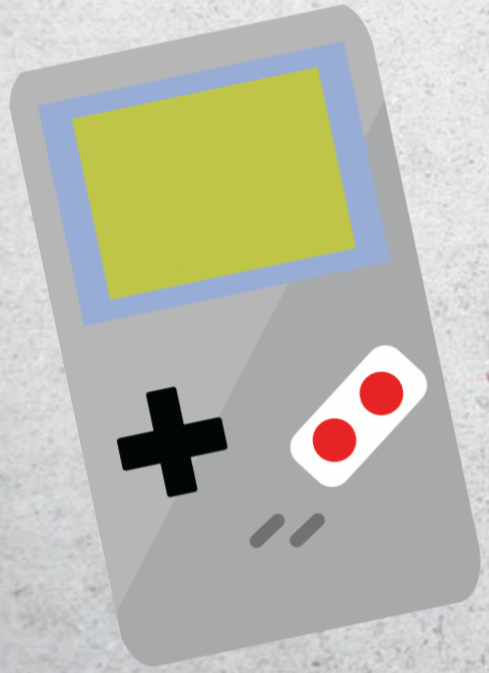
Makefile

Makefile

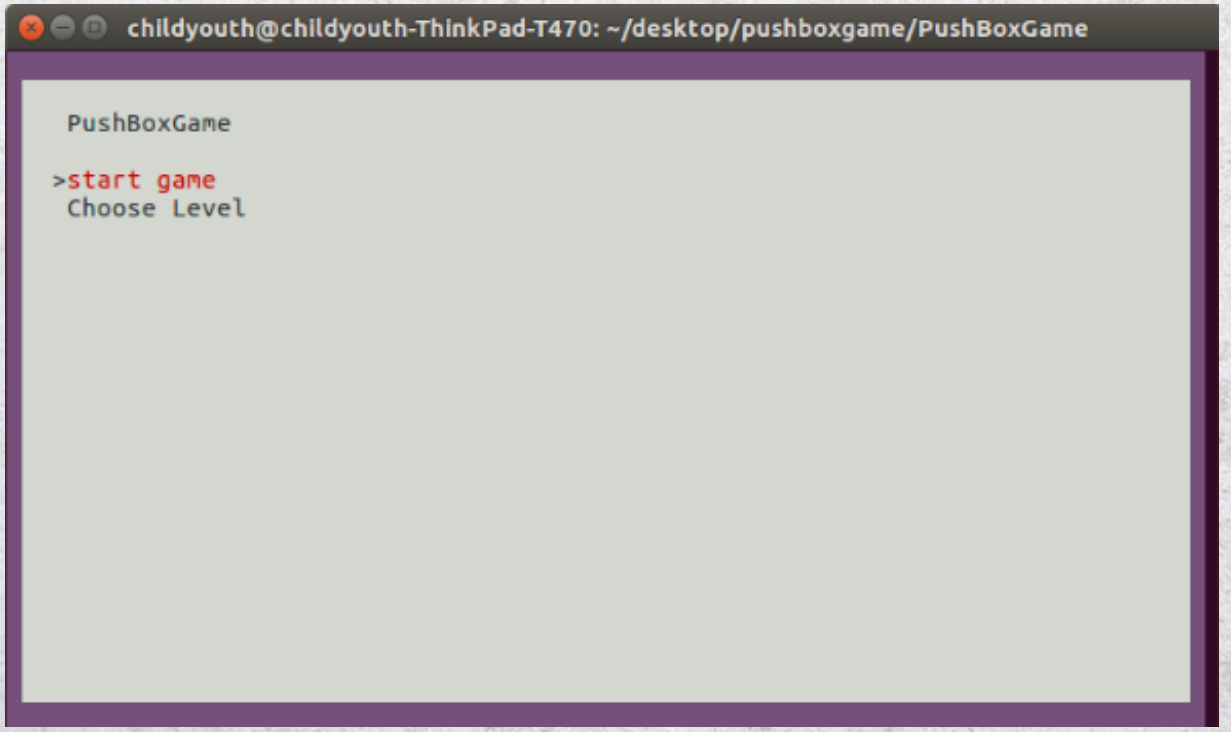
```
1   CC=g++
2   CPPFLAGS=-W -Wall -g
3   LFlag=-lncurses
4
5   all:pushbox
6
7   pushbox:gameobject.o map.o game.o main.o
8       $(CC) -o pushbox gameobject.o map.o game.o main.o $(LFlag)
9
10  gameobject.o: gameobject.cpp gameobject.h
11      $(CC) -c -o gameobject.o gameobject.cpp $(LFlag)
12
13  map.o: map.cpp map.h
14      $(CC) -c -o map.o map.cpp $(LFlag)
15
16  game.o : game.cpp game.h map.h
17      $(CC) -c -o game.o game.cpp $(LFlag)
18
19  main.o:main.cpp
20      $(CC) -c -o main.o main.cpp $(LFlag)
21
22  clean:
23      rm -rf *.o
```

실행화면

- 시작 페이지
- 게임 페이지
- 추가 구현

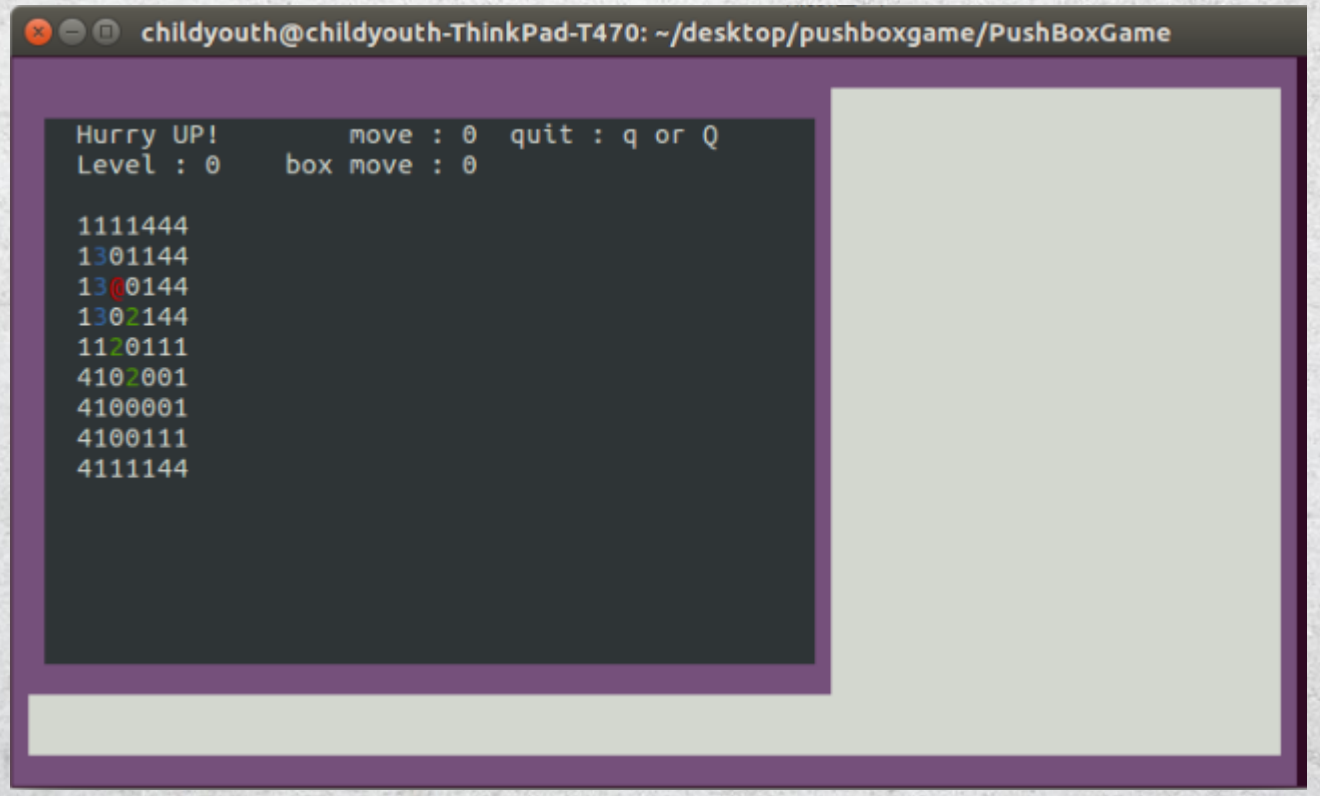


시작 화면



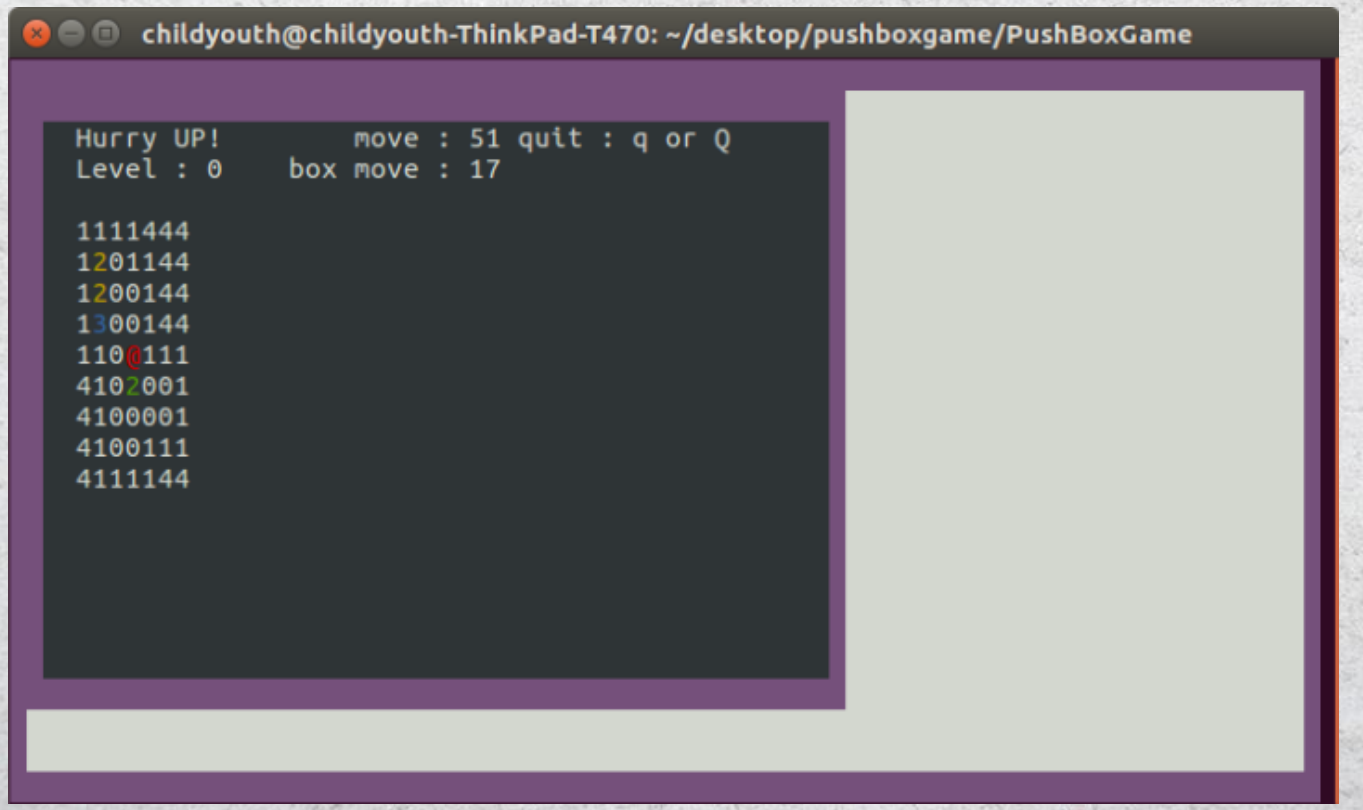
PushBoxGme – 완성

Start Game – 1단계 시작



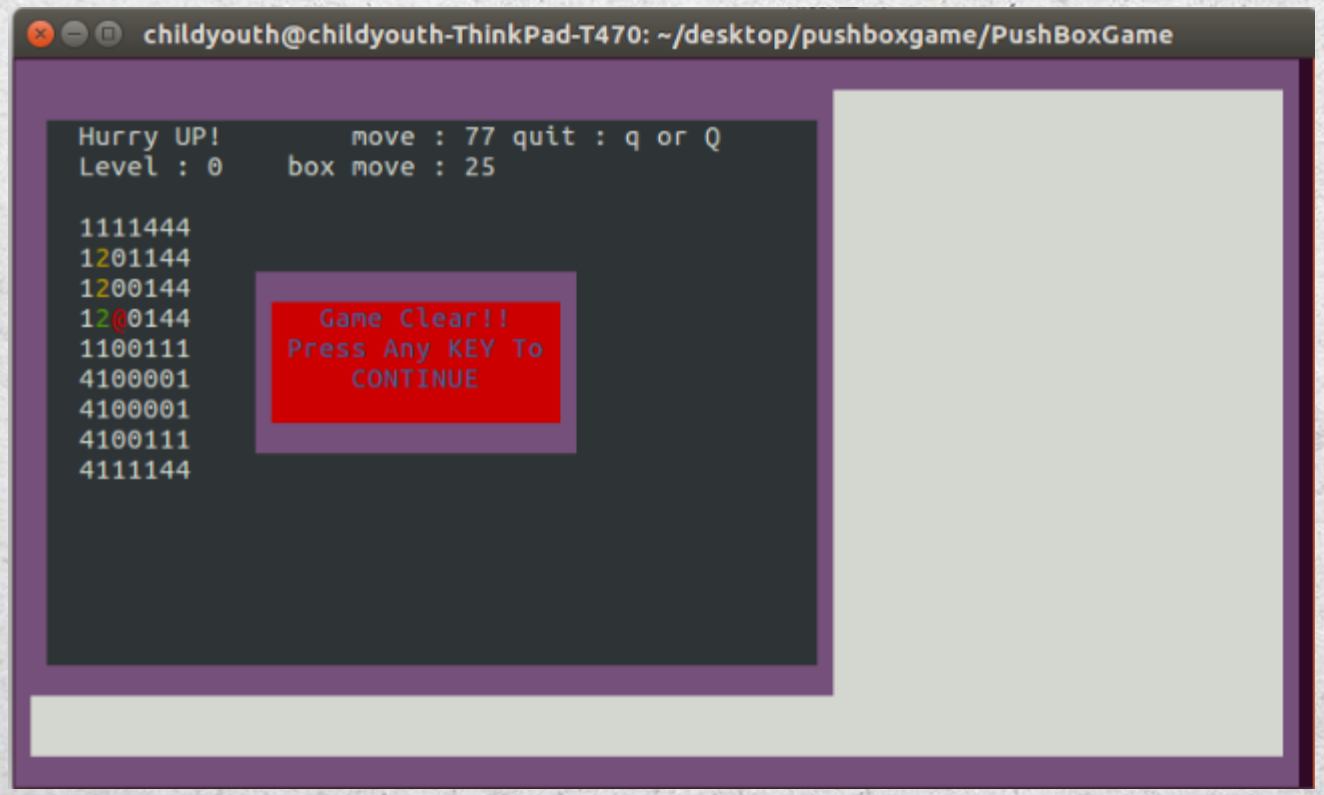
PushBoxGme – 완성

- 상자가 제자리에 가면 노란색으로 변하는 모습
- 캐릭터의 움직임, 상자의 움직임 나타내기



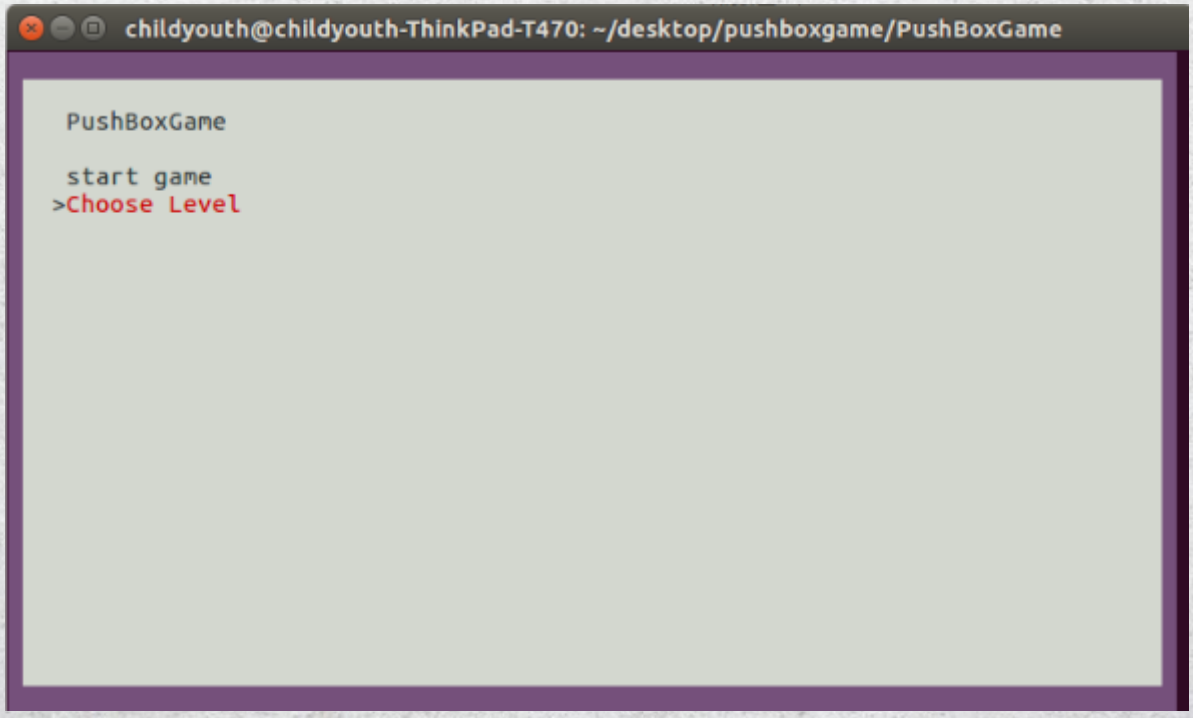
PushBoxGme – 완성

성공한 모습



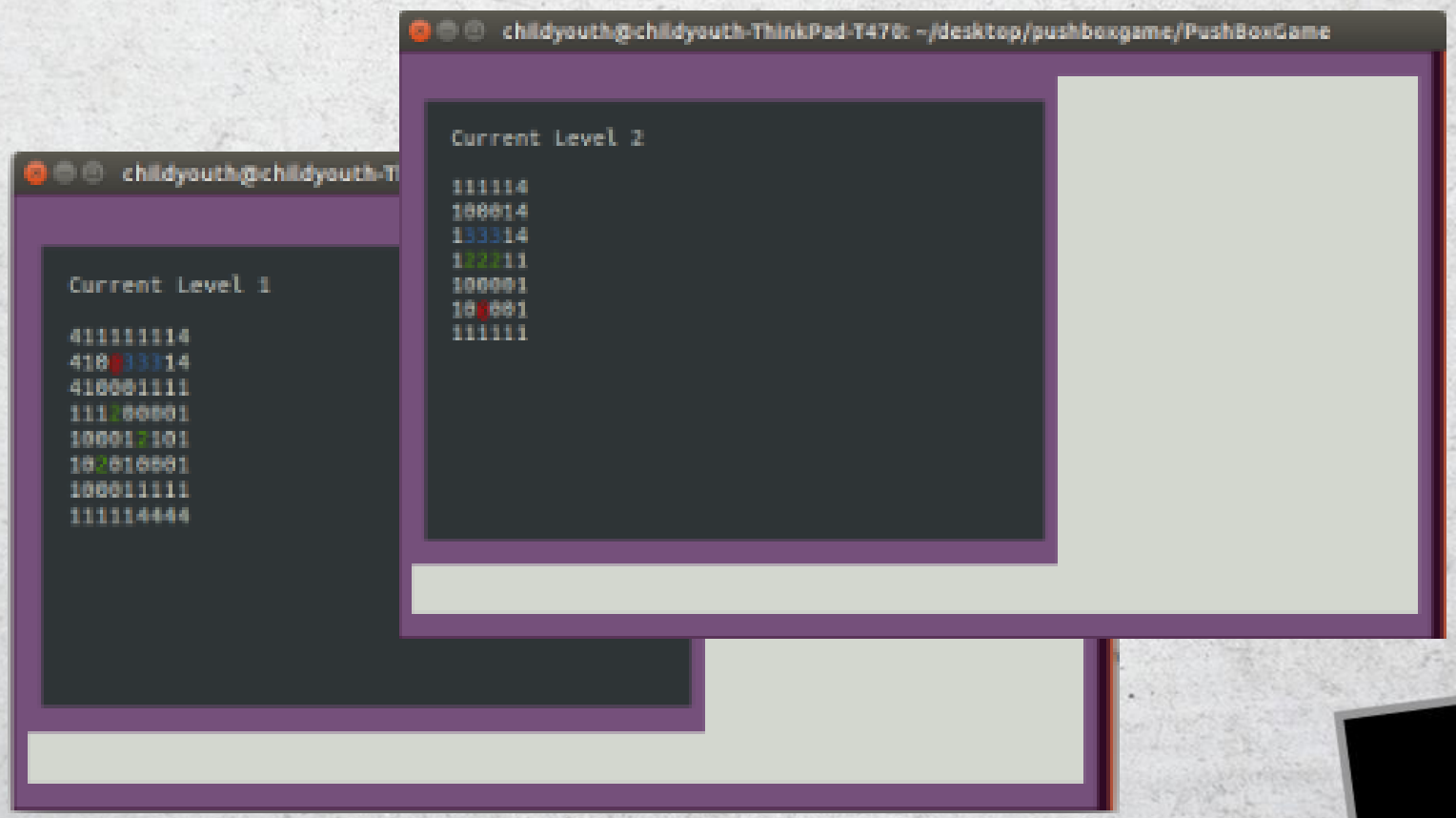
아무 키나 누르면 다음 단계로 넘어간 모습

Choose Level

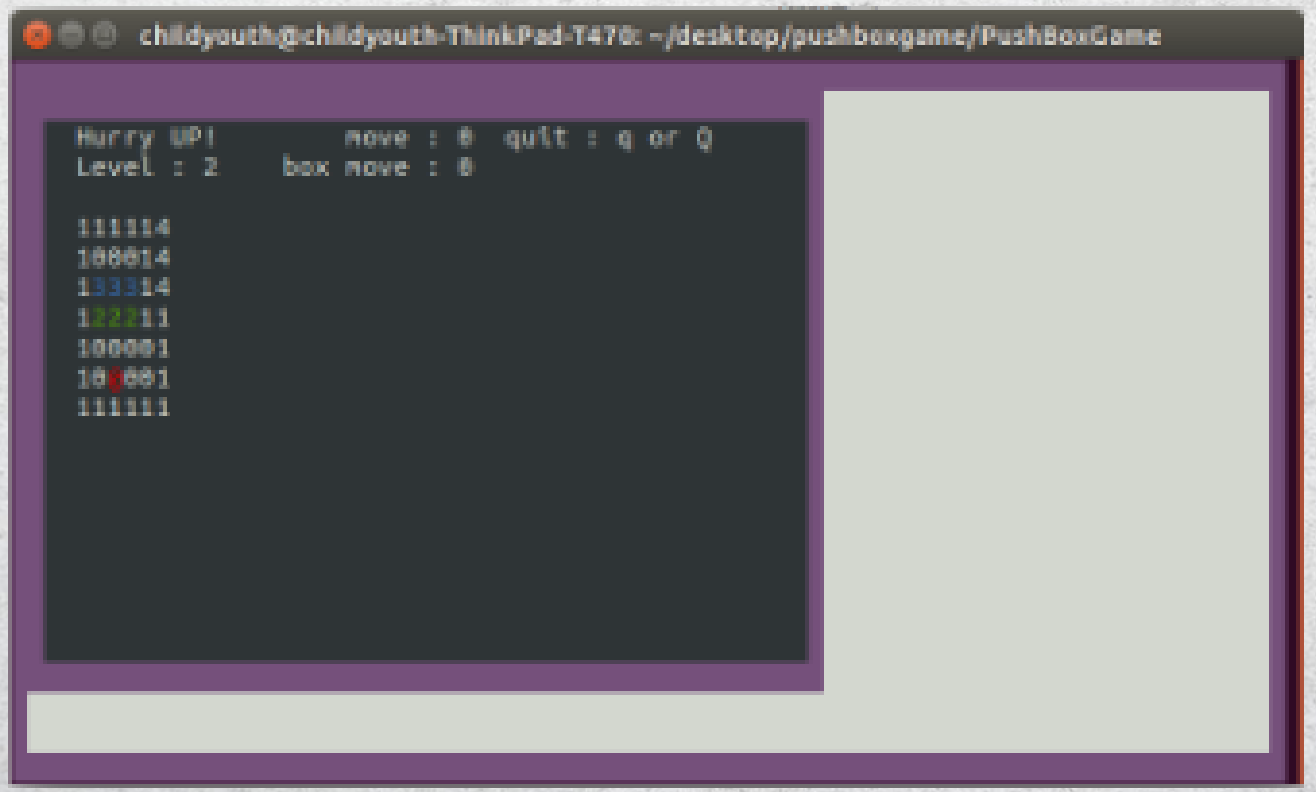


PushBoxGme – 완성

방향키를 통해 레벨 선택 가능 – 6레벨까지

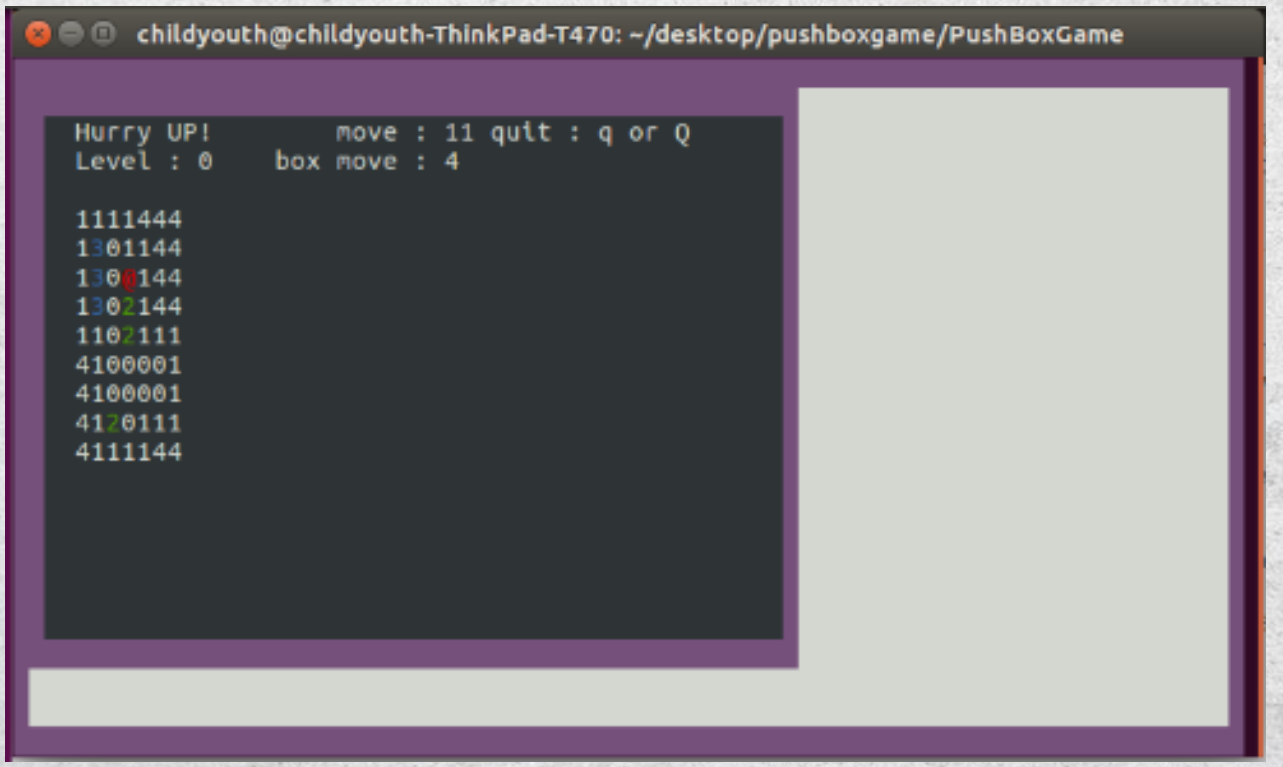


Enter키를 누르면 시작 가능



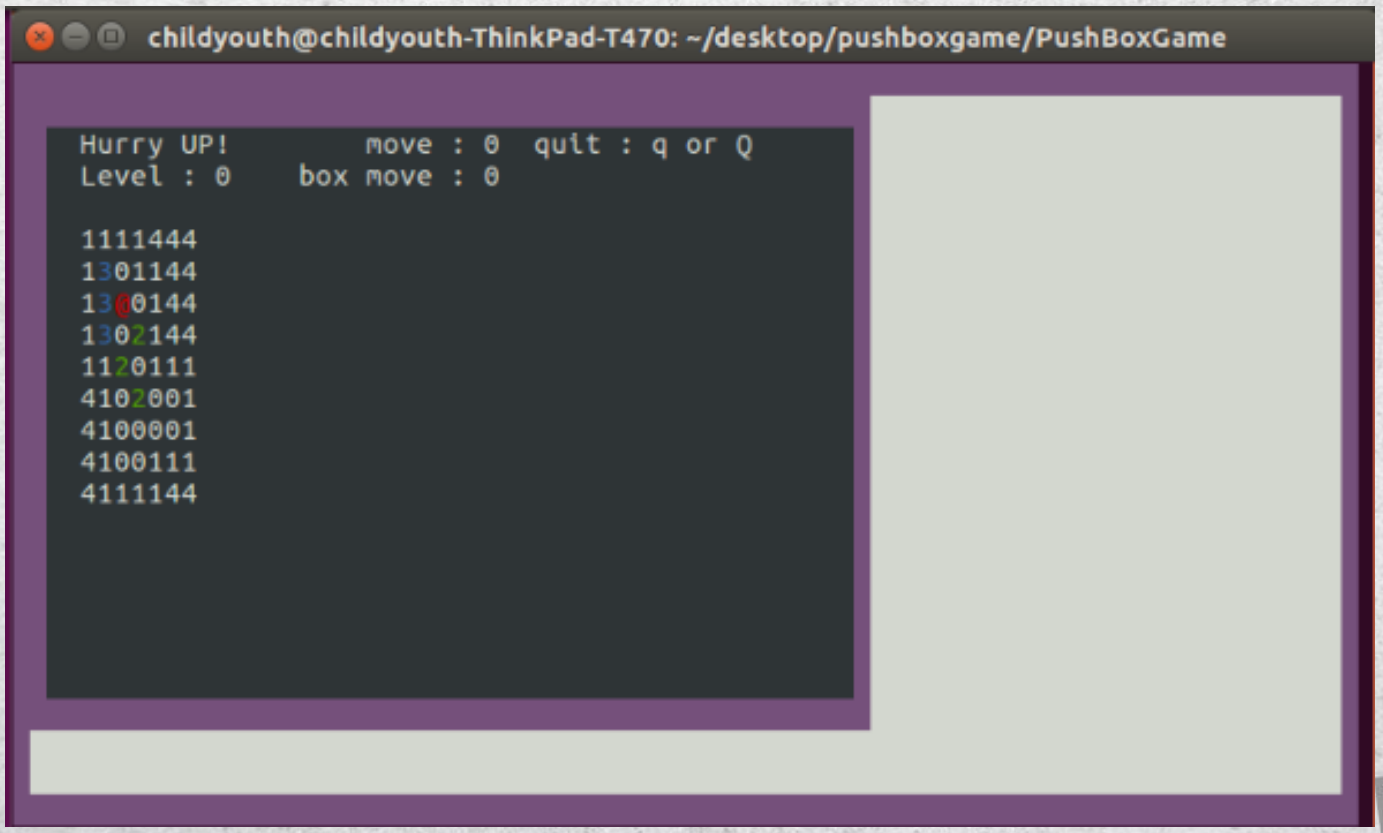
PushBoxGme – 완성

움직일 수 없을 때



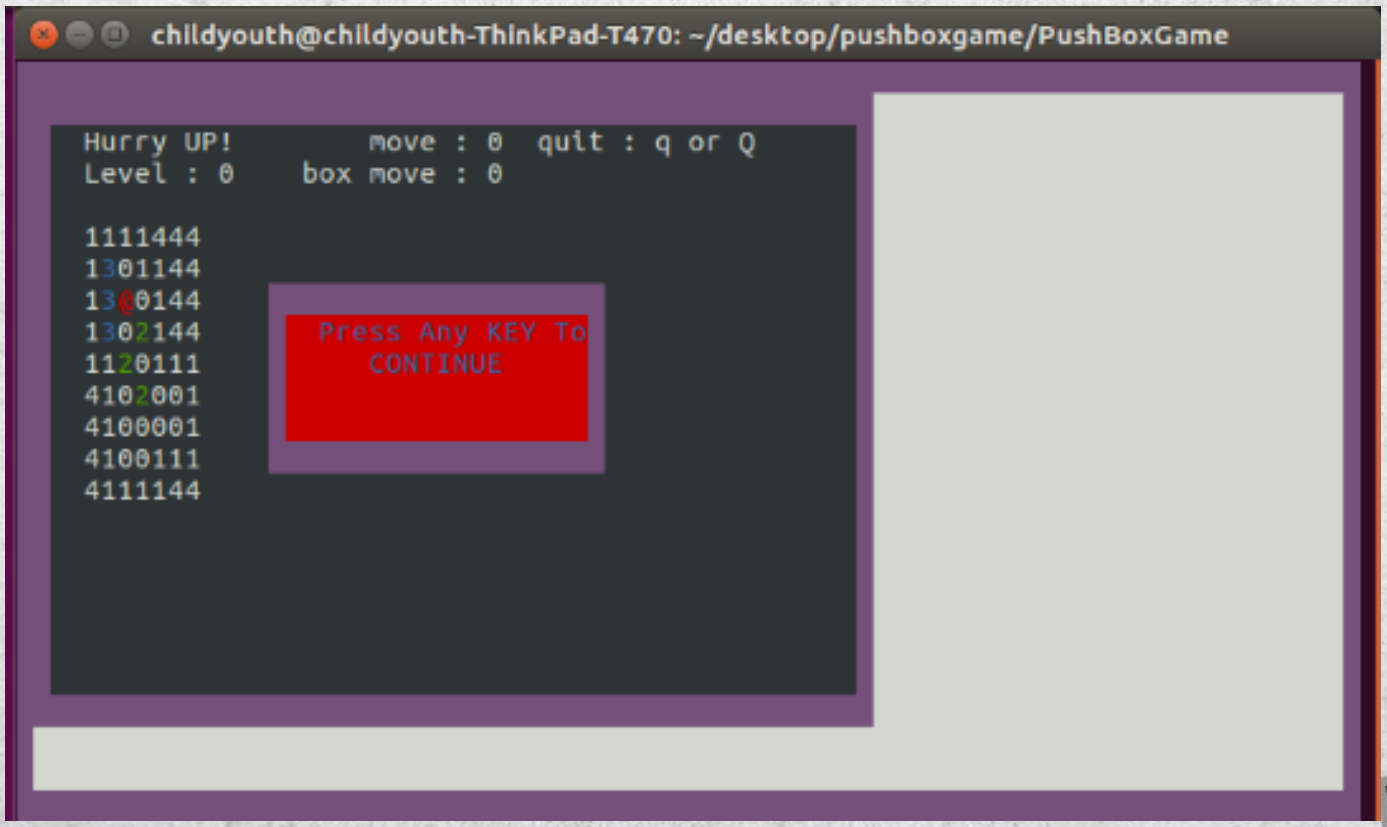
PushBoxGme – 완성

R키를 통해 다시 시작 가능



PushBoxGme – 완성

P버튼을 통해 일시정지 가능







THE END

THANK U

INSERT COINS
TO CONTINUE