

CS246 Final Project: Detailed Design Document

Renee Jingling Li, Jordan Henck

April 29, 2014

1 Introduction

In this project Jordan and Renee created the game called flow or alternatively titled Numberlink. In this game the user is given a square grid with an even number of dots. Each dot relates to another dot. The goal is to match the all the corresponding dots with each other while leaving no white space. Normally there is a pre-set board given by the games creator. However to have more boards then possible when using predetermined boards and solutions we will use randomly generated boards so that there is a seemingly infinite amount of fun. Since not every randomly generated board will have a solution, before the board is given to the user, an algorithm checks to see if it has at least one solution. If a solution is found the user is then be given the board to solve. The user must use the a,s,w and d keys to move the number along the grid to connect it with the corresponding pair. The program checks to see if the user has won the game by filling up all of the spaces in the grid and connecting each number.

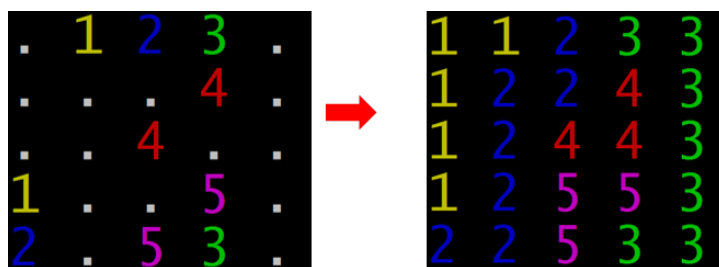


Figure 1: An example of the our game in console

2 How to play

The user should select which number he/she wants to connect. Then connect that number to its corresponding number by going left right up or down. When one number is connected move on to the next number until all the numbers are connected and the board is completely filled. The rules for the game are displayed below:

- I User selects which number they would like to connect
- II Use the keys a,w,d and s to connect the number with its corresponding number
- III To win the grid must be completely filled and each number must be connected to its corresponding number
- IV Each space may only be occupied by one line or number
- V Each time when the user wants to use a spot that is already occupied the user will be given the option to clear that entire path or not

3 Function prototypes

I main.h

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct {
    int size_of_board;
    char board[8][8];
    int size_of_Num;
    int num[19];
}Board;

/*****
* This function tests to see if the user input is what is expected
* if not the function asks the user re enter the input
*****/
char getUserinput(char[],int);

```

II setup.h

```

#include <stdio.h>
#include <stdlib.h>
typedef struct map{
    unsigned long day: 64;
} map;

/*****
* checks to see if the board has a solution
*****/
int solution(Board);

/*****

```

```

* creates the array of specified size and number of pairs randomly
*****/
Board createArray(int, int);

/*****
* prints the board
*****/
void print(Board);

/*****
changes the two dimensional char array into an unsigned long bitmap
*****/
map* exchange(char [][][8], int);

/*****
* a recursive method that finds if there is a solution which is
* called by solution
*****/
unsigned long pointTpoint(unsigned long, int, int, int, int, const int,
    const int, const unsigned long, const int[]);

```

III play.h

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RESETCOLOR "\033[0m"
#define RED "\033[0;31m"
#define GREEN "\033[0;32m"
#define YELLOW "\033[0;33m"
#define BLUE "\033[0;34m"
#define MAGENTA "\033[0;35m"
#define CYAN "\033[0;36m"
#define WHITE "\033[0;37m"
#define LIGHTRED "\033[1;31m"
#define LIGHTGREEN "\033[1;32m"
#define LIGHTMAGENTA "\033[1;35m"

typedef struct Path{
    int num; //which number does this path represent
    int way; //its current way
    struct Path* next;
    struct Path* prev;
} Path;

/*****
accept the board created, ask which number to start (by default using the

```

```

    first position).
call play to start and store the successful path for each number(using a
    structure)
Check isWin after each path successfully connected.
*****/
void start(Board);

/*****
accept and check user input for movement and decision(e.g. q, restart),
calls isvalid first to check the movement, and then if the movement is valid,
conduct the movement.
Return either null or the successful path being created;
*****/
Path* play(char[][8], int, int, Path*[], int[], int, int);

/*****
1) if the movement will go off the boundary, not valid. return 0
2) if the movement will cross into another number's path,
the program will ask first and a) continue the movement and clean off the
    other path or
b) return not valid and go back to function play
*****/
int isvalid(char[][8], char, int, int, int[], int, int);

/*****
* This function prints the board in a user friendly format
*****/
void printBoard(char[][8], int);

/*****
* returns a path structure which stores the current path and
* points to previous path and points to next path which is null
*****/
Path *makenode (int, Path*);

/*****
* cleans the path of the specified number
*****/
Path *cleanNode(char[][8], Path*);

/*****
check if every point are connected properly, if yes,
check if the board is full, if not, return false;
*****/
int isWin(Board, Path*[]);

/*****

```

```
*This function changes the color of the text based on what character is given
*****/
void changeColor(char);
```

4 Dependency graph

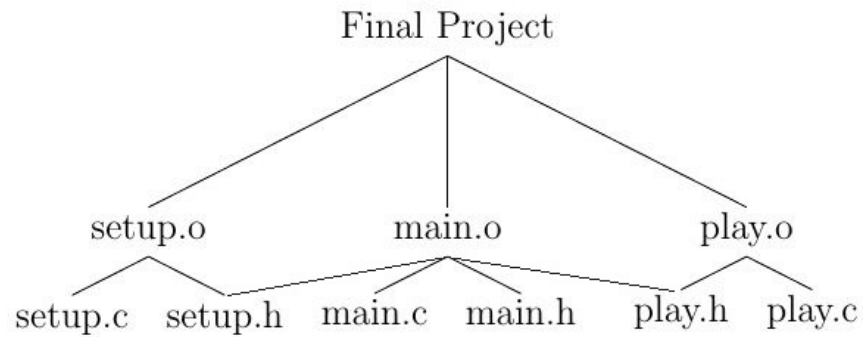


Figure 2: Dependency graph for the final design