Dots&Boxes Game

• Description & User manual

- The game consists of a grid of some boxes divided into lines.
- Each player in his turn selects one line
- The target is to close a box
- The player who close a box his score increases by one and he have another turn
- The game ends when all lines are selected

• Main struct

```
typedef struct{
    char name[100];
    int color;
    int score=0;
}player;
player p1,p2;
```

- The struct has three variables Player name ,his color and his score .
    - In Case of computer p2name is computer and p2color is yellow.

- Main arrays

```
char usedChars[60];
```

  - It stores the chosen characters.

```
int turns[60];
```

  - It stores who chose each character in (usedChars) array.

```
char boxes[30],boxes2[30];
```

  - The first one stores the complete box with player's number.
  - The second one is used to check if the player closed a box .

```
int undoarr[25][3][60]={0};
```

  - A 3 Dimensional array which take a snapshot of last played move and it is used in (undo) function.

```
int redoarr[25]={0};
```

  - It stores the last undo char and is used in (redo) function.

```
vector<char> dfss(0);
```

  - This vector is used when applying DFS to store the lines which has to be colored in it.

• Assumptions

- The player select the line he wants by choosing the character corresponding to it.
- When player close a box his number is put in the box.
- When player choose computer mode , then the player number is 1 ,computer number is 2 and computer color is yellow.
- In computer turn , computer plays in the box who has 3 sides and if there isn't he plays in a systematic way mentioned in Algorithms
- Total score of the winner player increase every game he wins by 1.
- It is supposed to be two files in the folder of the game (winners and saved Games).
- Time is started when the player play the first line.
- When load a game time starts from zero when the player who in turn plays.
- We supposed that chain must start and end with a box who have three side and boxes in between them have 2 sides and all chain borders are closed.
- When player play a line that results in chain and he wants to undo he has to play the next line and then he can undo move by move till the first move which resulted in the chain.

• Main algorithm &Game flow

When the game starts the player has to choose mode, level and enter players details. The player who in turn enter the character of the line the he wants to choose then after checking its validity it is sent to “print Grid” function which store it in “usedChars” array and store the number of the player who played it in “turns” array, then it prints the grid which consist of 12 line for beginners OR 60 line for experts and each chosen line is colored by player’s color ,and white lines represent unchosen character yet.

While printing a check of closing boxes takes place and if there is a box closed the number of player who closed the box is printed in the middle of the box.

After choosing the line the player has the ability to choose if he wants to undo , save&quit or continue playing.
If player choose to continue we print the following information after the grid:

- Each player score. “The score of the player is the number of boxes he closed”.
- Remaining lines .
- Number of moves each player made.
- Time from starting the game.
- Which player takes the next turn.

If the player’s move resulted in chain “Chain is defined in Assumptions” then all chain lines are played automatically in order not to be boring for players to close all chain line by line ,But if the player wanted to undo chain he plays a 1 move more then he can undo line by line till his first move.

If player choose Save&quit he is asked to enter the name of the game which shouldn’t be used before then all game details are stored and when he wants to return to game any time later he can choose load game from main menu and enter the game’s name then continue playing.

The game has two modes one VS one and human VS computer the player choose the mode he wants at the start of the game.

Computer’s playing algorithm is that he searches for boxes that have one missed side and he playes this side if he didn’t found he playes in a systematic way of choosing the vertical line which in turn till finishing all virtical line then he playes in horizontal lines.

• Main functions.

```
int chooseLevel(){ ···
```

- It determines the size of the grid (5*5 or 2*2)

```
int choosemode(){ ···
```

- It determines the mode one VS one or human VS computer.

```
void enterplayers1(){ ···
```

- It is used to enter players data when mode is 1.

```
void enterplayers2(){ ···
```

- It is used to enter player data when mode is 2.

```
int choosen(char a){ ···
```

- It determines if the character is already choosen or not.

```
void printPlayer(){ ···
```

- It prints the player's name with his color.

```
int isBox1(char c,int size){ ···

void isBox2(char c,int size){ ···
```

- They determine if there is a closed box and store who closed it.

```
void whoIsLast(char c,int def,int size){ ···
```

- It determines the last side to close box and use it for computer logic.

```
void computerLogic(char c,int size){ ···
```

- The function that are responsible for playing for computer.

```
> void printHorizontel(char x,int size){···
> void printNumInBox(int i,int q){···
> void printVertical(char x,int size,int i){···
> void printGrid(char x,int size){···
```

- They are responsible for printing the grid of the game.

```
bool repeated(char x){···
```

- If the player entered a chosen number, it prints already taken.

```
bool valid(char x,int size){···
```

- It returns if the choosen character is valid or not.

```
bool filled(){···
```

- It checks if the player closed a box or not.

```
void score1(){···
void score2(){···
```

- They calculate the score of each player.

```
void printscores(){···
```

- It prints the score of each player.

```
void calculateMoves(){···
```

- It calculates the number of moves of each player.

```
void printmoves(){···
```

- It prints the number of moves of each player.

```
void printremaininglines(int size){···
```

- It prints how many remaining lines in the grid.

```
void save(){···
```

- It saves the data of game to continue it in another time.

```
void load(){···
```

- It prints the name of saved games and player choose what he want to continue it.

```
int continueGame1(int size){···
```

- It continues the saved game if mode is 1.

```
int continueGame2(int size){···
```

- It continues the saved game if mode is 2.

```
void topTen(){···
```

- It prints the name of top 10 who won most number of games.

```
void userManual(){···
```

- It prints the rule of the game.

```
int winner(int ans,char name[],int len){···
```

- It prints the score and the rank of the winner.

```
void menu(){···
```

- It prints the menu and player chooses what he wants from it.

```
void printwinner(){···
```

- It prints the winner in the game if exist or draw if not.

```
int mode1(int size){···
```

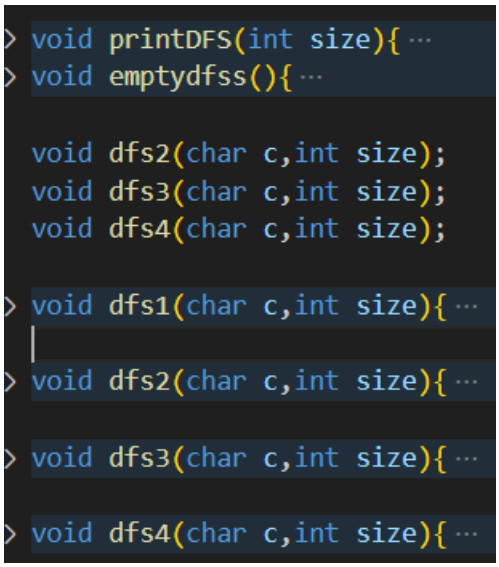
- It starts the game if mode is 1.

```
int mode2(int size){···
```

- t starts the game if mode is 2.

```
void newGame(){···
```

- It starts a new game.



- They are responsible for DFS.

- It responsible for undo last move

- It responsible for redo last undo

- Sample game play
  - Game1: one VS one in beginner mode

Video :
https://drive.google.com/file/d/1y8-4Hvfn49AtO7X6pVw4V9agL5XQGPki/view?usp=sharing

Photos:
https://drive.google.com/file/d/133Uneg2m799ldJvF5OeJzpxTuzlqwzWp/view?usp=sharing

  - Game2: human VS computer in expert mode

Video :

https://drive.google.com/file/d/11gPjEciFEx-SrQJItDojQ5fttprGyKDS/view?usp=sharing

- References

We used this code to print colors in terminal

- ANSI-color-codes.h
https://gist.github.com/RabaDabaDoba/145049536f815903c79944599c6f952a?permalink_comment_id=4050376