



DOTS & BOXES

CSED23 Project (Term 3)



AHMAD WALEED OTHMAN 18 01 5028
AHMAD MOHAMED ABDELMONEM 18 01 0225
ABDERAHMAN KHALED GABER 18 01 0924

Game description:

Dots and Boxes is a game for two players starting with an empty grid of dots, two players take turns adding a single horizontal or vertical line between two un-joined adjacent dots. The player who completes the fourth side of a 1x1 earns one point and takes another turn.

In this application we have designed a version of this game with the ability to try different sizes of grids and other gaming features like (how to play, save, load, undo, redo... etc.). It was implemented using the C programming language.

Features:

1. Player can choose to play beginner, expert or custom modes where each one differs from the other in the number of squares.
2. Player can choose between playing with another player (two players) or with the computer.
3. The computer is programmed to close a box if available, then to play in a box with 3 sides left, 4 sides left then 2 sides left on that order.
4. Player can return to the previous menu any time.
5. in-game menu contains undo, redo, save and exit as well as exit to main menu without saving.
6. time updates every second.
7. which turn it is, total moves left, turns played by each player and each player's score are all displayed along with the time elapsed since beginning of game under the grid.
8. numbers are put on the grid in a way that allows the user to choose where to play with only two numbers separated by a comma.
9. player can undo all moves till beginning of game and then redo all of them if they wish to do so.
10. there are 3 different saved games at any given time and the user can choose to overwrite any of them if they chose to save the game.
11. player can load any of the 3 games from main menu and continue playing where he left off.
12. from the main menu, the player can choose to view the top 10 scores along with their usernames.
13. if a player achieves a high score, they are prompted to input their username in order to put in the top 10.

14. usernames are case-insensitive; meaning that if “aHmaD” is in the top 10 with a score of 8 while the least score in the top 10 is 3 for example, a new username “Ahmad” won’t make it to the top 10 unless his score is above 8. Note that the user name is taken from user after game is finished, that means in the previous example, if a player scored 5 he will be prompted to enter his username, however, he will not be put in the top 10 if he inputs “Ahmad” because he is considered the same user “aHmaD” with a higher score of 8.
15. top 10 are displayed after a player achieves high scores and enters their name.
16. there is a “how to play” section where the player can see how to choose the desired column/row. However, this section assumes the player is familiar with the rules of the game. If not, a user manual is present in this document for how to play any game of dots and boxes.
17. the game is designed not to crash given any input. This is supposed to work for any number of inputs even if in a single line.
18. player can alternate between undo and redo in a streak as many times as he wants but all the redo is deleted once he plays a new move.

Design Overview:

The player is first greeted with the main menu when he can choose between starting a new game, loading a previous one, view the top 10 players or learn how to play. The `start game` section the player can choose beginner size grid or expert size. They can also choose custom to input a custom size.

Assumptions:

1. The game is designed to look like the photo given in the .pdf provided by you.
2. It is assumed that the user will see the list of top 10 so as to not to input a username as a present one; since they will both be assumed to be the same player.
3. the player name is only taken if a high score is achieved.
4. it is assumed that only one list of rankings is required. i.e. not one for each of the grids.
5. we also assumed the user will not change any of the .txt files

Data structures:

We have used different kinds of data structures.

Arrays: like “grid” which is the main array in the app that saves the dots and lines in the game mode and “boxes” which corresponds every box in the grid and how many sides are available to draw lines.

Structures: like “player”; we have used this structure to save all players’ information like score, and moves

Description of files:

Note that only the important functions are stated below.

-grid.h:

Contains functions related to the grid like creating the grid, printing it, drawing a line in the grid and assigning a box in the grid to a certain player.

-gamePlay.h:

This contains the structure Player as well as multiple other functions related to the game play like `movesLeft` which returns the number of moves left essential to know when to end game, `checkbox` which checks if the line drawn resulted in a box being closed and therefore needs to be assigned to the player using `assignBox` from grid.h, `printBar` which prints the bar containing info. Below the grid. It also contains the most important function for playing: `play`; which uses all of the other functions to loop until no moves left. There is also the function `getInput` which makes sure the user inputs a valid input.

-rankings.h:

It contains function needed for the Top 10 section. `readTop10` reads the rankings from a file. `checkHigh` score checks if the new score is a high score. `updateTop10` then puts the new name in the list.

-undo.h:

It contains some functions that are the inverse of some above like `deleteLine`, `assignBoxAgain` (with a blank space), `checkBoxAgain` to see if the undo will decrease the score of a player well as some other functions like `addUndo`, `addRedo`, `undoPlay` and `redoPlay`.

-computer.h:

Contains functions that enables the computer to play not only a valid play but also a good one. ``compSearchNum`` is used to search for a specific number of sides left in all the boxes then ``compSearchRc`` searches which side to play in that given box. The function ``compChoose`` is the one implementing both previous functions.

-save.h:

It has the functions essential for the process of saving and loading. The functions save the main grid, array of boxes, all the other data like the moves played by each player, moves left and the scores.

It also contains all the functions necessary to load all the saved info.

Pseudo code:

MAIN

Print game interface, get the user input and loop until it is a valid input

Case based on user input

Case 1 start the game

Get user input and loop until it is valid

Case based on user input

Case 1 beginner mode set size to 2

Get user input

IF it is 1 then its human VS computer

Start “play” function and loop until there are no moves left

IF it is 2 then it is human VS human

Start “play” function and loop until there are no moves left

Case 2 expert mode set size to 5

Get user input

IF it is 1 then its human VS computer

Start “play” function and loop until there are no moves left

IF it is 2 then it is human VS human

Start “play” function and loop until there are no moves left

Case 3 custom mode get size from user

Get user input

IF it is 1 then its human VS computer

Start “play” function and loop until there are no moves left

IF it is 2 then it is human VS human

Start “play” function and loop until there are no moves left

Case 4 back to previous menu

Case 2 get file number from user

IF it is 1 then load saved game1

Load grid size, moves left, and players’ scores from the chosen file

Start “play” function and loop until there are no moves left

IF it is 2 then load saved game2

Load grid size, moves left, and players’ scores from the chosen file

Start “play” function and loop until there are no moves left

IF it is 3 then load saved game3

Load grid size, moves left, and players’ scores from the chosen file

Start “play” function and loop until there are no moves left

Case 3 print top 10

Case 4 start “how to” function

Case 5 exit game

PLAY

Pass in: char array “grid”, size, computer state, loaded state and loaded moves

IF loaded is 1 or 2 or 3

THEN load specific data from text file using “loadBoxes”, ”loadDatat” functions and start the game with them

ELSE

Start the game with default data

WHILE moves left dose not equal zero

Print the grid with the drawn lines

Print information bar

IF it is computer mode and computer turn

THEN let computer choose where to put the line

ELSE

Get user input while updating the time every second

IF user input is 'e' or 'E'

THEN exit to main menu

ELSE IF user input is 'u' or 'U'

THEN undo the last move

ELSE IF user input is 's' or 'S'

THEN ask user for file number then save

WHILE the chosen line is not valid

Print "invalid"

Get user input while updating the time every second

IF user input is 'e' or 'E'

THEN exit to main menu

ELSE IF user input is 'u' or 'U'

THEN undo the last move

ELSE IF user input is 's' or 'S'

THEN ask user for file number then save

IF it is player 1's turn

Draw line

IF the line completes a box then increase player 1 score

Else

Draw line

IF the line completes a box then increase player 2 score

Print grid

Print information bar

IF computer score > player score

Print “computer has won the game”

ELSE IF player’s 1 score dose not equal player’s 2 score

Print who has won the game

IF it is new high score

Get user’s name and update top 10 list

ELSE

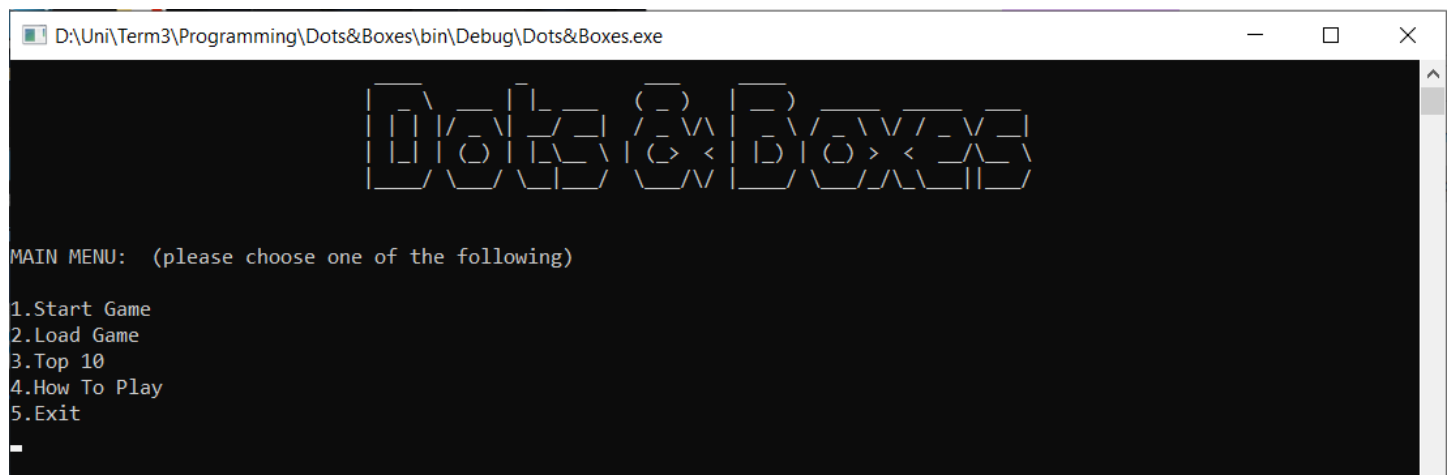
Print “it’s a tie!”

User Manual:

The game starts with an empty grid of dots. Usually two players take turns adding a single horizontal or vertical line between two not joined adjacent dots. A player who completes the fourth side of a 1×1 box earns one point and takes another turn. The game ends when no more lines can be placed. The winner is the player with the most points.

In our game the player chooses the column where they wish to play then the row separated by a comma. For more information on how to choose the column and row please refer to the “How to play” option from the main menu.

Sample Runs:



```
D:\Uni\Term3\Programming\Dots&Boxes\bin\Debug\Dots&Boxes.exe

  [ ] [ ] [ ] [ ] [ ]
  [ ] [ ] [ ] [ ] [ ]
  [ ] [ ] [ ] [ ] [ ]
  [ ] [ ] [ ] [ ] [ ]
  [ ] [ ] [ ] [ ] [ ]

MAIN MENU: (please choose one of the following)
1.Start Game
2.Load Game
3.Top 10
4.How To Play
5.Exit
_
```


Beginner Mode

```
D:\Uni\Term3\Programming\Dots&Boxes\bin\Debug\Dots&Boxes.exe

Dots & Boxes

  1   2   3   4   5
1  o-----o       o
2
3  o       o-----o
4
5  o       o       o

Player 1's turn    Total moves left:    8    Time elapsed:  0 minute(s) 17 seconds

Player 1:    played 2 turns    Score = 0
Computer:    played 2 turns    Score = 0

U: undo    R: redo    S: save and exit    E: exit to main menu

Please choose column then row separated by a comma:  _
```

```
D:\Uni\Term3\Programming\Dots&Boxes\bin\Debug\Dots&Boxes.exe

Dots & Boxes

  1   2   3   4   5
1  o-----o-----o
2  |       |       |
   |   B   |   R   |
   |       |       |
3  o-----o-----o
4  |       |       |
   |   B   |   B   |
   |       |       |
5  o-----o-----o

Total moves left:    0    Time elapsed:  1 minute(s)  3 seconds

Player 1:    played 7 turns    Score = 3
Computer:    played 5 turns    Score = 1

U: undo    R: redo    S: save and exit    E: exit to main menu

Player 1 has won the game

New High Score!
Please enter username (20 character max. without spaces): myAmazingTestName
```

```
D:\Uni\Term3\Programming\Dots&Boxes\Dots&Boxes.exe

Dots & Boxes

TOP 10
=====
1.AhmadWaleed      500
2.AbelrahmanKhaled 500
3.AhmedAbdelmonem  500
4.youssefWaleed    6
5.myAmazingTestName 3
6.placeholder2      0
7.placeholder3      0
8.testName          0
9.yetAnotherTestName 0
10.3aashYaReggala   0

Press Enter to return to main menu
```

Expert Mode

```
D:\Uni\Term3\Programming\Dots&Boxes\Dots&Boxes.exe

Dots & Boxes

  1  2  3  4  5  6  7  8  9 10 11
1  o      o      o      o      o
2      o      o      o      o      o
3  o      o      o      o      o
4  o      o      o      o      o
5  o      o      o      o      o
6  o      o      o      o      o
7  o      o      o      o      o
8  o      o      o      o      o
9  o      o      o      o      o
10 o      o      o      o      o
11 o      o      o      o      o

Player 1's turn    Total moves left: 32    Time elapsed: 1 minute(s) 52 seconds
Player 1: played 16 turns    Score = 4
Computer: played 12 turns    Score = 0
U: undo  R: redo  S: save and exit  E: exit to main menu
Please choose column then row separated by a comma: _
```

References:

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>