Final project
Dots and Boxes

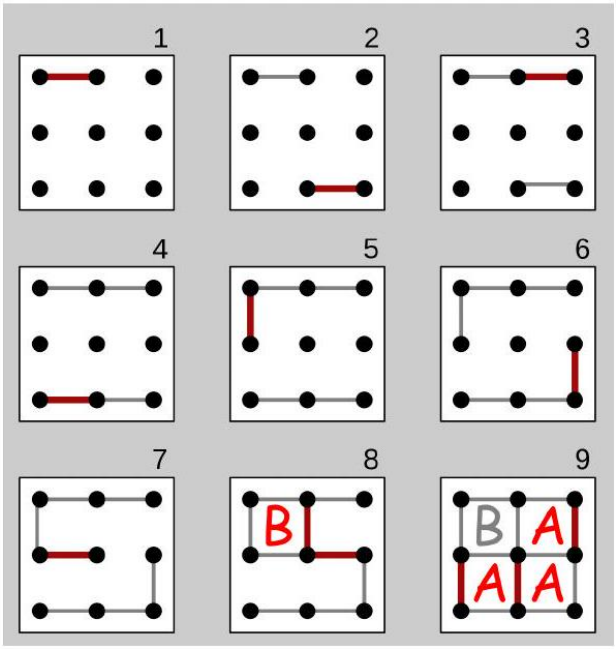

Figure 1: Example on dots and boxes of 2×2 board

1 Game Description

Dots and Boxes is a game for two players (or more). It was first published in the 19th century by Edouard Lucas, who called it originally la pipopipette. Starting with an empty grid of dots, two players take turns adding a single horizontal or vertical line between two unjoined adjacent dots. The player who completes the fourth side of a 1x1 box (or groups of one or more adjacent boxes) earns one point (s) and takes another turn. In this project you will make your own version of the game. You can read more about Dots and Boxes on wikipedia.

2 Overview

The goal of the game is to connect dots and take as much boxes as you can and mark them with your color. The board may be of any size. When short on time, a 2x2 board (a square of 9 dots) is good for beginners. A 5×5 is good for experts.

Score is calculated based on the number of boxes you have. The game ends when no more lines can be placed. The winner is the player with the most points.

The map is a two dimensional grid where it has no walls at all, the players should connect dots (build walls) to win a box (s).

When a player connects dots, it may produce a chain, a chain is a groups of adjacent boxes

in which any move gives all the boxes in the chain to the player.

3 Game Actions

In Dots and Boxes there is only one action, which is connecting between two dots (draw a line) in order produce a box or a chain. You should color the box or chain with the player's color.

You should enable a turn based game, and if a player win a box or a chain, he should take another turn.

4 Game Mode

Your game should have two game modes:

- Two player mode: human vs human
- One player mode: human vs computer

Computer player can play with any strategy even a random computer player that chooses any valid random row/column for the lines.

5 End Game

Dots and Boxes will end when no more lines (connection between two dots) can be places. The player with the highest score is the winner.

6 Grid Display

You should display a grid filled with only dots (no lines) at the beginning of the game. All the cells will be by default empty, the cell color will be changer during the play:

- Connecting two dots: place a border (line) with the player's color between the two dots.
- Completing a box: change the color of the cell and its borders to the player's color.
- Completing a chain of boxes: change the color of the chain and its outer borders only to the player's color.
- End game : keep the board and print the name of the winner, then handle his rank (section 8). Ask the player to either go back to the menu or exit.

Each row and each column should start with a number to make it easy for the player to choose a cell in the gird.

Hint: check ascii codes, colored text in console.

7 User Interface

During the game play you should print extra information beside the grid.

- one field -> Whose player turn is.
- two fields -> Number of moves for each player.
- two fields -> Scores of each players.
- one field -> Number of remaining dots.
- one field -> Time passed in minutes and seconds.

All the above fields should be updated each time any player makes a move.
You should color the above information with the color of each player to make it more readable.

Taking input from the user shall be like: row row col col. Example on the figure (section upper left sub image) is 3312 .

8 User's Rank

When a player successfully wins a game, You should take his name and display his score and rank. If it's the first time for this user, then add him to the users list (a file). Else, update the score if the current score is higher than the saved score. User name should be case insensitive. You shouldn't take name of the second player. You should print top ten players in sorted order based on their scores. You don't have to save them sorted in the file.

9 Save and Load

- At any time, the game could be saved to a file.
- A saved game could be loaded and continued.
- Players names and scores should be saved to a file.
- No need to save undo data.
- You should have 3 saved files for 3 different games.

Hint: The saved file could be a binary file, instead of a text file.

10 Undo and Redo

- The game could be undone till the first move.
- After undo, if the player make a new move, all the redo should be cleared.

11 Chain of Boxes

When the player connects two dots that results in a chain of boxes, you should check all the adjacent boxes and find all the boxes until you hit a border, then you should fill them all with the player's color. This can be solved using recursion with depth first search algorithm. (DFS).

The Algorithm :

While doing a DFS, we maintain a set of visited cells. Initially this set is empty. When DFS is called on any cell (say u), first that cell is marked as visited and then for every adjacent cell v, such that v is unvisited, we call DFS on v. Finally, we return when we have exhausted all the adjacent cells.

12 Main Menu

The main menu of your program should contain the following options :

- start game

  choose beginner or expert

  choose game mode vs human or vs computer
- load game

  choose the saved game to load
- top 10 players
- exit

  You can add other options.

You should provide some sort of an in game menu in order to handle undo, redo, save, and exit.

13 Game Flow

1. Ask the user for beginner or expert
2. Ask the user for one or two players
3. Loop till the end of the game:

(a) Read a move from Player 1
(b) Check if it is valid input and if the selected dots are empty
(c) If not valid, ask for another move
(d) Do the move
(e) Update the UI
(f) If this move completes a box or a chain, give the player another move.
(g) Repeat the same logic for player 2. If it is "human vs. computer", then instead of reading the move as input, you should execute some logic for the computer's turn.

4. Update list of high scores if any and display the sorted list of high scores at the end of the running game.

14 Implementation Notes

- You are required to implement the game using the C programming language.
- Top Priority Your program MUST NOT crash under any circumstances, even against malicious users!
- All users' inputs should be validated. Any missed validation will reduce your marks.
- Take care of the following:

  Naming conventions.

  Code organization and style.

  Code comments
- You are welcome to add extra features to your game. Good extra work will be graded as bonus.

15 Bonus

- Split your project on different files and use header files (different files for: ranking, game loop, computer turn, user interface, undo, ...).
- Use git to co-work with your teammate.
- When the user is idle for a minute, update the time.
- Implementing a convenient computer turn algorithms (AI, ...).

Deliverables

- Complete project and source files
- Executable version of your project (the *.exe file)
- Report containing the following:
- Description: You should briefly describe the application you implemented.
- Features: Write down the features of the application.
- Design Overview: Make a small overview on your design and the overall structure with no details.
- Assumptions: Tell us if you had taken any assumption during the design or the implementation phase. necessary to be clarified.
- Data Structure: Write down the data structure you have used in the program, like arrays, structures, ...
- Description of the important functions/modules: Describe the important functions you implemented. Also, write down the header files you created (if any) and What is each header file used for?
- Flow chart and pseudo code: Write the main algorithms used in the program, like game loop, DFS, ...
- User Manual: You should make a user manual to teach the user how to use the application.
- Sample runs: You should provide some screenshots showing the program while running through various cases, like winning, playing, losing, ...
- References: Any copied material from anywhere should be referenced. Any discovered unreferenced copied material will result in sever reduction in your grade.

17 Notes

- You are required to work in groups of two.
- Take your time to design the project and discuss your design.
- No time will be allowed in the lab for any modifications, your program should be executed from a copy of your CD during your 15 min discussion.
- All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to all participating students.
- Start in the project early and come forward with your questions.
- Late submissions are not accepted.
- Prior to discussion, you should be prepared to illustrate your work and answer any questions about it. Failing to do so is a strong indication to copying / cheating.
- It is better to play the actual game so you can understand it better. However, the only difference is in case of chain of boxes, the player has to close box by box (boring!), here you will close all of the boxes for him using DFS.

Good Luck isA :)