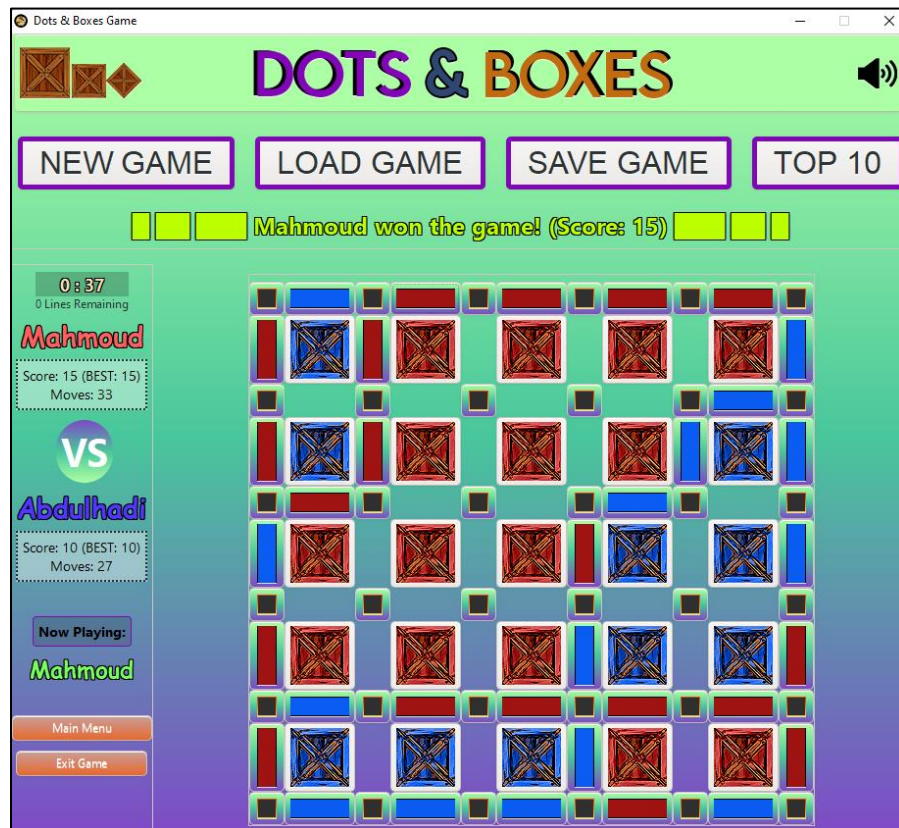


## Final Project

Project Name	Dots and Boxes
Repository	<a href="https://github.com/membaby/Dots-n-Boxes">https://github.com/membaby/Dots-n-Boxes</a>
Student Name	Mahmoud Tarek Mahmoud Embaby
Student ID	20011800
Supervisors	Dr. Nagia M. Ghanem & Eng. Mohamed Al-Mansour



# DOTS & BOXES

## 1. Description:

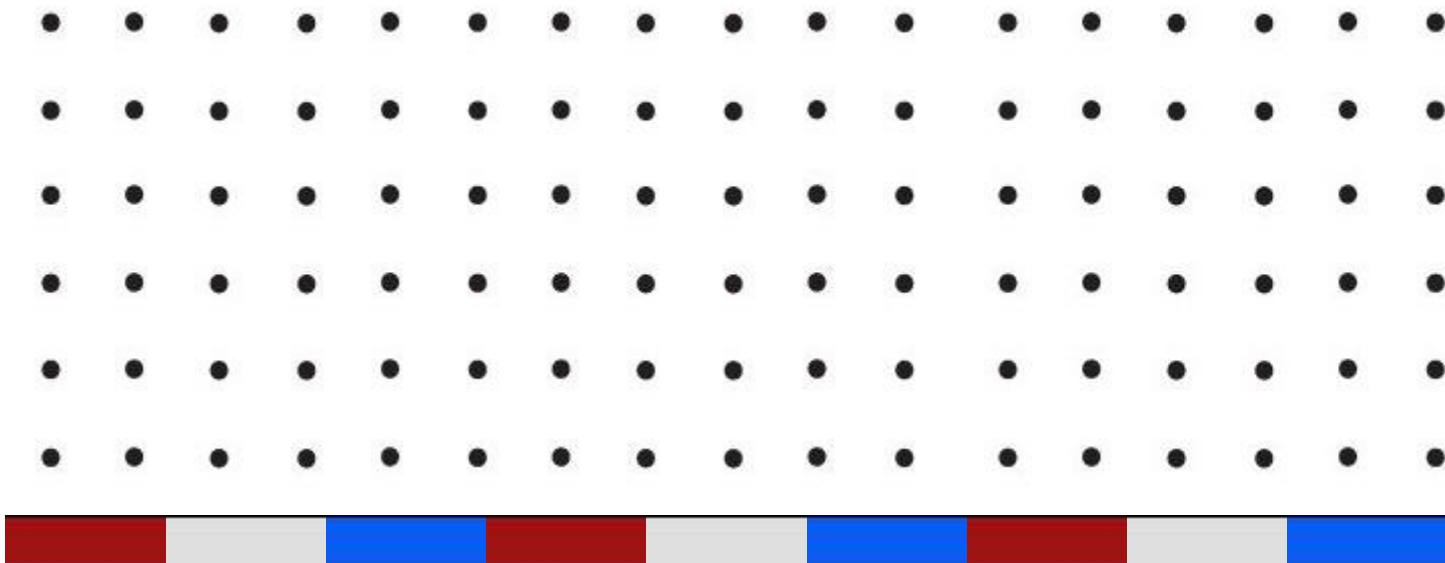
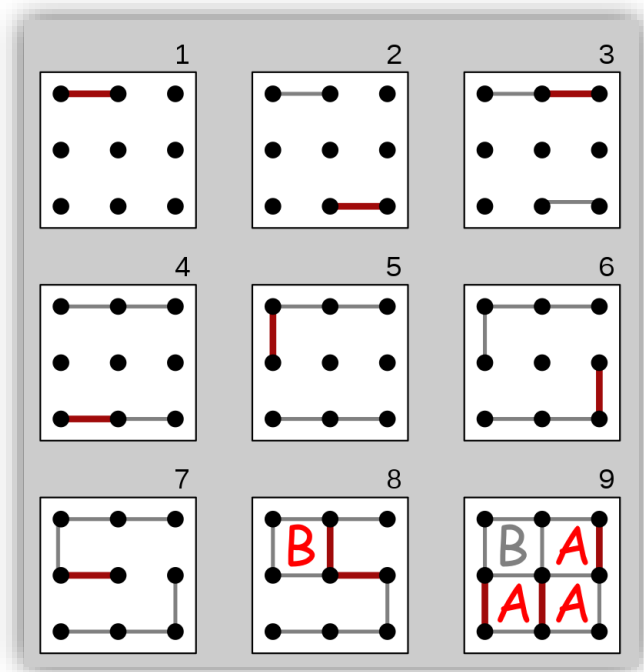
Dots and Boxes is a popular game that is played by two or more players. It was first published in the 19th century by French mathematician [Édouard Lucas](#), who called it la Pipo pipette. The game starts with an empty grid of dots. Two players take turns adding a single horizontal or vertical line between two un-joined adjacent dots. A player who completes the fourth side of a 1×1 box earns one point and takes another turn. A point is typically recorded by placing a mark that identifies the player in the box, such as an initial. The game ends when no more lines can be placed. The winner is the player with the most points. – [Reference](#).

This project aims to create a simple game that follows the same technique used in the original Dots and Boxes game. However, instead of using pencils and papers players would only need a computer and a mouse to enjoy the game.

We used [C programming language](#) to write most of the game functions and techniques, it's such a great language that has a variety of resources and libraries that were very useful during the development of the game. Additionally, CSS - [Cascading Style Sheets](#) – language which is responsible for making the GUI look nicer.

The game is a GUI-based software, thanks to [GTK library](#) and its wonderful documentation that allowed us to build such a nice interface with least effort and time.

Finally, this is the first release of the game. We made sure the players won't have issues running and playing the game. However, like all software, it's expected to find bugs. Please let us know in case you find one or would like to add more features to the game. Enjoy!

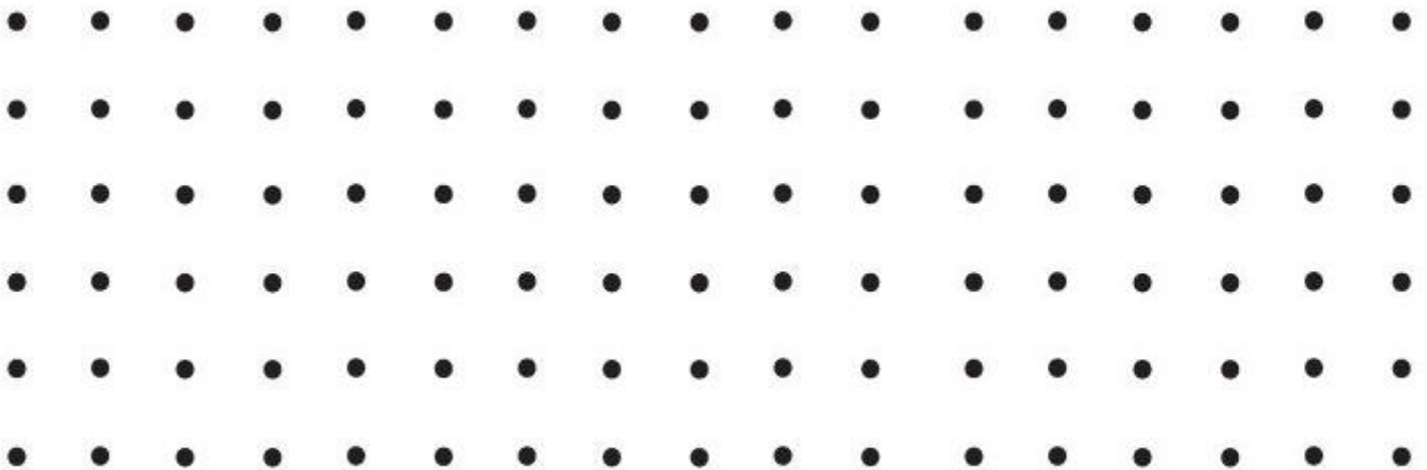


## 2. Features:

TITLE	DESCRIPTION				
DIFFICULTY LEVELS	<p><b>Beginner</b> In beginner mode, a 2x2 game board is created with four boxes. User can select that choice when he wants to play a fast game or test game features.</p> <p><b>Expert</b> In expert mode, a 5x5 game board is created with 25 boxes and 60 lines. User can compete with his friends or with the computer!</p>				
GAME MODES	<p><b>Single Player</b> Play against the computer. Notice that computers play randomly and are very weak. The game is originally created to be multiplayer.</p> <p><b>Multiplayer</b> Play against your friend. ONLINE will be available in next releases!</p>				
SAVE & LOAD	Users have THREE saving slots, meaning that they can save three games at the same time. After saving the game, they can load it and continue the match.				
UNDO & REDO	Users can undo the game till the first move while the game is running. Undone game can be redone if you don't make a new move.				
SCORING	<p>For every box the user completes, he gets a point score.</p> <p><b>Winner</b> Top scoring player wins the game when all boxes are created.</p>				
TOP 10	Top 10 players who won with best scores have their name listed on the top 10 tab. List updates dynamically every time the user clicks on the tab.				
GRAPHICAL USER INTERFACE	<p>GTK-3 is used to build the GUI.</p> <table><tr><td><b>NEW GAME</b> Where user starts a new game with customized settings.</td><td><b>LOAD GAME</b> Where user loads a previously saved game.</td><td><b>SAVE GAME</b> Where user saves the current game</td><td><b>TOP 10</b> Where top 10 scoring players names are listed.</td></tr></table>	<b>NEW GAME</b> Where user starts a new game with customized settings.	<b>LOAD GAME</b> Where user loads a previously saved game.	<b>SAVE GAME</b> Where user saves the current game	<b>TOP 10</b> Where top 10 scoring players names are listed.
<b>NEW GAME</b> Where user starts a new game with customized settings.	<b>LOAD GAME</b> Where user loads a previously saved game.	<b>SAVE GAME</b> Where user saves the current game	<b>TOP 10</b> Where top 10 scoring players names are listed.		
CASCADING STYLE SHEETS	CSS is used to make the GUI look beautiful and more entertaining. It is responsible for all styles in the GUI.				



SQL DATABASE	<p>SQLITE3 is used to store saved games and players scores. MySQL will be used in next versions when the game ONLINE feature is added.</p>
LOGGING	All actions are logged in a text file for debugging purposes
FILES	Application code is split into multiple files for organization purposes.
MUSIC	<p>Background music can be turned on or off during the game by clicking on the speaker icon on top right of the window.</p> <p>When music is turned off, user starts to hear nice sound effects on each move.</p>
GAME ICON	A nice-looking icon is used. It represents a dot with a box inside.
BUTTONS EFFECTS	When a user hovers over or clicks on a button, its style changes to maximize the entertaining and visual satisfaction.
CUSTOM CURSOR	To give users a better experience, we added a custom cursor that looks like a box.
MULTI-USAGE BUTTONS	<p><b>NEW GAME BUTTON</b> Changes to `Continue` when a game is in progress and user clicks on a tab button. When he clicks on `Continue` it takes him back to the game.</p> <p><b>REDO &amp; UNDO BUTTONS</b> When the game finishes, the buttons change to `Main Menu` and `Exit Game`</p>
POP-UP MESSAGES	Error and info messages are displayed in a pop-up label below tab buttons.



### 3. Design Overview:

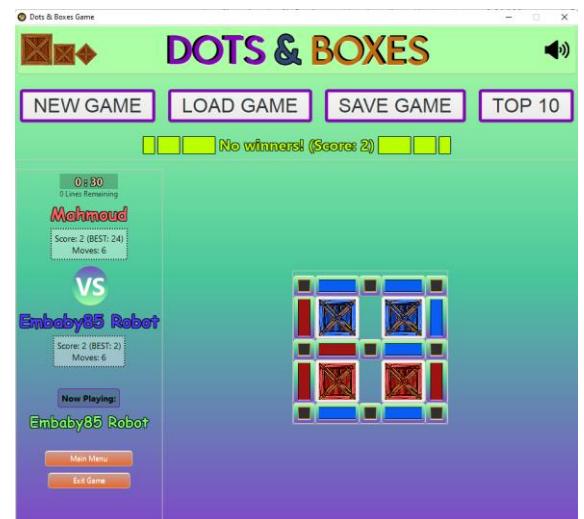
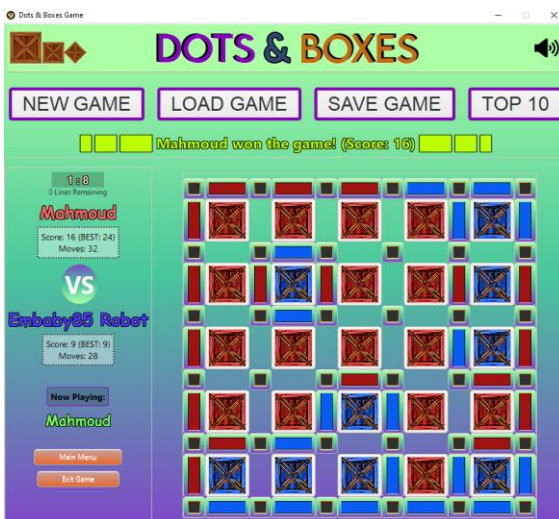
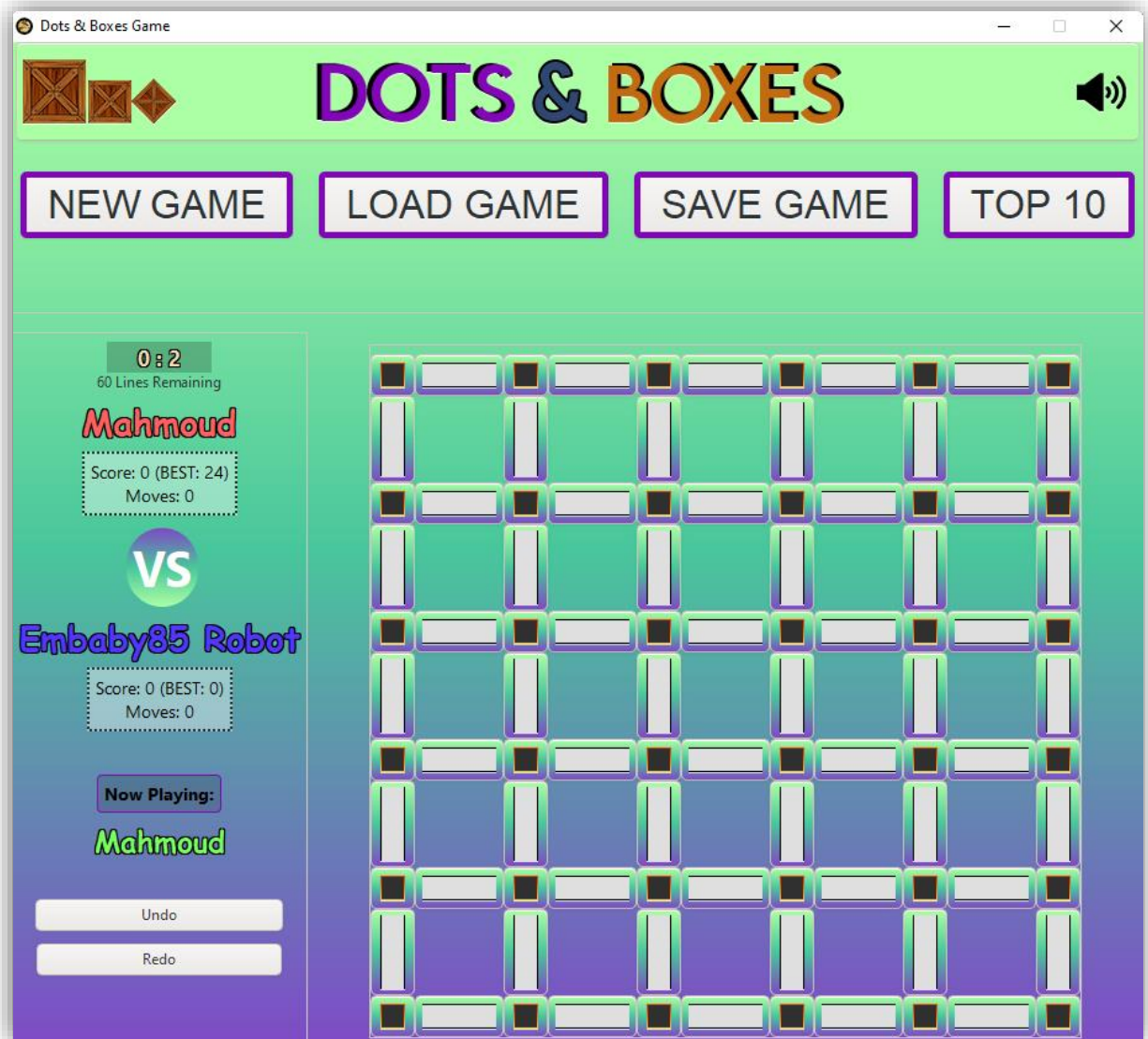
[GTK library](#) is used to build the program GUI, and [CSS](#) is responsible for the beauty!

## Main Window

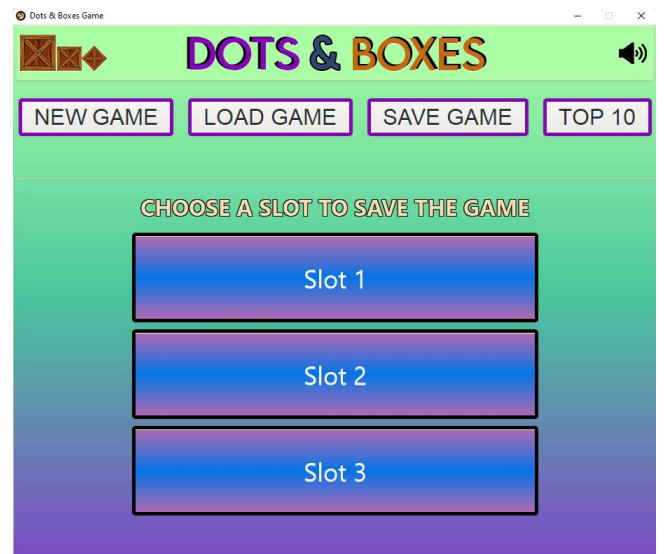
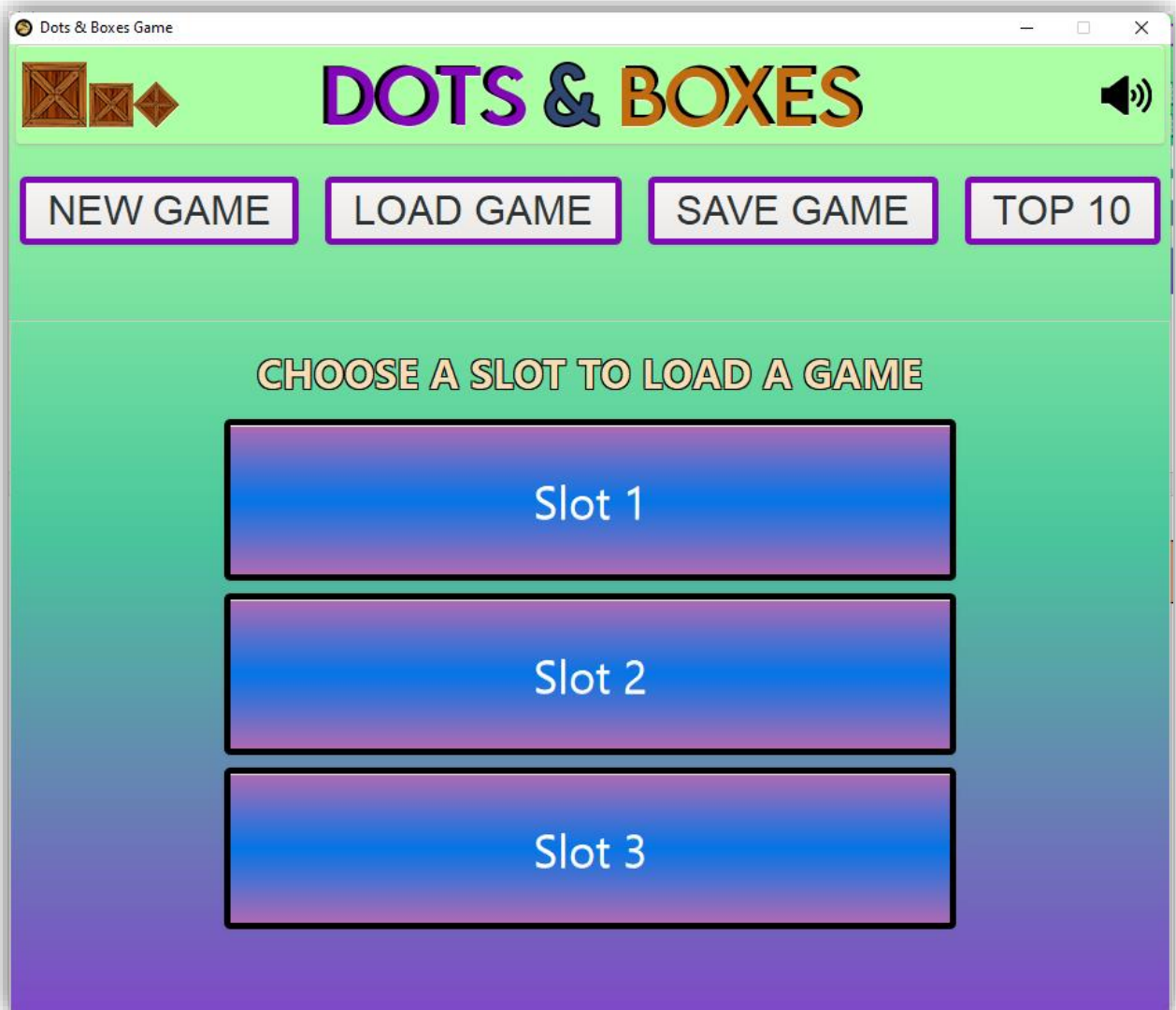




# Game Tab




# Load & Save Tabs




# Top 10 Tab

Dots & Boxes Game



# DOTS & BOXES



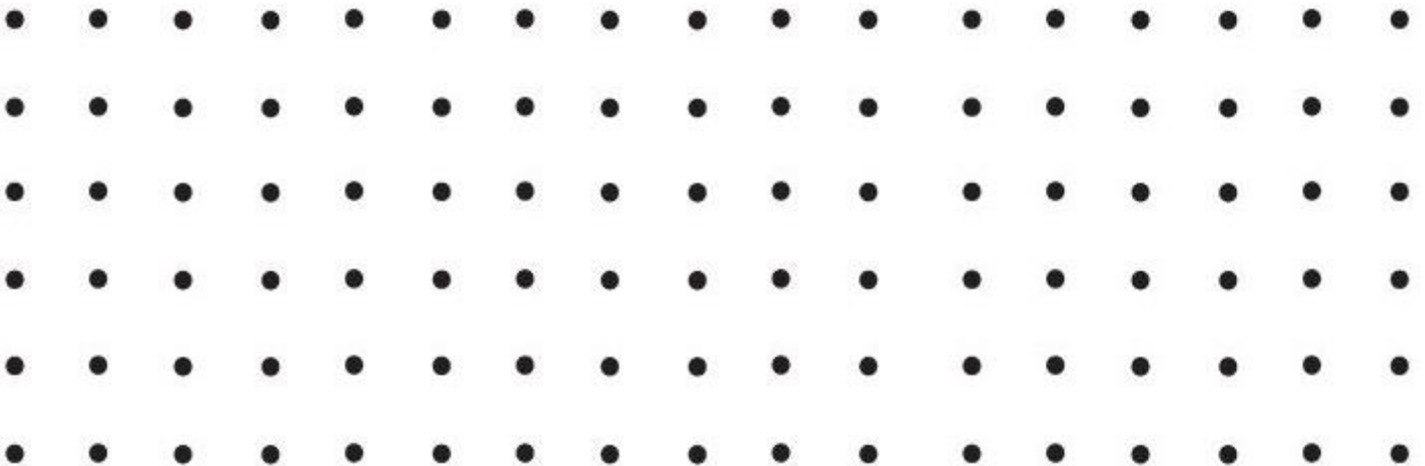
NEW GAME

LOAD GAME

SAVE GAME

TOP 10

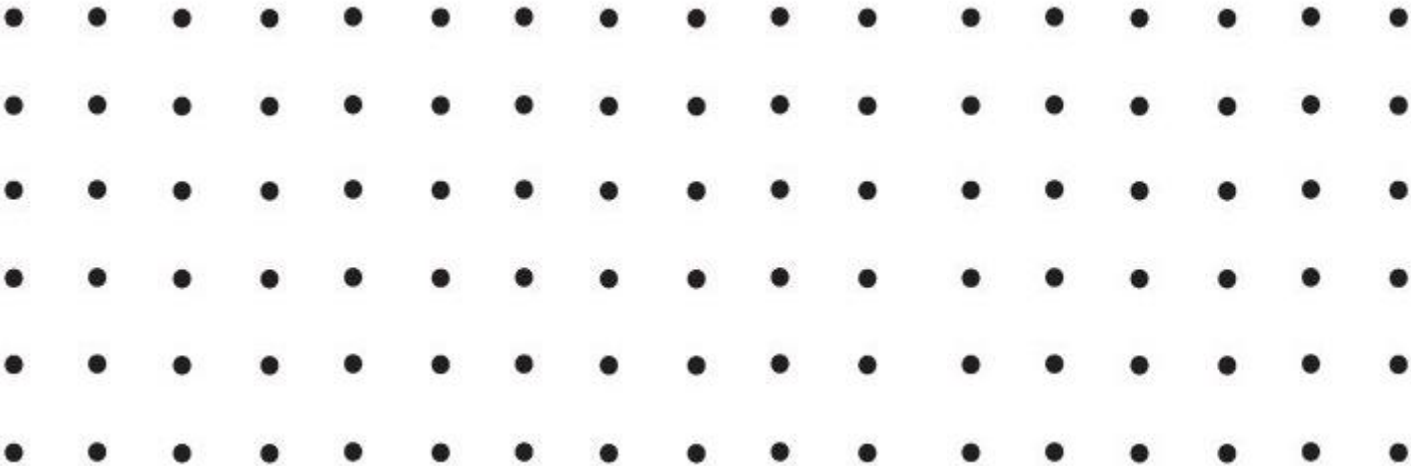
1	Mahmoud	24
2	Emad	20
3	Tarek	19
4	Nadia	15
5	Alaa	13
6	Sascha	11
7	EmbaBy85 Bot	10
8	Eric	3
9	Joseph	2
10	Wael	2





4. Assumptions:

TITLE	ASSUMPTION
OPERATING SYSTEM	<p>The user runs the game on a computer running</p> <ul style="list-style-type: none"><li>- Windows 7, 8, 10 or 11 OS.</li><li>- Linux OS.</li></ul> <p>MacOS is not supported on this version.</p>
NAME	<p>The user enters letters, numbers, or characters.</p> <p>Arabic, Chinese, and Latin characters are not supported on this version.</p> <p>Players names are not similar.</p>
REQUIRED LIBRARIES	<p>The user has installed the required libraries to run the software.</p> <ul style="list-style-type: none"><li>- GTK-3.0</li><li>- Pthread</li><li>- Libgio-2.0</li><li>- Libglib-2.0</li></ul>
FILES	<p>The user doesn't delete or move any file from project or software directory.</p>
COMPILATION	<p>OTHER LINKER OPTIONS ==&gt; <code>-lsqlite3 `pkg-config --libs gtk+-3.0`</code></p>



## 5. Data Structure:

### SQLITE3 DATABASE

Saved games and players scores are saved in a SQL database file, each in a table structured as the following:

Name	Type	Schema
Tables (2)		
game		CREATE TABLE `game` (id INT NOT NULL, difficulty INT, player1 TEXT, player2 TEXT, array TEXT, playingBIT INT, gametime INT, PRIMARY KEY (`id`))
id	INT	"id" INT NOT NULL
difficulty	INT	"difficulty" INT
player1	TEXT	"player1" TEXT
player2	TEXT	"player2" TEXT
array	TEXT	"array" TEXT
playingBIT	INT	"playingBIT" INT
gametime	INT	"gametime" INT
scores		CREATE TABLE scores(name text, score int)
name	text	"name" text
score	int	"score" int
Indices (0)		
Views (0)		
Triggers (0)		

Table Name	Structure		
`game`	ID	INT	Slot Number
	Difficulty	INT	Beginner = 2 Expert = 5
	Player1	TEXT	Player1 Name
	Player2	TEXT	Player2 Name
	Array	TEXT	1D array containing the game bits.
	playingBIT	INT	0 -> Player1's turn 1 -> Player2's turn
	gametime	INT	Game time in seconds
`scores`	Name	TEXT	Player Name
	score	INT	Player Score



## GLOBAL VARIABLES

GtkWidget	*window, *table, *logo, ...	GTK Objects	
CONST CHAR	BOTNAME	Computer's Name	
CONST CHAR	DBNAME	DB File Name	
INT	nowPlayingBIT	0 = Player 1	1 = Player 2
INT	boxesY / boxesX	Number of boxes in X and Y axis.	
INT	Game_level	2 = BEGINNER	5 = EXPERT
INT	Game_mode	1 = SINGLE	2 = MULTIPLAYER
INT	Game_in_progress	Bitwise 1 / 0	
INT	Game_time	Time in seconds	
INT	Game_bits[250][250]	Array of game bits	
INT	storedMoves[100][100][100]	Array of arrays containing the whole game bits on each bit to be used when user UNDO or REDO the game	
INT	playingHistory	Stores who played each game, used when the user UNDO or REDO the game.	
SQLITE3	*db	Database Instance	
Pthread_t	musicThread	Thread where background music can loop infinitely without affecting GTK loop.	
struct	player	Stores data for each player: <ul style="list-style-type: none"> <li>- currentScore</li> <li>- bestScore</li> <li>- name</li> <li>- Number of moves</li> </ul>	

Global variables are initially created in the file `variables.h` which is called before the program functions are initiated. Then they are used in functions and work together to make the program work without crashing.



## 6. Main Functions:

### FILE: Main.c

#### Main

Initializes the game window and all settings, calls necessary functions to start the game and listens to user's inputs and selections to update the window status (objects visibility, colors, etc.) according to it.

#### Menu\_Button\_Click SaveLoad\_Button\_Click Selection\_Click

Callback functions, called when user interacts with GUI buttons to perform certain action or make a selection.

#### Timer

Called every second to update the time of game, it increments the time by one and updates the label on game tab every time it is called.

The main function of Main.c is starting and maintaining the game while it the application is open. It has the main GTK loop. It is the heart of the application interface and functions.

### FILE: database.h

#### getScore (char \*name)

Fetches score of a player from DB and returns an integer.

#### updateScore (char \*name, int score)

Updates score of a player in DB if new score is greater than initial score.

#### saveGame (int id, int gamelevel, char array[], int nowPlaying)

Saves the current game data into DB to be reloaded later by the user.

#### loadGame (int id)

Fetches game data from DB and loads it.

#### createPlayer (char \*player1\_name, char \*player2\_name)

Updates Players struct with new players names and scores.  
Plays an important role in stopping case sensitivity in players names.

The main function of database.h is to create and prepare the database and tables to be ready to save, load or read data from it when other functions require them.



## FILE: utilities.h

**load\_css**  
(void)

Applies CSS file codes to GTK window, called on software initialization.

**play\_music**  
(int i)

For I = 1: background music is turned on or off.  
For I = 2: Sound effects are played.

**storeMove**

Called on each move to store game data to an array to be used when the user decides to UNDO or REDO the game.

**undoRedo**  
(GtkWidget \*widget, int undo)

Callback function of UNDO and REDO clicking buttons.  
For I = 1: Undo takes place.  
For I = 0: Redo takes place.

**updateGameInfo**  
(int stop)

Updates labels in game (Players names, scores, remaining bricks, turn, etc.)  
For I = 1: Stops the game and announces the winner.

**nextTurn**

According to the algorithm, it chooses the player who should play next turn. If it is computer's turn, it plays the game.

**toggleBrick**  
(GtkWidget \*widget, int x)

Callback function when clicking on a line, toggles the line and updates boxes if required.

**createGame**

Responsible for creating game grid, getting all settings ready and making the game playable. All according to user's selections in main window.

The main function of utilities.h is to create helping and necessary functions required to make the game playable and help the main function by supplying these functions and running them when required to keep the game flow.

## FILE: logging.h

**Logging**  
(char \*type, char \*text)

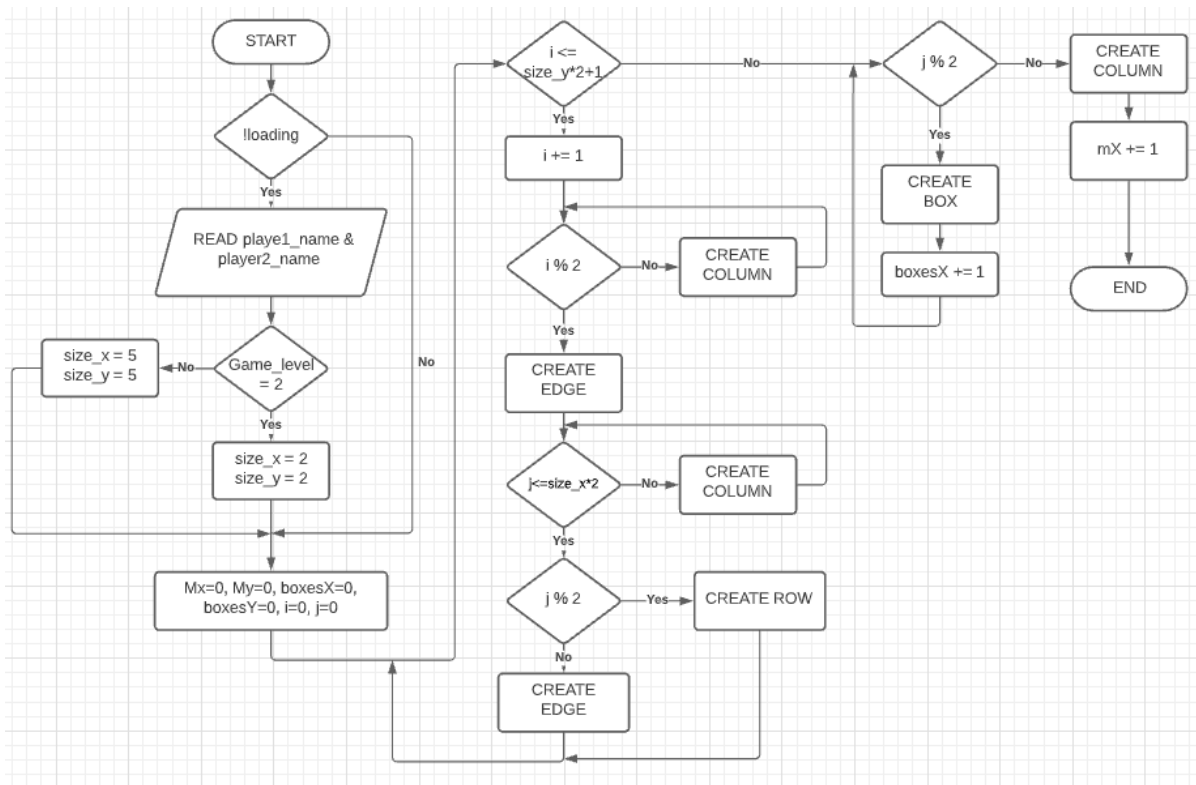
Logs playing history into a text file for debugging purposes.

The main function of logging.h is to maintain the function `logging()` and run it when required to keep the game logged so that we can trace errors and fix them.





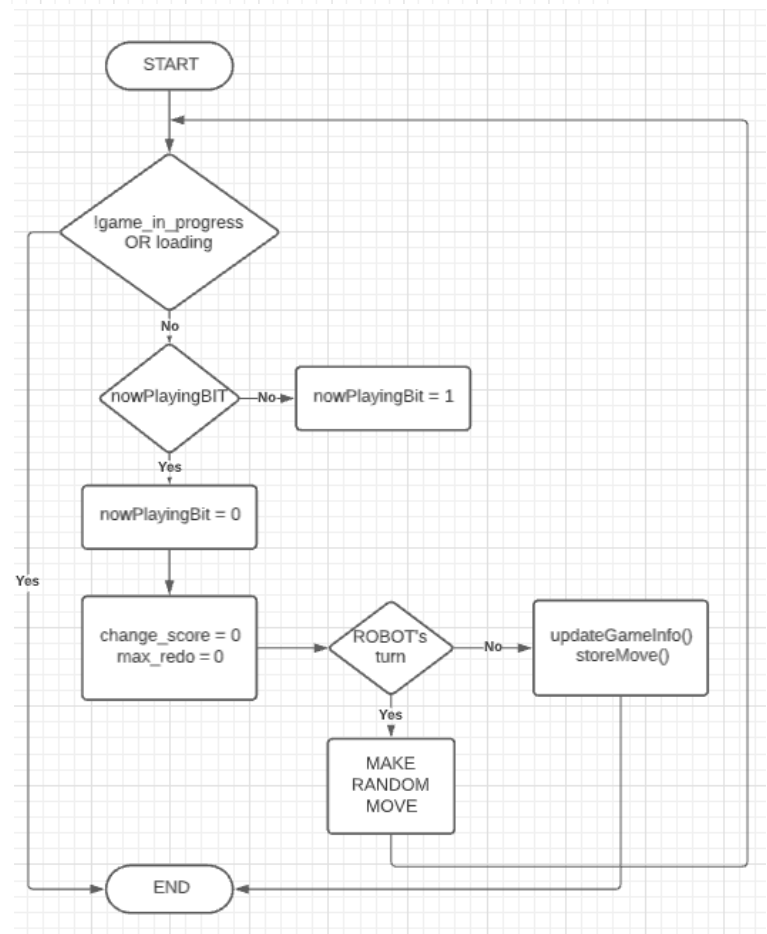
## 7. Flow Charts:

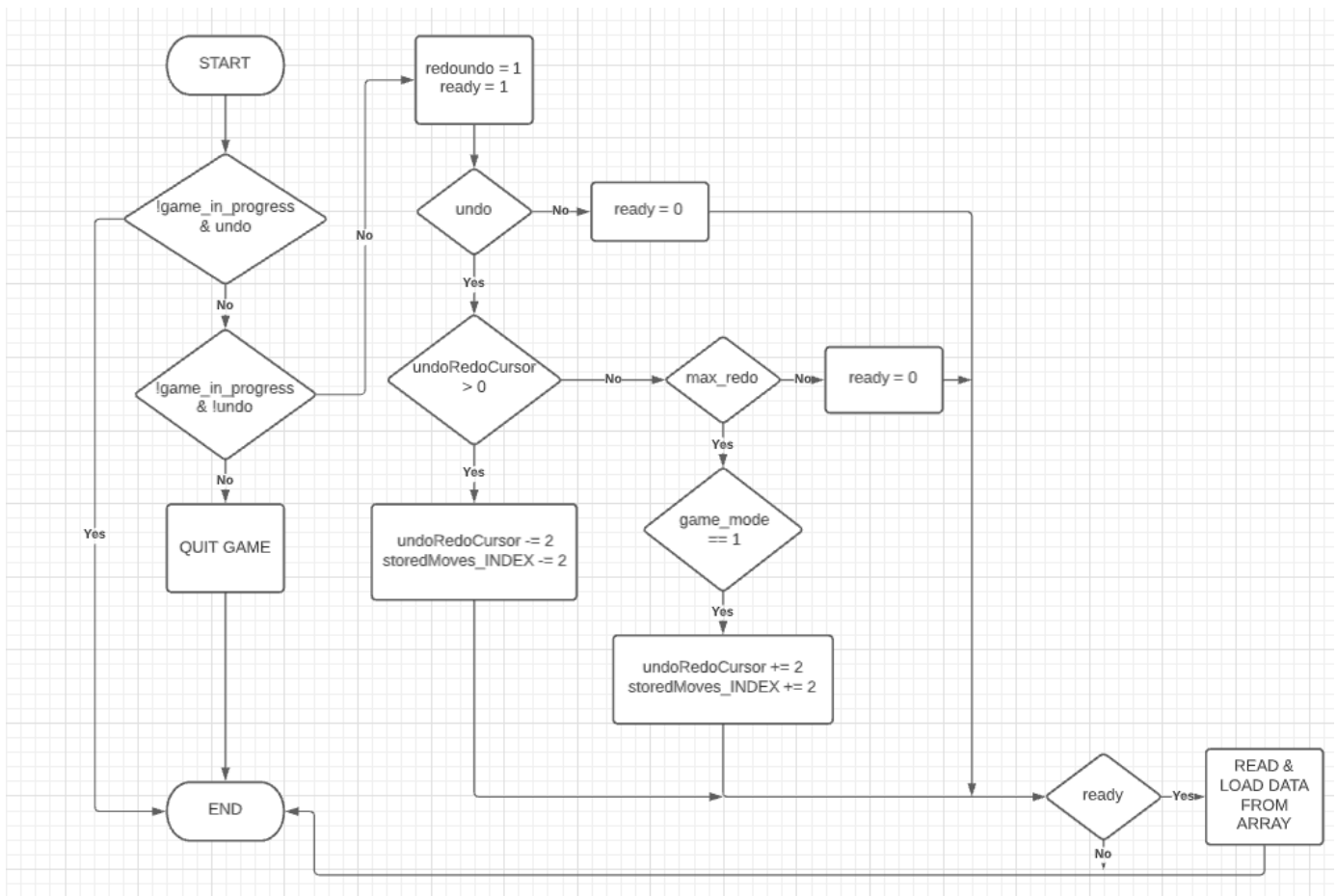


The **upper flowchart** shows the game creation process. Some variables are initiated then nested loops take place and create the nice-looking grid with all buttons, images and boxes!

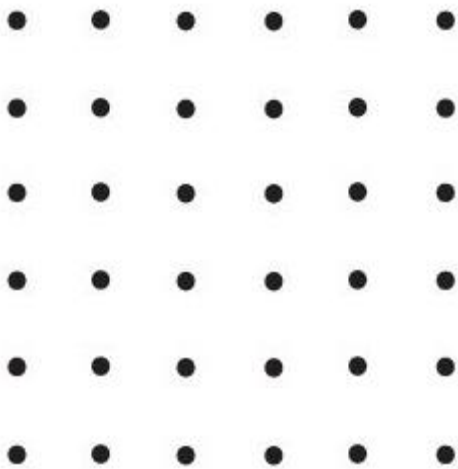


The **right flowchart** shows the function that alternates players turns and plays for the computer a random move if it's computer's turn.

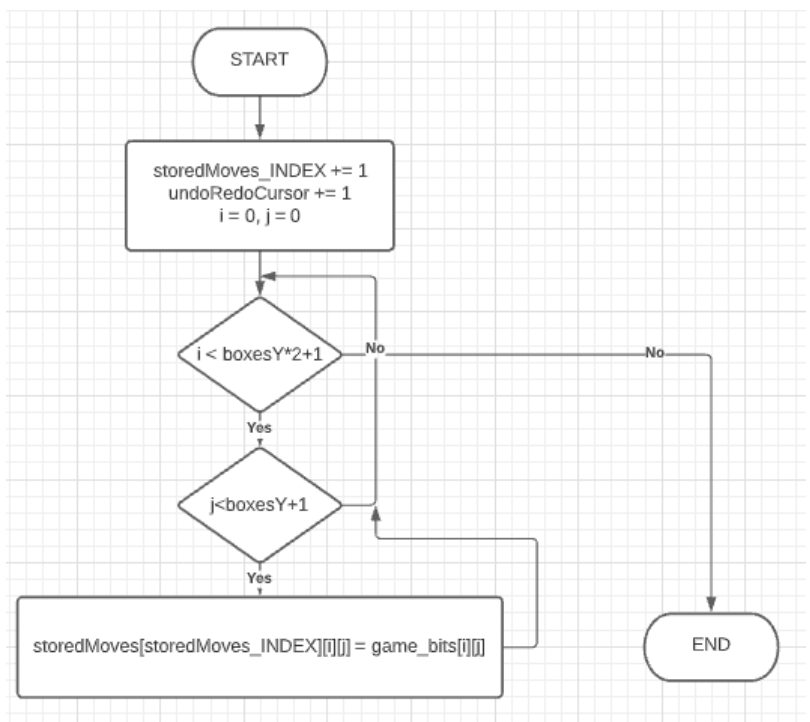


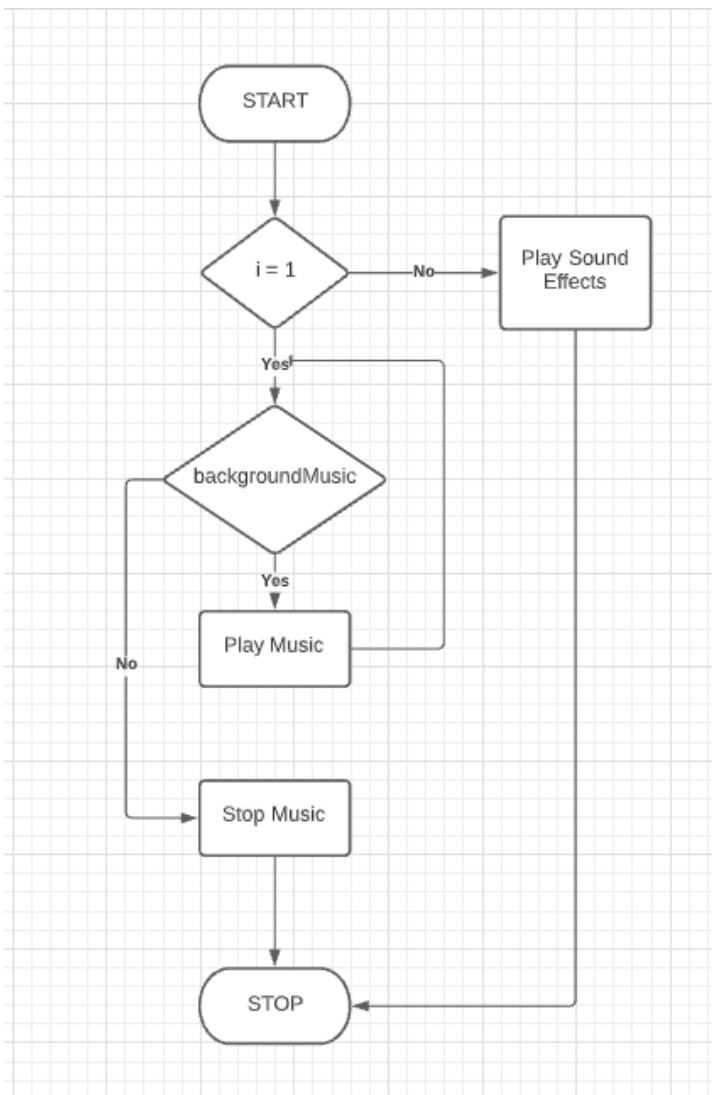


This flowchart shows the Redo & Undo function. It first selects the index of data needed to be loaded then loads the data in a nested loop.



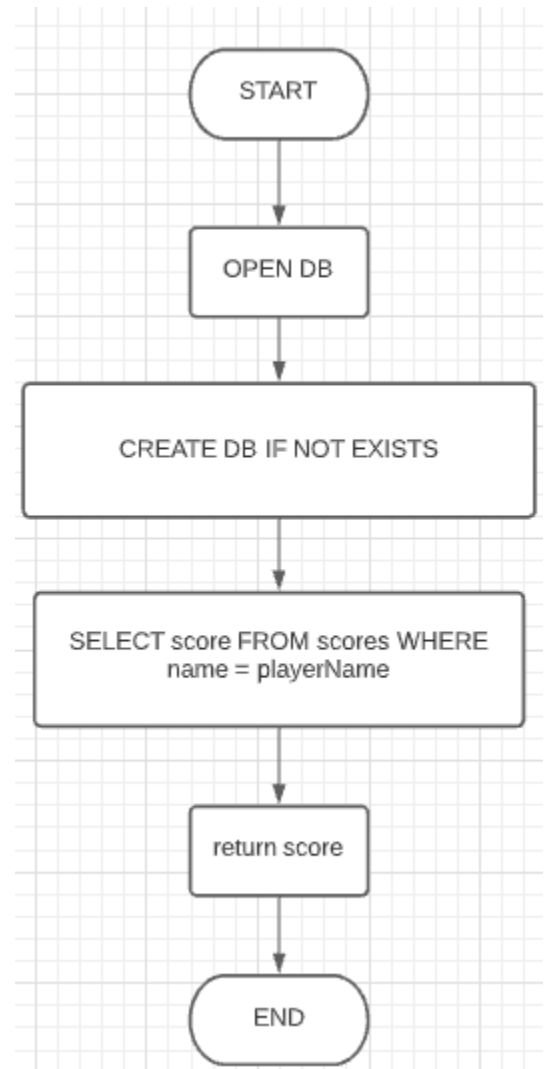
While the flowchart on the **right side** shows the function that saves the game to an array of arrays to be used when needed for redo and undo functions.





The flowchart on the left side shows the function that toggles music on and off.

The flowchart on the right side shows how the function ``getScore(playerName)`` fetches player scores from database



# USER MANUAL

## GAME SETTINGS

### 1. Game Level:

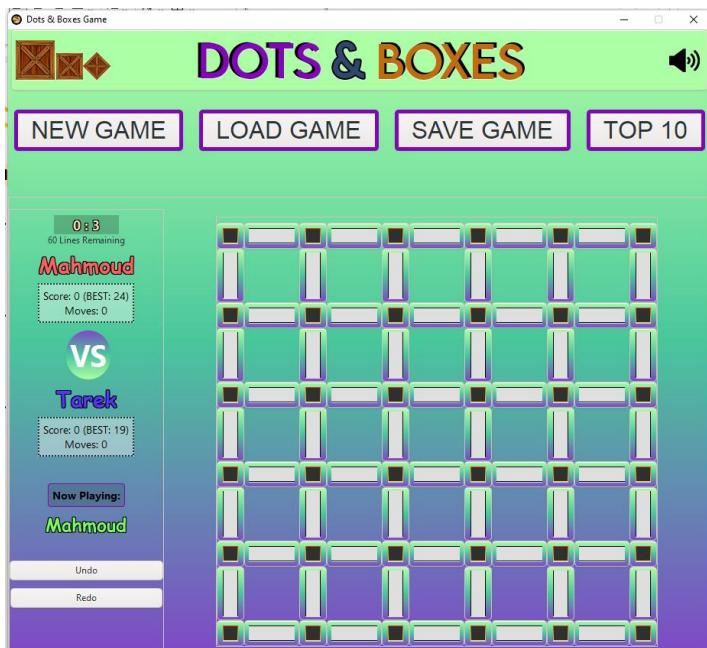
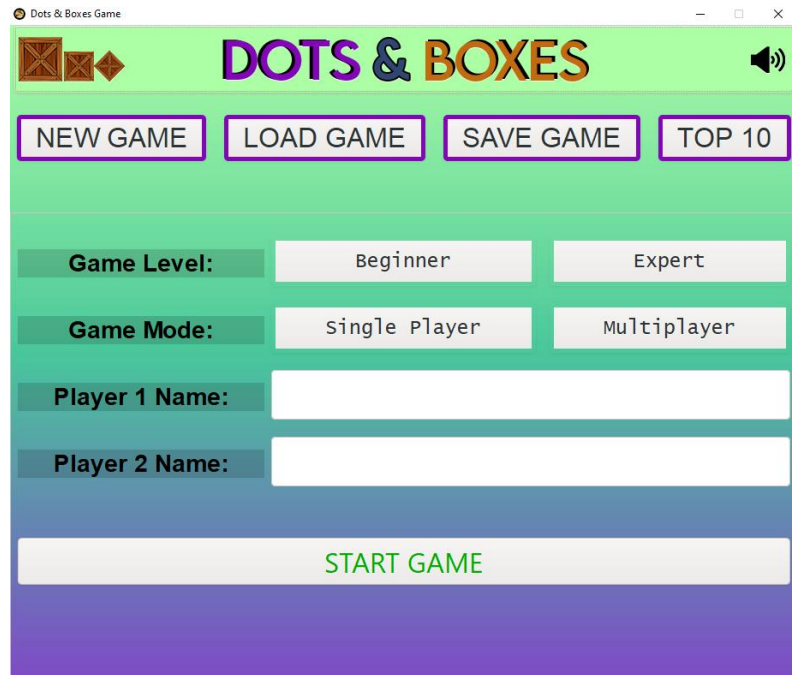
- Beginner (2 X 2 game)
- Expert (5 X 5 game)

### 2. Game Mode:

- Single Player VS Bot
- Multiplayer VS player

### 3. Players Names

Only English letters, digits and characters are allowed.



## How to play?

Dots and boxes is a simple game with a simple goal: whoever "owns" the most boxes at the end of the game wins. You and your opponent take turns drawing horizontal or vertical lines to connect the boxes. When someone draws a line that completes a box, you write your initial inside to win the box. Once all the dots have been connected, you can count up the boxes and find the winner

Each turn, draw one horizontal or vertical line to connect two dots. Early on this will be mostly random, as there are not enough lines to win any boxes. Each line simply goes from one dot to it's neighboring dot either above, below, left, or right. There are no diagonal lines.

Draw the 4th wall of a box to win it for yourself. Each box is worth one point, so write your initial in the completed box to score it for yourself. If you have two different colored pens, you can also scribble your color in to mark it as well. – [Reference](#).





## How to save and load the game?

When a game is running, go to SAVE GAME menu and select a slot to save your game into.

To load a saved game, go to LOAD GAME menu and select the slot you want to load.

You only load a game if you have at least one saved game.

Maximum number of slots is THREE.

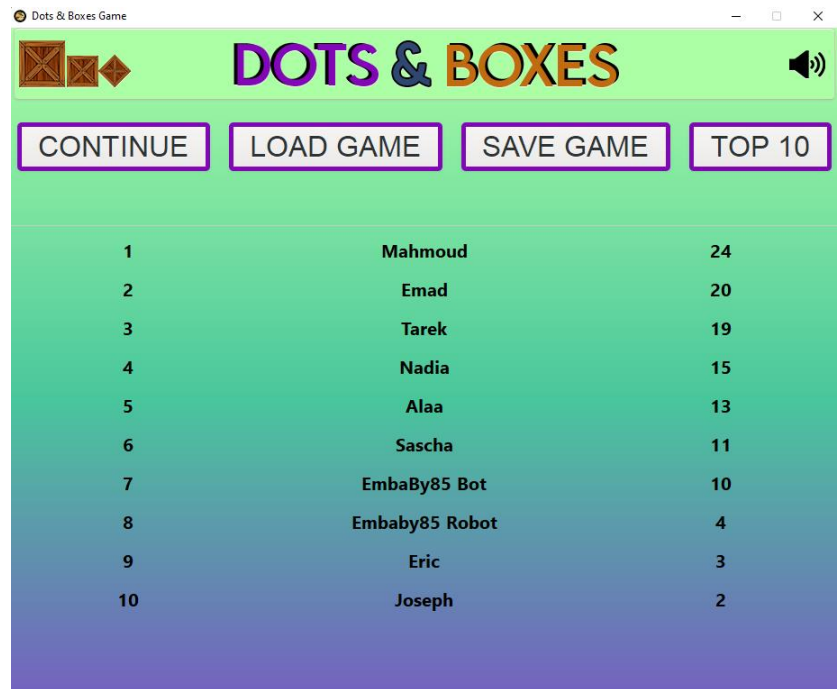
## How to put your name in TOP 10 list?

When you win a game, your score is stored in the database if the new score is greater than the score previously stored.

Clicking on TOP 10 button, the 10 players names with highest scores are displayed on the list.

Accordingly, you are required to win a game with a score higher than the 10<sup>th</sup> player to replace him and be placed in the list.

The higher you rank, the better!



Finally, I hope you enjoy the game as much as I enjoyed creating it. For me, the track wasn't easy, but I really enjoyed going through it. I learnt new technologies and algorithms that I wouldn't have learnt without taking the opportunity to work on this project.

Excited to hear your opinions!

Mahmoud Tarek Embaby.





## 9. Sample Runs:

### WINNING



### NO WINNERS

