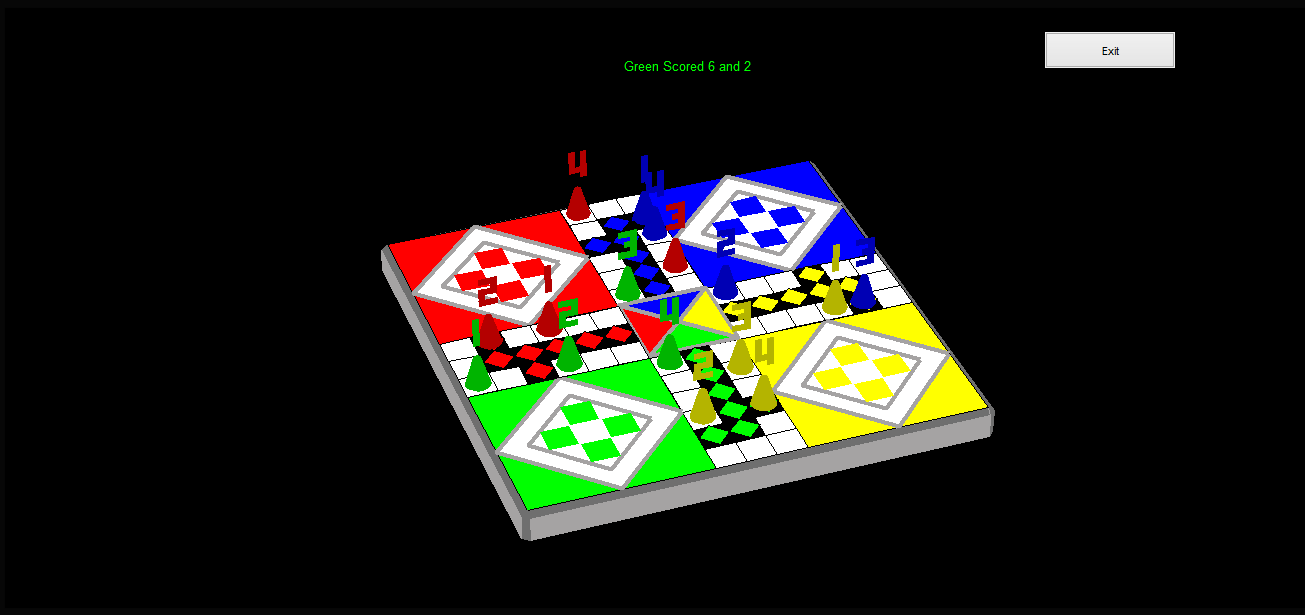
**MATLAB Project :** Ludo

**By:** Abdullah Irfan

**Roll Number:** 2018-10-0081



**Summary:** I have produced a program that simulates the game of Ludo. It is fully interactive and allows for 2-4 players to enjoy a game of Ludo. The computer randomly generates a score and the user chooses a piece to move. If no moves are possible, the user must skip the turn. Players can capture opponent pieces by landing on their spot. The game ends when one of the players successfully guides all of his/her pieces to their home.

**Usage:** The program starts by running **runme.m**. The user may either select the number of players or go with the default configuration for 4. The game is started by pressing the 'Play' button.

**Options:** The user may choose the number of players to start the game with. Once the game starts, the user has the liberty to choose which player he wants to move with a particular die score. The pieces are labelled '1', '2', '3' and '4' and the user may likewise press one of the numbers from 1 to 4 on the keyboard in order to get the game going. Alternatively, the user may press the 's' key to skip the turn in case no moves are possible.

**Features:** The die score is displayed over the Ludo board. If a 6 is scored, the player gets another turn in which case more than one scores are displayed. A player may move different pieces with different die scores in the same turn. If three 6's are scored in a row in a turn, the turn is skipped as per the game's rule. A sound is played at times when a die rolls, a piece is moved, a piece is captured or a player wins. If an opponent piece is captured, the text over the board notifies the user of this and the opponent's piece is returned to its base. Safe zones spread across the board however make a piece immune to attacks from all other players. When a turn cycles, the board rotates to the base of the player to whom the turn has shifted. When close to victory, a player cannot move a piece if the score on the die is more than the number of steps to home. When all pieces of a particular colour have reached home, the player wins and the game subsequently ends.

**Theory:** The backbone of this project is simply the list of rules of a normal Ludo game.

**Programming: Unusual Features:**

In order to cut down on lengthy processing times in loading the GUI's music, I have sampled the frequency of the music and saved it in a matrix file so that instead of having to convert the music file into numerical data for processing every time the game is run, MATLAB merely has to load the variables containing the sample rate from a saved matrix file. This saves a huge chunk of time whenever the game is run.

**Structure:**

* The main program is started by **runme.m**, which calls a script called **GUI.m**.
* The script **GUI.m** contains the code for the game's GUI and runs several media files. It is executed in **runme.m**
* The script **Graphics.m** contains the entire code for the 3D graphics of the game and creates the game's figure. The graphics are entirely hardcoded and no external images have been used in rendering the visuals. This script is called in the function **Gameplay.m**
* The script **Gameplay.m** contains all of the main variables required to run the code and is called in the function **go.m**. It also contains the main while loop that keeps the game running till a player wins.
* The script **greenMove.m** runs the green player's turn and is called in the script **Gameplay.m**
* The script **gScore.m** displays the green player's score and is called in **greenMove.m**
* The script **gCapture.m** checks if a green piece has captured any of the opponent pieces. This script is executed in the script **greenMove.m**
* The script **redMove.m** runs the red player's turn and is called in the script **Gameplay.m**
* The script **rScore.m** displays the red player's score and is called in **redMove.m**
* The script **rCapture.m** checks if a red piece has captured any of the opponent pieces. This script is executed in the script **redMove.m**
* The script **blueMove.m** runs the blue player's turn and is called in the script **Gameplay.m**
* The script **bScore.m** displays the blue player's score and is called in **blueMove.m**
* The script **bCapture.m** checks if a blue piece has captured any of the opponent pieces. This script is executed in the script **blueMove.m**
* The script **yellowMove.m** runs the yellow player's turn and is called in the script **Gameplay.m**
* The script **yScore.m** displays the yellow player's score and is called in **yellowMove.m**
* The script **yCapture.m** checks if a yellow piece has captured any of the opponent pieces. This script is executed in the script **yellowMove.m**
* The file **theme.mat** contains the sample rate of the GUI music and is loaded in **GUI.m**
* The file **gui.mat** contains the variable 'p' which determines the number of players and is saved in **go.m**.
* The function **back.m** is the call back function of the 'Exit' button within the game and is used in **Graphics.m**. It gives the variable 'done' a value of 1, which marks the end of the game and exits it.
* **edit\_pos.m** is a function that alters the position of a playing piece graphically by editing properties of specific handles and is called in **greenMove.m**, **redMove.m**, **blueMove.m** and **yellowMove.m.**
* The **go.m** function is a call back of the 'Play' button on the GUI and is called in the script **GUI.m**. It stops the GUI music and saves the number of players chosen by the user as the variable 'p' in the file **gui.mat**.
* The function **list.m** contains the list of options available in the drop down menu on the GUI. This function is called in **GUI.m** and it records the number of players chosen by the user.
* The function **out.m** is the call back function of the 'Exit' button on the GUI and is called in **GUI.m**. It simply closes the GUI window.
* **1.jpg** is a picture with a die showing a score of 1 and is processed in **GUI.m**.
* **6.jpg** is a picture with a die showing a score of 6 and is processed in **GUI.m**.
* The picture **bp.png** is the blue playing piece seen on the GUI. It is processed in **GUI.m**.
* The picture **rp.png** is the red playing piece seen on the GUI. It is processed in **GUI.m**.
* The picture **Dice.jpg** contains instructions on playing the game for the user. The picture is called to be displayed in **gameplay.m**.
* The title of the GUI window is contained in the file **Ludo.jpg** which is displayed in **GUI.m**.
* The sound file **capture.wav** is the sound effect played when a piece captures an opponent piece. The sound is processed in **Gameplay.m**.
* The sound file **move.mp3** is the sound effect played whenever a piece moves. The sound is processed in **Gameplay.m**.
* The sound **roll.mp3** is the sound effect played when a die roll occurs. The sound is processed in **Gameplay.m**.
* The **victory.wav** file contains the sound effect played when the game is over and one the players has won. It is processed in **Gameplay.m**.