University Of Dhaka

Department of Compuer Science and Engineering

CSE-1211: Fundamentals of Programming Lab

Name of the Project:

Carrom

Team Members:

1. Ahmed Nesar Tahsin Choudhury (20)
2. Mehrajul Abadin Miraj (20)
3. Jannatin Tajri (42)

Introduction

This report discusses the results of the work done in development of “Carrom” on Linux platform using C++ programming language along with SDL2 library and its extensions. Carrom is a 2D board game. The game features three modes, namely single player, two player and four player mode. In the single player mode, the carom pieces are arranged in a certain manner initially, and the player has to pot all the pieces in the shortest amount of time using the striker. In the other modes, there are two teams. The first team has to pot the white pieces, and the second team has to pot the black pieces. The team managing to pot all their pieces first wins.

Objectives

The main goal of this project for the developers was to become more familiar with the fundamentals of computer programming and development, as well as to make use of libraries like SDL2 and its extension libraries.
Our goal was to produce reusable, simple-to-implement modules. The game is simple-to-play, challenging, and quite engaging.

Project Features

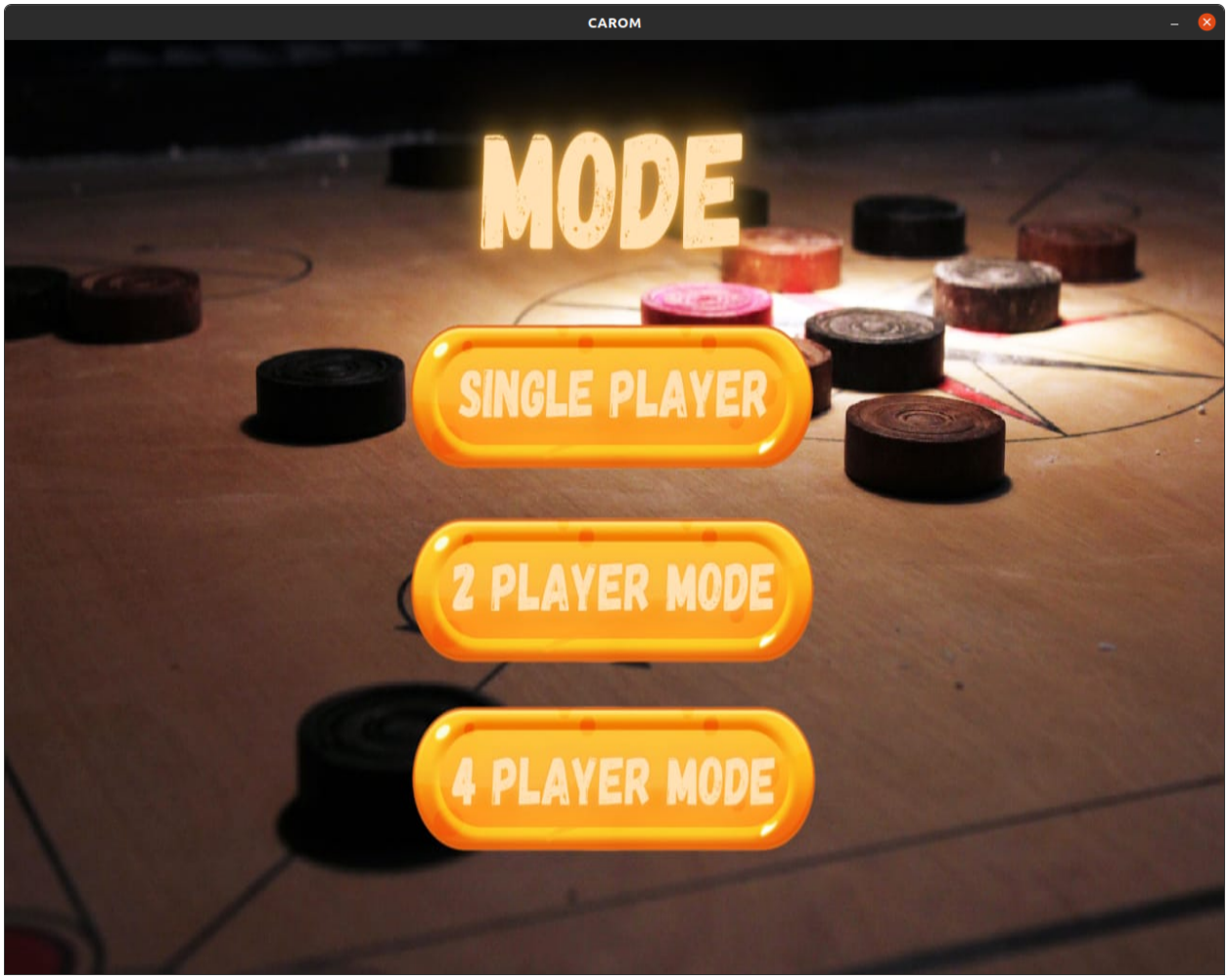
The project provides the following features:

1. Single Player Mode (Rapid)
2. Two Player Mode
3. Four Player Mode
4. Graphical User Interface
5. Leaderboard

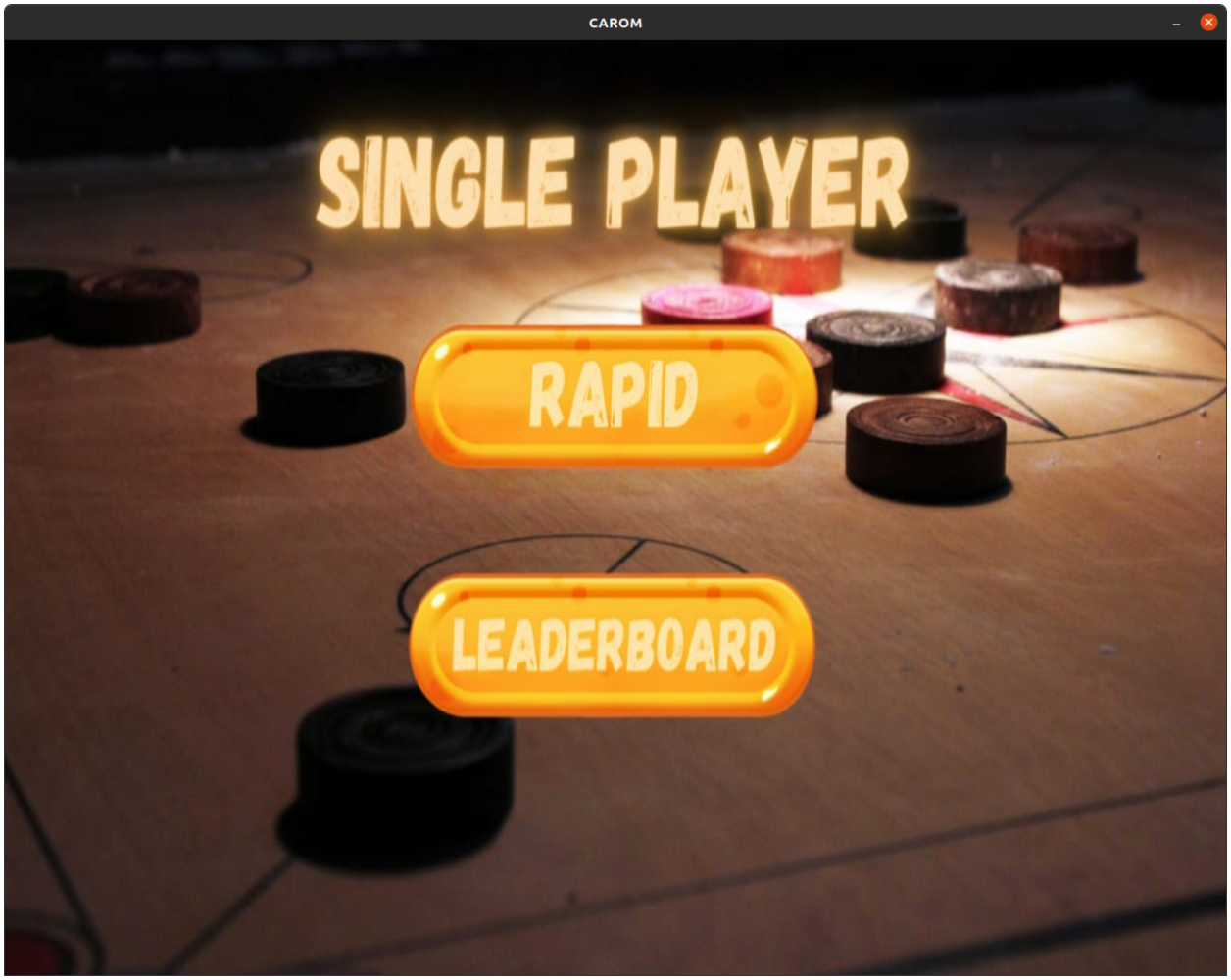
Different features are described below:

When the game starts, the main menu is displayed:

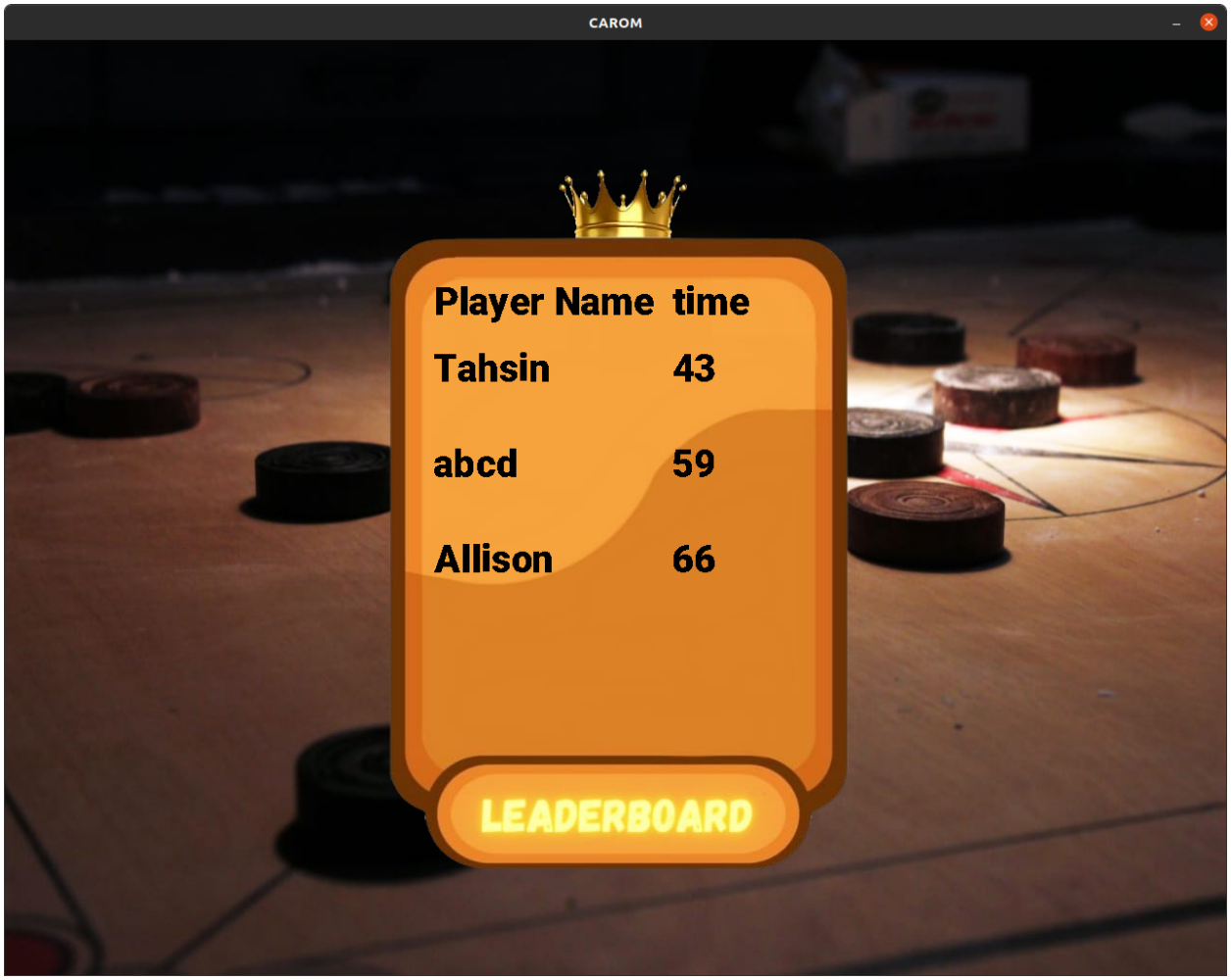
The PLAY button is glowing since the cursor is on it. The PLAY button has to be pressed to choose the game mode. Clicking on it leads to the following window:



From this window, the player has to choose which mode to play. Clicking on on SINGLE PLAYER makes the following window appear:

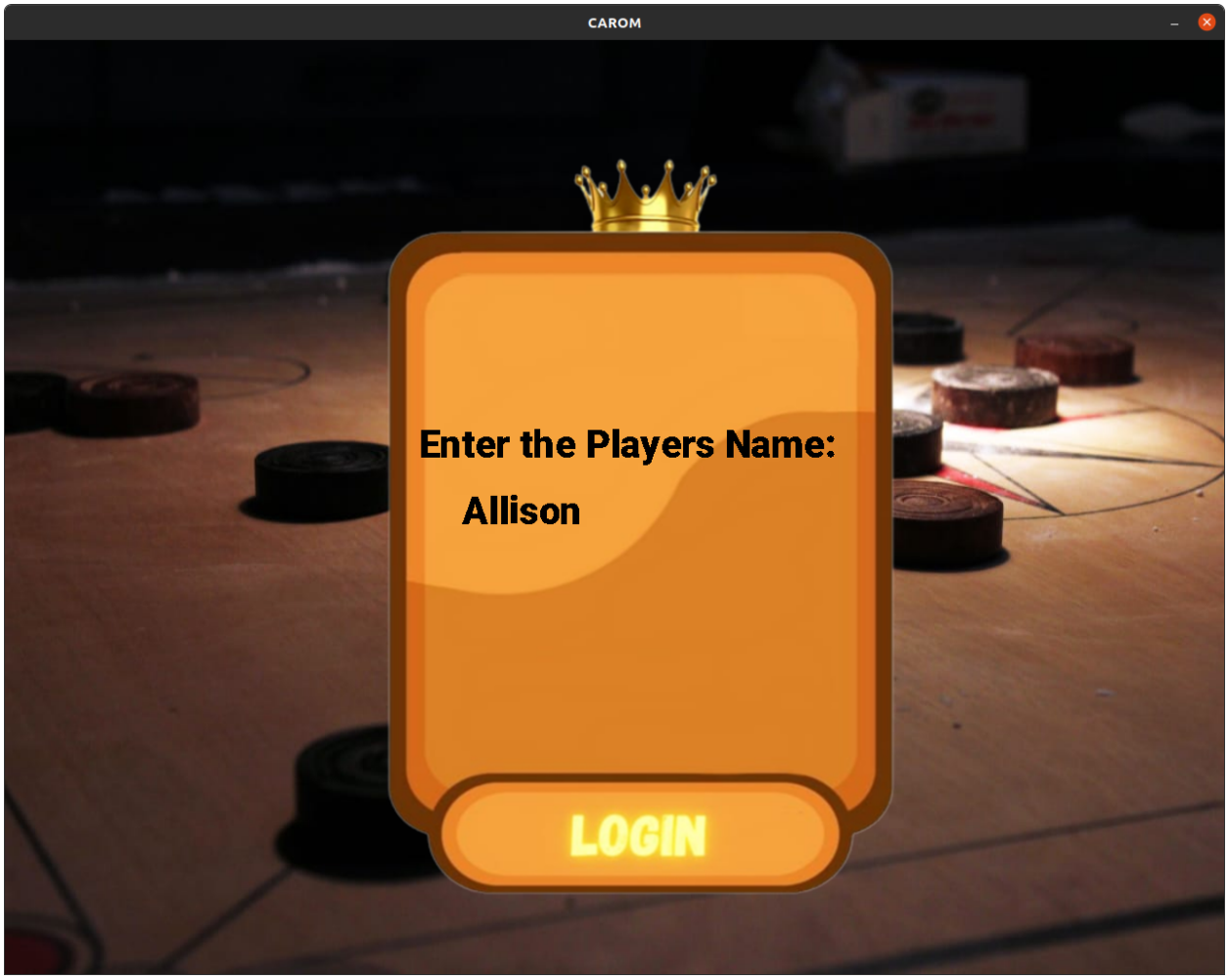


Now, the player has two options: he/she can choose to play the rapid mode, or check out the current leaderboard. Pressing on LEADERBOARD makes the following window appear:

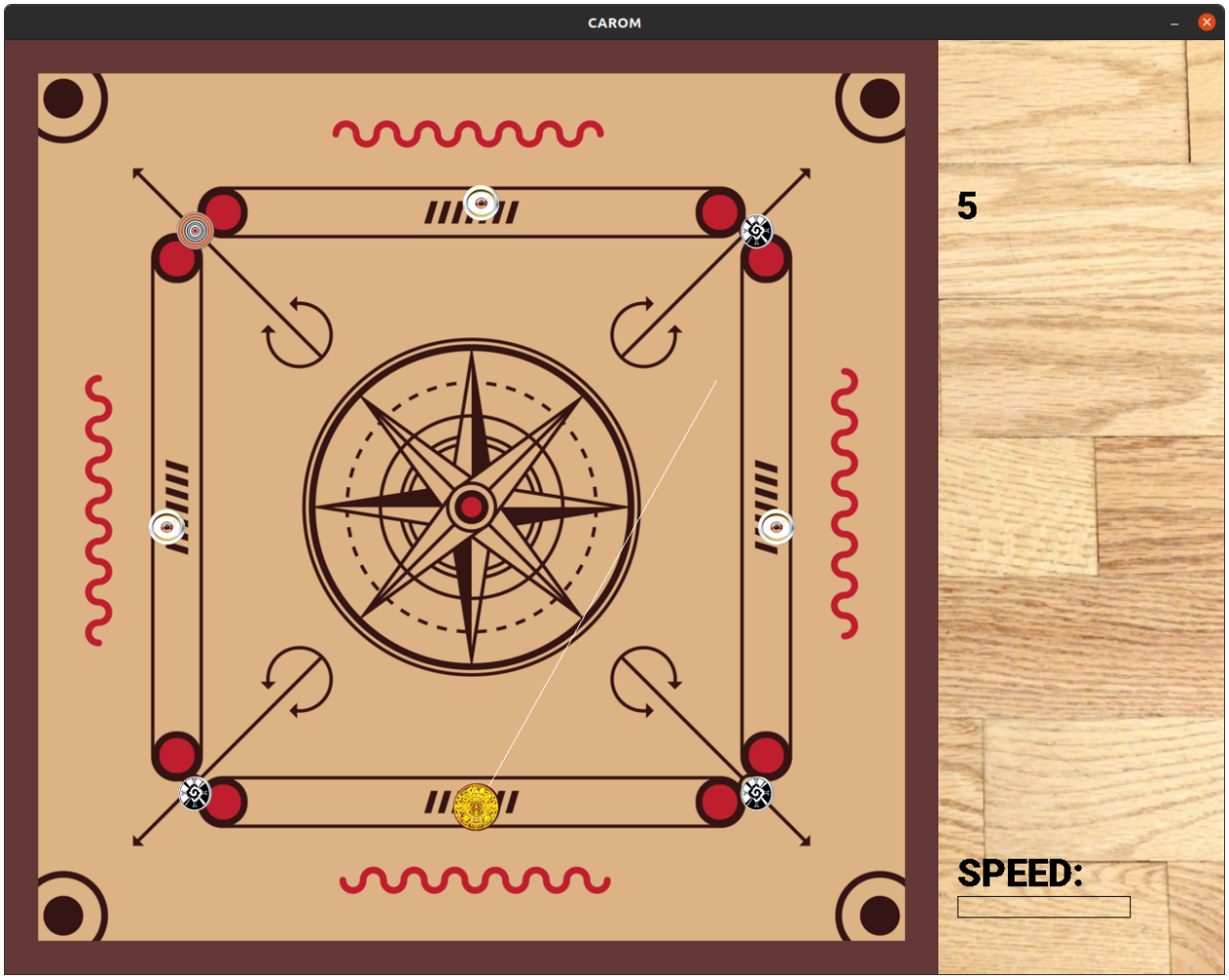


The top three players of the SINGLE PLAYER RAPID MODE are displayed here. Pressing on ESC, we can return. Then pressing on RAPID mode makes the game ask

for introduction:

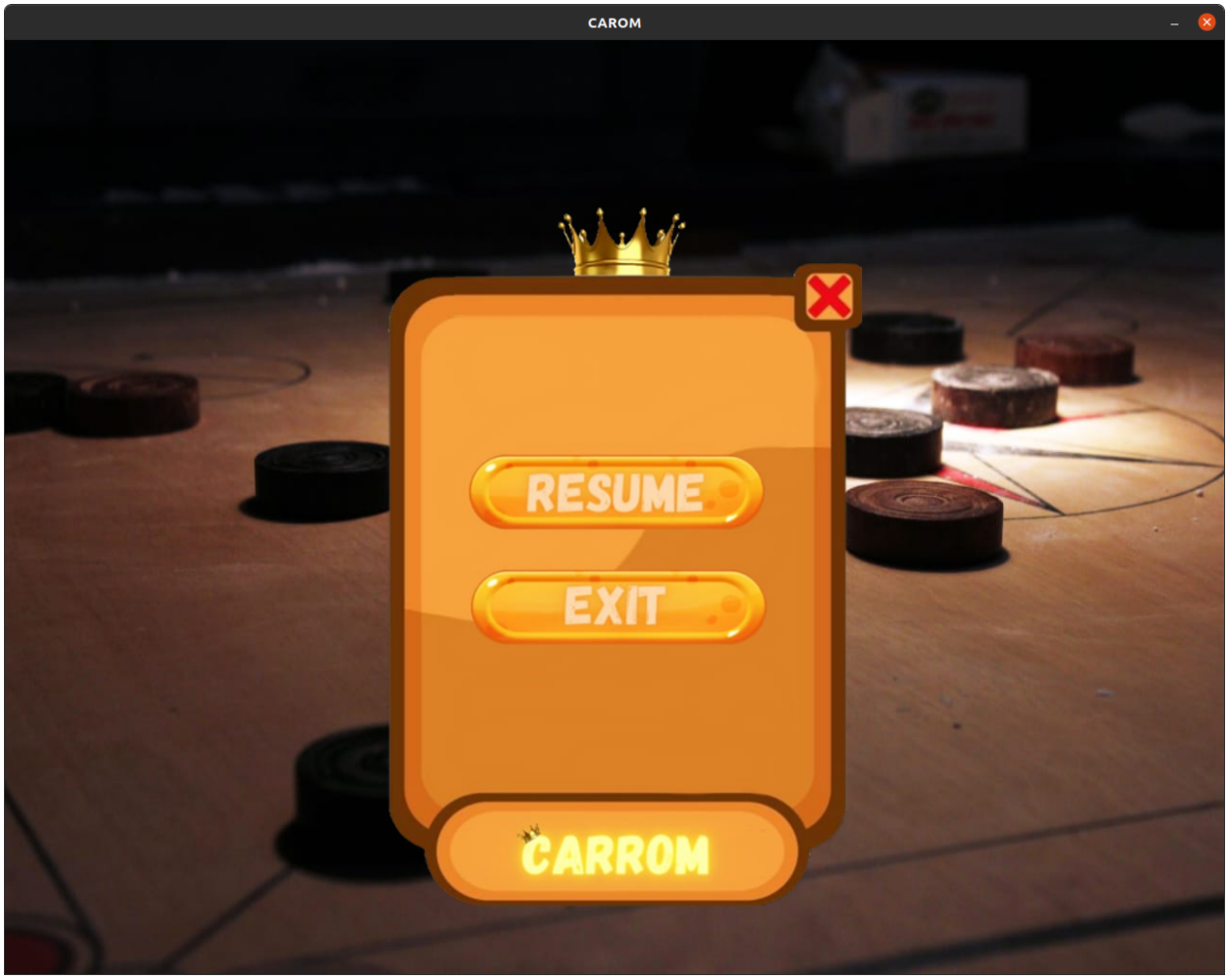


The player’s name has to be entered to start the game.



This is what the gameplay looks like. The striker can be moved by pressing LEFT or RIGHT arrow key. The direction of shooting is controlled by the mouse, and the direction can be seen by the white line, as shown in the picture above. In the right bottom of the screen, the speed bar can be seen, which indicates the speed with which the striker can be shoot. The longer the SPACE button is pressed, the higher the speed.

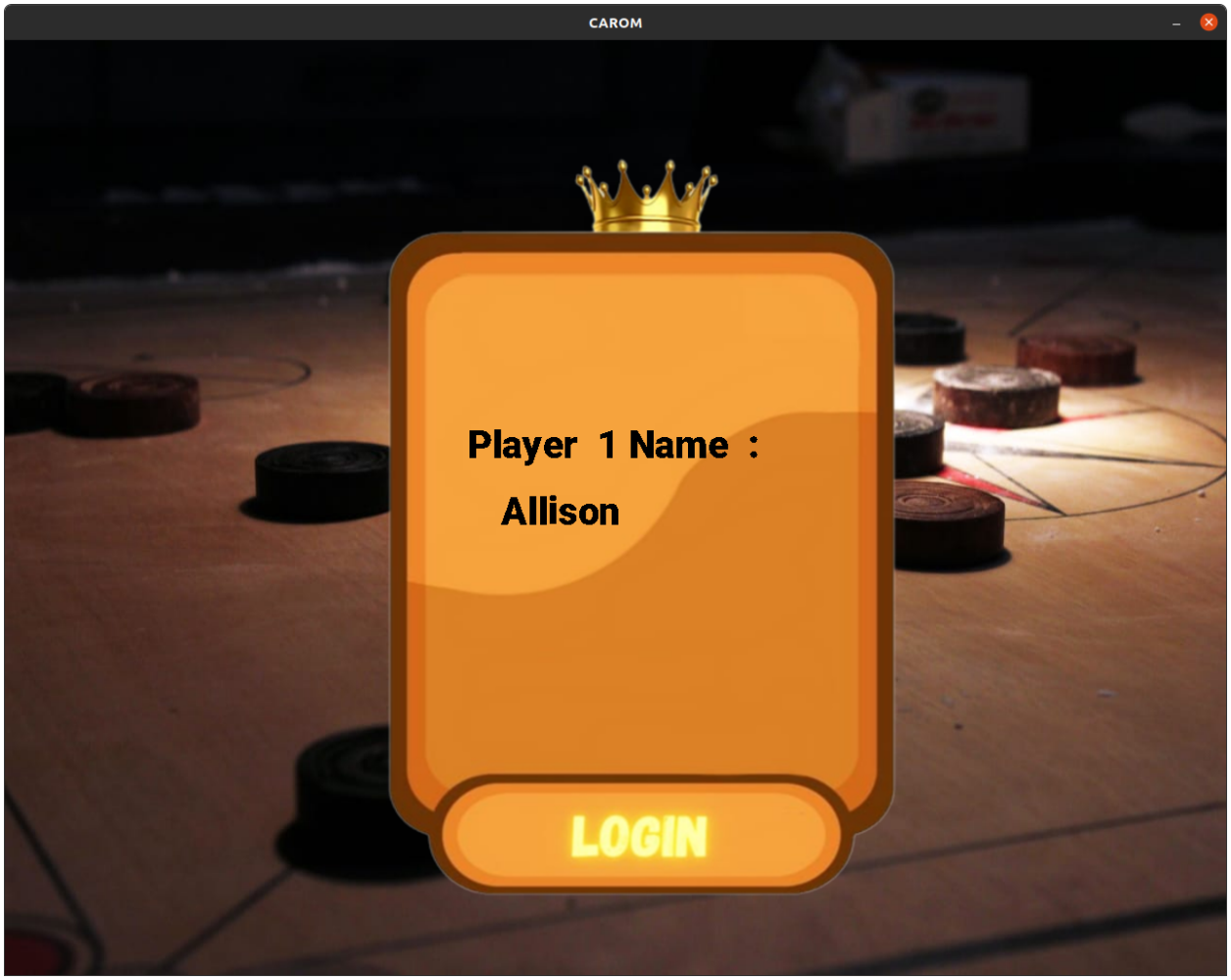
In the top right, the time taken till now can be seen.

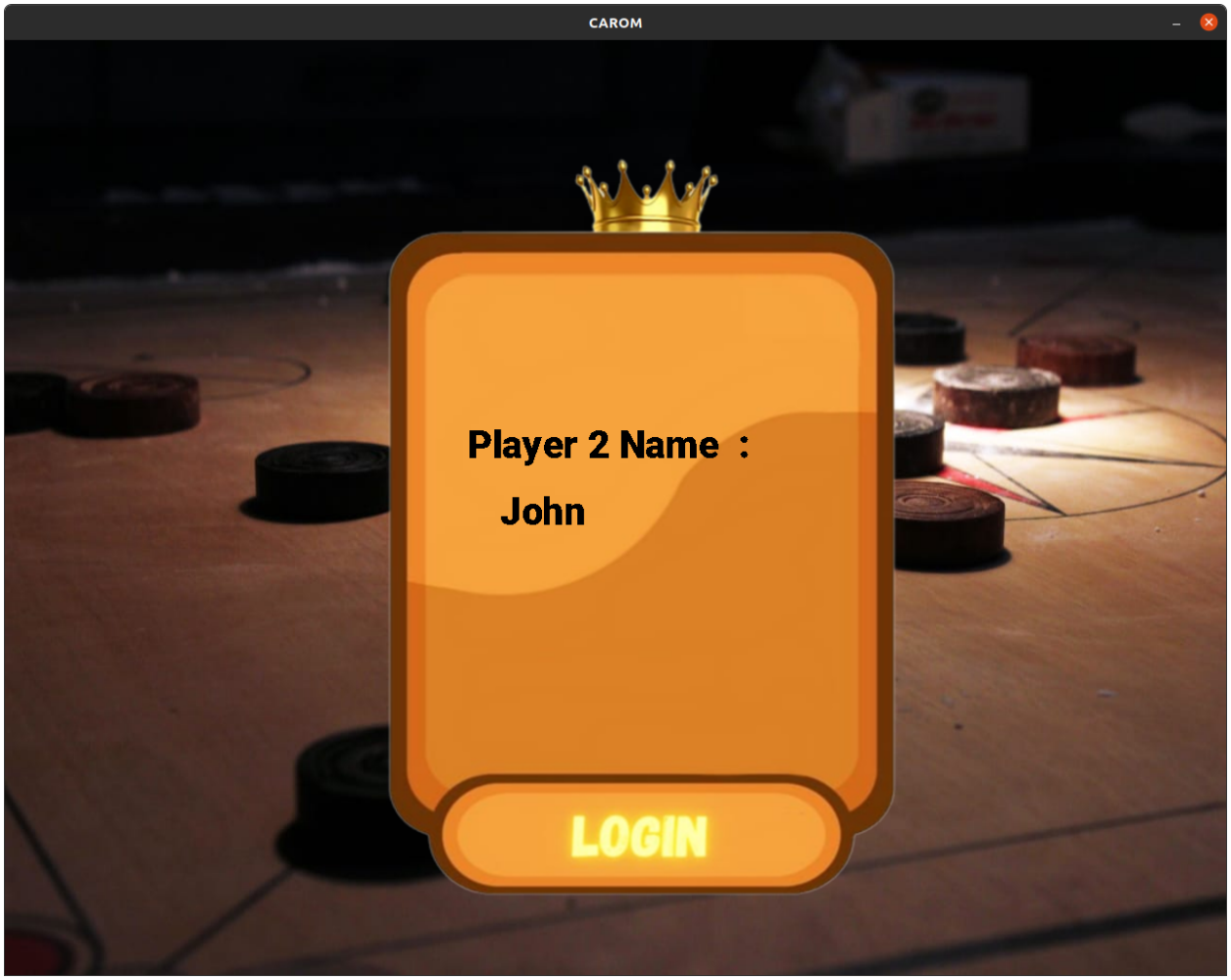


Pressing on ESC, the game can be paused. The RESUME button is used to resume the game, and the EXIT button is used to return to the main menu.

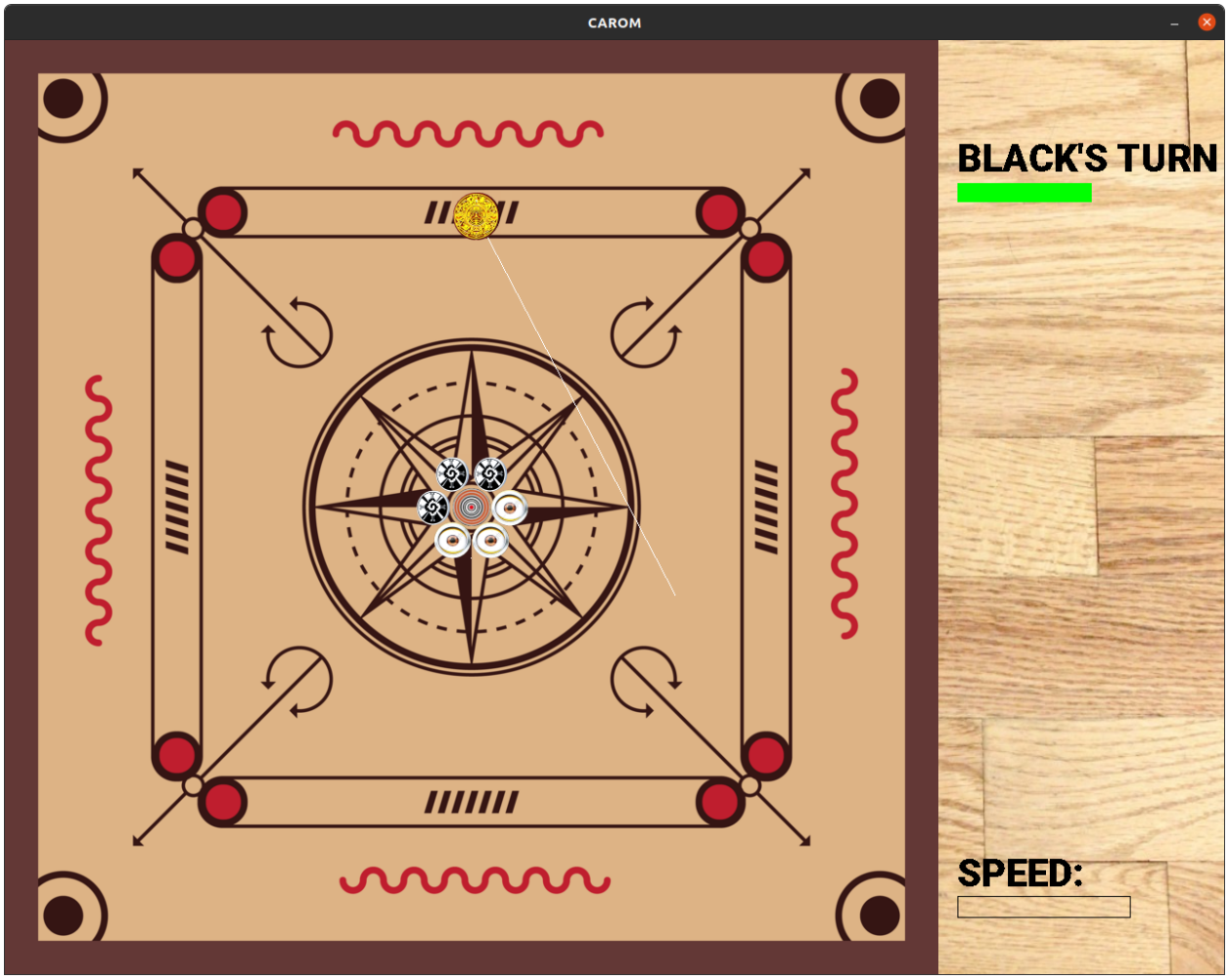
When the game ends, the result window appears. For instance, here, a player named “Oppenheimer” has finished the game in 38 seconds:

Then, if TWO PLAYER MODE is entered from all modes, then the game asks for the introduction of two players:


CAROM
Player 1 Name :
Allison
LOGIN


CAROM
Player 2 Name :
John
LOGIN

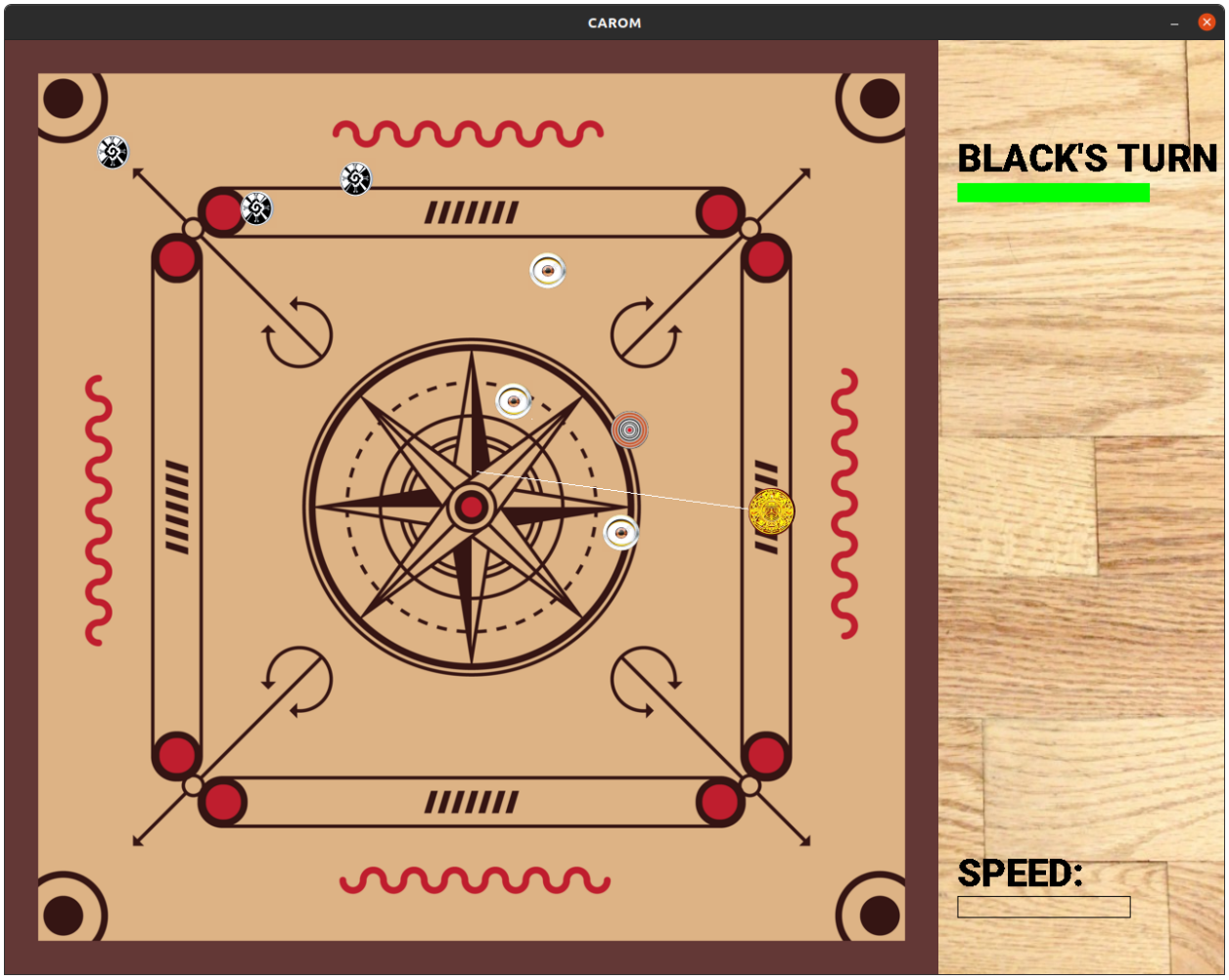
After the names are entered, the game begins:



The top right corner says which player has to strike now. The green bar below the writing indicates the remaining time for the current player to strike. Each player has to strike within fifteen seconds. The speed bar in the bottom right works as before. When a player called John wins the game, the following window is displayed:

When the FOUR PLAYER MODE is entered, the four players have to introduce themselves, and begin the game. The following is what the screen looks like when it’s

the second player's turn:



Project Modules

- **2_player_mode.h:** This module serves as a backbone to other modes of the game. It not only contains some functions that control the two player mode of the game, but there are some other functions also that are used in single player and four player modes. The functions are discussed below:

  initVariables() and initAllPieces() are used to initialize the variables and position of the pieces when the game begins.
  handle_2PlayerMode() moves the pieces and calls the rendering functions after the game begins.
  handleEvent_2PlayerMode() handles the events.

The functions renderAllPieces(), renderTurn(), renderRemainingTime(), renderSpeedBar() handle rendering as evident from their names.
checkGameOver() checks if the game is over.
handleFailedCover() handles the scenario where a player has failed to cover the red piece.
gameWinLogic() checks who won the game according to the rules of the game.
handleNextMove() is used to prepare all the pieces for the next move.
handleInvalidPot() handles if a piece has been potted in an invalid manner.
placeStriker() is used to place the striker for the next move.

- **allModes.h:** This module contains some functions that control the window where all the modes are shown and the players can choose which mode to play. The function handleEvent_allModes() handles events, and the function handle_allModes() does rendering.

- **circle.h:** This module contains some functions related to circles. The struct circle is defined here.

- **fourPlayerIntro.h:** When players enter the four player mode, they are asked to introduce themselves. This module contains the necessary functions that facilitate the introduction. The function handleEvent_fourPlayerIntro() handles events, and the function handle_fourPlayerIntro() does rendering.

- **gFunctions.h:** This module contains some functions that initializes SDL2, loads media, close SDL2, checks if two circles are colliding etc.

  The C++ enumeration is defined here, which is used to understand which window the game is currently on.
  The functions init(), loadMedia() and close() initialize SDL, load all media, and close SDL libraries respectively.
  The function checkCollision() is used to detect if two circles are colliding, and the function motionOver() is used to see if all pieces have stopped moving.

- **leaderboard.h:** This module contains functions that are used to implement the leaderboard of the game.
  The functions handle_leaderboard() and handleEvent_leaderboard() are used to render and event handling in the leaderboard window.
  The function searchForLeaders() is used to sort the leaderboard.
  The function addToTheLeaderboard() is used to add new data to the leaderboard

after a new game is finished.

- **libHeaders.h:** This includes all the library header files that this project uses, like C/C++ libraries, SDL2, SDL2_image, SDL2_ttf etc.

- **mainMenu.h:** This module contains functions that are used to implement the main menu window of the game.
  The functions handleEvent_mainMenu() and handle_mainMenu() are used to handle events and rendering respectively in the Main Menu window.

- **maths.h:** This module contains some mathematical functions that are needed for the game.
  Given two points, the functions distanceSq() and dist() calculates the square of the distance and the distance between two points.

- **music.h:** This module contains some functions that are used to play music.

- **pauseMenu.h:** This module contains functions that are used to implement the window that comes up when the game is paused.
  The functions handleEvent_pauseMenu() and handle_pauseMenu() are used to handle events and rendering respectively in the Pause Menu window.

- **piece.h:** This module contains the struct called piece, which controls all the carom pieces. This has functions that are used to initialize the position of the pieces, move the pieces, handle events, detecting collisions, adjusting position and velocity of the pieces after collisions etc.

  The function initialize() is used to set the position of the piece.
  The function handleEvent() is used to change the attributes of the piece according to an event.
  The function move() is used to move the pieces every frame.
  The function render() is used to render the texture associated with the piece.
  The function shiftColliders() is used to change the position of the colliders associated with the pieces when the position of a piece changes.
  The function handleCollision() deals with the physics of the collision. It changes the attributes associated with the pieces as necessary when a collision takes place.
  The function isPotted() checks if a piece has been potted.
  The function handleBeforeStrikeMovement() handles the movement of the striker before it is striked.

The function adjustVelocity_KeyReleased() deals with the velocity of the striker before striking. while moving around on the baseline.

- **result.h:** This module contains functions that are used to implement the window that comes up after the game is ended.

- **rules.h:** This module contains functions that are used to implement the window that displays the rules.

- **singlePlayerIntro.h:** This module contains functions that are used to implement the window that asks the player to introduce himself/herself before playing the single player mode.

- **singlePlayerMode.h:** This module contains functions that are used to implement the single player mode of the game.

- **texture.h:** This module contains the struct LTexture. It contains functions that are used to load SDL surfaces, convert them into textures, converting text to texture, deallocating the textures, rendering them etc.

  The constructor is used to initialize the variables.
  The function loadFromFile() is used to load a SDL surface from a given path, and convert it to a texture.
  The function loadFromRenderedText() is used to convert a string into a texture.

- **timer.h:** This module contains the struct LTimer. This module implements a timer that is used as a stopwatch during all modes of the game.

  The functions start(), stop(), pause(), unpause() are used to start, stop, pause and unpause the timer respectively.
  The function getTicks() is used to get the time elapsed on the timer.
  The functions isStarted() and isPaused() are used to see if the timer has been started or has been paused, as evident from the names.

- **twoPlayerIntro.h:** This module contains functions that are used to implement the window that asks the player to introduce himself/herself before playing the two player mode.

Team Member Responsibilities

---

1. **Ahmed Nesar Tahsin Choudhury:**
   2_player_mode.h
   piece.h
   gFunctions.h

2. **Mehrajul Abadin Miraj:**
   fourPlayerIntro.h
   leaderboard.h
   pauseMenu.h
   playerIntro.h
   result.h
   singlePlayerMode.h
   singlePlayerIntro.h
   twoPlayerIntro.h

3. **Jannatin Tajri:**
   allModes.h
   circle.h
   fourPlayerMode.h
   libHeaders.h
   mainMenu.h
   maths.h
   music.h
   rules.h
   Timer.h
   texture.h

Platform, Library and Tools

As of now, this game is available for the Linux Operating System. It is developed with the help of SDL2 library, and its extension libraries: SDL_Image, SDL_TTF and SDL_Mixer. The tools used are gimp and Adobe Photoshop.

Limitations

There are some features that have the potential to make the game provide a better experience for the player. They are:

1. Single Player Mode against Computer: If there was a mode to play against the computer, the game would provide a better experience for the player.
2. The game lacks animations and smooth transitions between windows.
3. The game is available only on the Linux Operating System.
4. Better quality images could have made the visuals more enjoyable, but it is difficult to get good images on the internet.

Conclusion

We have learnt quite a lot throughout the development of the project. Before the project, we usually coded in a disorganized manner, and often did not use good naming conventions to save time. While doing the project, that lead to a lot of problems. It was difficult to understand what happened where, and communication was difficult. We understood the how significant the readability of code is when working in groups.

We learnt how to communicate with each other, and how to structure our code so that it is easier for another person to use our code.

We learnt how to create functions that can be used in other modules as a black-box. The black box model helped us a lot to break complicated tasks into simpler models, and then implementing them one by one.

We learnt how to work with big files, and maintain code. Debugging the project was extremely difficult for us since we never had to deal with so many lines of code at once. The project enabled us to learn a lot about being able to debug large blocks of code. The importance of commenting was understood when we had to return to a large block of code written a while ago, because we did not remember what we did. The comments helped us to understand previously written code efficiently.

Learning to use Github for sharing code was a very good experience too.

To conclude, the project helped us understand why writing clean and readable code is

important, how to share code, how to debug large files of code, how to go through documentations, and how to communicate efficiently with teammates.

Future Plan

We are eager to introduce our game to the Android, iOS and Windows Platform. We also aspire to include the mode against computers, online features, intend to make further improvements to this game and provide the user with an even more refined experience.

Repositories

Github: https://github.com/sheldor944/Carrom/tree/master
YouTube: https://youtu.be/pIaXJZSV2Ik

References

---

1. Official SDL2 Documentation: https://wiki.libsdl.org/SDL2/FrontPage
2. LazyFoo Tutorials: https://lazyfoo.net/tutorials/SDL/