**Arimaa**

**Project Description:**

The goal of developing this project is to understand the working of Android activity life cycle and the working of Custom View for development of Board Games using Canvas.

**Files / Pages:**

This project consists of few .java files (for same activity but different classes) and 3 .xml files for UI layout.

XML files involved are:

1. activity\_main.xml
2. activity\_game\_info.xml
3. activity\_game\_board.xml

Java files involved are:

1. ActivityMain.java
2. CreateDB.java
3. GameInfo.java
4. GameBoard.java
5. GameBoard\_CustomView.java
6. Actions.java

**activity\_main.xml**

activity\_main.xml is the home page of Arimaa. This layout consists of Game logo and 3 buttons: 1) To start a new game. 2) To view Game information. 3) To Exit the game. This layout also contains 1 additional button which will be viewed only if previously played game is stored within SQLite database table. This button is named as Resume Game. Below are the 2 images of activity\_mail.xml with both the types of views i.e., with Resume button and other one without the same.

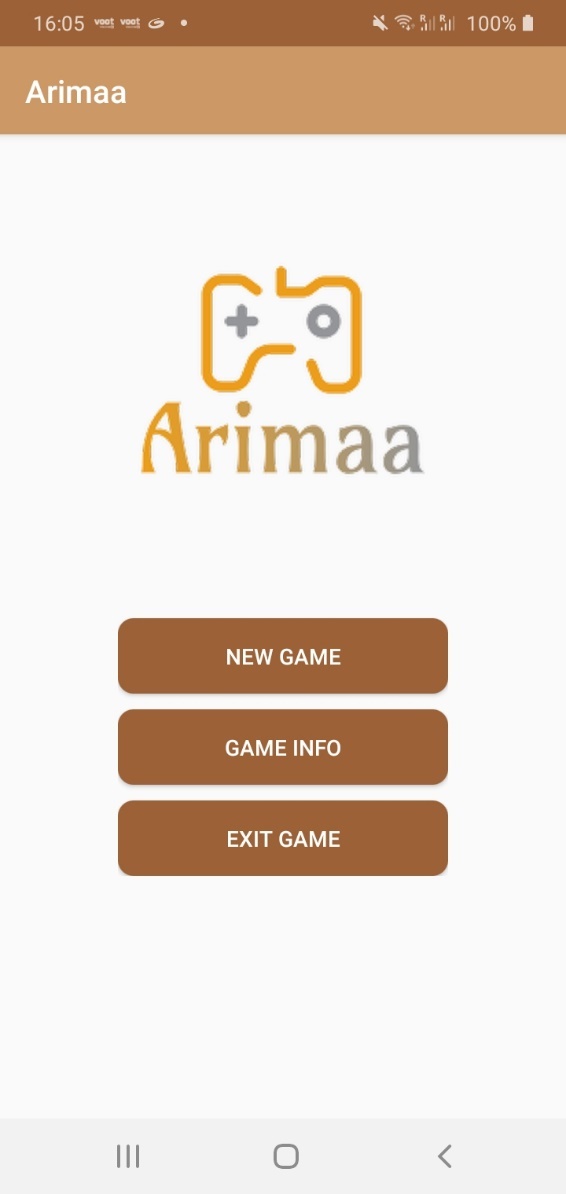
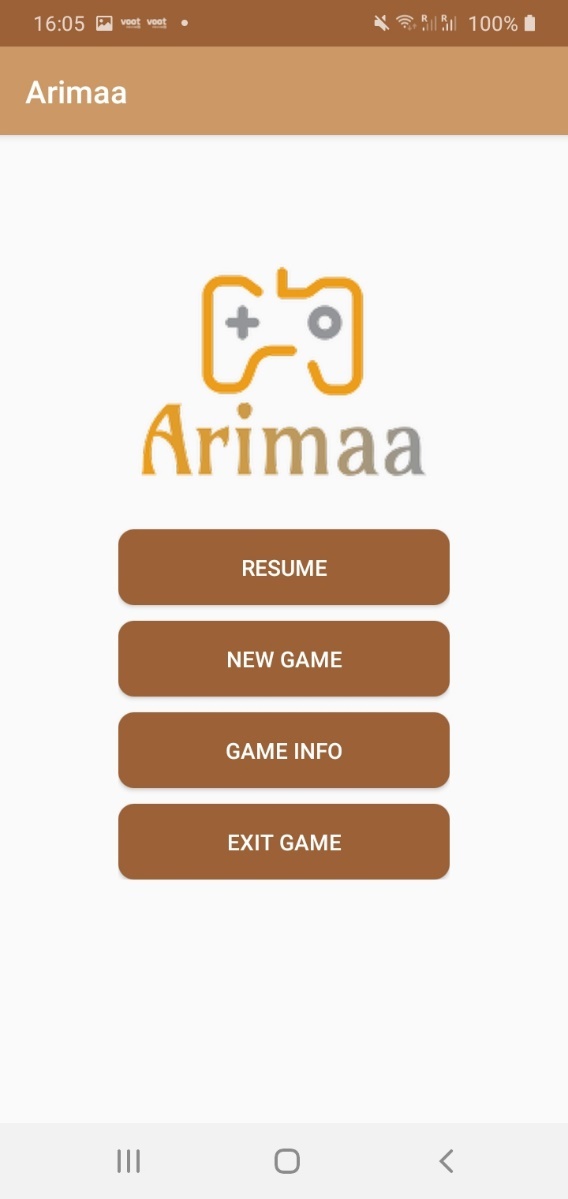
 

Figure: 1.1 Figure: 1.2

As shown in figure 1.1, as a game is loaded for the first time or in case there are no previously saved games, Resume button will not be displayed.

As shown in figure 1.2, Resume button will be displayed only in case there are previously stored game which player may want to resume back or they can start a new game.

**MainActivity.java**

MainActivity.java consist of the main class responsible to run this program i.e., onCreate() method of home activity. It also consist of below methods responsible to handle button onClick events on home page of the Arimaa Game.

1. onCreate() method:

onCreate method is responsible to check if the database table exist or not. In case game is freshly installed, there does not exist any database table. Hence, this method calls the function to generate a blank database table to be used further while required to store data in the same.

This method is also responsible to check if the database table already exist, does it contain any previously stored data to resume previous game or not. In case there are any records in database table, it enables the button on UI to allow players resume previously quit or left game from the state it was left.

It also consists of a logic to display Game Logo on the top of the home page. Hence, fetching, resizing and setting an image is also handled in onCreate method.

1. resumeGame() method:

resumeGame method will be triggered when player clicks on Resume button from home screen of game. This method will redirect player to another screen of Game (GameBoard.class from backend).

1. newGame() method:

newGame method will be triggered when player clicks on New Game button from home screen of game. This method will first check if there are any records in database table or not. In case there are records, it will delete them all and then redirect player to screen of Game (GameBoard.class from backend).

1. gameInfo() method:

gameInfo method will be triggered when player clicks on Game Info button from the home screen of game. This method will redirect player to Game Info screen (GameInfo.class from backend) and close the existing activity (i.e MainActivity.class).

1. exitGame() method:

exitGame method will be triggered when player clicks on Exit Game button from the home screen of game. This method will terminate the game by closing the existing activity (i.e MainActivity.class).

1. onBackPressed() method:

onBackPressed() method is overridden here to handle game termination when back is pressed from home page. In such case, game will never be terminated unless players click on exit game button.

**CreateDB.java**

CreateDB class is extending SQLiteOpenHelper class. Hence, we need to define blocks for few methods like:

1. onCreate() method:

We are writing a logic to create a database table if it does not exist on device database. In case it already exists, this method will not create a new table and continue to use existing one. This method will be triggered in onCreate method of MainActivity.class.

1. onUpgrade() method:

This method will be triggered when there is an update in database versions. In such cases, a logic is written to handle safe deletion of database and recreation of the same.

**GameInfo.xml**

GameInfo will be called from Game home screen. A player will be redirected here if they click on Game Info button provided on the home screen of the game. Below is the image of Game Info screen in Arimaa.

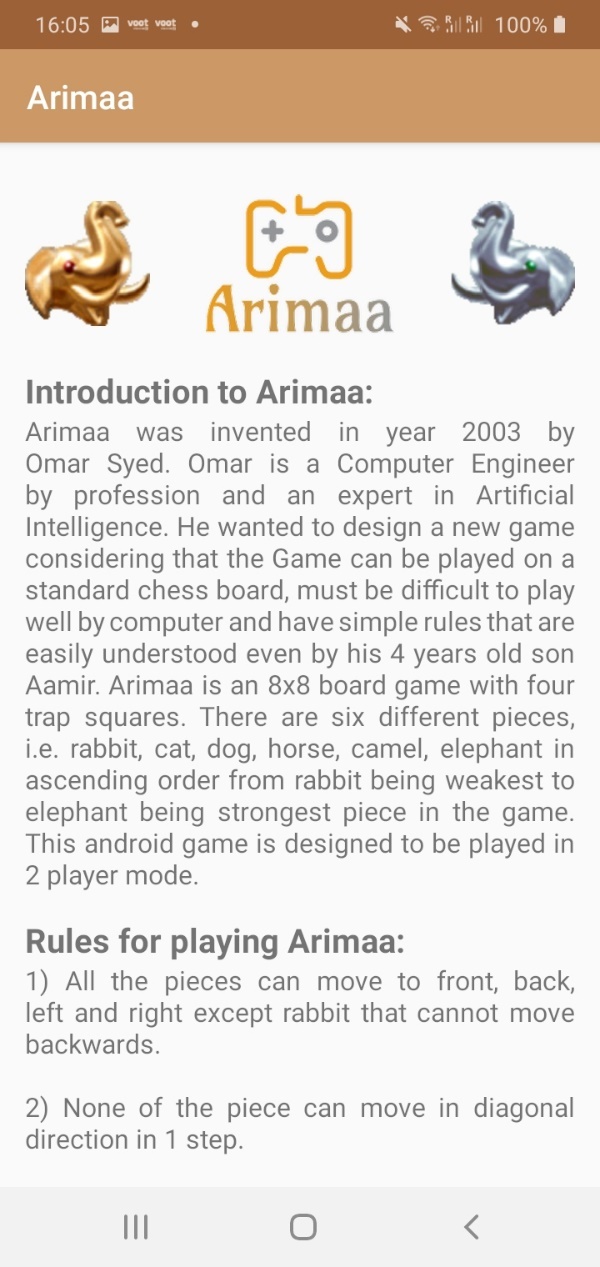
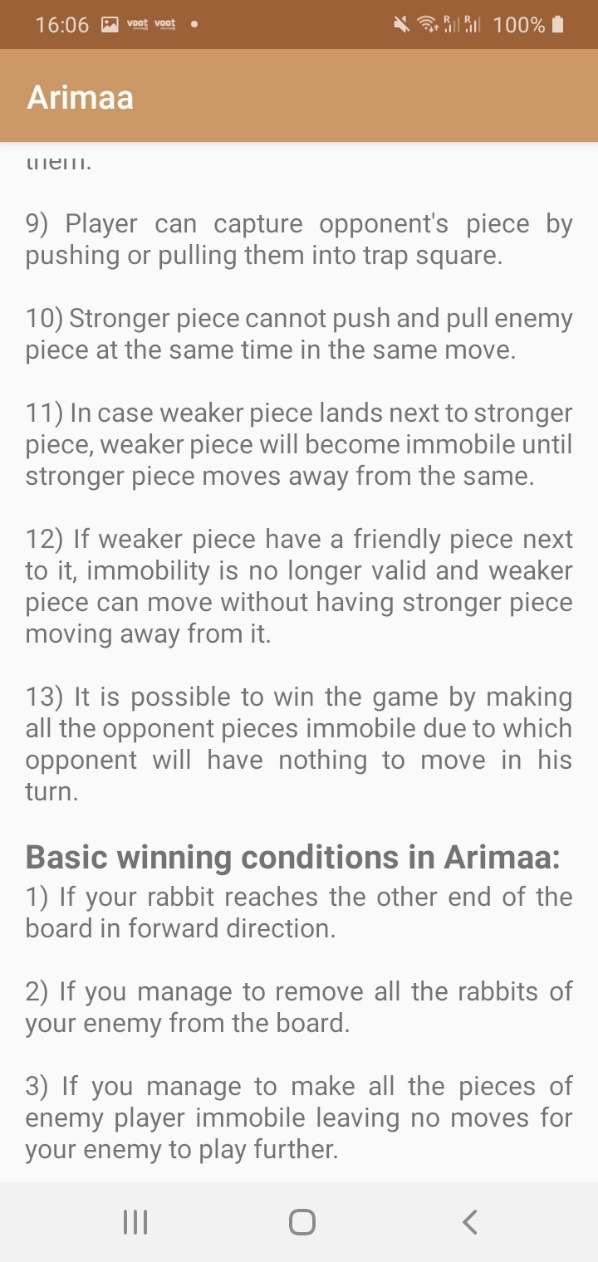
 

Figure: 2.1 Figure: 2.2

**GameInfo.java**

GameInfo.java contains a logic to render images on the top of the screen. This file also contains a logic to set text of rules and other information provided on this screen.

**GameBoard.xml**

This screen is the actual screen where Game board will be displayed. The UI design of this activity is in such that everything seems to be self-explanatory. All the buttons are perfectly placed at the top of the screen followed by message TextView and then the Game board. A counter indicating number of steps left to end the turn is displayed at the bottom of the screen along with Player turn. Figure 3.1 displays the Game board screen.

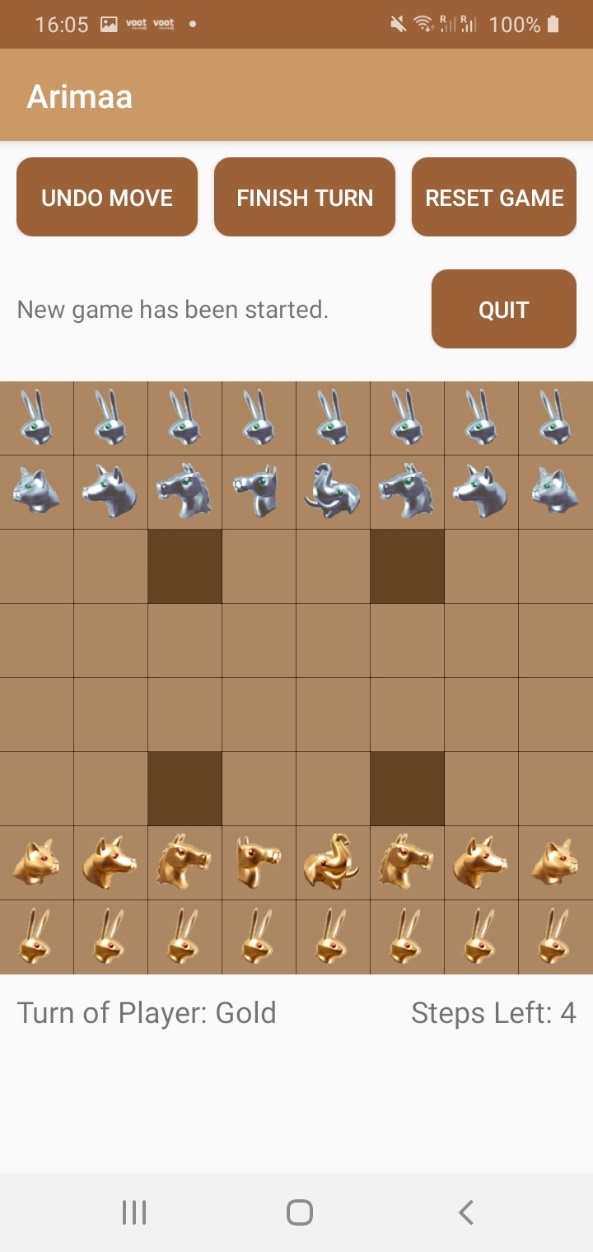


Figure: 3.1

**GameBoard.java**

GameBoard.java consist of below mentioned functions:

1. onCreate() method:

onCreate method is responsible for linking all the TextViews, initializing Context of this activity and creating all objects of required classes.

1. resetGameButton() method:

The resetGameButton method is responsible for resetting the existing game board. This function will be called once user clicks on Reset Game button from the UI. The function also contains a logic to pop up an alert box with options to reset game or go back. Depending on user’s input, further program will execute. In case user choose to reset game, a function to truncate db table will be called followed by resetting winner decision and finally the reset game.

1. onBackPressed() method:

A logic has been written to display a dialog box and handle the same based on decision of user’s input.

1. goBack() method:

goBack method will be called from onBackPressed method. This method is responsible for finishing the current activity.

1. updateMessage() method:

updateMessage method is responsible to set content of view with the message on UI which is passed to updateMessage method.

1. updateStepsLeft() method:

updateStepsLeft method is responsible to update the steps left after every move player makes. This method will be called from custom view class.

1. updatePlayerTurn() method:

updatePlayerTurn method is responsible to update player turn. This will be called from custom view class after every 4 moves or after player finishes their turn.

1. finishTurn() method:

This method will be called once user clicks on Finish Turn button. finishTurn method contains a logic to identify if game is already finished or not. In case game is finished, a dialog will be popped up displaying an appropriate message. In case game is not finished, a method in custom view class will be called to finish the turn of current player.

1. finishGame() method:

finishGame method will be called when game is finished using one of the 3 methods of ending a game that are defined in custom view class. A logic is defined to handle a dialog box that displays appropriate message to user along with Okay button.

1. pushPosition() method:

This method contains a logic to handle a dialog box which displays a message for user to select the position on board user wish to push to enemy piece to.

1. isPull() method:

This method will be called from custom view class of Game Board. This contains a logic to identify if user wish to continue without pulling enemy piece or without the same. In case user wish to continue with pulling an enemy piece, an appropriate dialog box will be called. In case user choose not to pull logic will handle to safely close dialog box.

1. pullPosition() method:

pullPosition method will be called from isPull method in case user opts to pull enemy piece. A logic to handle another dialog is defined.

1. onUndoMoveButtonClicked() method:

This method will be called once user clicks on Undo Move button from the UI. This method contains a logic to identify if game is already finished or not. In case game is finished, a dialog will be popped up displaying an appropriate message. In case game is not finished, a logic is defined to undo moved until whole turn is undone.

1. undoMove() method:

This method will be called either from onUndoMoveButtonClicked method or from Game Board Custom view class. This method is responsible to check if there are any records in db table and if yes, delete the last record and resume the game from 2nd last record in db table. In case there are no records in db table, a proper message will be displayed in message text view.

**Actions.java**

Actions.java consist of key functions responsible to perform CRUD operations on database. Below mentioned are the functions in this java class:

1. recordPresent() method:

recordPresent method is responsible to check in case database table contains any records or not. This will be called from Mainactivity class and from Custom view of Game Board to resume the game in case of undo and to implement the reset function.

1. truncateDBTable() method:

This method will be called when reset game button is clicked from an UI. This method also contains a logic to delete all the records if any from the database table.

1. getStringArray() method:

This method will be called when game resumes from any of the undo called by user from UI. This method is responsible to restore game state from string to 2-dimensional string array.

1. getPlayerTurn() method:

This method is responsible to return player turn at any state of the board or any timestamp saved in db table.

1. getStepsLeft() method:

This method is responsible to return number of steps left at any state of the board or any timestamp saved in db table.

1. convertToString() method:

This method is responsible to convert a 2-dimensional array to string format. This method will be called in Actions class and from Custom View class before storing board state in db table.

1. storeData() method:

This method will be called from Custom View class to store all the board and players data in db table. This method contains a logic defined to perform insertion operation in database table.

1. deleteLastRecord() method:

This method contains a logic to delete last record in db table. This method will be called when user press Undo Move button from UI to undo the last turn. This will also be called from GameBoard Custom View when a move is invalid or in case move cannot be accepted.

1. getBoardStateStatus() method:

This method will be called to identify in case board state is already seen in the game or not. In case the returned state is seen before, move will not be accepted.

**GameBoard\_CustomView:**

GameBoard\_CustomView is responsible to render the actual GameBoard. All the operations performed on GameBoard are performed in this class. Below are the methods defined in GameBoard\_CustomView class:

1. onMeasure() method:

This method consists of a logic that is responsible to force the Game Board of 8x8 blocks to be square shaped. The size of the board is calculated based on screen width and screen height. The minimum length from the 2 is accepted to be the maximum size of Game board. This method also calls init method to initialize all the required Game Board variables before rendering the board on screen.

1. init() method:

init method is responsible to initialize all the required variables before rendering the game board. This also contains a logic to identify if game is already ended or not. This is to handle invalidate method calls. In case game is ended, Game is not required to reinitialize board piece variables but in case game is not ended, all the Game piece variables will be initialized.

1. resetGame() method:

This method will be called from init method and is responsible to reset all game variables. Variables include playerTurn, stepsLeft and a 2-dimensional array which stores all the piece data.

1. onDraw() method:

onDraw is a default method of android that is overridden in our custom view. This accepts canvas as a parameter and is responsible to draw a Game board on the same. This method consists of a logic that handles drawing of light and dark boxes on board along with rendering an image on the same at appropriate positions. This also contains a logic to handle selected piece and possible moves for selected piece.

1. onTouchEvent() method:

onTouchEvent method will be called once user makes a touch on game board. This method contains a logic to check in case game is finished or not. In case game is finished, the touch will not be accepted else touch is acceptable. This method also contains a logic to identify type of touch. Touch type are different for identifying 1st touch which is selection of player piece, 2nd touch which is destination block where piece needs to be shifted, 3rd touch in case player wish to push enemy piece and 4th touch in case player wish to pull the enemy player piece. This method also contains logic to invalidate the game board.

1. pieceMoveStart() method:

pieceMoveStart method will be called from onTouchEvent method. This method contains a logic to identify the touch position on the board. This method return true in case touch is valid or not.

1. pieceMoveEnd() method:

pieceMoveEnd method will be called from onTouchEvent method. This method contains 3 logics. 1st is to identify end position to move the piece. In case end position is using push to enemy piece, 2nd logic is to identify position to push enemy piece. 3rd logic is to identify end position to pull the enemy piece in game move.

1. getStepsLeft() method:

getStepsLeft method will be called from Main class of GameBoard. This method will return the number of steps left after each move.

1. identifyPiece() method:

This method will be used to identify piece touched on game board by player. This method returns the bitmap of piece type touched. In case an invalid piece is touched, this method returns null.

1. highlightPossibleMoves() method:

highlightPossibleMoves method is used to identify all the possible moves near any selected piece. This method Contains logic to identify possible moved for 1st time touched piece and the moved for push piece.

1. setPull() method:

setPull method will be used to make pullTouch variable true. This method will be called from GameBoard main class.

1. isWeakOrNull() method:

This method will be used to identify if there are any positions containing enemy’s weak piece or null positions near selected piece. This will be called from highlightPossibleMoves method.

1. nearStrongerPiece() method:

This method will be used to identify if the selected piece contains any enemy piece stronger than itself or not. This method will be called from onTouchEvent method.

1. nearFriendlyPiece() method:

nearFriendlyPiece method will be used to identify if selected piece contains any friendly piece near the itself or not. This method will be called from onTouchEvent method.

1. nearWeakerEnemy() method:

This method will be used to identify if there are any weaker enemy piece near the selected piece or not. It will be called from pieceMoveEnd method.

1. confirmFinishTurn() method:

confirmFinishTurn method will be used to finish turn of existing player. This method will be called from game board main class when a button of Finish Turn will be clicked from UI.

1. checkTrapBlocks() method:

This method will be used to check if there are any piece in all the 4 trap blocks of game board or not. In case there are, eliminate the same from board. A logic is mentioned to identify in case rabbit is removed from board, decrement rabbit count by 1. This method also calls saveGameData method and checkGameStatus method.

1. saveGameData() method:

This method will be used save game data in db table. This method calls storeData method from Actions class to perform insertion operation on db table.

1. checkGameStatus() method:

This method will be used to check 3 possible conditions on winning. 1st condition is rabbit of any 1 player from both reaches the extreme end of the board in forward direction of either player. 2nd condition of check is count of rabbits present for both the players. In case any one player’s rabbits are removed from board, end the game. 3rd condition is to check if currently moving player have more steps to play game or not. In case there are no more steps left to play, end the game declaring an appropriate winner of the same.

1. haveMoves() method:

This method is used to identify if piece at given position have any positions to move or not. This method will be called from checkGameStatus method to check 3rd condition of winning game.

1. checkCanMove() method:

This method is used to identify if piece at given position can move to passed position or not. This method will be called from haveMoves method.

1. checkStrongEnemy() method:

This method will be used to identify if piece is having any strong enemy piece near itself or not. This method will be called from haveMoves method.

1. setWinnerDecided() method:

This method will be used to reset decided winner in last game. This method will be called when a game is reset by user from UI.

1. getWinnerDecided() method:

This method will be used to fetch winner data to display the same at the end of the game.

**Below is the mathematical calculation for plotting points on graph:**

The only mathematical calculation we are performing is we are setting the piece image size to boxSize – boxSize/10. This is to make the piece image visible on the blocks of Arimaa game in custom view game board.

**Data structure:**

We have used SQLite database to store all the game states after each move. We are storing 4 possible data in database. They are timestamp in dd/MM/yyyy hh:mm:ss:SS format, Player turn, Number of steps left and lastly Game board state. Game board state is a string containing all the data of 2-dimensional array converted into string.

Database table name is “gamestate” and is contains below key fields:

1. Timestamp: This is the primary key in our database. No timestamp can be ever same for any number of moves at any time of game instance. Hence, we have selected timestamp to be primary key and the data type of this field is Text as we are going to store String data with special characters like ‘/’ and ‘:’.
2. PlayerTurn: We have set the data type of player turn to varchar. This is because we are going to store the player turn after each move in this data field and the length of data is maximum of 6 characters long.
3. StepsLeft: We have set the data type of stepsLeft to int. This is because we are going to store the number of steps left to end the turn by each player in this field. The maximum integer value we are going to store is 4 and the minimum integer value stored is 1.
4. State: This field is the most important field of our interest as we are going to store the state of game board after each move. The data type of this field is varchar as we are going to store String data of limited length.

Apart of above database table, we are using multiple variables of different data types like String 1-dimensional array, String 2-dimensional array, String, Integer, Boolean, Double, etc. to store required data during runtime.

**Design Decisions:**

1. We have tried to keep the UI self-explanatory and easy to understand by users. We have added Button with appropriate and self-explanatory text to indicate their use to users without having any second thought about the same.
2. We have also placed the text views for message, steps left and player turn at appropriate positions to make it easy for users to understand the game state at any point.
3. We are updating the message text view at each step to make users understand what actions are expected from them.
4. We have also used a proper combination of colour codes to make the UI attractive and distinguishable. All the blocks except the trap blocks are light brown in colour and the trap blocks are dark brown in colour.
5. Once the game is started, users have an option to Reset or Quit game. They can resume the game from the state they have left it the last time before quitting the same.