CS154 PROJECT REPORT

180050060 Missula Meghana
180050087 Ritika
180050048 Khushi Kapadia

λUDO

INTRODUCTION

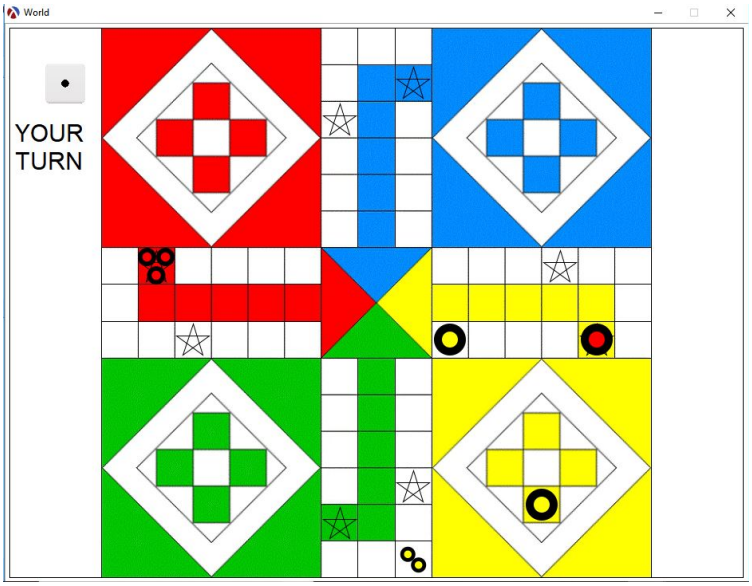
We have created the famous “λudo” aka Ludo game in the Racket language(GUI based). It covers all features of typical Ludo games like variable number of players, safe places, home squares etc. We introduced AI based computer players which takes the best possible moves.

DESCRIPTION

1. **Rules** :-

1) The game begins for any player's piece when the player rolls 1/6 on the dice.
2) The number obtained by rolling the dice will give the next position like a usual die game. Getting a 6 gives an extra chance.
3) A player reaching the success region or killing an opponent’s piece gets an extra move.
4) A third consecutive six is not counted for the final moves.
5) A piece will be safe if its at one of the safe position i.e coloured and starred squares. It is also safe if it is with other coins of the same color at the same place.
6) A piece can kill other player's piece if his piece pre-occupies the non-safe place where there is no blockade. In this case the other player’s piece will go back to home.
7) Reaching success region by all four pieces of a player is the ultimate victory for the player. After a player wins, the game plays carries on with the rest of the players who are yet to reach victory.
8) If no move is possible for any player (human or AI) then the game skips it’s turn on its own.

9) Our game doesn't respond if player tries to make a move which isn't possible (ex: if a coin is at home and the throw of the dice is not 1 or 6 and player tries to move it , or if a coin can reach the success region in 2 moves but the throw of the dice is more than 2 and the player tries to move it.)



AI-strategy

While making the AI strategy, we have considered the following features and have assigned scores to them:

1) **AI-risky**: It checks that how many coins our coin will overtake if it makes the move, which puts it in a risky position now as the other coins can kill it. We have given it a negative score and given a 0 score if it crosses a same-coloured coin which poses no threat to it.

2) **AI-safe**: We have given a positive score to a coin if it reaches a safe position after making the move.

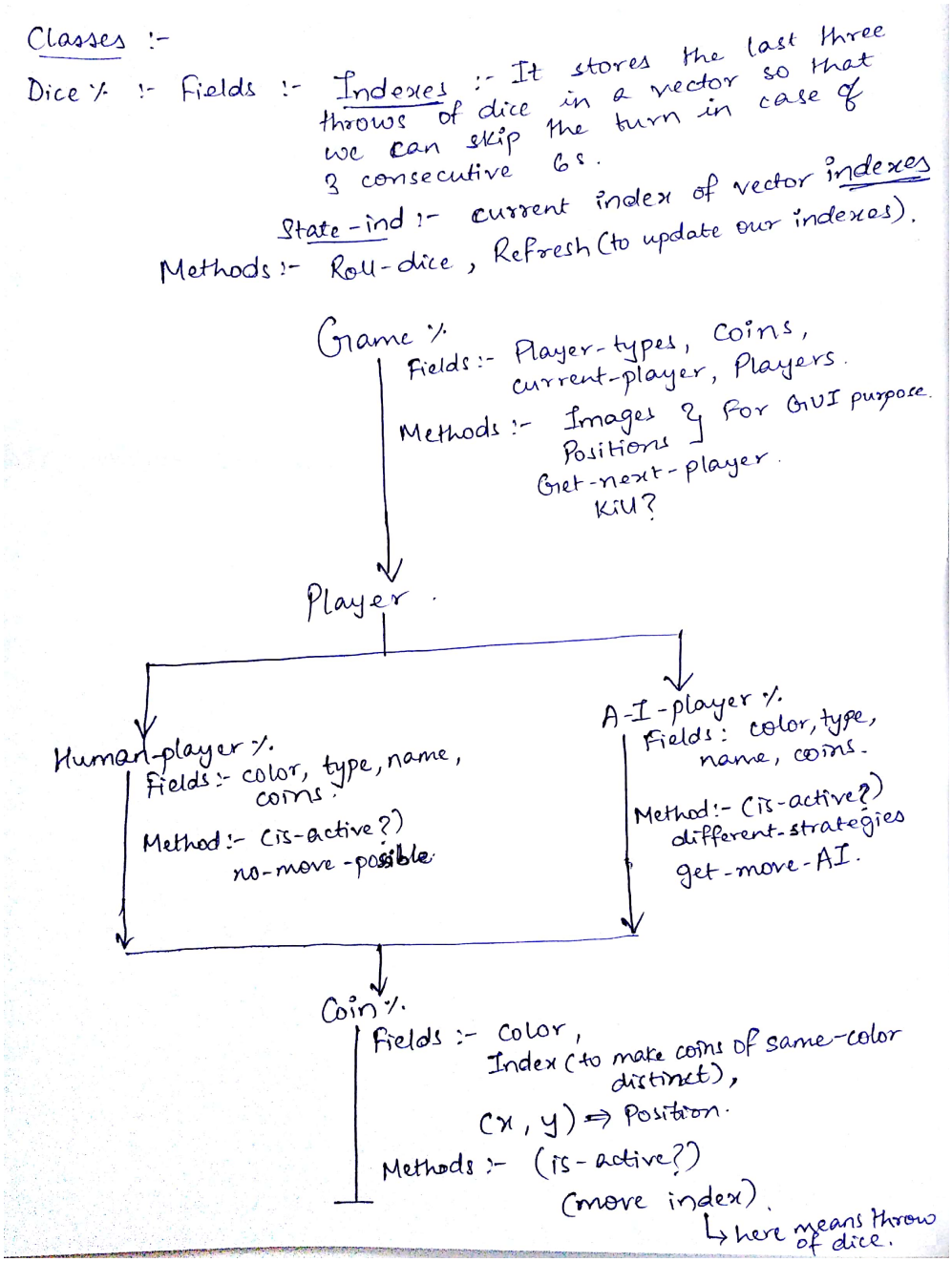
3) **AI-kill**: We have given a positive score to the coin if it kills other coins after making the move.

4) **AI-success**: We have given a positive score to a coin if it enters the success region as then the player gets an extra chance.

5) We have also given a score to the coin if it is initially in its home and gets out after getting 1 or 6. The function **get-move-AI** uses **best-possible-move** function which takes the sum of scores of all these five features for each coin of the player and chooses the coin with the maximum score to move it.Our AI does the dice-click and coin-move on its own. The function get-move-AI

HIGHLIGHTS

1. Our game manifests the various benefits of **object-oriented programming** and the file ***classes.rkt*** contains the classes of all objects of our game. We have made a flow chart showing the classes we have made.



2. We have used **abstractions** for setting the positions of our coins. For eg, we have made the formula which gives the start positions of the coins by taking color and index of coin as input in the functions ***(getstartX)*** and ***(getstartY)***.
3. We have ***maintained a state*** in variables like **indexes** (in Dice class, refer flow chart), **current-player**(in Game), **active**(in Coin), etc.
4. GUI is designed so as to display multiple coins in the same square in a fashionable manner(as shown in the picture).
5. **constants.rkt** : A separate file containing all the constants.
6. **gui-AI-human.rkt** : Consists of all the states and the corresponding screens displayed as a part of the game state. Big-bang has been used to call for the game to begin. Functions like to-draw and on-mouse decide which events take place and what is displayed as per that. Stop-when has been used to terminate the game.
7. We have also added ***dice and click sounds*** in our game.

INPUT-OUTPUT

All possible user interactions and corresponding displays: clicking on the dice is equivalent to rolling it and generates a number and the following game take its course by the (get-move) function call. The user decides which coin is to be moved by clicking on it. Our world now draws the modified arrangement of pieces on the board (output) and the game proceeds.

LIMITATIONS

1) Due to time constraints we weren't able to add the LAN feature because of which players could have played on different PCs.