Ludo game using GUI and random numbers

END-TERM REPORT

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By:

S.no.	***Name***	***Roll No.***	***Registration no.***
1.	*B. Naga prasanth reddy*	*62*	*11907524*
2.	*A.Trilok kumar reddy*	*47*	*11908754*

Courses Code: INT213

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Objective

The primary objective of this project is to implement what we’ve learnt throughout our course of Python programming and use the ludo game GUI and random numbers.this project also aims act providing a user friendly interface to the users to let them easily and attractive using graphics in ludo game. In an aggressive stratgy, a player always gives high priorty to move the piece which can eliminate n opponentspiece, whenever possible.

Introduction

Ludo Game project is written in Python. The project file contains python scripts (game.py, run.py, painter.py, go recorder.py). This is a simple console based strategy board game which is very easy to understand and use. Talking about the gameplay, all the playing rules are the same just like we play in real time ludo. Here at first, the user has to select players i.e either human or computer. After selecting human, the player has to enter details such as name and select color(red, green, yellow and blue). the player can also start the game within two players if he/she wants.

After starting the game, a console based ludo board appears, other rules are the same. First, the computer or the player has to roll the dice. The main thing in this console based game is that the player just has to press “Enter Key” to roll the dice. At the top of the board, it displays a dice with the number. The system keeps on rolling until there’s a possible pawn to move. All the game movements are performed automatically. Whenever the player tries to end the game, there’s an option available to save the unfinished game so that he/she can continue it later. A simple console GUI is provided for the easy gameplay. The gameplay design is so simple that user won’t find it difficult to use and understand.
How to play Ludo King:

Ludo King is an easy to play a strategy board game which is largely automatic, with a player's only choice is to roll a dice and select a token to move forward. And once there is a token that you can move, the computer automatically does it for you.

Movement

To begin, a player must roll a six to move a piece out of the base and onto the start position. That piece is then in play. The player cannot make any other moves until at least one piece is in play.

If a player has a piece or pieces in play, they can move any one of their pieces 1 to 6 spaces along the path according to the number they roll.

Rules of the 6's.

- If a six is rolled, the player can choose to either move a piece out of his base onto the start position or move a piece that is in play.
- Anytime a six is rolled, the player gets an extra roll after his move.
- If a six is rolled three times in a row, the player loses his turn.

Landing on a shared square

If a player's piece lands on an opponent's piece, the opponent's piece is sent back to their base where he must roll a six again in order move it out onto the starting square.

If a player lands on a space occupied by one of his own pieces, that space becomes blocked. A blocked space cannot be passed or landed on by an opponent's piece.

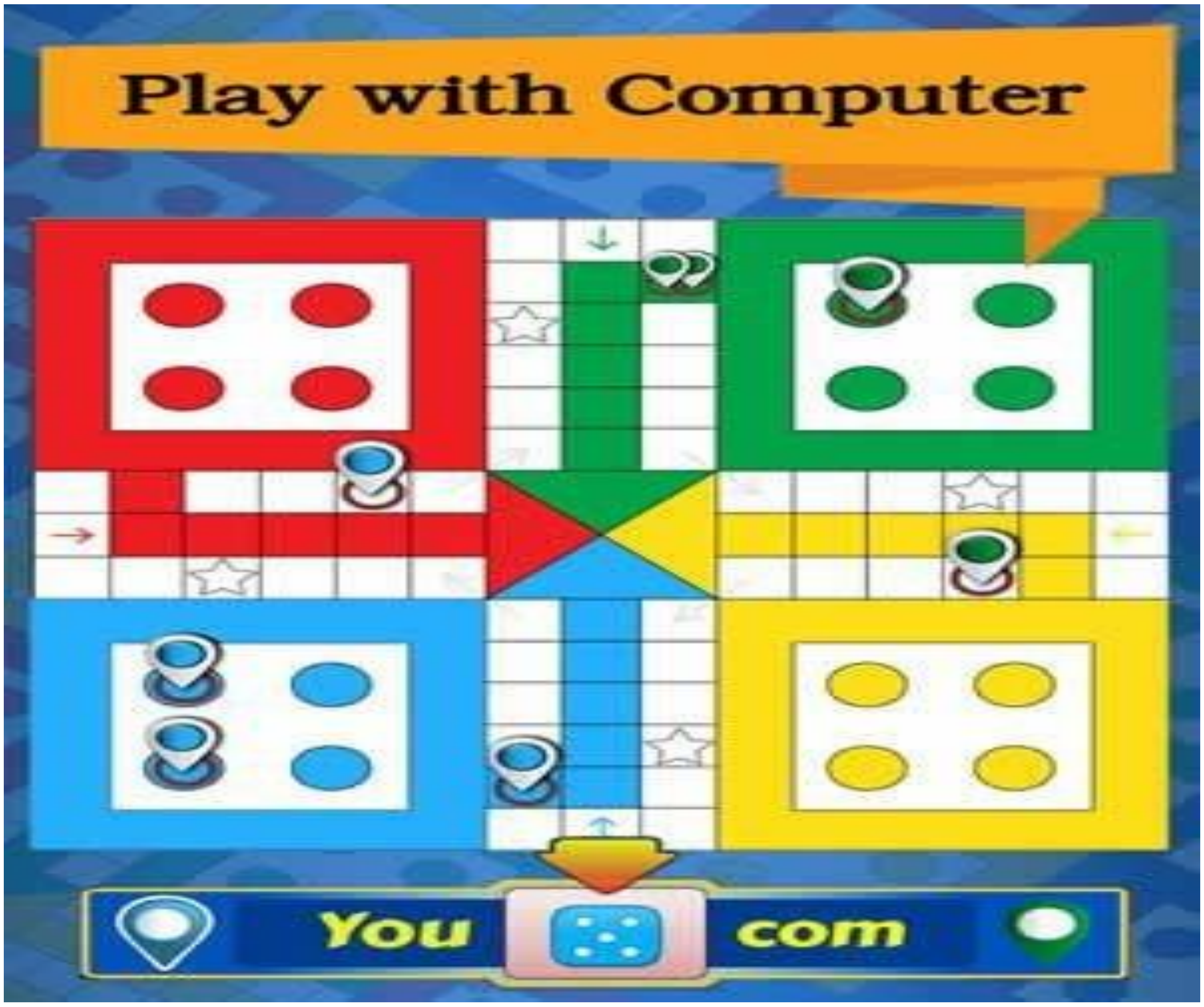
Winning the Game

When a player's piece has reached the home column of its own color, the piece continues its moves toward the center to its home triangle. When a player's die roll lands its piece on the home triangle, that piece has completed its journey. A piece can only be moved to the home triangle with an exact roll.

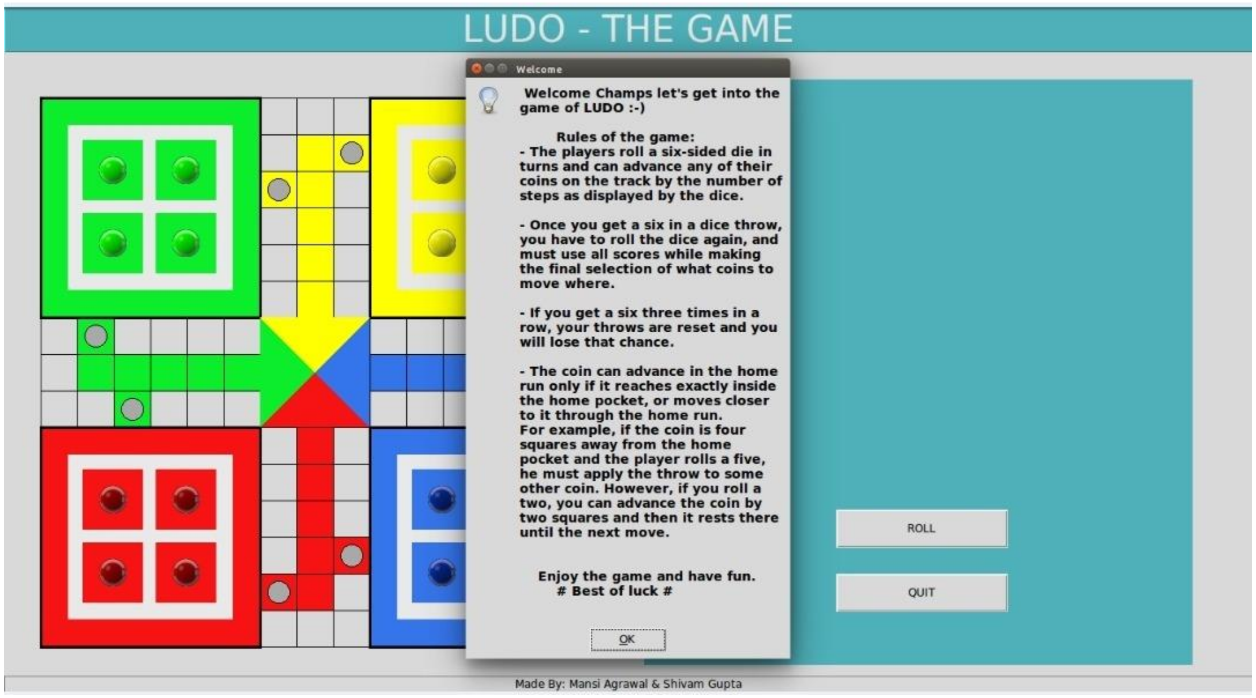
The first player to have all four of his pieces finish their journeys wins. The remaining players continue the game to determine the runner-ups.

- **GUI SCREEN SHOTS**

Ludo game using GUI and random numbers



Rules of the game



Welcome Champs let's get into the game of LUDO :-)

Rules of the game:

- The players roll a six-sided die in turns and can advance any of their coins on the track by the number of steps as displayed by the dice.

- Once you get a six in a dice throw, you have to roll the dice again, and must use all scores while making the final selection of what coins to move where.

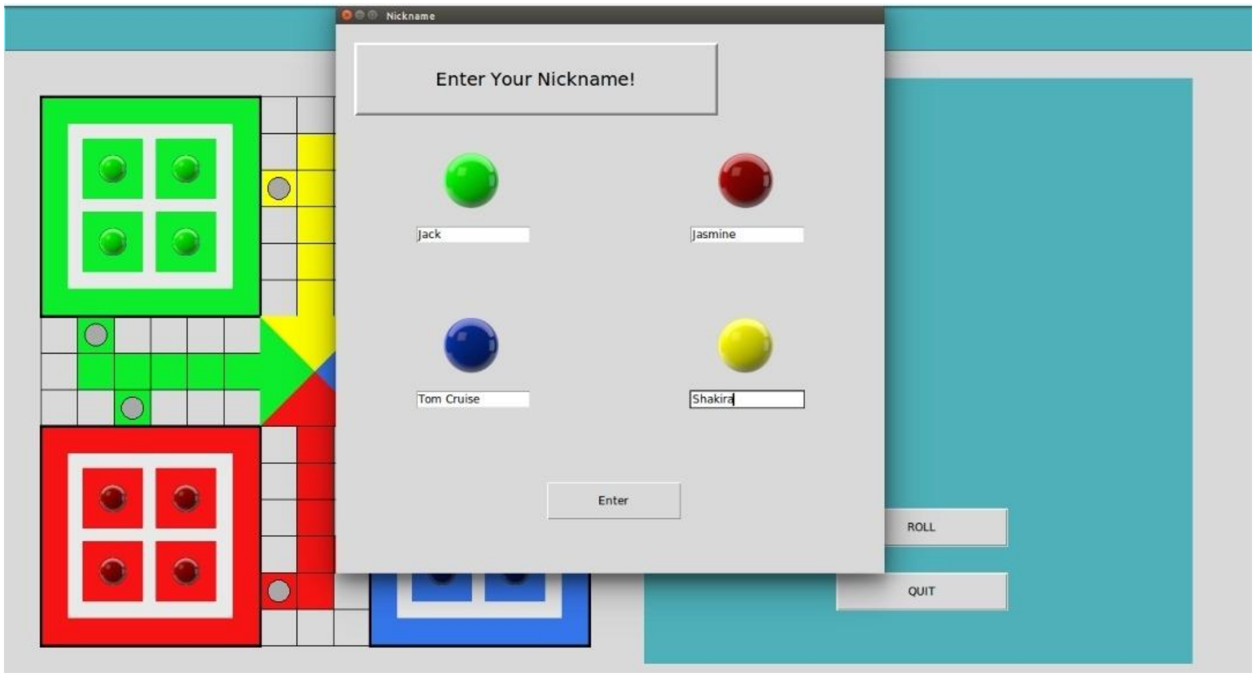
- If you get a six three times in a row, your throws are reset and you will lose that chance.

- The coin can advance in the home run only if it reaches exactly inside the home pocket, or moves closer to it through the home run. For example, if the coin is four squares away from the home pocket and the player rolls a five, he must apply the throw to some other coin. However, if you roll a two, you can advance the coin by two squares and then it rests there until the next move.

Enjoy the game and have fun.
Best of luck

Made By: Mansi Agrawal & Shivam Gupta

- **<u>Entering the nick names</u>**



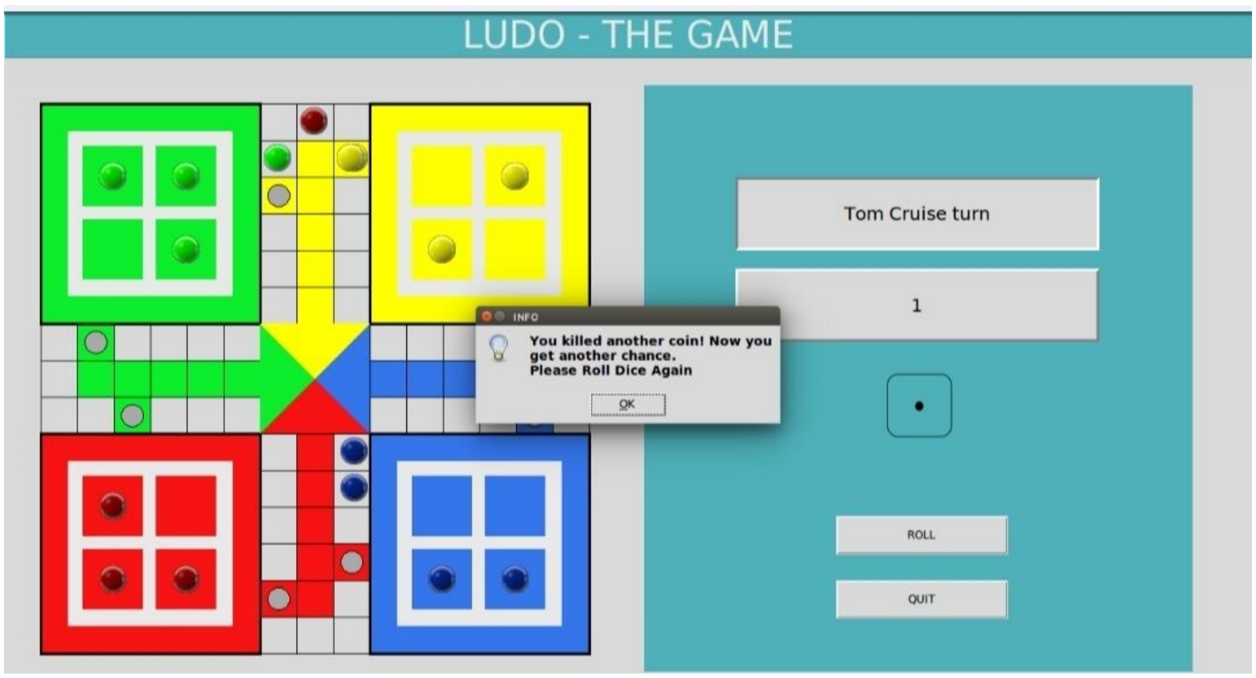
Jack

Jasmine

Tom Cruise

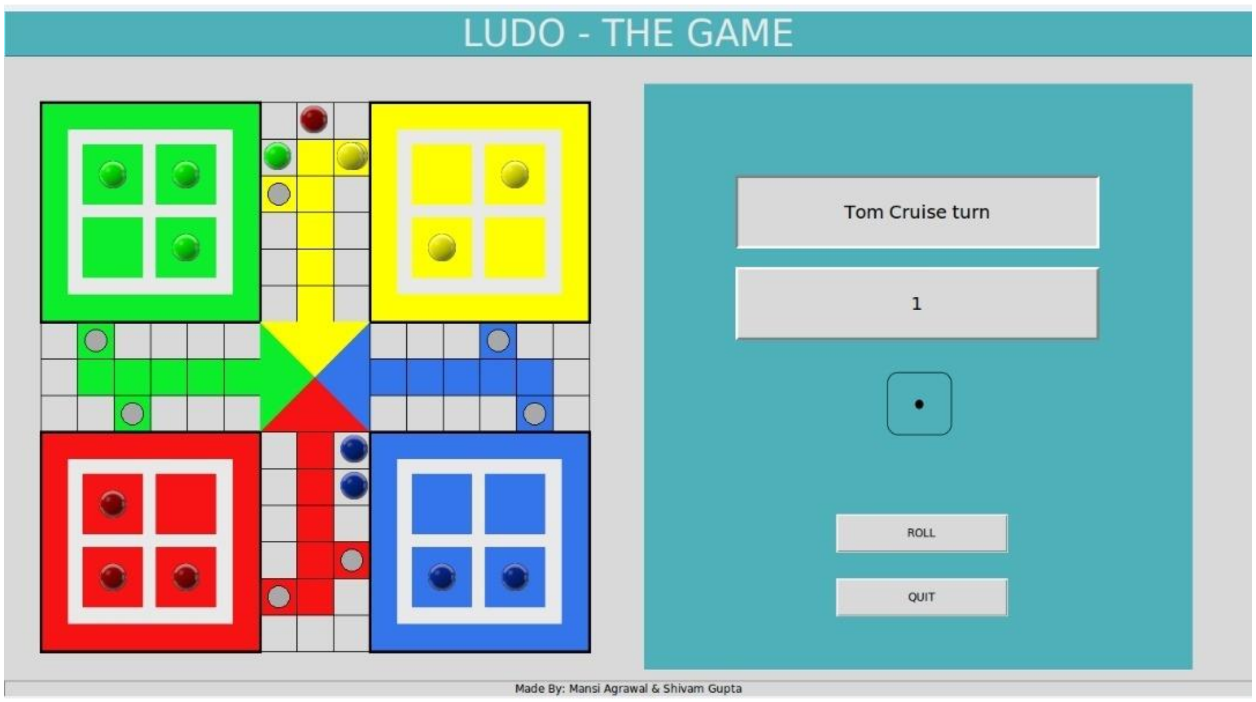
Shakira

Getting the chance by killing token



You killed another coin! Now you get another chance.
Please Roll Dice Again

Rolling the dice



- **<u>SOURCE CODE</u>**

```
from tkinter import *  # Tkinter is used as the GUI.
from tkinter import messagebox
import sys
import os
import random
import tkinter.messagebox

root = Tk()

root.resizable(width=False, height=False)  # The window size of the game.
root.geometry('1000x750')
root.configure(background='green')
root.title("Checkers")
```

```
logo = PhotoImage(file="whitebox.gif")      # Loading all the image files that are required
in the game.
logo2 = PhotoImage(file="red side.gif")     # Loading all the image files that are required
in the game.
logo3 = PhotoImage(file="red.gif")          # Loading all the image files that are required in
the game.
logo4 = PhotoImage(file="blue side.gif")
logo5 = PhotoImage(file="green side.gif")
logo6 = PhotoImage(file="yellow side.gif")
logo7 = PhotoImage(file="center.gif")
logoxx = PhotoImage(file="test.gif")
logog = PhotoImage(file="greenbox.gif")
logogs = PhotoImage(file="greenstop.gif")
logoy = PhotoImage(file="yellowbox.gif")
logoys = PhotoImage(file="yellowstop.gif")
logob = PhotoImage(file="bluebox.gif")
logobs = PhotoImage(file="bluestop.gif")
logor = PhotoImage(file="redbox.gif")
logors = PhotoImage(file="redstop.gif")
logoh = PhotoImage(file="head.gif")
logot = PhotoImage(file="tail.gif")
logoh1 = PhotoImage(file="head1.gif")
logot1 = PhotoImage(file="tail1.gif")
logoh2 = PhotoImage(file="head2.gif")
logot2 = PhotoImage(file="tail2.gif")
logoh3 = PhotoImage(file="head3.gif")
logot3 = PhotoImage(file="tail3.gif")
logoab= PhotoImage(file="blue.gif")
logoay= PhotoImage(file="yellow.gif")
logoag= PhotoImage(file="green.gif")

Label(image=logo2, width=298, height=298).place(x=-1, y=-1)              #setting up
board images
Label(image=logo4, width=300, height=300).place(x=(-2), y=(448))
Label(image=logo5, width=296, height=296).place(x=(450), y=(0))
Label(image=logo6, width=294, height=294).place(x=(450), y=(450))
Label(image=logo7, width=150, height=150).place(x=(298), y=(298))

c = 0                          #initializing variable and flags that are to be used in the game
lx = 0
bb =0
nc = 0
rollc = 0
rolls = []
RED = True
BLUE = False
```

```
GREEN = False
YELLOW = False
TURN = True
REDKILL = False
BLUEKILL = False
GREENKILL = False
YELLOWKILL = False


def board():                          #Drawing the board, piece by piece.

                                      #Splash Screen.
    tkinter.messagebox.showinfo(title=None, message="TO START GAME PRESS
OKAY & TO EXIT PRESS CROSS UP IN THE WINDOW")
    v = 0
    z = 0

    while (v != 300):          #Drawing White boxes
        z = 0
        while (z != 150):
            Label(image=logo, width=46, height=46).place(x=(300 + z), y=(0 + v))
            z = z + 50
        v = v + 50

    z = 0
    v = 0
    while (v != 300):         #Drawing White boxes
        z = 0
        while (z != 150):
            Label(image=logo, width=46, height=46).place(x=(0 + v), y=(300 + z))
            z = z + 50
        v = v + 50

    #####################

    v = 0
    z = 0

    while (v != 300):          #Drawing White boxes
        z = 0
        while (z != 150):
            Label(image=logo, width=46, height=46).place(x=(300 + z), y=(450 + v))
            z = z + 50
        v = v + 50

    z = 0
```

```
v = 0
while (v != 300):        #Drawing White boxes
    z = 0
    while (z != 150):
        Label(image=logo, width=46, height=46).place(x=(450 + v), y=(300 + z))
        z = z + 50
    v = v + 50

v = 0
while (v != 250):     #Drawing Green boxes
    Label(image=logog, width=46, height=46).place(x=(350), y=(50 + v))
    v = v + 50

Label(image=logog, width=46, height=46).place(x=(300), y=(100))
Label(image=logogs, width=46, height=46).place(x=(400), y=(50))

v = 0
while (v != 250):     #Drawing Yellow boxes
    Label(image=logoy, width=46, height=46).place(x=(450 + v), y=(350))
    v = v + 50

Label(image=logoy, width=46, height=46).place(x=(600), y=(300))
Label(image=logoys, width=46, height=46).place(x=(650), y=(400))

v = 0
while (v != 250):    #Drawing Red Boxes
    Label(image=logor, width=46, height=46).place(x=(50 + v), y=(350))
    v = v + 50

Label(image=logor, width=46, height=46).place(x=(100), y=(400))
Label(image=logors, width=46, height=46).place(x=(50), y=(300))

v = 0
while (v != 250):    #Drawing Blue Boxes
    Label(image=logob, width=46, height=46).place(x=(350), y=(450 + v))
    v = v + 50

Label(image=logobs, width=46, height=46).place(x=(300), y=(650))
Label(image=logob, width=46, height=46).place(x=(400), y=(600))

Label(image=logoh, width=46, height=46).place(x=250, y=400)        #Drawing arrows
Label(image=logot, width=46, height=46).place(x=300, y=450)
Label(image=logoh1, width=46, height=46).place(x=400, y=450)
Label(image=logot1, width=46, height=46).place(x=450, y=400)
Label(image=logoh2, width=46, height=46).place(x=450, y=300)
Label(image=logot2, width=46, height=46).place(x=400, y=250)
```

```
    Label(image=logoh3, width=46, height=46).place(x=300, y=250)
    Label(image=logot3, width=46, height=46).place(x=250, y=300)

class YBox:                              #Class of yellow box
    rap = None

    def __init__(self, num=-1, x=0, y=0, x0=0, y0=0, double=False, ):
        self.num = num                #no of gamepiece acc to box
        self.x = x                  #initial and final co-ordinates of the boxes
        self.y = y
        self.x0 = x0
        self.y0 = y0
        self.rap = Label(image=logoay, width=20, height=20)       #image of game piece.
        self.double = double                                  #if one game piece on top of another.

    def swap(self):                    #Swaps the position of gamepiece according to the
number on dice.
        self.rap.place(x=self.x0 + 13, y=self.y0 + 14)

class GBox:                            #Class of green box
    rap = None

    def __init__(self, num=-1, x=0, y=0, x0=0, y0=0, double=False, ):
        self.num = num
        self.x = x
        self.y = y
        self.x0 = x0
        self.y0 = y0
        self.rap = Label(image=logoag, width=20, height=20)
        self.double = double

    def swap(self):
        self.rap.place(x=self.x0 + 13, y=self.y0 + 14)

class BBox:                          #Class of Blue box
    rap = None

    def __init__(self, num=-1, x=0, y=0, x0=0, y0=0, double=False, ):
        self.num = num
        self.x = x
        self.y = y
        self.x0 = x0
        self.y0 = y0
        self.rap = Label(image=logoab, width=20, height=20)
        self.double = double
    def swap(self):
        self.rap.place(x=self.x0 + 13, y=self.y0 + 14)

class Box:                          #class of red box
    rap = None

    def __init__(self, num=-1, x=0, y=0, x0=0, y0=0, double=False, ):
        self.num = num
        self.x = x
        self.y = y
        self.x0 = x0
        self.y0 = y0
        self.rap = Label(image=logo3, width=20, height=20)
        self.double = double

    def swap(self):
        self.rap.place(x=self.x0 + 13, y=self.y0 + 14)


def main():                                 # Main game function.

    global box, redbox, bluebox, greenbox, yellowbox, redhome, bluehome, yellowhome,
greenhome
    global red, blue, yellow, green, rap, RED, BLUE, GREEN, YELLOW, dice, nc, TURN,
bb

    if c == 0:                              #constructs the game pieces first time the code is ran.

        board()

        box = [Box() for i in range(52)]  # list of co-ordinates of all the outer boxes

        redbox = [Box() for i in range(57)]  # list of co-ordinates of all the colored boxes,
excluding home and stop.
        bluebox = [Box() for i in range(57)]
        greenbox = [Box() for i in range(57)]
        yellowbox = [Box() for i in range(57)]

        redhome = [Box() for i in range(4)]  # list co-ordinates of all the home positions
        bluehome = [Box() for i in range(4)]
        greenhome = [Box() for i in range(4)]
        yellowhome = [Box() for i in range(4)]

        red = [Box() for i in range(4)]  # list of co-ordinates of all the game pieces in their
initial state
```

```
        blue = [BBox() for i in range(4)]  # that is equal to their respective home co-
ordinates.
        green = [GBox() for i in range(4)]
        yellow = [YBox() for i in range(4)]

        for i in range(2):                      #Populates list of homeboxes, colored boxes,
gamepieces and white boxes
            redhome[i].x = (100 + (100 * i))
            redhome[i].y = 100
            red[i].x0 = redhome[i].x
            red[i].y0 = redhome[i].y
            red[i].x = (red[i].x0) + 25
            red[i].y = (red[i].y0) + 25

            bluehome[i].x = (100 + (100 * i))
            bluehome[i].y = (550)
            blue[i].x0 = bluehome[i].x
            blue[i].y0 = bluehome[i].y
            blue[i].x = (blue[i].x0) + 25
            blue[i].y = (blue[i].y0) + 25

            yellowhome[i].x = (550 + (100 * i))
            yellowhome[i].y = (550)
            yellow[i].x0 = yellowhome[i].x
            yellow[i].y0 = yellowhome[i].y
            yellow[i].x = (yellow[i].x0) + 25
            yellow[i].y = (yellow[i].y0) + 25

            greenhome[i].x = (550 + (100 * i))
            greenhome[i].y = (100)
            green[i].x0 = greenhome[i].x
            green[i].y0 = greenhome[i].y
            green[i].x = (green[i].x0) + 25
            green[i].y = (green[i].y0) + 25

        for i in range(2, 4):
            redhome[i].x = (100 + (100 * (i - 2)))
            redhome[i].y = 200
            red[i].x0 = redhome[i].x
            red[i].y0 = redhome[i].y
            red[i].x = (red[i].x0) + 25
            red[i].y = (red[i].y0) + 25

            bluehome[i].x = (100 + (100 * (i - 2)))
            bluehome[i].y = (650)
            blue[i].x0 = bluehome[i].x
```

```
    blue[i].y0 = bluehome[i].y
    blue[i].x = (blue[i].x0) + 25
    blue[i].y = (blue[i].y0) + 25

    yellowhome[i].x = (550 + (100 * (i - 2)))
    yellowhome[i].y = (650)
    yellow[i].x0 = yellowhome[i].x
    yellow[i].y0 = yellowhome[i].y
    yellow[i].x = (yellow[i].x0) + 25
    yellow[i].y = (yellow[i].y0) + 25

    greenhome[i].x = (550 + (100 * (i - 2)))
    greenhome[i].y = 200
    green[i].x0 = greenhome[i].x
    green[i].y0 = greenhome[i].y
    green[i].x = (green[i].x0) + 25
    green[i].y = (green[i].y0) + 25

for i in range(6):
    box[i].x = 300
    box[i].y = (700 - (50 * i))

for i in range(6, 12):
    box[i].x = (250 - (50 * (i - 6)))
    box[i].y = (400)

box[12].x = 0
box[12].y = 350

for i in range(13, 19):
    box[i].x = (0 + (50 * (i - 13)))
    box[i].y = (300)

for i in range(19, 25):
    box[i].x = (300)
    box[i].y = (250 - (50 * (i - 19)))

box[25].x = 350
box[25].y = 0

for i in range(26, 32):
    box[i].x = (400)
    box[i].y = (0 + (50 * (i - 26)))

for i in range(32, 38):
    box[i].x = (450 + (50 * (i - 32)))
    box[i].y = (300)

box[38].x = 700
box[38].y = 350

for i in range(39, 45):
    box[i].x = (700 - (50 * (i - 39)))
    box[i].y = (400)

for i in range(45, 51):
    box[i].x = (400)
    box[i].y = (450 + (50 * (i - 45)))

box[51].x = 350
box[51].y = 700

# teshh
lx = 14
for i in range(52):
    redbox[i].x = box[lx].x
    redbox[i].y = box[lx].y
    lx = lx + 1
    if lx > 51:
        lx = 0

lx = 50
for i in range(7):
    redbox[lx].x = (0 + (50 * i))
    redbox[lx].y = 350
    lx = lx + 1
# blue
lx = 1
for i in range(52):

    bluebox[i].x = box[lx].x
    bluebox[i].y = box[lx].y
    lx = lx + 1
    if lx > 51:
        lx = 0

lx = 50
for i in range(7):
    bluebox[lx].x = 350
    bluebox[lx].y = (700 - (50 * i))
    lx = lx + 1
# yellow
```

```
        lx = 40
        for i in range(52):
            yellowbox[i].x = box[lx].x
            yellowbox[i].y = box[lx].y
            lx = lx + 1
            if lx > 51:
                lx = 0

        lx = 50
        for i in range(7):
            yellowbox[lx].x = (700 - (50 * i))
            yellowbox[lx].y = (350)
            lx = lx + 1

        # green
        lx = 27
        for i in range(52):

            greenbox[i].x = box[lx].x
            greenbox[i].y = box[lx].y

            lx = lx + 1
            if lx > 51:
                lx = 0

        lx = 50
        for i in range(7):
            greenbox[lx].x = 350
            greenbox[lx].y = (0 + (50 * i))
            lx = lx + 1

        for i in range(4):
            red[i].swap()
            blue[i].swap()
            green[i].swap()
            yellow[i].swap()                    #Population of all list is completed. Now game can
begin


    else:  # HERE ALL THE GAME OCCURS ... IF WAGHAIRA, MOVEMENT IDHAR
HOGI !!!

        if c >= 1:                              #This condition is true when a click is made.

            if RED == True and TURN == False:      #Red players turn
                print("Red's Turn")
        print("moves available: ", rolls)
        la = "RED"
        if (movecheck(red, redhome, redbox, la)) == False:  #Checks if player can
take a turn.
            BLUE = True
            RED = False
            clear()                              #clears variable, next players turn

        if RED == True:                               # searches if click is made on a red
game piece.
            for i in range(len(red)):
                if ((((cx > red[i].x0 + 13) and (cx < red[i].x + 13)) and (
                    (cy > red[i].y0 + 14) and (cy < red[i].y + 14)))
                    and (red[i].x0 == redhome[i].x) and (red[i].y0 == redhome[i].y)):
                    print("woila ")

                    if rolls[0 + nc] == 6:                #If a six occurs and gamepiece is in
home
                                                   #Game piece is moved onto the home box
                        red[i].x0 = redbox[0].x
                        red[i].y0 = redbox[0].y
                        red[i].x = redbox[0].x + 25
                        red[i].y = redbox[0].y + 25
                        red[i].num = 0
                        red[i].swap()
                        nc = nc + 1

                        if nc > len(rolls) - 1:          # check if all moves are made. so next
players turn.
                            BLUE = True
                            RED = False
                            clear()
                        break

                if ((((cx > red[i].x0 + 13) and (cx < red[i].x + 13)) and (       #if gamepiece
is outside home
                    (cy > red[i].y0 + 14) and (cy < red[i].y + 14)))
                    and ((red[i].x0 > 270) or (red[i].y0 > 270))):
                    print("woila ")
                    bb = ((red[i].num) + rolls[0 + nc])
                    # Winning condition

                    if bb > 57:                      #prevents moves greater than allowed
number
                        break
                        #bb = ((red[i].num) + rolls[0 + nc]) - 57
                    kill(redbox,blue,yellow,green,bluehome,yellowhome,greenhome)
#checks if a kill can be made.

                    red[i].x0 = redbox[bb].x
                    red[i].y0 = redbox[bb].y
                    red[i].x = redbox[bb].x + 25
                    red[i].y = redbox[bb].y + 25
                    red[i].swap()
                    red[i].num = bb
                    doublecheck(red)                  #checks if the gamepiece can be
made as a double.

                    nc = nc + 1
                    if bb == 57:                      #checks if game has traversed all the
blocks
                        # del red[i]
                        red.remove(red[i]);

                    if nc > len(rolls) - 1:
                        BLUE = True                   #next players turn.
                        RED = False
                        clear()
                    break


                # BLUES TURN!!!!!!!!!!!!!!!!!!!!

          if BLUE == True and TURN == False:                #same as REDS CODE
        print("Blue's Turn")
        print("moves available: ", rolls)
        la="BLUE"
        if (movecheck(blue, bluehome, bluebox, la)) == False:
            print("NO MOVES SIR JEE")
            BLUE = False
            YELLOW = True
            clear()

        if BLUE == True:

            for i in range(len(blue)):
                if ((((cx > blue[i].x0 + 13) and (cx < blue[i].x + 13)) and (
                    (cy > blue[i].y0 + 14) and (cy < blue[i].y + 14)))
                    and (blue[i].x0 == bluehome[i].x) and (blue[i].y0 == bluehome[i].y)):
                    print("woila ")
                    if rolls[0 + nc] == 6:

                        blue[i].x0 = bluebox[0].x
                        blue[i].y0 = bluebox[0].y
                        blue[i].x = bluebox[0].x + 25
                        blue[i].y = bluebox[0].y + 25
                        blue[i].num = 0
                        blue[i].swap()
                        nc = nc + 1

                        if nc > len(rolls) - 1:
                            YELLOW = True
                            BLUE = False
                            clear()
                        break

                if ((((cx > blue[i].x0 + 13) and (cx < blue[i].x + 13)) and (
                    (cy > blue[i].y0 + 14) and (cy < blue[i].y + 14)))
                    and ((blue[i].x0 > 270) or (blue[i].y0 < 470))):
                    print("woila ")
                    bb = ((blue[i].num) + rolls[0 + nc])
                    if bb > 57:
                        break
                        # bb= ((blue[i].num) + rolls[0 + nc]) - 52

                    kill(bluebox,red,yellow,green,redhome,yellowhome,greenhome)

                    blue[i].x0 = bluebox[bb].x
                    blue[i].y0 = bluebox[bb].y
                    blue[i].x = bluebox[bb].x + 25
                    blue[i].y = bluebox[bb].y + 25
                    blue[i].swap()
                    blue[i].num = bb
                    doublecheck(blue)
                    nc = nc + 1
                    if bb == 57:
                        # del red[i]
                        blue.remove(blue[i]);

                    if nc > len(rolls) - 1:
                        YELLOW = True
                        BLUE = False
                        clear()
                    break

                # YELLOWS TURN!!!!!!!!!!!!!!!!!!!!
```

```
        if YELLOW == True and TURN == False:                    #Same as RED's code
            print("Yellows's Turn")
            print("moves available: ", rolls)
            la="YELLOW"
            if (movecheck(yellow, yellowhome, yellowbox,la)) == False:
                print("NO MOVES SIR JEE")
                YELLOW = False
                GREEN = True
                clear()

            if YELLOW == True:

                for i in range(len(yellow)):
                    if ((((cx > yellow[i].x0 + 13) and (cx < yellow[i].x + 13)) and (
                            (cy > yellow[i].y0 + 14) and (cy < yellow[i].y + 14)))
                        and (yellow[i].x0 == yellowhome[i].x) and (yellow[i].y0 ==
yellowhome[i].y)):
                        print("woila ")

                        if rolls[0 + nc] == 6:

                            yellow[i].x0 = yellowbox[0].x
                            yellow[i].y0 = yellowbox[0].y
                            yellow[i].x = yellowbox[0].x + 25
                            yellow[i].y = yellowbox[0].y + 25
                            yellow[i].num = 0
                            yellow[i].swap()
                            nc = nc + 1

                            if nc > len(rolls) - 1:
                                YELLOW = False
                                GREEN = True
                                clear()
                            break

                    if ((((cx > yellow[i].x0 + 13) and (cx < yellow[i].x + 13)) and (
                            (cy > yellow[i].y0 + 14) and (cy < yellow[i].y + 14)))
                        and ((yellow[i].x0 < 470) or (yellow[i].y0 < 470))):
                        print("woila ")
                        bb = ((yellow[i].num) + rolls[0 + nc])
                        if bb > 57:
                            break
                            #bb = ((yellow[i].num) + rolls[0 + nc]) - 52

                        kill(yellowbox,blue,red,green,bluehome,redhome,greenhome)
                        yellow[i].x0 = yellowbox[bb].x
                        yellow[i].y0 = yellowbox[bb].y
                        yellow[i].x = yellowbox[bb].x + 25
                        yellow[i].y = yellowbox[bb].y + 25
                        yellow[i].swap()
                        yellow[i].num = bb
                        doublecheck(yellow)
                        nc = nc + 1
                        if bb == 57:
                            # del red[i]
                            yellow.remove(yellow[i]);

                        if nc > len(rolls) - 1:
                            YELLOW = False
                            GREEN = True
                            clear()
                        break


                    # GREENS TURN!!!!!!!!!!!!!!!!!!!

            if GREEN == True and TURN == False:                    #Same as RED's code
                print("Green's Turn")
                print("moves available: ", rolls)
                la="GREEN"
                if (movecheck(green, greenhome, greenbox,la)) == False:
                    print("NO MOVES SIR JEE")
                    GREEN = False
                    RED = True
                    clear()

                if GREEN == True:

                    for i in range(len(green)):
                        if ((((cx > green[i].x0 + 13) and (cx < green[i].x + 13)) and (
                                (cy > green[i].y0 + 14) and (cy < green[i].y + 14)))
                            and (green[i].x0 == greenhome[i].x) and (green[i].y0 ==
greenhome[i].y)):
                            print("woila ")

                            if rolls[0 + nc] == 6:

                                green[i].x0 = greenbox[0].x
                                green[i].y0 = greenbox[0].y
                                green[i].x = greenbox[0].x + 25
                    green[i].y = greenbox[0].y + 25
                    green[i].num = 0
                    green[i].swap()
                    nc = nc + 1
                    print("green x.y: ", green[i].x0, green[i].y0)

                    if nc > len(rolls) - 1:
                        GREEN = False
                        RED = True
                        clear()
                    break

              if ((((cx > green[i].x0 + 13) and (cx < green[i].x + 13)) and (
                        (cy > green[i].y0 + 14) and (cy < green[i].y + 14)))
                  and ((green[i].x0 < 470) or (green[i].y0 < 470))):
                  print("woila ")
                  bb = ((green[i].num) + rolls[0 + nc])
                  if bb > 57:
                      break
                      # bb = ((green[i].num) + rolls[0 + nc]) - 52

                  kill(greenbox,blue,yellow,red,bluehome,yellowhome,redhome)

                  green[i].x0 = greenbox[bb].x
                  green[i].y0 = greenbox[bb].y
                  green[i].x = greenbox[bb].x + 25
                  green[i].y = greenbox[bb].y + 25
                  green[i].swap()
                  green[i].num = bb
                  nc = nc + 1
                  doublecheck(green)
                  if bb == 57:
                      # del red[i]
                      green.remove(green[i]);

                  if nc > len(rolls) - 1:
                      GREEN = False
                      RED = True
                      clear()
                  break


main()    #Main functin is called once when c==0 to intialize all the gamepieces.


def leftClick(event):  # Main play function is called on every left click.
```

```
    global c, cx, cy, RED, YELLOW
    c = c + 1
    cx = root.winfo_pointerx() - root.winfo_rootx()  # This formula returns the x,y co-
ordinates of the mouse pointer relative to the board.
    cy = root.winfo_pointery() - root.winfo_rooty()

    print("Click at: ", cx, cy)

    main()          #Main function called on every click to progress the game


root.bind("<Button-1>", leftClick)


def turn():   #Prints whoose turn is it

    if RED == True:
        L2 = Label(root, text="   Red's Turn    ", fg='Black', background='green',
font=("Arial", 24, "bold"))
        L2.place(x=770, y=50)

    if BLUE == True:
        L2 = Label(root, text="   Blue's Turn   ", fg='Black', background='green',
font=("Arial", 24, "bold"))
        L2.place(x=770, y=50)

    if GREEN == True:
        L2 = Label(root, text="Green's Turn  ", fg='Black', background='green',
font=("Arial", 24, "bold"))
        L2.place(x=770, y=50)

    if YELLOW == True:
        L2 = Label(root, text="Yellow's Turn", fg='Black', background='green', font=("Arial",
24, "bold"))
        L2.place(x=770, y=50)


def roll():   #Rolling function that rolls a dice, goes again if its a six
    global rollc, dice, dice1, dice2, TURN, rolls

    if TURN == True:

        rollc = rollc + 1
        print("roll: ", rollc)
        if rollc == 1:
            dice = random.randint(1, 6)
            L1 = Label(root, text=dice, fg='Black', background='green', font=("Arial", 24,
"bold"))
            L1.place(x=800, y=200)
            print("dice: ", dice)
            rolls.append(dice)
            if dice != 6:
                rollc = 0
                TURN = False

        if rollc == 2:
            if dice == 6:
                dice1 = random.randint(1, 6)
                L3 = Label(root, text=dice1, fg='Black', background='green', font=("Arial", 24,
"bold"))
                L3.place(x=800, y=250)
                rolls.append(dice1)
                if dice1 != 6:
                    rollc = 0
                    TURN = False

        if rollc == 3:
            if dice1 == 6:
                dice2 = random.randint(1, 6)
                L4 = Label(root, text=dice2, fg='Black', background='green', font=("Arial", 24,
"bold"))
                L4.place(x=800, y=300)
                rolls.append(dice2)
                rollc = 0
                TURN = False


def clear():        #clears all the variable prior to next player's turn
    global nc, rolls, TURN, L1, L3, L4
    nc = 0
    del rolls[:]
    TURN = True
    L1 = Label(root, text="        ", fg='Black', background='green', font=("Arial", 24, "bold"))
    L1.place(x=800, y=200)
    L3 = Label(root, text="        ", fg='Black', background='green', font=("Arial", 24, "bold"))
    L3.place(x=800, y=250)
    L4 = Label(root, text="        ", fg='Black', background='green', font=("Arial", 24, "bold"))
    L4.place(x=800, y=300)
    print("cleared")
    turn()
```

```
def movecheck(r, rh, rb, la):       #Check if the player can make a move

    if (dice == 6 and dice1 == 6 and dice2 == 6):
        return False

    win=True                                                  #Checking if the game is won or the player
can make any moves.
    for j in range(4):
        if (r[j].x0 != rb[56].x) and (r[j].y0 != rb[56].y):
            win=False

    if win == True:                                          #If all gamepieces home, prints that the
player has won
        print("YOU HAVE WON")
        L2 = Label(root, text=(la + "Wins"), fg='Black', background='green', font=("Arial",
24, "bold"))
        L2.place(x=770, y=500)
        return False

    if win == False and dice != 6:                    #if its not a 6 and all game pieces inside
home, then next players turn
        for i in range(len(r)):
            if(r[i].num != -1):
                (print("good hai"))
                return True
        print("jani all in")
        return False

def kill(a,b,c,d,bh,ch,dh):   #function that determines if a gamepiece can be killed

    #if the game piece is not on a stop
    if ((a[bb].x0 != box[1].x and a[bb].y0 != box[1].y) and (a[bb].x0 != box[14].x and
a[bb].y0 != box[14].y) and
        (a[bb].x0 != box[9].x and a[bb].y0 != box[9].y) and (a[bb].x0 != box[22].x and
a[bb].y0 != box[22].y) and
        (a[bb].x0 != box[27].x and a[bb].y0 != box[27].y) and (a[bb].x0 != box[35].x and
a[bb].y0 != box[35].y) and
        (a[bb].x0 != box[40].x and a[bb].y0 != box[40].y) and (a[bb].x0 != box[48].x and
a[bb].y0 != box[48].y)):


        #if the game piece of another color and its on the same block and it is not a
double, a kill is made
        for i in range (len(b)):
            if (b[i].x0 == a[bb].x and b[i].y0 == a[bb].y and (b[i].double == False)):
                b[i].x0 = bh[i].x
                b[i].y0 = bh[i].y
                b[i].x = bh[i].x + 25
                b[i].y = bh[i].y + 25
                b[i].num=-1
                b[i].swap()
                break

        for i in range (len(c)):
            if (c[i].x0 == a[bb].x and c[i].y0 == a[bb].y and (c[i].double == False)):
                c[i].x0 = ch[i].x
                c[i].y0 = ch[i].y
                c[i].x = ch[i].x + 25
                c[i].y = ch[i].y + 25
                c[i].num=-1
                c[i].swap()
                break

        for i in range (len(d)):
            if (d[i].x0 == a[bb].x and d[i].y0 == a[bb].y and (d[i].double == False)):
                d[i].x0 = dh[i].x
                d[i].y0 = dh[i].y
                d[i].x = dh[i].x + 25
                d[i].y = dh[i].y + 25
                d[i].num=-1
                d[i].swap()
                break

def doublecheck(a):        #makes a double is two or more gamepieces on top of another.

    for k in range (len(a)):
        a[k].double = False

    for i in range (len(a)):
        for j in range (len(a)):
            if (a[i].num == a[j].num) and (i != j):
                a[j].double = True
                a[i].double = True



turn()            #prints the "red player's turn" initially

button = Button(root, text="   ROLL   ", relief="raised", font=("Arial", 20),
                command=roll)  # call roll function evertime this button is clicked
button.place(x=805, y=120)
```

```
root.mainloop()
```

Reference

https://github.com/trilokkumarreddy/ludo-game-using-GUI-AND-RANDOM-NUMBERS.git