PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Computer Science & Engineering

Dice Rolling Simulator

Python For Computational Problem Solving

(UE21CS111A)

Submitted by

PES1UG21EC123 – K N CHETAN

PES1UG21CS182 – DILIP H

PES1UG21CS210 – GOWTHAM S

TABLE OF CONTENTS

ABSTRACT

The use of dice has been an integral part of tabletop roleplaying games since the beginning. Ancient Chinese and Egyptian tombs date back to 2000 BCE, and cubic dice with markings nearly identical to modern dice have been found. Children will learn about probability while playing dice games, get to use math calculation skills, improve their visual perception and manual dexterity, as well as improve their social skills. When a person throws a dice and searches for the outcome, they are usually curious about the outcome, and this also helps them relax and unwind.

In this project titled "**Dice Rolling Simulator**" we have developed two games Single Player and Multi Player. These games are implemented using the Python modules Tkinter, random, PIL, playsound and pyttsx3.

INTRODUCTION

Dice have been an integral part of tabletop roleplaying games since the beginning. Cubical dice with markings practically equivalent to those of modern dice have been found in Chinese excavation from 600 BCE and in Egyptian tombs dating from 2000 BCE.

Playing the games with dice expose the kids to the wonder of probability, put math calculation skills to use, improve visual perception and manual dexterity, also while playing games children will improve social skills. While throwing a dice and looking for output, most will be in the curiosity of the output, this also helps them to relax from stress.

Dice are implements used for generating random numbers in a variety of social and gambling games. Known since antiquity, dice have been called the oldest gaming instruments. They are typically cube-shaped and marked with one to six dots on each face. The most common method of dice manufacture involves injection molding of plastic followed by painting.

Dice are thrown onto a surface either from the hand or from a container designed for this (such as a cup or tray). The face of the die that is uppermost when it comes to rest provides the value of the throw.

The result of a die roll is determined by the way it is thrown, according to the laws of classical mechanics. A die roll is made random by uncertainty in minor factors such as tiny movements in the thrower's hand; they are thus a crude form of hardware random number generator.

One typical contemporary dice game is craps, where two dice are thrown simultaneously and wagers are made on the total value of the two dice. Dice are frequently used to introduce randomness into board games, where they are often used to decide the distance through which a piece will move along the board (as in backgammon and Monopoly).

In the future, dice manufacturers will concentrate on increasing sales and improving the production process. To increase sales, dice marketers will be involved in developing new games that utilize different types of dice. These games will require new types of dice that may have different shapes, sizes, and plastic compositions. From a production standpoint, future improvements will focus on increasing manufacturing speeds, minimizing chemical waste, and reducing overall costs.

In this project we used the modules such as Tkinter, random, PIL, playsound and pyttsx3.

From module Tkinter we used 'title' to design the title of project, 'Canvas' is used to add a structured graphics, 'Label' is used to write the heading of project "DICE ROLLIING STIMULATOR". 'ImageTk.PhotoImage' to add an image on Parent window of the project.

Under parent window we have designed a two-child window, one for one player game and other for two player game. We had designed each game by using a two dice system.

Game 1:

In Single player game we have set a random target from range 2 to 12, let that number be called X. In this game the player wins if he/she gets sum of two dice as X. To get a required number, a player is given 10 chances, if he/she is not able to get the given target score within 10 chances, the Player will lose the game.

We have designed a start button and restart button, after starting game we get a chance to roll two dices, the output of result of dice is giving by images using unicodes. If the Player is unable to get given target score with in 10 chances, we show the message from messagebox as "YOU LOST!!", else the message will be "YOU WON!!"

If the Player continuous to play, he/she can play again by clicking on restart button or can quit the game.

Game 2:

This is a multi-player game which is played by two players. Here, each player is allowed to role a pair of dice for a number of five times, i.e., each player is given five chances. During each player's turn, sum of the two dice is calculated and stored as their score. After both the players are done with their turns, their total score of all the five chances is compared with each other. Say, if Player 1 gets a total score more than Player 2, then Player 1 is declared as the winner, else Player 2 is the winner. And, if both the Players score the same, then the result will be a tie, and the result will be declared accordingly.

CODE IMPLEMENTATION

```
import tkinter as tk

import random

from playsound import playsound

from tkinter import RAISED, messagebox as mb

from PIL import Image,ImageTk

import pyttsx3




root=tk.Tk()

root.title("GAME")

root.geometry("800x600")

c1=tk.Canvas(root, width=800, height=600,bg='black',borderwidth=0)

c1.pack()

label1=tk.Label(root,text="DICE ROLLING
SIMULATION",font=('Times',25,'bold'),fg='red',bg='black')

c1.create_window(430,50,window=label1)

photo=ImageTk.PhotoImage(Image.open("singledie.jpg"))

photo1=ImageTk.PhotoImage(Image.open("twodice.jpg"))



def childwindow1():

    root.iconify()
```

```
r1=tk.Toplevel(root)
r1.geometry('800x600')
r1.title('Roll Dice')
z=random.randint(2,12)
c = tk.Canvas(r1, width=800, height=600,bg='black')
c.pack()


def start():
    global bttn_clicks
    bttn_clicks= 0
    label1.configure(text="")
    label2.configure(text="Not rolled yet")
    button1.config(text="Roll",command=roll_dice,bg='black')
    playsound("click.mp3")
    button2.config(state='normal')
def restart():
    global bttn_clicks
    nonlocal z
    bttn_clicks= 0
    z=random.randint(2,12)
    label3.config(text=f"Winning rule: The player wins if he/she gets a sum of {z} on rolling the dice,within 10 chances\nTARGET={z}")
```

```
    label1.configure(text="")
    label2.configure(text="Not rolled yet")
    ldice.config(text="")
    label4.config(text="",bg="black")
    button1.config(state="normal")


def roll_dice():
    nonlocal z
    global bttn_clicks
    dice = ['\u2680', '\u2681', '\u2682', '\u2683', '\u2684', '\u2685']
    d = {'\u2680':1, '\u2681':2, '\u2682':3, '\u2683':4, '\u2684':5, '\u2685':6}
    playsound("dice roll.mp3")
    die1 = random.choice(dice)
    die2 = random.choice(dice)
    ldice.configure(text=f'{die1} {die2}')
    c.create_window(350, 250, window=ldice)
    res = d[die1] + d[die2]
    label2.configure(text="You got  "+str(res))
    bttn_clicks += 1
    label1['text'] = "Dice rolled: " + str(bttn_clicks) + " times"
    if (bttn_clicks == 10 and res != z):
        button1.configure(state='disabled')
```

```
            mb.showinfo("RESULT","YOU LOSE!!")
            button1.config(state="disabled")
            label4.config(text="You lost!",bg="red")

        elif (res==z):
            button1.configure(state='disabled')
            mb.showinfo("RESULT","YOU WON!!")
            button1.config(state="disabled")
            label4.config(text="You won!",bg="green")

    def quit_func():
        if mb.askyesno("verify","QUIT ?"):
            engine1.say("Thank you!")
            engine1.runAndWait()
            engine1.stop()
            r1.destroy()
            root.destroy()

    ldice = tk.Label(r1, text='', font=('Times', 200),fg='green',bg='black')
    c.create_window(480,200,window=ldice)
    button1 = tk.Button(r1, text='Start', font=('times',
20,"bold"),background="black",foreground='white',height=1, width=15,
command=start,borderwidth=10,relief=RAISED)
```

```
    c.create_window(430, 50, window=button1)

    button2 = tk.Button(r1, text='Restart', font=('times',
20,"bold"),background="black",foreground='white',height=1, width=15,
command=restart,borderwidth=10,relief=RAISED)

    c.create_window(150, 50, window=button2,state='disabled')

    button3 = tk.Button(r1, text='EXIT', font=('times',
20,"bold"),background="black",foreground='white',height=1, width=10,
command=quit_func,borderwidth=10,relief=RAISED)

    c.create_window(670, 50, window=button3)

    label1 = tk.Label(r1, text='', font=('Times',20,'bold'),fg='red',bg='black')

    c.create_window(180, 410, window=label1)

    label2 = tk.Label(r1, text='Not rolled yet', font=('Times',20,'bold'),bg='black',fg='blue',width=12)

    c.create_window(630, 410, window=label2)

    label3 = tk.Label(r1, text=f"GAME: The player wins if he/she gets a sum of {z} on rolling the
dice,within 10 chances\nTARGET={z}", font=('Times',14,'bold'),fg='purple',bg="black")

    c.create_window(400, 500, window=label3)

    label4=tk.Label(r1,text='',font=('Times',25,'bold'),fg='white',width=10,bg='black')

    c.create_window(430,450,window=label4)

    r1.mainloop()


def childwindow2():

    root.iconify()

    r=tk.Toplevel(root)

    r.title('DICE ROLLING SIMULATION')
```

```
r.geometry("800x600")

c = tk.Canvas(r, width=800, height=600,bg='black')

c.pack()

sum_h=0

sum_k=0


def player1():

        global bttn_clicks1

        nonlocal sum_h

        dice = ['\u2680', '\u2681', '\u2682', '\u2683', '\u2684', '\u2685']

        d = {'\u2680':1, '\u2681':2, '\u2682':3, '\u2683':4, '\u2684':5, '\u2685':6}

        startbutton.config(state='disabled')

        playsound("dice roll.mp3")

        h = random.choice(dice)

        i=random.choice(dice)

        ldice.configure(text=f'{h}{i}')

        c.create_window(350, 250, window=ldice)

        res = d[h]+d[i]

        sum_h+=d[h]+d[i]

        label4.configure(text="Player1's score: "+str(sum_h),fg='blue')

        label2.configure(text="You got "+str(res))

        bttn_clicks1 += 1
```

```
        label1['text'] = "Dice rolled: " + str(bttn_clicks1) + " times\n(PLAYER 1)"
        startbutton.config(state='normal')
        if(bttn_clicks1==5):
            startbutton.config(command=player2)
            label4.configure(text="Player1's total: "+str(sum_h),fg='blue')
            mb.showinfo("","Player1's Score: "+str(sum_h)+"\nPress the roll button to continue")
            ldice.configure(text="")
            label1.configure(text="PLAYER2",fg='red')
            label2.configure(text="",font=('Times',14,'bold'),fg='red')

def player2():
        global bttn_clicks2
        nonlocal sum_k
        dice = ['\u2680', '\u2681', '\u2682', '\u2683', '\u2684', '\u2685']
        d = {'\u2680':1, '\u2681':2, '\u2682':3, '\u2683':4, '\u2684':5, '\u2685':6}
        playsound("dice roll.mp3")
        k = random.choice(dice)
        l=random.choice(dice)
        ldice.configure(text=f'{k}{l}')
        res1 = d[k]+d[l]
        sum_k+=d[k]+d[l]
        label5.config(text="Player2's score: "+str(sum_k),fg='blue')
```

```
        label2.configure(text="You got "+str(res1),font=('Times',20,'bold'),fg='white')
        bttn_clicks2 += 1
        label1['text'] = "Dice rolled: " + str(bttn_clicks2) + " times\n(PLAYER 2)"
        if sum_k>sum_h:
            mb.showinfo("RESULT",f"PLAYER2 won")
            label6.configure(text="RESULT:Player 2 has
won",font=('Times',16,'bold'),fg='yellow')
            startbutton.config(state="disabled")
            label5.configure(text="Player2's total: "+str(sum_k),fg='blue')



        if(bttn_clicks2==5):
            label5.configure(text="Player2's total: "+str(sum_k),fg='blue')
            mb.showinfo("","Game is over \nPlayer2's Total: "+str(sum_k))
            a=sum_h>sum_k
            c=sum_h==sum_k
            startbutton.config(state="disabled")
            if a:
                mb.showinfo("RESULT",f"PLAYER1 won by {sum_h-sum_k} points")
            if c:
                mb.showinfo("RESULT",f"It's a TIE with {sum_h} points")
```

```
            if a:
                label6.configure(text="RESULT:Player 1 has
won",font=('Times',16,'bold'),fg='yellow')
            if c:
                label6.configure(text="RESULT:It's a tie",font=('Times',16,'bold'),fg='red')


    def start():
        global bttn_clicks1
        global bttn_clicks2
        bttn_clicks1= 0
        bttn_clicks2= 0
        label1.configure(text="")
        label2.configure(text="Not rolled yet")
        startbutton.configure(text='Roll',command=player1,bg="black",fg="white")
        playsound("click.mp3")
        button1.configure(state="normal")


    def restart():
        nonlocal sum_h,sum_k
        sum_h=0
        sum_k=0
        global bttn_clicks1,bttn_clicks2
```

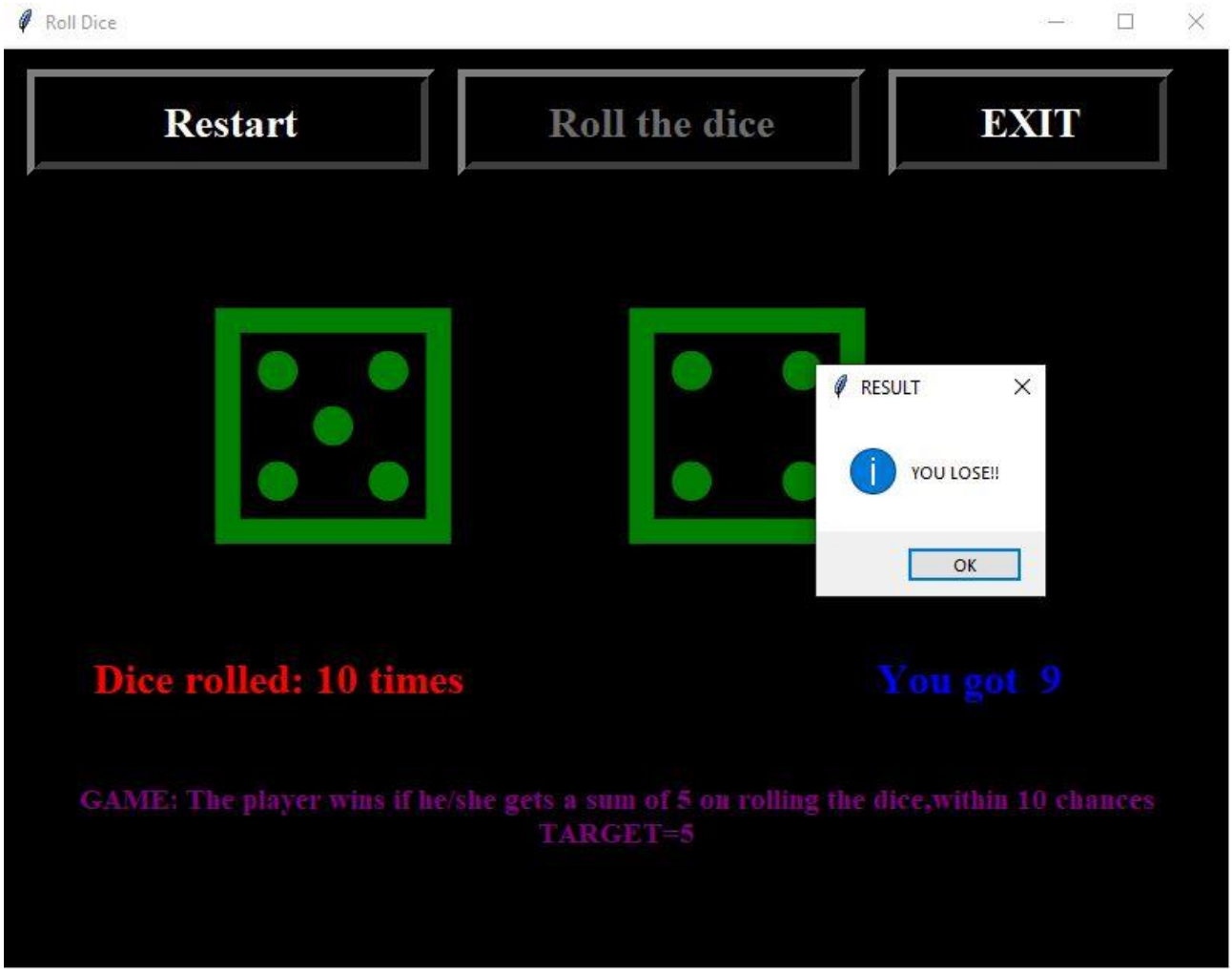
```
        bttn_clicks1= 0
        bttn_clicks2= 0
        ldice.configure(text="")
        label1.configure(text="")
        label2.configure(text="Not rolled yet")
        label4.configure(text="DICE ROLLING SIMULATOR",fg='red')
        label5.configure(text="")
        label6.configure(text="")
        startbutton.configure(state='normal',command=player1)


    def quit_func():
        if mb.askyesno("verify","QUIT ?"):
            engine1.say("Thank you!")
            engine1.runAndWait()
            engine1.stop()
            r.destroy()
            root.destroy()


    ldice = tk.Label(r, text='', font=('Times', 200),fg='green',bg='black')
    c.create_window(900,180,window=ldice)
    button1 = tk.Button(r, text='Restart',state='disabled',font=('times',
20,"bold"),background="black",foreground='white',height=1, width=15, command=restart)
```

```
    c.create_window(140, 50, window=button1)

    startbutton=tk.Button(r,text="START",font=('times',
20,"bold"),background="black",foreground='white',height=1, width=15,command=start)

    c.create_window(400,50,window=startbutton)

    button2=tk.Button(r,text="EXIT",command=quit_func,font=('times',
20,"bold"),background="black",foreground='white',height=1, width=15)

    c.create_window(670,50,window=button2)

    label1 = tk.Label(r, text='', font=('Times',18,'bold'),fg='red',bg='black')

    c.create_window(190, 410, window=label1)

    label2 = tk.Label(r, text='Not rolled yet', font=('Times',20,'bold'),bg='black',fg='white',width=12)

    c.create_window(640, 410, window=label2)

    label3 = tk.Label(r, text="Rule:The player who get's the highest score(5 ROLLS) wins the
game",font=('Times',15,'bold'),bg='black',fg='purple')

    c.create_window(400, 590, window=label3)

    label4 = tk.Label(r, text="DICE ROLLING SIMULATOR",
font=('Times',16,'bold'),fg='red',bg='black')

    c.create_window(400, 500, window=label4)

    label5 = tk.Label(r, text="", font=('Times',16,'bold'),fg='white',bg='black')

    c.create_window(400,530,window=label5)

    label6 = tk.Label(r, text="", font=('Times',15,'bold'),fg='white',bg='black')

    c.create_window(400,560,window=label6)

    r.mainloop()
```

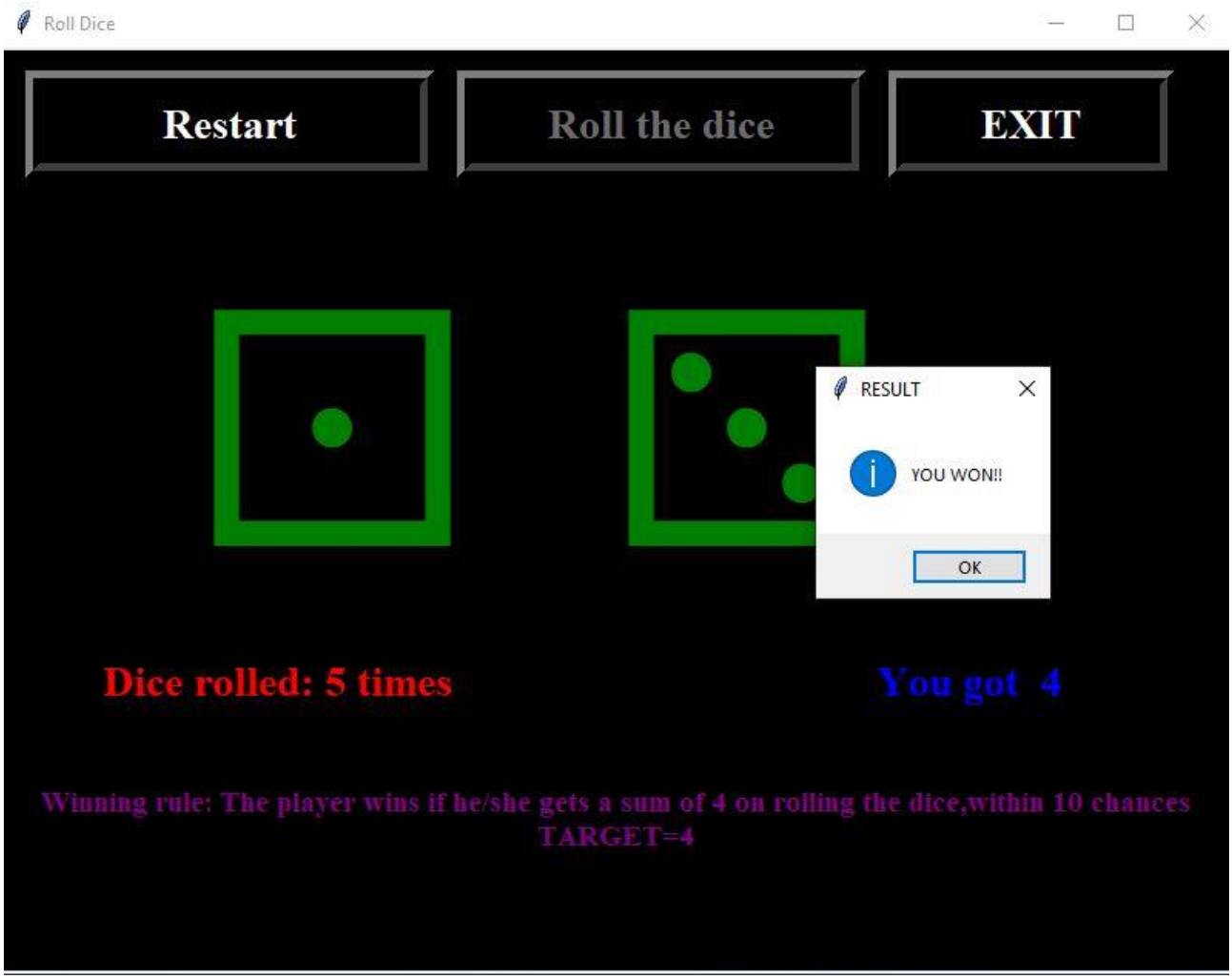
```
b1=tk.Button(root,text='game 1',font=('times',
20,"bold"),background="white",foreground='white',height=200,
width=200,command=childwindow1,image=photo,borderwidth=10,relief=RAISED)

c1.create_window(400,200,window=b1)

b2=tk.Button(root,text='game 2',font=('times',
20,"bold"),background="white",foreground='white',height=200,
width=200,command=childwindow2,image=photo1,borderwidth=10,relief=RAISED)

c1.create_window(400,450,window=b2)

label2=tk.Label(root, text="GAME 1",font=('times',20,'bold'),bg='black',fg='purple')

c1.create_window(400,290,window=label2)

label3=tk.Label(root, text="GAME 2",font=('times',20,'bold'),bg='black',fg='purple')

c1.create_window(400,570,window=label3)

engine = pyttsx3.init()

engine.setProperty('rate', 125)

engine1 = pyttsx3.init()

engine1.setProperty('rate', 125)

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[0].id)

engine.say("welcome to dice rolling simulator")

engine.runAndWait()

engine.stop()

root.mainloop()
```

TESTING

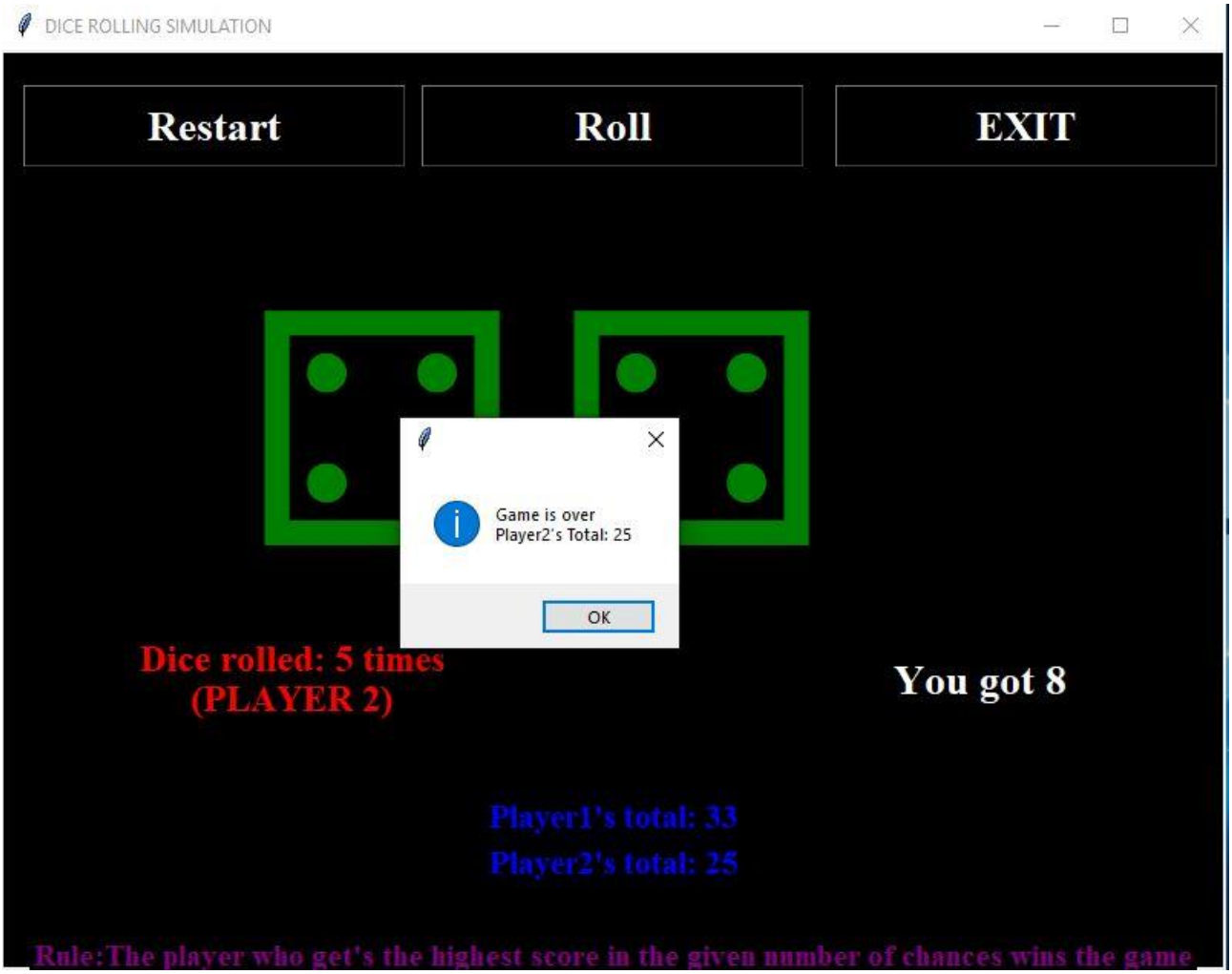
The testing case is usually a single step, or occasionally a sequence of steps, to test the correct behavior, functionality, features of a system. An expected result or expected outcome is usually given.



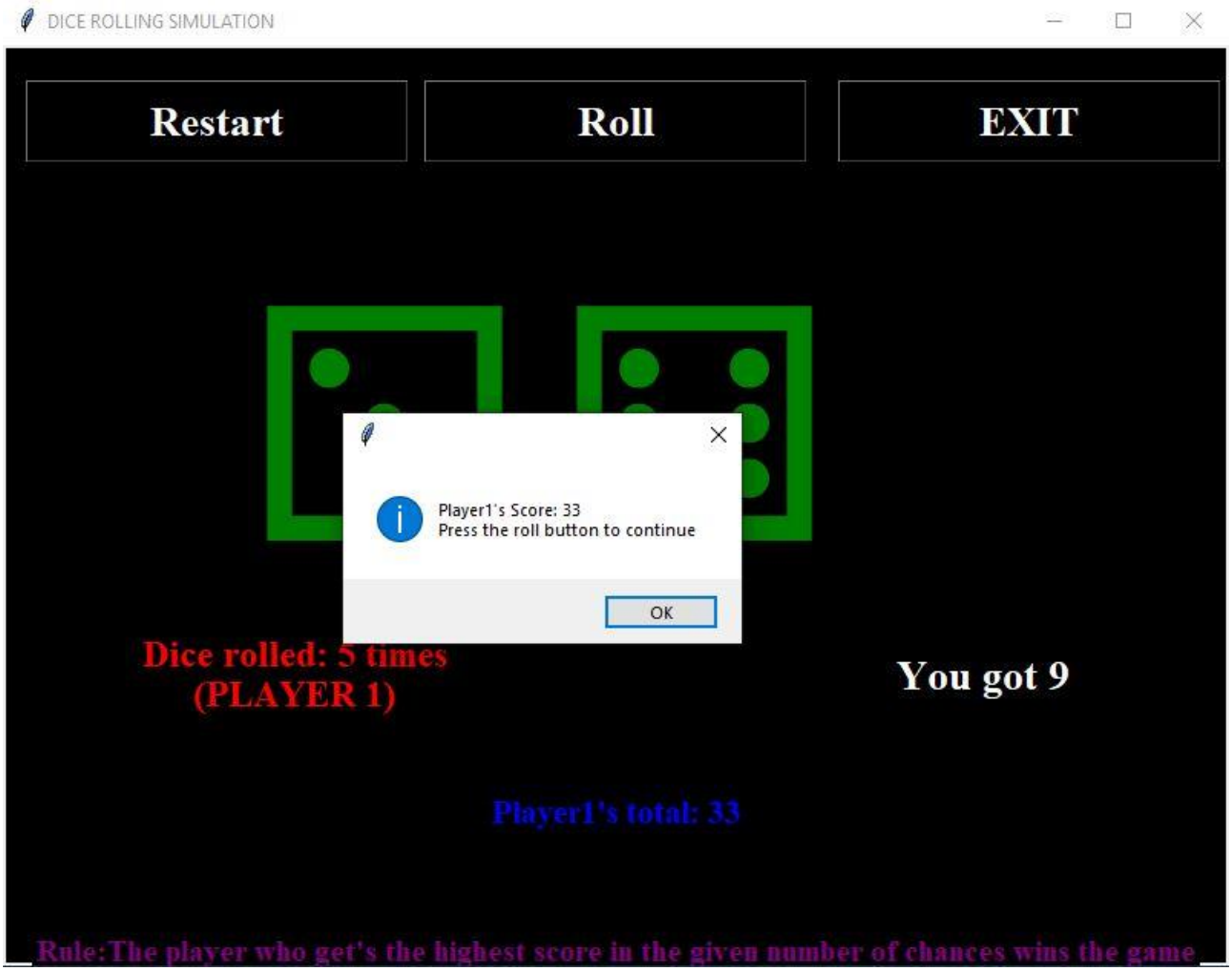
GAME 1: Message box pop up when the player exhaust his chances and loses the game



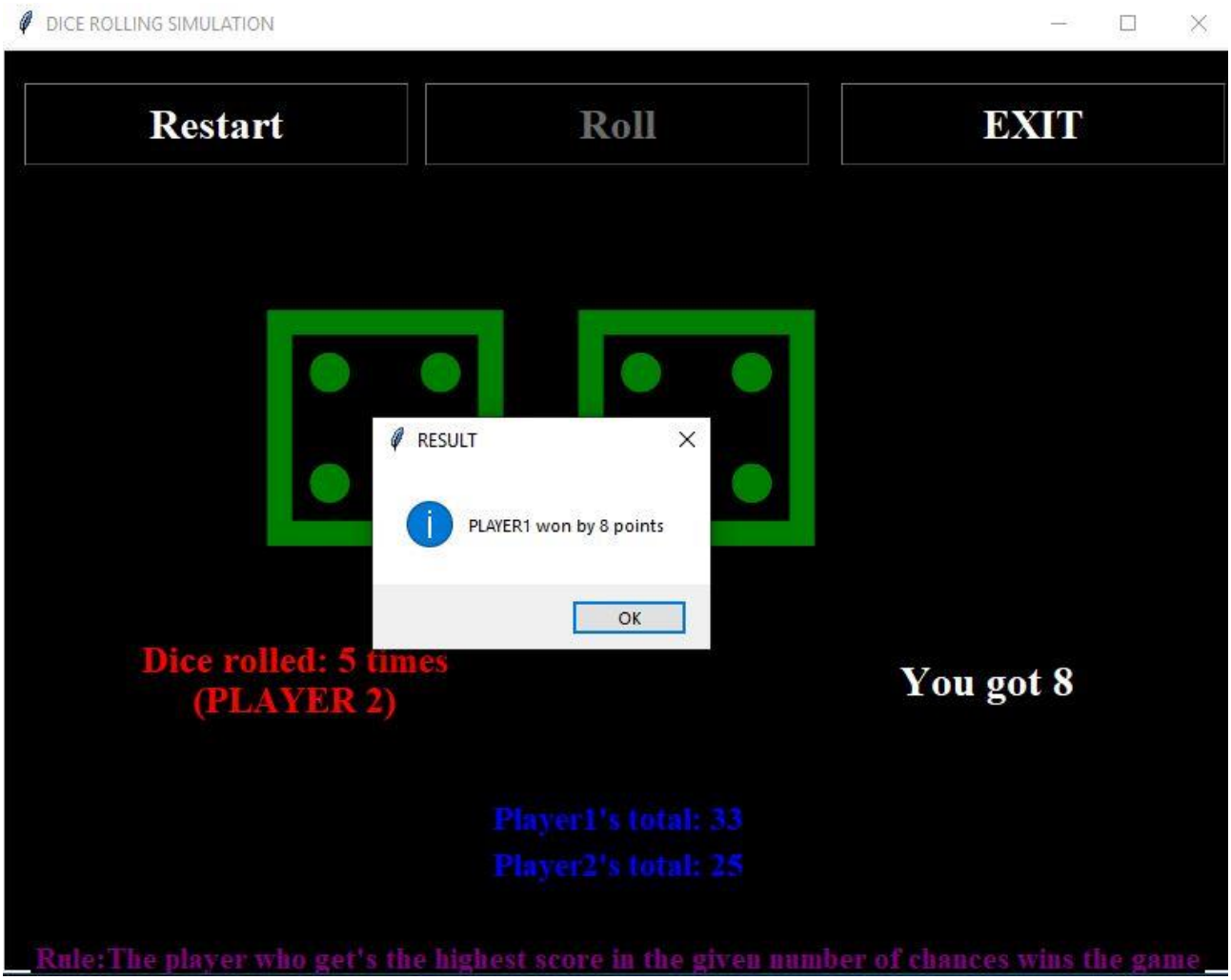
GAME 1: Message box pop up when the player wins the game



GAME 2: Message box pop up when player 2 finish his play



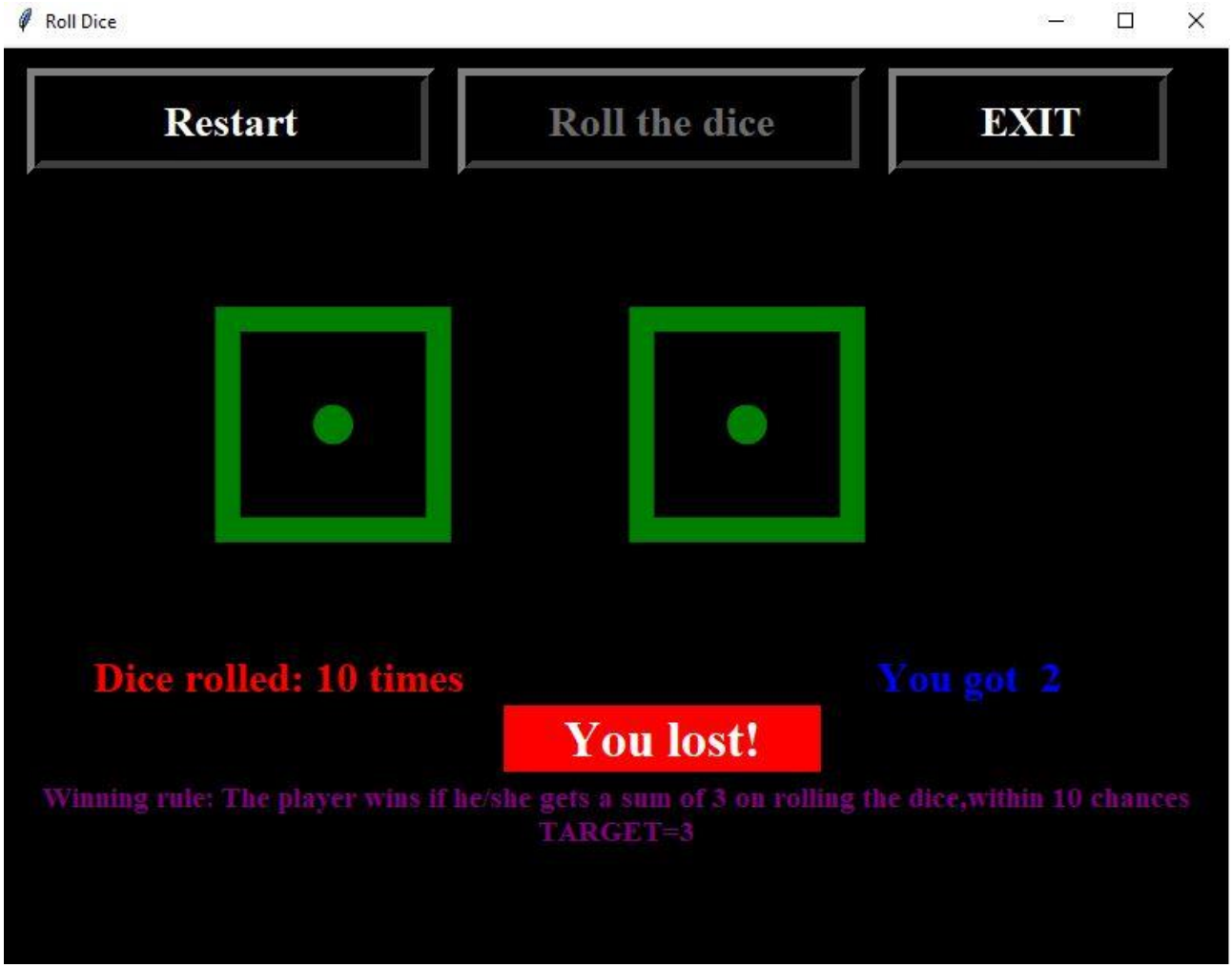
GAME 2:Message box pop up when the player 1 finish his play



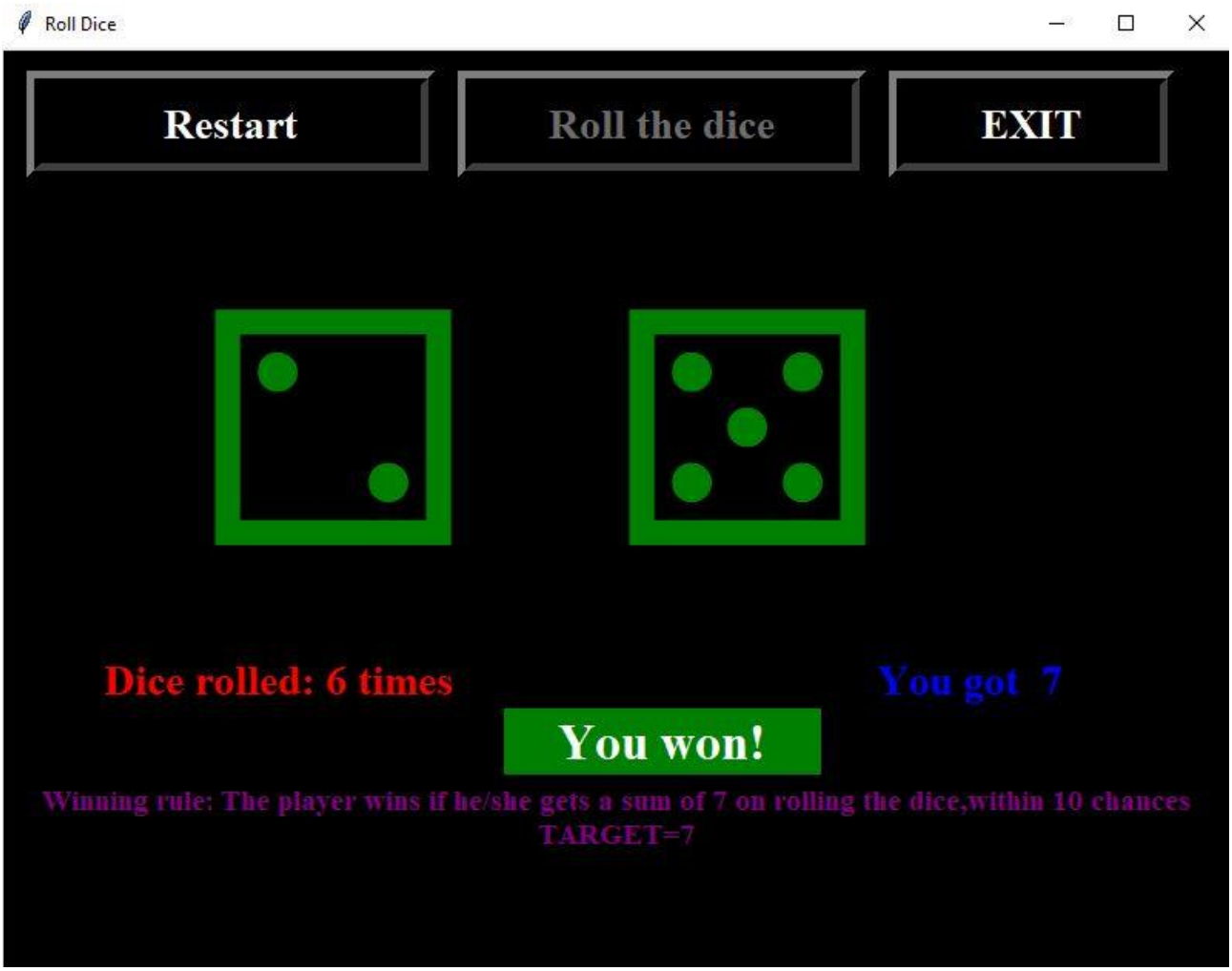
GAME 2: Message box pop up when the game is over

Window opened upon clicking 'exit' button

RESULT AND ANALYSIS

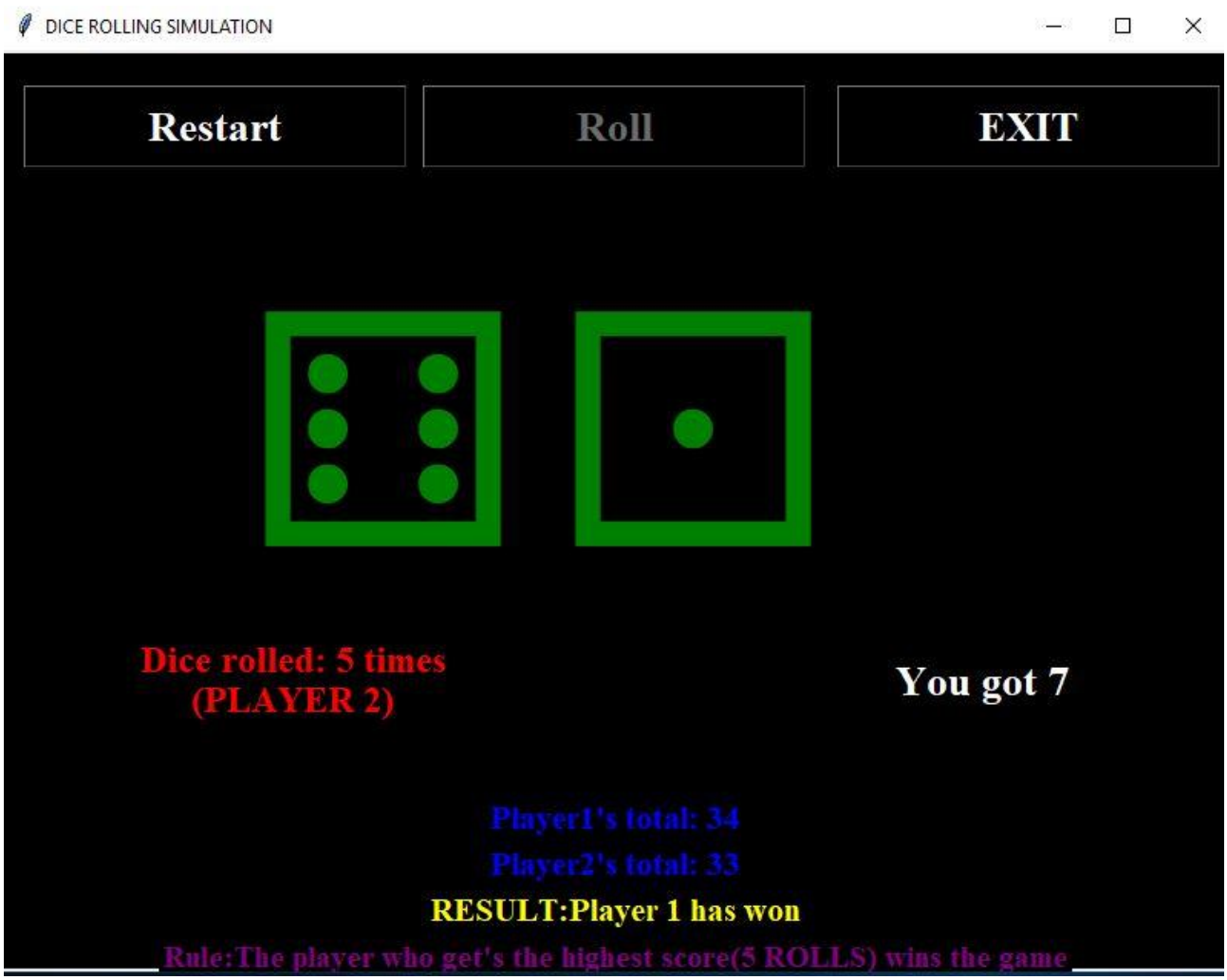


GAME 1: Output when the player loses the game



GAME 1: Output when the player wins the game

GAME 2: Output when player 2 wins the game against player 1



GAME 2: Output when player 1 wins the game against player 2

CONCLUSIONS & FUTURE ENHANCEMENTS

In this project we have developed two games using Dice Rolling Simulator. This project is implemented using Python modules such as Tkinter, random, PIL, playsound and pyttsx3.

Game 1:

In the single-player game, we've chosen a random target number between 2 and 12, which we'll refer to as X. In this game, the person who gets the sum of two dice as X wins. A player is given 10 attempts to attain the desired number; if he or she does not get the target score within those 10 chances, the player forfeits the game.

Game 2:

This is a two-player game with a multi-player mode. Each player is allowed to roll a pair of dice five times, i.e., each player has five chances. The sum of the two dice is calculated and kept as each player's score during their turn. After both players have completed their turns, their cumulative score for all five opportunities is compared. If Player 1 has a higher total score than Player 2, Player 1 is proclaimed the winner; otherwise, Player 2 is declared the winner. If both players have the same score, the result will be a draw, and the result will be announced as such.

FUTURE ENHANCEMENTS

We intend to make this game more user-friendly in the future, with an improved GUI and the ability to accommodate a larger number of players.

REFERENCES

[1]. Charles Dierbach, “Introduction to Computer Science Using Python: A Computational Problem Solving Focus”, Wiley India Edition, John Wiley, 2015.

[2]. Fabrizio Romano, “Learn Python Programming”, Second Edition, Packt Publishing, 2018.

[3]. John V Guttag, “Introduction to Computation and Programming Using Python: With Application to Understanding Data”, MIT Press, 2016.

Web Links:

[1]. https://www.python.org/

[2]. https://www.geeksforgeeks.org/

[3]. https://pypi.org/project/pyttsx3/

[4]. https://pypi.org/project/playsound/

[5]. https://pypi.org/project/Pillow/

[6]. https://www.pythontutorial.net/tkinter/