Joy of Electronics

(CS3001)

(Course Project)

Creation of Digital Board Game

NAMAN SHARMA AU19B1002

Bachelor of Technology

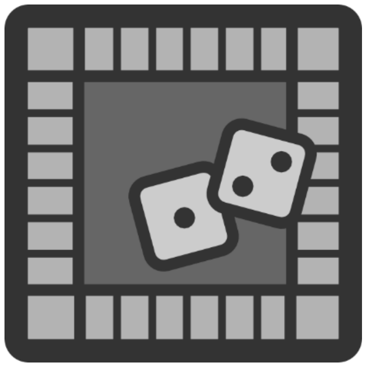
(2020 – 21)

BLACK HOLE (DIGITAL BOARD GAME)

Problem Statement

Creating a digital board game (Computing application)

- The game should be original, computationally accurate, entertaining, and educational. Make it neat (Representative), logical, algorithmic, and creative!



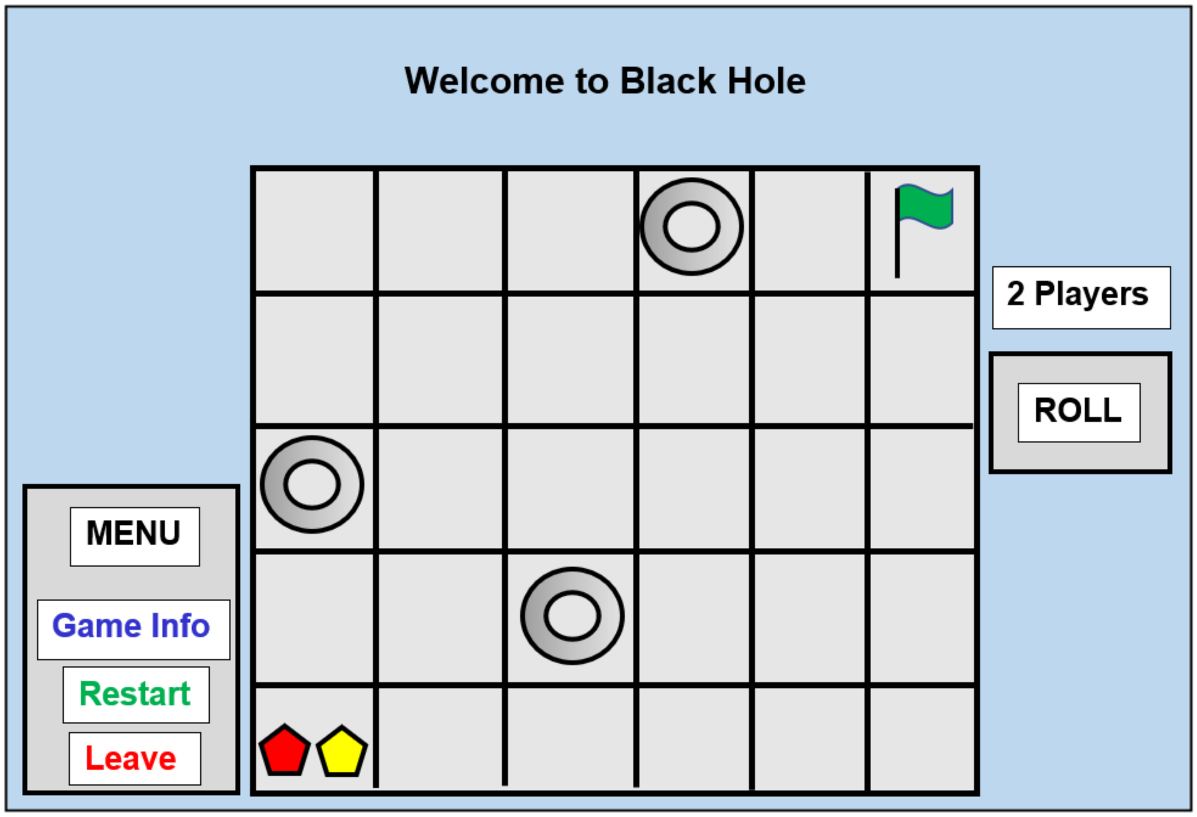
Game Introduction

Black Hole is the simple digital board game. So basically, this game is based on simple tile and can be played on the grid i.e., 6×5. So, we are having 30 tiles to play on. The black holes will be randomly placed on that board. Once a player reaches to that black hole the player has to restart from the start point. Until and unless he reaches the end point this will be repeated. It is the two-player game and the player who reaches first crossing the hurdles reaches to the end point will win.

Abstraction

The goal was to create the digital board game as a computer application which has some basic features like **board, game info, exit, start button,** etc. The combinations of these things make it more fun and interesting. For this problem to be solved we followed it by **Computational thinking** and creating **Interface** with **Interactions** and also **Decomposition** of the problem, **Abstraction**, **Algorithm** and finally the **Pattern Recognition.** Using **Python** as the preferred language and **tkinter** as the GUI package we had programmed the whole game mechanism.

Game Layout

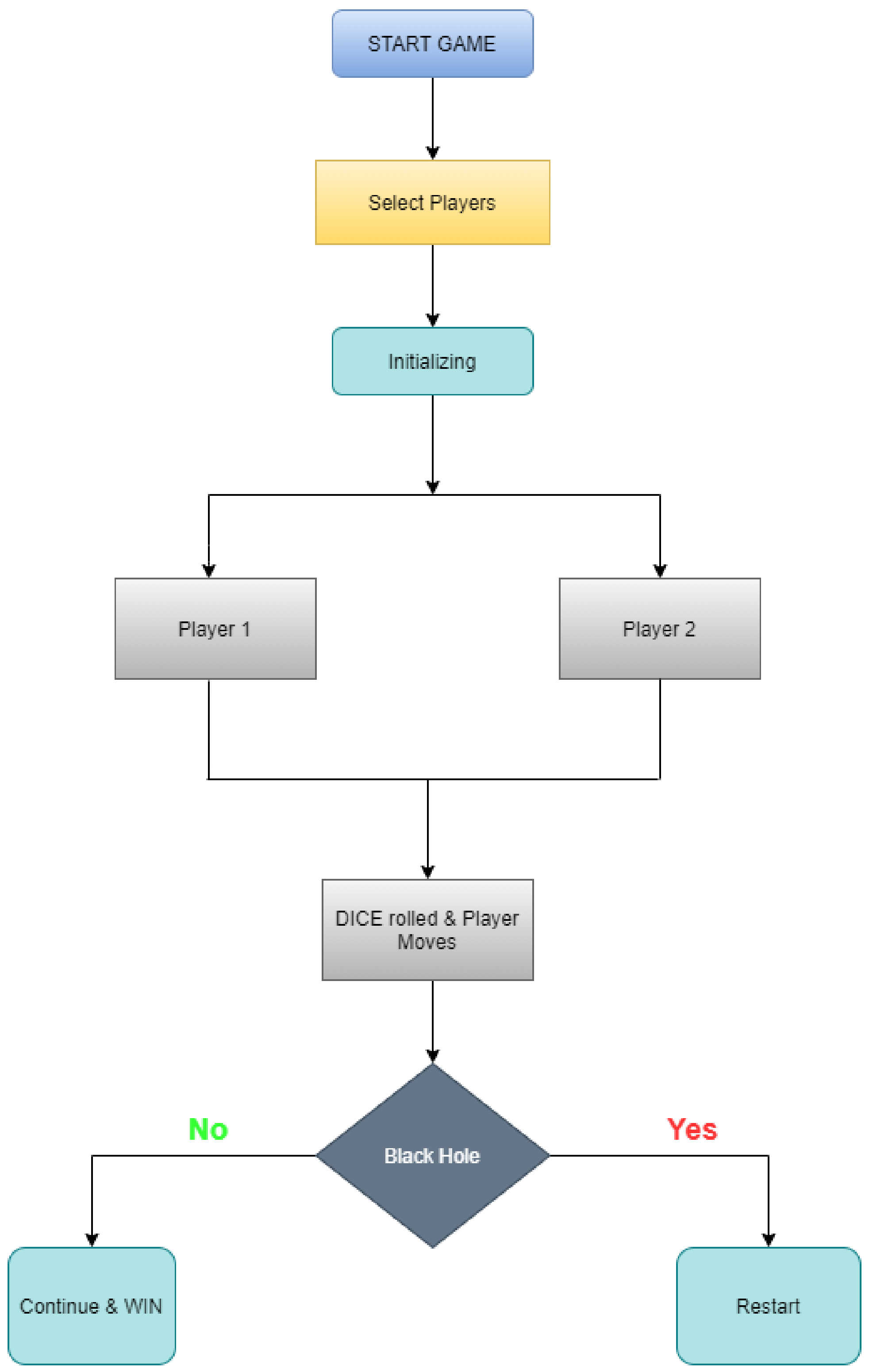


** This is a projected image. Actual preview can be differed.

Game Rules

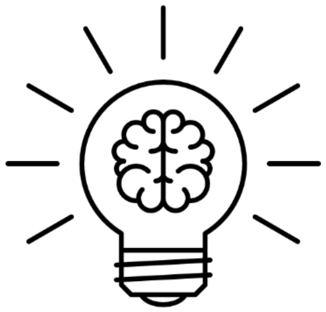
1. This game will be started from the first tile in the grid i.e., start point.
2. There will be a button by which the random will generate between 1 to 4.
3. The value of the number which has been generated, the same value player will be move forward.
4. It will move as per the directions in the diagram above.
5. If the player the player gets on the black hole coordinate, he will lose and the game will be restarted.
6. If the player successfully reached to the end point without getting caught in the black hole, he will win.

Algorithms Flowchart



Logical Aspects

By starting with the code we are starting by importing the packages like tkinter, random, os than after importing package we had defined some class Display, In the class we had defined all the functions like init, leave, restart program, game info, start game, get choice, dice move and gameplay at last with all the variables used in it by declaring it with self. In class we had declared canvas size, image, colors and all the required variables. After that linking all the functions into one code.



Decomposition of Problem

For this problem we had decomposed this problem into several constituent subproblems.

I had distributed this whole game into three major parts i.e., GUI, Interactions, logistics.

As started by the GUI I had subdivided this part into more smaller chunks like, creating blank canvas, adding frames, buttons, and more widget to it.

Likewise, GUI I had similarly done with interactions like creating functions for each and every interaction and implemented to it.

Logistics part is also divided into smaller subsequent problems like writing a logic behind every move, turns, etc.

Pattern Identification

In pattern identification we had created a logical, efficient and well-described steps to solve the problem and achieve the goal easily.

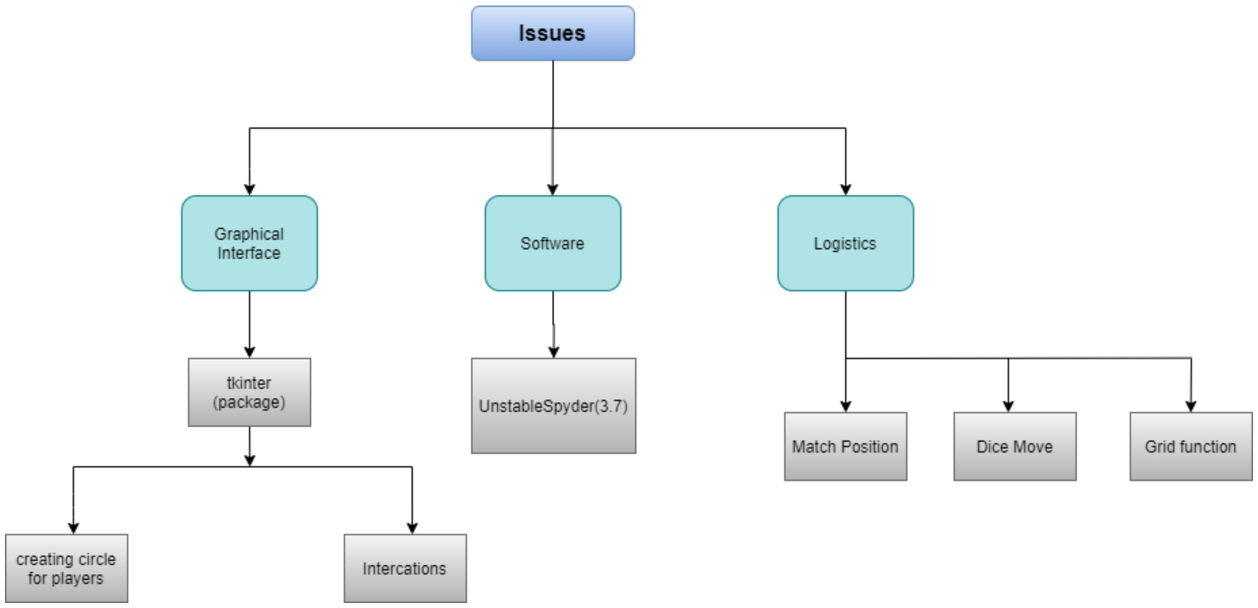
While observing the game it will be founded that if the player reaches to black hole than it will be restarted from the starting point. So, it is likely following certain pattern again and again.

Time Chart

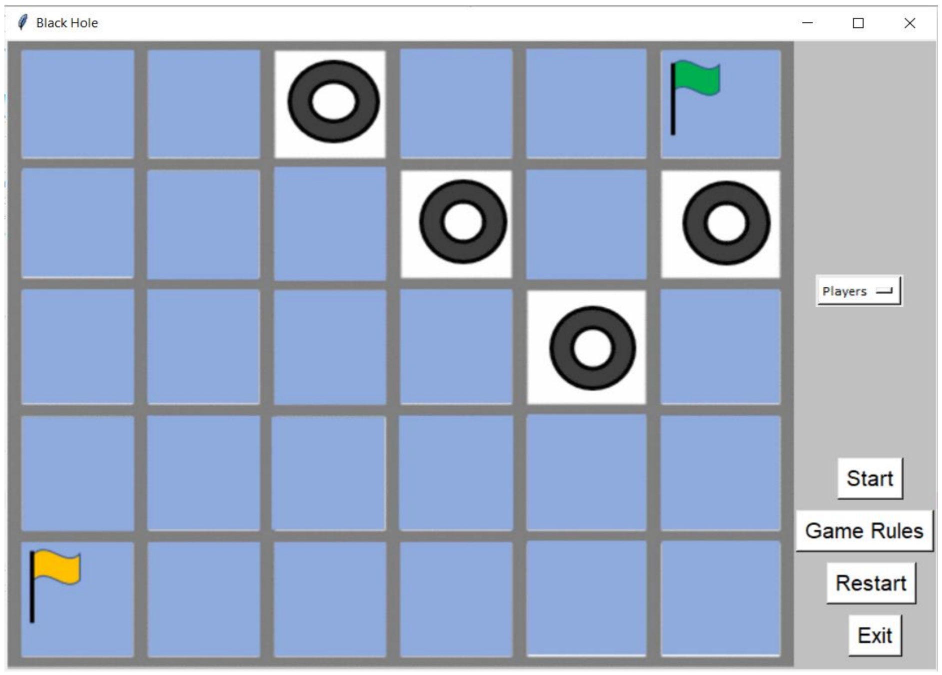
Day	Work
27-08-2020	Understanding the problem statement and started thinking about the game
28-08-2020	Proposed plan to the sir for finalizing the game and created some layouts for it.
31-08-2020	As inputs given by sir started to make basic interface for the game.
01-09-2020	Adding buttons and frames to interface of the game.
02-09-2020	Implementing interactions to all the buttons and finishing of the game.
03-09-2020	Creating reports and presentation according to submission guidelines.

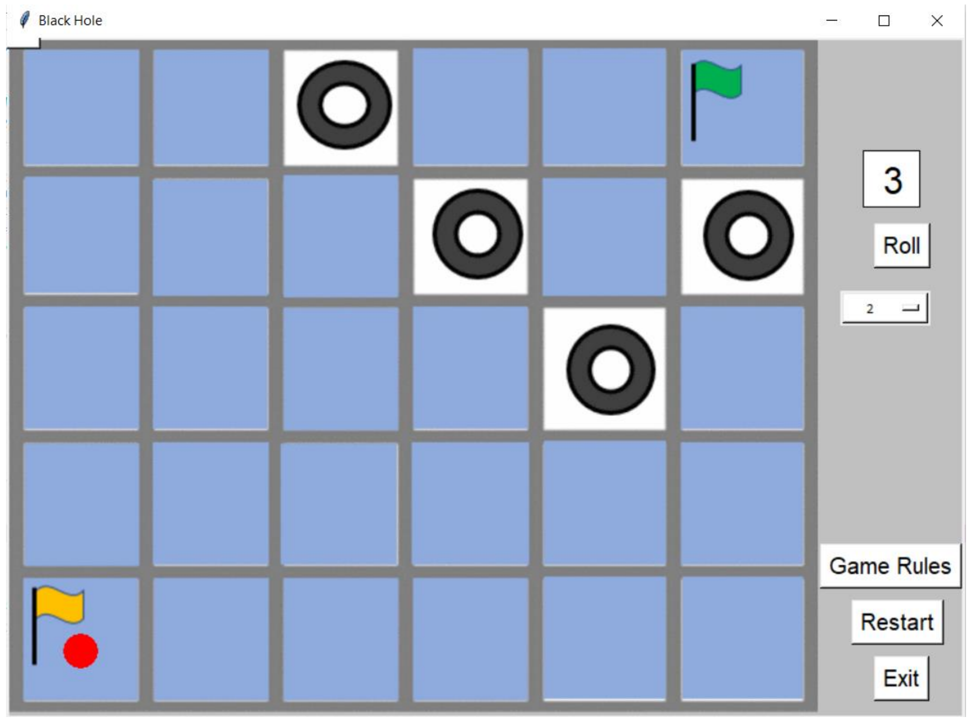
MECE ANALYSIS : -

Issue Tree



Source Code Output Preview





Source Code

```
from tkinter import *
import random
import os


class Display(object):
    def __init__(self,root,img):

        #Create board for Black Hole
        canvas_width = 850
        canvas_height = 600
        self.color = ["#FF0", "#F00"]
        self.canvas = Canvas(root, width = canvas_width, height =
canvas_height, bg = "silver")
        self.canvas.grid(padx=0, pady=0)
        self.canvas.create_image(360,300,anchor=CENTER, image = img)

        self.x = 65
        self.y = 510
        self.m = []
        self.num_player = "Players"
        self.player = []
        self.position = []
        self.i = 0
        self.block=[]
        self.move = 1
        self.turn = 0


        #Drop Menu
        OPTIONS = ["Players", "2", ]
        variable = StringVar(root)
        variable.set(OPTIONS[0]) # default value
        w = OptionMenu(self.canvas, variable, *OPTIONS,
command=self.get_choice)
        w.pack()
        w.place(x=740, y=225)
        w.config(font=('calibri',(10)),bg='white',width=5)

        #Defining the buttons
        self.startGame = Button(self.canvas, text="Start",
background='white', command = self.startGame, font=("Helvetica"))
        self.startGame.place(x=760, y=400)

        self.game_info = Button(self.canvas, text = "Game
Rules",background='white', command = self.game_info, font=("Helvetica"))
        self.game_info.place(x=722, y=450)
        self.restart_program = Button(self.canvas, text = 
"Restart",background='white', command = self.restart_program, 
font=("Helvetica"))
        self.restart_program.place(x=750, y=500)

        self.leave = Button(self.canvas, text = "Exit",background='white', 
command = self.restart_program, font=("Helvetica"))
        self.leave.place(x=770, y=550)



    def leave():
        ## for exiting the program
        root.destroy()

    def restart_program(self):
        ## for restarting the program
        python = sys.executable
        os.excel(python,python, *sys.argv)
        return restart_program()


    def game_info(self):
        ## for displaying the game rules
        messagebox.showinfo("Game Rules",
                         "1. The game will be started from the first tile in 
the grid i.e., start point and end at the end point.\n\
                        \n2. There will be a button by which the random will 
generate between 1 to 4.\n\
                        \n3. If the player the player gets on the black hole 
coordinate, he will lose and the game will be restarted.\n\
                        \n4. If the player successfully reached to the end 
point without getting caught in the black hole, he will win")

    def startGame(self):
        if(self.num_player == "Players"):
            pass
        else:
            #Dice
            #Screen
            self.canvas.create_rectangle(810, 150, 760, 100, fill='white', 
outline='black')
            self.canvas.pack(fill=BOTH, expand=1)
            #Button
            self.diceRoll = Button(self.canvas, 
text="Roll",background='white',command = self.gamePlay, font=("Helvetica"))
            self.num_player = int(self.num_player)
            self.diceRoll.place(x=770, y=165)
            self.create_peice()
            self.startGame.place(x=-30, y=-30)


    def get_choice(self, value):
```

```python
        self.num_player = value


    def diceMove(self, position, turn):
        move = dice()
        #move = 1
        #Print Dice Value to screen
        dice_value = Label(self.canvas, text=str(move),background='white',
font=("Helvetica", 25))
        dice_value.pack()
        dice_value.place(x=775, y=105)


        self.x, self.y = position[0], position[1]
        if(move+self.block[turn] > 30):
            return [self.x, self.y]


        self.block[turn] += move

        self.canvas.delete(self.player[turn])

        return [self.x, self.y]


    def create_peice(self):
        ## for creating the peice in the game
        for i in range(int(self.num_player)):
            if(i==3):
                self.x += 35
                self.y -= 105
            self.player.append(self.canvas.create_circle(self.x, self.y, 15,
fill=self.color[i], outline=""))
            self.position.append([self.x, self.y])
            self.m.append(1)
            self.block.append(1)
            self.y += 35



    def gamePlay(self):
        if(self.move == 6):
            turn = self.turn
        else:
            turn = self.i%self.num_player
            self.i += 1
            self.turn = turn
        self.position[turn] = self.diceMove(self.position[turn], turn)
        if(self.block[self.turn] >= 30):
            self.diceRoll.place(x=-30, y=-30)
            print("Won", self.turn+1)
            top = Toplevel()
            top.title("Black Hole")
            message = "Player " + str(self.turn+1) + " Won"
            msg = Message(top, text=message)
            top.geometry("%dx%d%+d%+d" % (100, 100, 250, 125))
            msg.pack()
            button = Button(top, text="Exit", command=top.destroy)
            button.pack()


def dice():
    ## for generating the random number dice
    move = random.randrange(1, 7, 1)
    return move


def _create_circle(self, x, y, r, **kwargs):
    ## for creating the circles of the dice
    return self.create_oval(x-r, y-r, x+r, y+r, **kwargs)
Canvas.create_circle = _create_circle



def main():
    root = Tk()
    root.title("Black Hole")
    root.geometry("850x605")
    img = PhotoImage( file = "board.gif")
    x = Display(root,img)
    root.mainloop()

main()
```

Conclusion

At last I can conclude by saying that overall, this project was very interesting, helpful and engaging. Starting from basic window to ending up at designed output. It was too interesting and engaging. While writing the code for this game I had learned too many new things like interaction with gui and all. It will be very helpful for me in future.