



# **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**

## **Title: Mastermind**

**Team member details:**

**PES1UG21CS649 – SURYA KUMARAK KANNAN**

**PES1UG21CS659 - T P SHRIAMBHIKESH**

**PES1UG21CS662 – TANMAY PRAVEEN UDUPA**

## Abstract

This project is a simple representation of a strategy-based game, Mastermind. We have made use of functions, lists and Tkinter-based GUI in python. The GUI consists of two canvas screens that show the entered colours and the hints for the respective guessed colours, buttons for choosing colours, help and quit. The GUI is simplified with the use of buttons because of which the user can guess the colours with ease.

## Table of Contents

<b>Index No.</b>	<b>Content</b>	<b>Page No.</b>
<b>1.</b>	<b>Introduction</b>	<b>3</b>
<b>2.</b>	<b>Design/Implementation</b>	<b>4</b>
<b>3.</b>	<b>Testing</b>	<b>21</b>
<b>4.</b>	<b>Result and Analysis</b>	<b>22</b>
<b>5.</b>	<b>Conclusion and Future Enhancements</b>	<b>24</b>
<b>6.</b>	<b>References</b>	<b>25</b>

## Introduction

Strategy based games allow exercising one's mind. By playing, one can experience a lot of mental stimulation, which is favourable in the case of handling plenty of monotonous work. Players of strategy games are accustomed to making quick decisions and thoroughly allocating their brain resources in different environments.

According to National Centre for Biotechnology Information (NCBI), five teachable moments during the play are:

- 1. Well-controlled experiments allow strong, specific conclusions*
- 2. Over-interpretation of data leads to a false conclusion.*
- 3. The value of negative data*
- 4. Good experimental design saves time in the long run*
- 5. Rather than seeking to confirm your hypothesis, test it as severely as possible. If a hypothesis is invalid, discard it immediately*

The user (code breaker) has to predict the correct colours and the position set randomly by the code maker (the computer) within the allowed number of guesses.

When the user successfully clicks four colours, the code breaker responds with a black colour to indicate that the colour chosen is the right colour and is in the exact position, white colour to show that the colour chosen is the right colour but in the wrong position and nothing to indicate a wrong colour that does not appear in the secret code.

When the user guesses the code, the program tells that the user has won the game and exits the window.

## Design/Implementation

```
from tkinter import *
```

```
from tkinter import messagebox
```

```
import random
```

```
#To create the main window
```

```
root = Tk()
```

```
root.geometry('1000x650')
```

```
root.configure(bg = 'black')
```

```
root.title("Mastermind")
```

```
p1 = PhotoImage(file = 'download.png')
```

```
root.iconphoto(False, p1)
```

```
l = Label(root,text = 'MASTERMIND',bg = 'black',fg='white', font = 14)
```

```
l.place(relx=0.4125)
```

```
colors = ["RED", "YELLOW", "GREEN", "BLUE", "ORANGE", "PURPLE", "PINK", "BRONZE"]
```

```
secret_colors = random.choices(colors, k = 4)
```

```
guess = []
```

```
hint = ["", "", "", ""]
```

```
count = 0
```

#To create the canvas which will contain user's input

```
board = Canvas(root, height = 544, width = 348, bg = "#ffffbf")
```

```
y = 68
```

```
while y <= 476:
```

```
    lineh = board.create_line(0, y, 348, y)
```

```
    y += 68
```

```
x = 87
```

```
while x <= 261:
```

```
    linev = board.create_line(x, 0, x, 544)
```

```
    x += 87
```

```
board.place(relx = 0.1, rely = 0.05)
```

#To create the canvas which will contain the hints

```
hints = Canvas(root, height = 544, width = 348, bg = "#00ffff")
```

```
Y = 68
```

```
while Y <= 476:
```

```
    lineh = hints.create_line(0, Y, 348, Y)
```

```
    Y += 68
```

```
X = 87
```

```
while X <= 261:
```

```
    linev = hints.create_line(X, 0, X, 544)
```

```
    X += 87
```

```
hints.place(relx = 0.5, rely = 0.05)
```

```
y1 = 486
```

```
y2 = 534
```

```
x1 = 10
```

```
x2 = 77
```

```
#This function Check() compares the four colors given by the user to the secretly chosen set of four colors
```

```
def Check():
```

```
    global x1
```

```
    global x2
```

```
    global y1
```

```
    global y2
```

```
    global guess
```

```
    global secret_colors
```

```
    global hint
```

```
    x1 = 10
```

```
x2 = 77
```

```
for i in range(4):
```

```
    if guess[i] == secret_colors[i]:
```

```
        circle = hints.create_oval(x1, y1, x2, y2, fill = "Black")
```

```
        hint[i] = "B"
```

```
        x1 += 87
```

```
        x2 += 87
```

```
    else:
```

```
        b = 0
```

```
        for j in range(4):
```

```
            if (guess[i] == secret_colors[j]) and (i != j):
```

```
                circle = hints.create_oval(x1, y1, x2, y2, fill = "White")
```

```
                hint[i] = "W"
```

```
                x1 += 87
```

```
                x2 += 87
```

```
                b = 1
```

```
                break
```

```
        if not b:
```

```
            hint[i] = "-"
```

```
            x1 += 87
```



```
x2 += 87
```

```
counter = 0
```

```
for i in range(4):
```

```
    if hint[i] == "B":
```

```
        counter += 1
```

```
if counter == 4:
```

```
    ok = messagebox.showinfo("Message", "Congratulations!!! You have guessed the correct color  
code")
```

```
    if ok == 'ok':
```

```
        root.destroy()
```

```
y1 -= 68
```

```
y2 -= 68
```

```
x1 = 10
```

```
x2 = 77
```

```
guess = []
```

```
if count >= 32 and counter != 4:
```

```
    ok = messagebox.showinfo("Message", "You are out of guesses")
```

```
    if ok == 'ok':
```

```
        root2 = Tk()
```

```
        root2.geometry("400x100")
```

```
        root2.iconbitmap("sol.ico")
```

```
root2.title("Solution")

board1 = Canvas(root2, height = 80, width = 320, bg = "#ffffbf")

X = 80

while X <= 240:

    lineh1 = board1.create_line(X, 0, X, 80)

    X += 80

X1 = 10

X2 = 70

for i in range(4):

    if secret_colors[i] == "RED":

        circle1 = board1.create_oval(X1, 10, X2, 70, fill = "Red")

        X1 += 80

        X2 += 80

    if secret_colors[i] == "YELLOW":

        circle1 = board1.create_oval(X1, 10, X2, 70, fill = "Yellow")

        X1 += 80

        X2 += 80

    if secret_colors[i] == "GREEN":

        circle1 = board1.create_oval(X1, 10, X2, 70, fill = "Green")

        X1 += 80
```

X2 += 80

if secret\_colors[i] == "BLUE":

circle1 = board1.create\_oval(X1, 10, X2, 70, fill = "Blue")

X1 += 80

X2 += 80

if secret\_colors[i] == "ORANGE":

circle1 = board1.create\_oval(X1, 10, X2, 70, fill = "Orange")

X1 += 80

X2 += 80

if secret\_colors[i] == "PURPLE":

circle1 = board1.create\_oval(X1, 10, X2, 70, fill = "Purple")

X1 += 80

X2 += 80

if secret\_colors[i] == "PINK":

circle1 = board1.create\_oval(X1, 10, X2, 70, fill = "Pink")

X1 += 80

X2 += 80

if secret\_colors[i] == "BRONZE":

circle1 = board1.create\_oval(X1, 10, X2, 70, fill = "#cd7f32")

X1 += 80

```
X2 += 80
```

```
board1.place(relx = 0.1, rely = 0.1)
```

```
root2.resizable(False, False)
```

```
root2.mainloop()
```

#This function is called when the "RED" button is clicked.

```
def Red():
```

```
    global x1
```

```
    global x2
```

```
    global y1
```

```
    global y2
```

```
    global count
```

```
    guess.append("RED")
```

```
    circle = board.create_oval(x1, y1, x2, y2, fill = "Red")
```

```
    x1 += 87
```

```
    x2 += 87
```

```
    count += 1
```

```
    if count % 4 == 0:
```

```
        Check()
```

#This function is called when the "YELLOW" button is clicked.

```
def Yellow():  
  
    global x1  
  
    global x2  
  
    global y1  
  
    global y2  
  
    global count  
  
    guess.append("YELLOW")  
  
    circle = board.create_oval(x1, y1, x2, y2, fill = "Yellow")  
  
    x1 += 87  
  
    x2 += 87  
  
    count += 1  
  
    if count % 4 == 0:  
  
        Check()
```

#This function is called when the "GREEN" button is clicked.

```
def Green():  
  
    global x1  
  
    global x2  
  
    global y1
```

```
global y2
```

```
global count
```

```
guess.append("GREEN")
```

```
circle = board.create_oval(x1, y1, x2, y2, fill = "Green")
```

```
x1 += 87
```

```
x2 += 87
```

```
count += 1
```

```
if count % 4 == 0:
```

```
    Check()
```

```
#This function is called when the "BLUE" button is clicked.
```

```
def Blue():
```

```
    global x1
```

```
    global x2
```

```
    global y1
```

```
    global y2
```

```
    global count
```

```
    guess.append("BLUE")
```

```
    circle = board.create_oval(x1, y1, x2, y2, fill = "Blue")
```

```
    x1 += 87
```

```
x2 += 87
```

```
count += 1
```

```
if count % 4 == 0:
```

```
    Check()
```

#This function is called when the "ORANGE" button is clicked.

```
def Orange():
```

```
    global x1
```

```
    global x2
```

```
    global y1
```

```
    global y2
```

```
    global count
```

```
    guess.append("ORANGE")
```

```
    circle = board.create_oval(x1, y1, x2, y2, fill = "Orange")
```

```
    x1 += 87
```

```
    x2 += 87
```

```
    count += 1
```

```
    if count % 4 == 0:
```

```
        Check()
```

#This function is called when the "PURPLE" button is clicked.

```
def Purple():  
  
    global x1  
  
    global x2  
  
    global y1  
  
    global y2  
  
    global count  
  
    guess.append("PURPLE")  
  
    circle = board.create_oval(x1, y1, x2, y2, fill = "Purple")  
  
    x1 += 87  
  
    x2 += 87  
  
    count += 1  
  
    if count % 4 == 0:  
  
        Check()
```

#This function is called when the "BRONZE" button is clicked.

```
def Bronze():  
  
    global x1  
  
    global x2  
  
    global y1
```



```
global y2
```

```
global count
```

```
guess.append("BRONZE")
```

```
circle = board.create_oval(x1, y1, x2, y2, fill = "#cd7f32")
```

```
x1 += 87
```

```
x2 += 87
```

```
count += 1
```

```
if count % 4 == 0:
```

```
    Check()
```

```
#This function is called when the "PINK" button is clicked.
```

```
def Pink():
```

```
    global x1
```

```
    global x2
```

```
    global y1
```

```
    global y2
```

```
    global count
```

```
    guess.append("PINK")
```

```
    circle = board.create_oval(x1, y1, x2, y2, fill = "Pink")
```

```
    x1 += 87
```

```
x2 += 87
```

```
count += 1
```

```
if count % 4 == 0:
```

```
    Check()
```

#This function is called when the "HELP?" button is clicked. It is used to display the rules of the game.

```
def Help():
```

```
    root1 = Tk()
```

```
    root1.configure(bg = 'Black')
```

```
    root1.title("Help")
```

```
    root1.iconbitmap("download1.ico")
```

```
    Objective = "The object of MASTERMIND is to guess a secret code consisting \
```

```
of a series of 4 colored pegs. Each guess results in feedback narrowing \
```

```
down the possibilities of the code.\n\n\
```

```
The computer, known as the Codemaker, secretly chooses any combination of \
```

```
4 colors from the collection of 6 colors. It can also choose 2 or more colors \
```

```
if it wishes.\n\n\
```

```
The user, known as the Codebreaker, attempts to duplicate the exact colors and \
```

```
positions of the secret code.\n\n\
```

```
The Codemaker responds by:\n\n\
```

- (a) A black color to indicate that the color chosen is the right color and is in the right position \ (without indication of which Code Peg it corresponds to).\n\
- (b) A white color to indicate that the color chosen is the right color but in the wrong position.\n\
- (c) Nothing to indicate a wrong color that does not appear in the secret code.\n\n"

```
t = Text(root1, wrap = WORD)
```

```
t.insert(END, Objective)
```

```
t.pack()
```

```
root1.resizable(False, False)
```

```
root1.mainloop()
```

#The below commands are for creating various buttons in the main window

```
jo = 0.025
```

```
guessR = Button(root, text = 'RED', bg = 'Red', font = ("Bold", 13), width = 7, command = Red)
```

```
guessR.place(relx = 0.05-jo , rely = 0.95)
```

```
guessY = Button(root, text = 'YELLOW', bg = 'Yellow', font = ("Bold", 13), width = 7, command = Yellow)
```

```
guessY.place(relx = 0.175-jo, rely = 0.95)
```

```
guessG = Button(root, text = 'GREEN', bg = 'Green', font = ("Bold", 13), width = 7, command = Green)
```

```
guessG.place(relx = 0.300-jo, rely = 0.95)
```

```
guessB = Button(root, text = 'BLUE', bg = 'Blue', font = ("Bold", 13), width = 7,command = Blue)
```

```
guessB.place(relx = 0.425-jo, rely = 0.95)
```

```
guessO = Button(root, text = 'ORANGE', bg = 'Orange', font = ("Bold", 13), width = 7, command = Orange)
```

```
guessO.place(relx = 0.550-jo, rely = 0.95)
```

```
guessP = Button(root, text = 'PURPLE', bg = 'Purple', font = ("Bold", 13), width = 7, command = Purple)
```

```
guessP.place(relx = 0.675-jo, rely = 0.95)
```

```
guessBr = Button(root, text = 'BRONZE', bg = '#cd7f32', font = ("Bold", 13), width = 7, command = Bronze)
```

```
guessBr.place(relx = 0.800-jo, rely = 0.95)
```

```
guessPk = Button(root, text = 'PINK', bg = 'Pink', font = ("Bold", 13), width = 7, command = Pink)
```

```
guessPk.place(relx = 0.925-jo, rely = 0.95)
```

```
help_ = Button(root, text = 'HELP?', bg = 'White', font = ("Bold", 13), width = 7, command = Help)
```

```
help_.place(relx = 0.9, rely = 0.05)
```

```
quit_ = Button(root, text = "QUIT", bg = "White", font = ("Bold", 13), width = 7, command = root.destroy)
```

```
quit_.place(relx = 0.9, rely = 0.15)
```

```
guessR.configure(activebackground='maroon', activeforeground='White')
```

```
guessY.configure(activebackground='#bf9000', activeforeground='White')
```

```
guessG.configure(activebackground='#274e13', activeforeground='White')
```

```
guessB.configure(activebackground='#01006a', activeforeground='White')
```

```
guessO.configure(activebackground='#995500', activeforeground='White')
```

```
guessP.configure(activebackground='#4f0080', activeforeground='White')
```

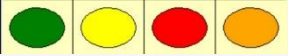




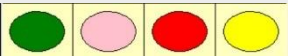
```
guessBr.configure(activebackground='#654321', activeforeground='White')
```

```
guessPk.configure(activebackground='#e75480', activeforeground='White')
```

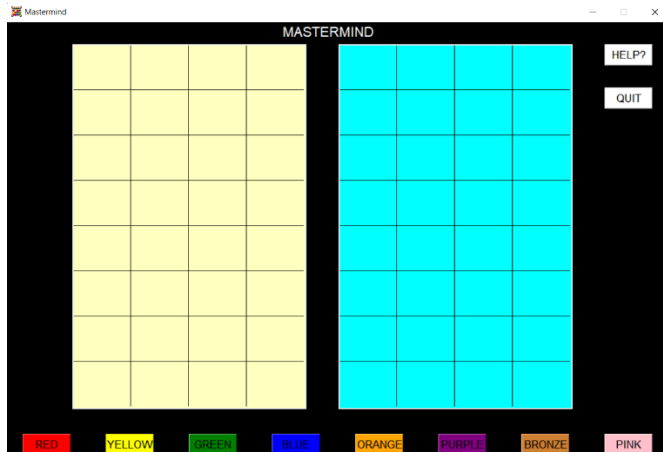
```
root.resizable(False, False)
```

```
root.mainloop()
```

## Testing

Sl. No.	Test Case	Produced Output	Expected Output	Status
1.				FAIL
2.				FAIL
3.				FAIL
4.				FAIL
5.				FAIL
6.				FAIL
7.				PASS

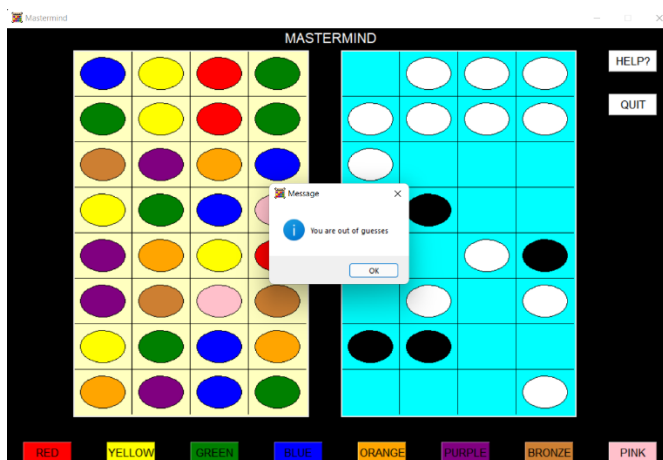
## Result and Analysis



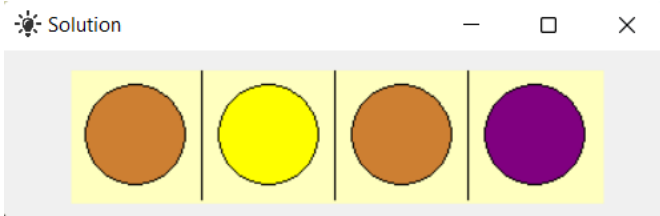
1) The GUI window



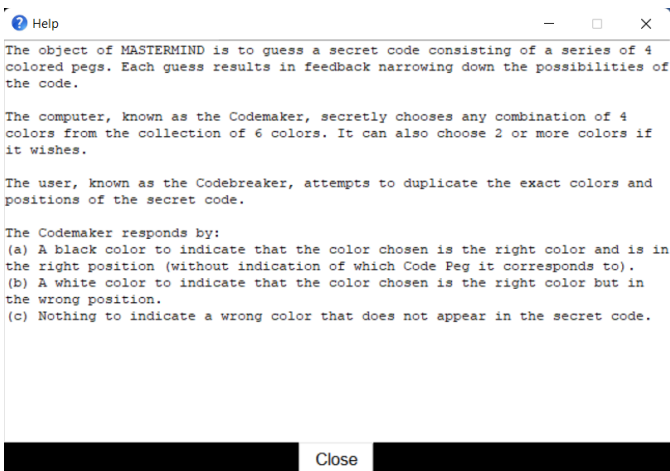
2) When the code has  
been guessed correctly



3) When number of guesses  
has exceeded the limit



4) The solution displayed  
after the user is out of  
guesses



5) The windows appearing on  
clicking help



## Conclusion and Future Enhancements

Mastermind helps you to train your problem-solving skills in a fun way. Use logic to solve the code in as few steps as possible. This mainly concerns the identification and evaluation before you take the next step. You regularly use your problem-solving abilities and logic in daily life.

In future, we could plan to use a timer so that the player can track the time taken by him to guess the colors.

Also, along with Player vs Computer, we can include an option of Player vs Player in which one of the players will be the Code Maker and the other, Code Breaker. After one round, the roles exchange and the player who has guessed the right set of colors in the least number of guesses wins.

## References

- <https://magisterrex.files.wordpress.com/2014/07/mastermindrules.pdf>
- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3022521/>
- <https://www.geeksforgeeks.org/>