**MEMORY PUZZLE GAME**

A

Mini Project Report

Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

NARSING SANDEEP 1602-20-737-103

KODAM JAYANTH KUMAR 1602-20-737-074

GUMMIDELLI SANDEEP 1602-20-737-102



Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Hyderabad-500 031

Department of Information Technology



DECLARATION BY THE CANDIDATE

We, Narsing Sandeep, Kodam Jayanth Kumar, and Gummidelli Sandeep, bearing hall ticket numbers, 1602-20-737-

103, 1602-20-737-074 aand1602-20-737-102, hereby declare that the project report entitled **“MEMORY PUZZLE GAME”** is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

NARSING SANDEEP

1602-20-737-103

KODAM JAYANTH KUMAR

1602-20-737-074

GUMMIDELLI SANDEEP

1602-20-737-102

(Faculty In-Charge) (Head,Dept of IT)

ACKNOWLEDGMENT

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators **DRL PRASANNA** madam and **RADHA** madam, for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.

**ABSTRACT**

This project in python language of Memory Puzzle is a simple console based GUI application using pygame module

In the Memory Puzzle game, several icons are covered up by white boxes. There are two of each icon. The player can click on two boxes to see what icon is behind them. If the icons match, then those boxes remain uncovered. The player wins when all the boxes on the board are uncovered. To give the player a hint, the boxes are quickly uncovered once at the beginning of the game.

This programme is made by using random module in which all the object are at different position than the previous game and changes every time we play the game.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.No | Topic | Page No. |
| 1. | Introduction | 6 |
| 2. | Technology | 8 |
| 3. | Design | 9 |
| 4. | Implementation | 14 |
| 5. | Testing | 25 |
| 6. | Additional Knowledge Acquired | 29 |
| 7. | Conclusion and Future work | 30 |
| 8. | References | 31 |

**INTRODUCTION**

**Overview:**

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. Python is developed by Guido van Rossum. Guido van Rossum started implementing Python in 1989. The syntax in

Python helps the programmers to do coding in fewer steps as compared to Java or C++. The Python is widely used in bigger organizations because of its multiple programming paradigms. They usually involve imperative and objectoriented functional programming. It has a comprehensive and large standard library that has automatic memory management and dynamic features.

Some of the many uses of Python are application development, implementation of automation testing process, allows multiple programming build, fully constructed programming library, can be used in all the major operating systems and platforms, database system accessibility, simple and readable code, easy to apply on complex software development processes, aids in test driven software application development approach, machine learning/ data analytics, helps pattern recognitions, supported in multiple tools, permitted by many of the provisioned frameworks, etc.

In this project we use pygame it is a is a cross-platform set of Python modules which is used to create video games. It consists of computer graphics and sound libraries designed to be used with the Python programming language. Many of the functions of the pygame are used in every game it also has many prebuilt libraries which are used to directly construct the screen unlike the normal package in this module we also have mixer with screen display with it we can able to add music to game and action music where we want music at particular event

Out Game start which the screen shows to select the difficulty level of the game they want to play. There are three different types of levels they can they are level 1, level 2, level 3 hardness increases as they choose the level 1,2,3 respectively. After the selection the game screen pops up and the user has to start the game. At the beginning the boxes will be revealed themselves for few seconds for user to remember the objects and there duplicates/same objects.

Then the game begins all the boxes will be covered at first the user has to click on one box after they click on the box the object behind that box will be revealed and remained open after that we should click on other box which has same object as the first revealed box if both the boxes matches then the both will remain open otherwise both boxes close. We have to continue this process until all boxes open and we should play until end after all the boxes remain open the the game will be finished.

Our game focuses on memorizing the icons present behind the blocks and try to locate the exact duplicate of it behind the other unopened blocks.

**PURPOSE:**

Making games more interactive with python.

Explore different modules and libraries in python

To improve critical thinking skills using python

**OBJECTIVE:**

The objective of this game is to make users have fun while playing the game and help them increase their concentration levels.

## SYSTEM REQUIREMENTS

**Hardware Requirements:**

Minimum Ram required: 512 mb

Minimum Disk Space required: 50 mb

Processor: core i3

Input devices: Mouse

Output devices: Monitor

**Software Requirements:**

IDLE Python (3.8 32 bit)

Python Launcher

Windows 8.1 or above **Modules used:**

pygame module

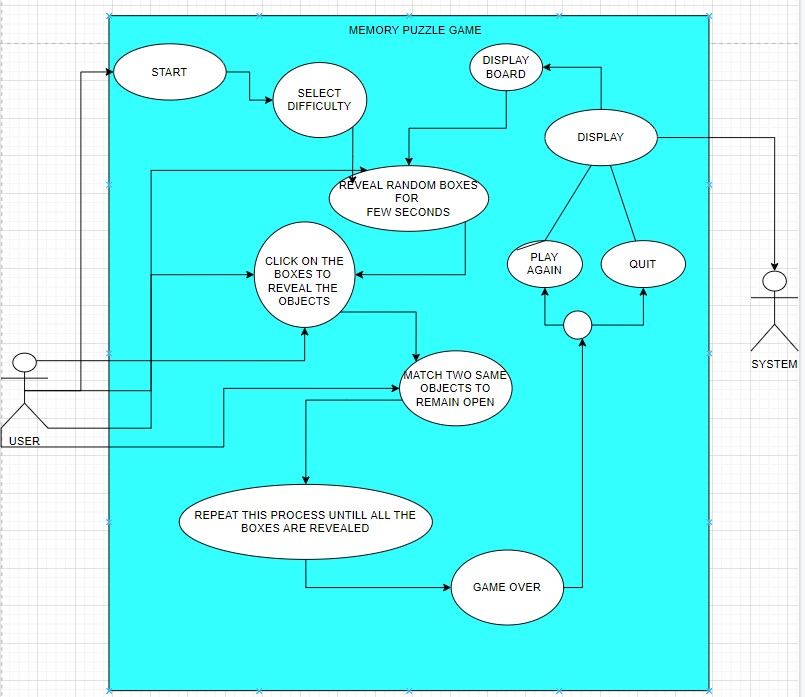
Random module

Time

**PROPOSED WORK**

**DESIGN**

USE CASE DIAGRAM -



**USE CASES :**

* Difficulty
* Start game
* Gaming Process
* End game Animation
* End menu

**USE CASE DESCRIPTIONS**

Use Case ID: UC01

Name: Difficulty

Actors: User

Description: Allows the user to select the difficulty

Pre-conditions: None

Post-conditions: Game with selected difficulty beginsMain Flow:

|  |  |
| --- | --- |
| **user** | **System** |
|  | **System displays the greetings and difficulty levels of the memory puzzle game** |
| **User should selet the game difficulty by entering number 1 or 2 or 3** |  |
| **If user enters difficulty level 1** | **The System displays the Easy mode game where user can complete quickly** |
| **If user enters difficulty level 2** | **The System displays the Normal level game where the game become hard than before** |
| **If user enters difficulty level 3** | **The system displays the Hard level game where**  **the user/player should have high concentration and memory power** |

User case ID: UC02

Name: Start Game

Actors: User

Description: Allow user to start the game

Pre-condition: User should be able to see the screen.

Post-condition: Start the game Main flow:

|  |  |
| --- | --- |
| User | System |
| Start the game by choosing difficulty |  |
|  | Display the game interface and reveal for few seconds and cover all the boxes. |

User case ID: UC03

Name: Gaming Process

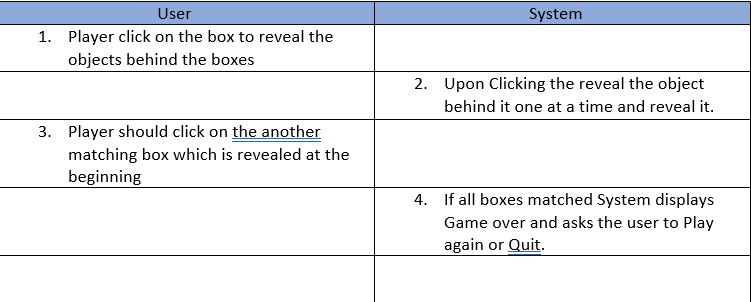
Actors: User

Description: User should be able to play the game and Complete the game

Pre-condition: The Game will be ready and playable

Post-condition: At the end of the game System displays game end animation options.

Main flow:



User case ID: UC04

Name: Gaming end animation

Actors: System

Description: User completes the game by reveal all the boxes

Pre-condition: The Game will be finished all the boxes are revelead

Post-condition: User goes to exit menu Main flow:

|  |  |
| --- | --- |
| User | System |
| The user Reveals all the boxes by gaming process | Displays game won animation by blinking lights |
|  | A game won text appers on the screen |

User case ID: UC05

Name: End Menu

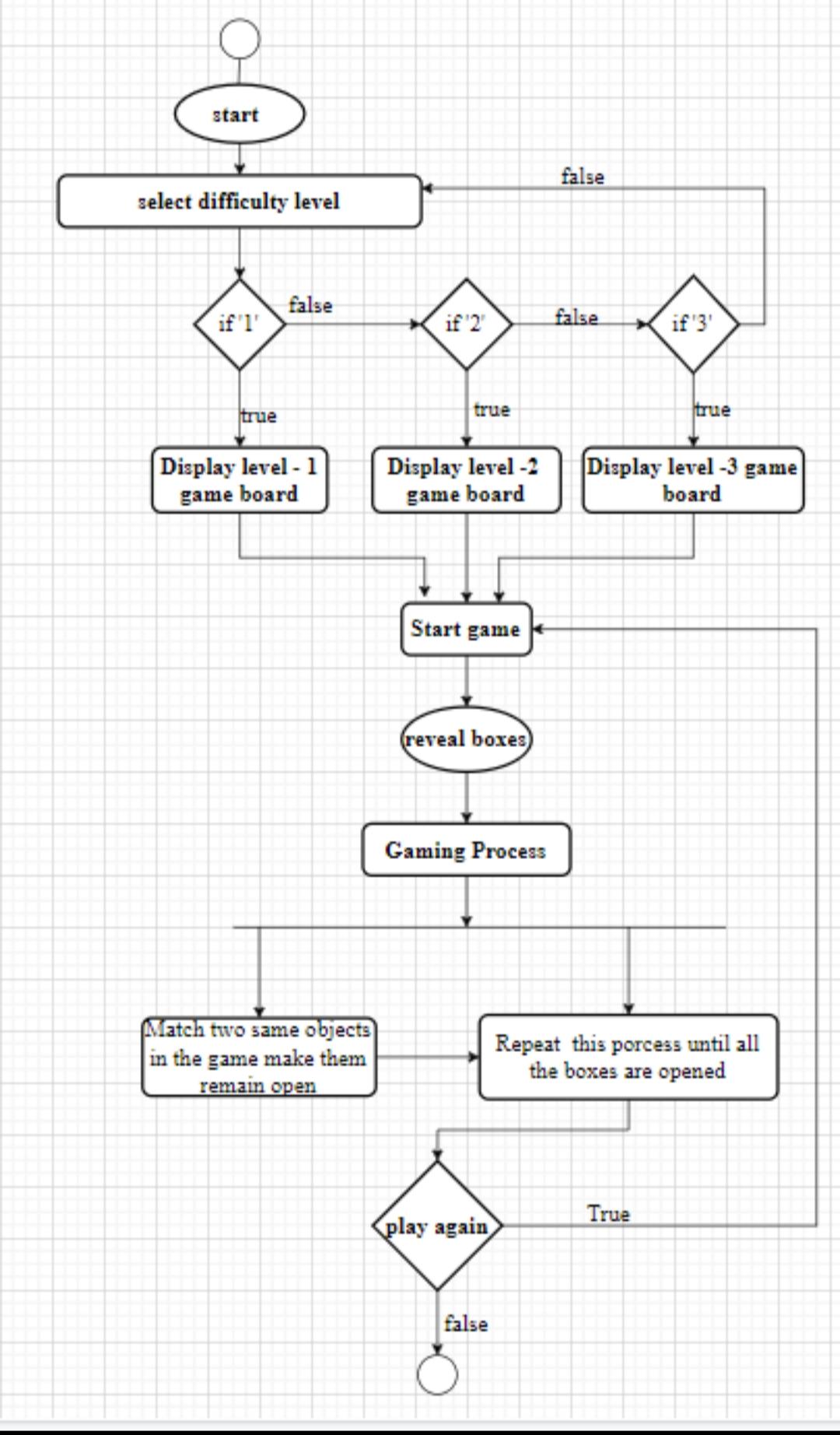
Actors: User

Description: User goes to End Menu option after game over

Pre-condition: Game won animation played Post-condition: User Plays the game again or Quit Main flow:

|  |  |
| --- | --- |
| User | System |
|  | System displays two option either to quit or play agian |
| If user clicks enter button | System displays the start game option |
| If user clicks exit button | Game closes |

**ACTIVITY DIAGRAM**

****

**Concepts Used:**

* **Functions**

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

* **Lists**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square 8 brackets. Important thing about a list is that items in a list need not be of the same type.

* **To build our project we used pygame module** **Modules Used:**
* Pygame is a cross-platform set of Python modules which is used to create video games.It consists of computer graphics and sound libraries designed to be used with the Python programming language.
* Random module The random module in Python is a built -in module to generate the pseudo-random variables. It can be used to perform some actions randomly such as to get a random number, selecting a random elements from a list, shuffle elements randomly , etc

**IMPLEMENTATION**

Source code: import random, pygame, sys,time from pygame.locals import \* from pygame import mixer

print(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ") print("\* WELCOME TO MEMORY PUZZLE GAME \* ")

print("\* THIS GAME IS DESIGNED TO INCREASE MEMORY AND CONCENTRATION POWER \* ") print("\* THERE ARE THREE DIFFICULTY LEVEL TYPES IN THIS \* ") print("\* Level-1 = EASY \* ") print("\* Level-1 = NORMAL \* ") print("\* Level-1 = HARD \* ")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ") print(" INSTRUCTIONS ")

print("1. In the Memory Puzzle game, several icons are covered up by white boxes.") print("2. There are two of each icon. The player can click on two boxes to see what icon is behind them.")

print("3. If the icons match, then those boxes remain uncovered.") print("4. The player wins when all the boxes on the board are uncovered.") print("5. To give the player a hint, the boxes are quickly uncovered once at the beginning of the game.")

level = int(input("\*Enter the diificulty level you want to play:"))

# R G B

GRAY = (100, 100, 100)

NAVYBLUE = ( 60, 60, 100)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

GREEN = ( 0, 255, 0)

BLUE = ( 0, 0, 255)

YELLOW = (255, 255, 0)

ORANGE = (255, 128, 0)

PURPLE = (255, 0, 255)

CYAN = ( 0, 255, 255)

BLACK = ( 0, 0, 0)

OLIVE = (128, 128, 0)

GOLD = (255, 215, 0)

FPS = 30 # frames per second, the general speed of the program

WINDOWWIDTH = 640 # size of window's width in pixels

WINDOWHEIGHT = 480 # size of windows' height in pixels

REVEALSPEED = 4 # speed boxes' sliding reveals and covers

BOXSIZE = 40 # size of box height & width in pixels GAPSIZE = 10 # size of gap between boxes in pixels if level == 1:

BOARDWIDTH = 4 # number of columns of icons

BOARDHEIGHT = 3 # number of rows of icons BGCOLOR = GREEN elif level == 2:

BOARDWIDTH = 5 # number of columns of icons

BOARDHEIGHT = 4# number of rows of icons

BGCOLOR = OLIVE

elif level == 3:

BOARDWIDTH = 6 # number of columns of icons

BOARDHEIGHT = 6 # number of rows of icons BGCOLOR = GOLD assert (BOARDWIDTH \* BOARDHEIGHT) % 2 == 0, 'Board needs to have an even number of boxes for pairs of matches.'

XMARGIN = int((WINDOWWIDTH - (BOARDWIDTH \* (BOXSIZE + GAPSIZE))) / 2)

YMARGIN = int((WINDOWHEIGHT - (BOARDHEIGHT \* (BOXSIZE + GAPSIZE))) / 2)

LIGHTBGCOLOR = GRAY

BOXCOLOR = WHITE

HIGHLIGHTCOLOR = BLUE

#shapes

DONUT = 'donut'

SQUARE = 'square'

DIAMOND = 'diamond'

OVAL = 'oval'

ALLCOLORS = (RED, NAVYBLUE, BLUE, YELLOW, ORANGE, PURPLE, CYAN) ALLSHAPES = (DONUT, SQUARE, DIAMOND, OVAL)

assert len(ALLCOLORS) \* len(ALLSHAPES) \* 2 >= BOARDWIDTH \* BOARDHEIGHT,

"Board is too big for the number of shapes/colors defined."

def mainMenu():

message\_to\_screen("start the

game",BLACK,50,[WINDOWWIDTH/8,WINDOWHEIGHT/8]) def generateRevealedBoxesData(val):

revealedBoxes = []

for i in range(BOARDWIDTH):

revealedBoxes.append([val] \* BOARDHEIGHT) return revealedBoxes

def getRandomizedBoard():

# Get a list of every possible shape in every possible color.

icons = [] for color in ALLCOLORS: for shape in ALLSHAPES:

icons.append( (shape, color) )

random.shuffle(icons) # randomize the order of the icons list numIconsUsed = int(BOARDWIDTH \* BOARDHEIGHT / 2) # calculate how many icons are needed icons = icons[:numIconsUsed] \* 2 # make two of each random.shuffle(icons)

# Create the board data structure, with randomly placed icons.

board = [] for x in range(BOARDWIDTH):

column = [] for y in range(BOARDHEIGHT):

column.append(icons[0])

del icons[0] # remove the icons as we assign them board.append(column) return board

def splitIntoGroupsOf(groupSize, theList):

# splits a list into a list of lists, where the inner lists have at # most groupSize number of items. result = [] for i in range(0, len(theList), groupSize): result.append(theList[i:i + groupSize]) return result

def leftTopCoordsOfBox(boxx, boxy):

# Convert board coordinates to pixel coordinates left = boxx \* (BOXSIZE + GAPSIZE) + XMARGIN top = boxy \* (BOXSIZE + GAPSIZE) + YMARGIN return (left, top)

def getBoxAtPixel(x, y): for boxx in range(BOARDWIDTH): for boxy in range(BOARDHEIGHT):

left, top = leftTopCoordsOfBox(boxx, boxy) boxRect = pygame.Rect(left, top, BOXSIZE, BOXSIZE) if boxRect.collidepoint(x, y):

return (boxx, boxy) return (None, None)

def drawIcon(shape, color, boxx, boxy):

quarter = int(BOXSIZE \* 0.25) # syntactic sugar half = int(BOXSIZE \* 0.5) # syntactic sugar

left, top = leftTopCoordsOfBox(boxx, boxy) # get pixel coords from board coords

# Draw the shapes

if shape == DONUT:

pygame.draw.circle(DISPLAYSURF, color, (left + half, top + half), half - 5) pygame.draw.circle(DISPLAYSURF, BGCOLOR, (left + half, top + half), quarter - 5) elif shape == SQUARE:

pygame.draw.rect(DISPLAYSURF, color, (left + quarter, top + quarter, BOXSIZE - half, BOXSIZE - half)) elif shape == DIAMOND:

pygame.draw.polygon(DISPLAYSURF, color, ((left + half, top), (left + BOXSIZE - 1, top

+ half), (left + half, top + BOXSIZE - 1), (left, top + half))) elif shape == OVAL:

pygame.draw.ellipse(DISPLAYSURF, color, (left, top + quarter, BOXSIZE, half))

def getShapeAndColor(board, boxx, boxy):

# shape value for x, y spot is stored in board[x][y][0] # color value for x, y spot is stored in board[x][y][1] return board[boxx][boxy][0], board[boxx][boxy][1]

def drawBoxCovers(board, boxes, coverage):

# Draws boxes being covered/revealed. "boxes" is a list # of two-item lists, which have the x & y spot of the box.

for box in boxes:

left, top = leftTopCoordsOfBox(box[0], box[1])

pygame.draw.rect(DISPLAYSURF, BGCOLOR, (left, top, BOXSIZE, BOXSIZE)) shape, color = getShapeAndColor(board, box[0], box[1]) drawIcon(shape, color, box[0], box[1]) if coverage > 0: # only draw the cover if there is an coverage

pygame.draw.rect(DISPLAYSURF, BOXCOLOR, (left, top, coverage, BOXSIZE)) pygame.display.update() FPSCLOCK.tick(FPS)

def message\_to\_screen(msg,color,size,pos): font = pygame.font.SysFont(None, size) screen\_text = font.render(msg, True, color)

DISPLAYSURF.blit(screen\_text, pos)

def revealBoxesAnimation(board, boxesToReveal):

# Do the "box reveal" animation. for coverage in range(BOXSIZE, (-REVEALSPEED) - 1, -REVEALSPEED):

drawBoxCovers(board, boxesToReveal, coverage)

def coverBoxesAnimation(board, boxesToCover):

# Do the "box cover" animation. for coverage in range(0, BOXSIZE + REVEALSPEED, REVEALSPEED):

drawBoxCovers(board, boxesToCover, coverage)

def drawBoard(board, revealed):

# Draws all of the boxes in their covered or revealed state. for boxx in range(BOARDWIDTH): for boxy in range(BOARDHEIGHT):

left, top = leftTopCoordsOfBox(boxx, boxy) if not revealed[boxx][boxy]:

# Draw a covered box.

pygame.draw.rect(DISPLAYSURF, BOXCOLOR, (left, top, BOXSIZE, BOXSIZE)) else:

# Draw the (revealed) icon.

shape, color = getShapeAndColor(board, boxx, boxy) drawIcon(shape, color, boxx, boxy)

def drawHighlightBox(boxx, boxy):

left, top = leftTopCoordsOfBox(boxx, boxy)

pygame.draw.rect(DISPLAYSURF, HIGHLIGHTCOLOR, (left - 5, top - 5, BOXSIZE + 10, BOXSIZE + 10), 4)

def startGameAnimation(board):

# Randomly reveal the boxes 8 at a time.

coveredBoxes = generateRevealedBoxesData(False) boxes = [] for x in range(BOARDWIDTH): for y in range(BOARDHEIGHT): boxes.append( (x, y) ) random.shuffle(boxes)

boxGroups = splitIntoGroupsOf(8, boxes)

drawBoard(board, coveredBoxes) for boxGroup in boxGroups:

revealBoxesAnimation(board, boxGroup) coverBoxesAnimation(board, boxGroup)

def gameWonAnimation(board):

# flash the background color when the player has won coveredBoxes = generateRevealedBoxesData(True) color1 = LIGHTBGCOLOR color2 = BGCOLOR for i in range(13):

color1, color2 = color2, color1 # swap colors

DISPLAYSURF.fill(color1)

#drawBoard(board, coveredBoxes) pygame.display.update() pygame.time.wait(300)

#time.sleep(1)

message\_to\_screen("YOU WON",

BLACK,50,[WINDOWWIDTH/12,WINDOWHEIGHT/12])

message\_to\_screen("->press escape key to exit.", OLIVE,30,[100,200]) message\_to\_screen("->press Enter key to play again.", OLIVE,30,[100,220]) time.sleep(1) pygame.display.update()

def hasWon(revealedBoxes):

# Returns True if all the boxes have been revealed, otherwise False for i in revealedBoxes: if False in i:

return False # return False if any boxes are covered.

return True

def main():

global FPSCLOCK, DISPLAYSURF

# mixer.music.load('indigo-946.ogg')

# mixer.music.play(-1)

pygame.init()

font = pygame.font.SysFont(None, 100)

FPSCLOCK = pygame.time.Clock()

DISPLAYSURF = pygame.display.set\_mode((WINDOWWIDTH, WINDOWHEIGHT)) mousex = 0 # used to store x coordinate of mouse event mousey = 0 # used to store y coordinate of mouse event pygame.display.set\_caption('Memory Game -BY IT-B')

mainBoard = getRandomizedBoard() revealedBoxes = generateRevealedBoxesData(False)

firstSelection = None # stores the (x, y) of the first box clicked.

DISPLAYSURF.fill(BGCOLOR) mainMenu() pygame.display.update() pygame.time.wait(50) time.sleep(2)

startGameAnimation(mainBoard)

while True: # main game loop mouseClicked = False

DISPLAYSURF.fill(BGCOLOR) # drawing the window drawBoard(mainBoard, revealedBoxes)

for event in pygame.event.get(): # event handling loop if event.type == QUIT or (event.type == KEYUP and event.key == K\_ESCAPE): pygame.quit() sys.exit() elif event.type == KEYDOWN:

main()

elif event.type == MOUSEMOTION:

mousex, mousey = event.pos elif event.type == MOUSEBUTTONUP:

mousex, mousey = event.pos mouseClicked = True

boxx, boxy = getBoxAtPixel(mousex, mousey) if boxx != None and boxy != None: # The mouse is currently over a box. if not revealedBoxes[boxx][boxy]: drawHighlightBox(boxx, boxy) if not revealedBoxes[boxx][boxy] and mouseClicked: revealBoxesAnimation(mainBoard, [(boxx, boxy)]) revealedBoxes[boxx][boxy] = True # set the box as "revealed" if firstSelection == None: # the current box was the first box clicked firstSelection = (boxx, boxy)

else: # the current box was the second box clicked # Check if there is a match between the two icons.

icon1shape, icon1color = getShapeAndColor(mainBoard, firstSelection[0], firstSelection[1])

icon2shape, icon2color = getShapeAndColor(mainBoard, boxx, boxy)

if icon1shape != icon2shape or icon1color != icon2color:

# Icons don't match. Re-cover up both selections.

pygame.time.wait(1000) # 1000 milliseconds = 1 sec coverBoxesAnimation(mainBoard, [(firstSelection[0], firstSelection[1]), (boxx, boxy)]) revealedBoxes[firstSelection[0]][firstSelection[1]] = False revealedBoxes[boxx][boxy] = False

elif hasWon(revealedBoxes): # check if all pairs found gameWonAnimation(mainBoard) pygame.time.wait(2000)

firstSelection = None # reset firstSelection variable

FPSCLOCK.tick(FPS)

if \_\_name\_\_ == '\_\_main\_\_':

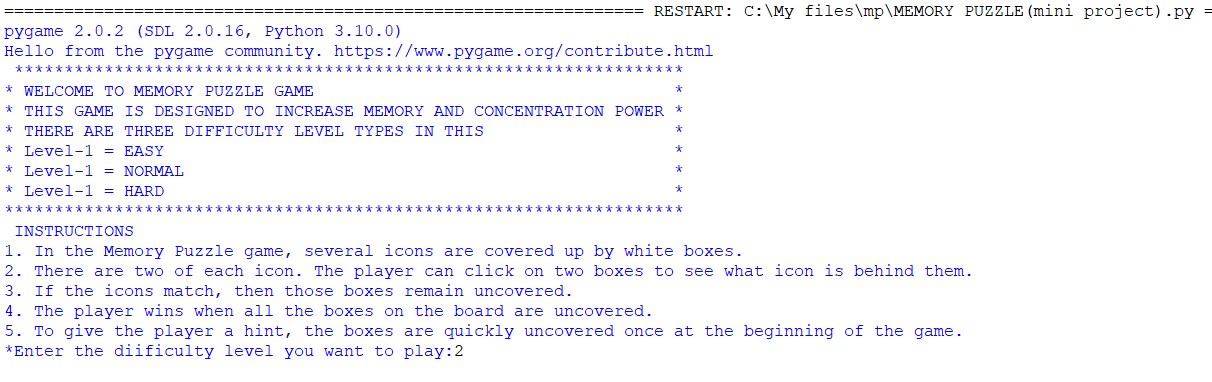
main()

GITHUB LINK:

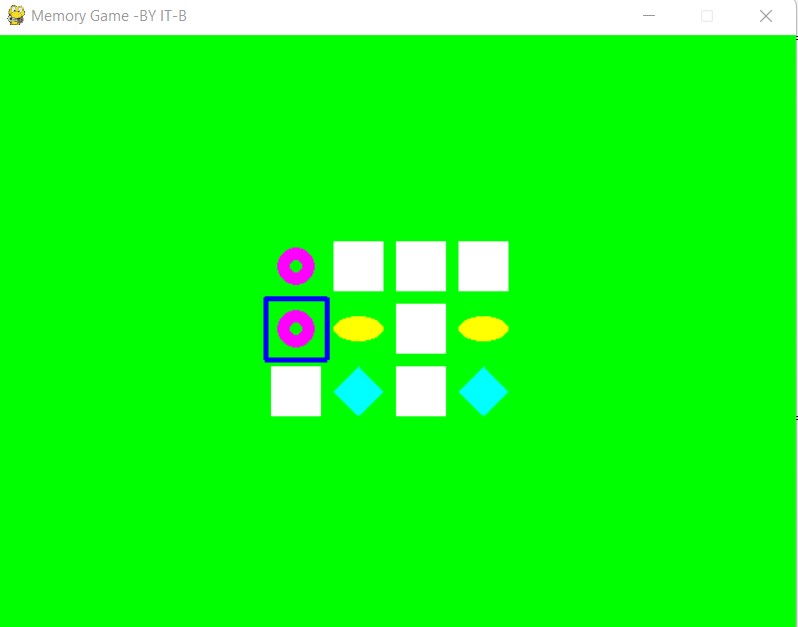
https://github.com/Jayanth02/Memory-puzzle.git

**SCREEN SHOTS OF GAME TEST CASES**

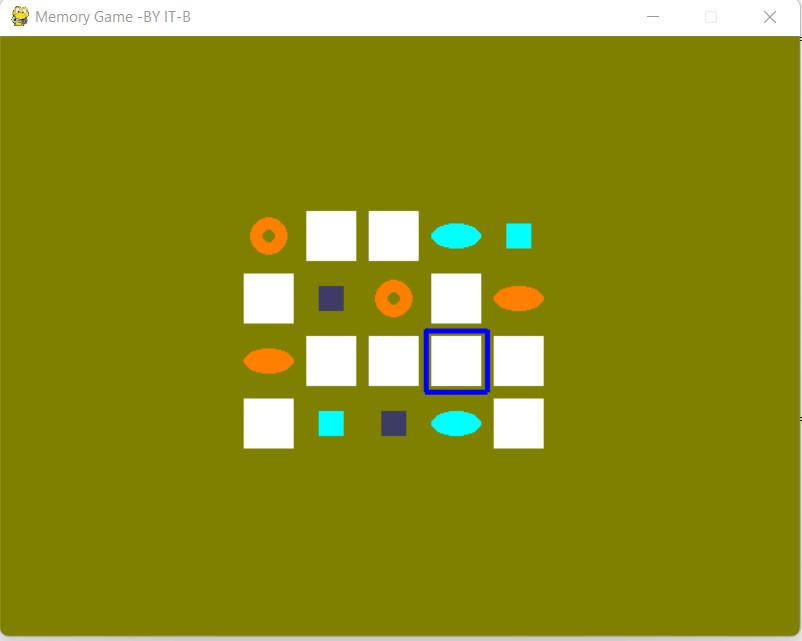
* **Instructions and choosing difficulty**



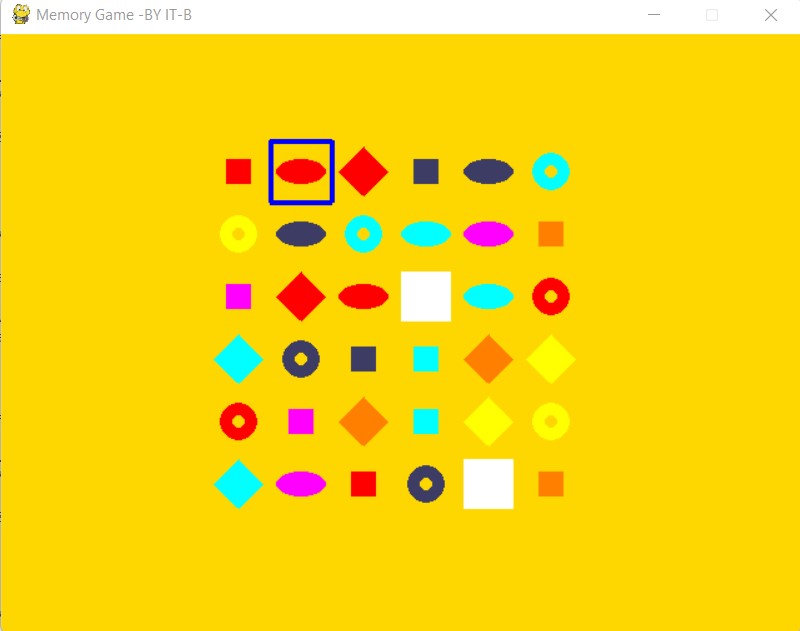
* **Difficulty level –1(Easy)**



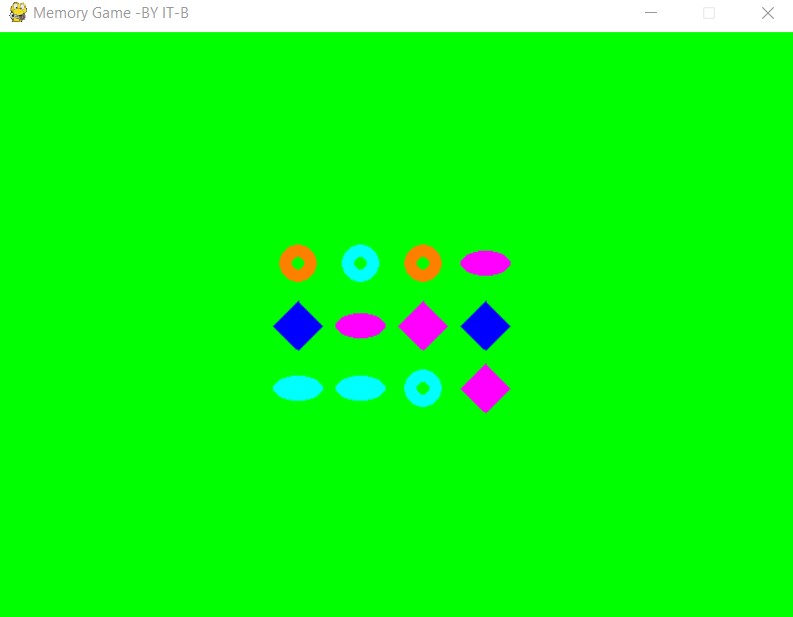
* **Difficulty level – 2(Normal)**



* **Difficulty level –3(Hard) and game before ending**



* **Game won/Game end**



* **End menu**



**ADDITIONAL KNOWLEDGE ACQUIRED**

* Python GUI programming using pygame module
* Adding music to the game and sound track in the particular gaming evnets using pygame mixer
* User case diagram
* Activity diagram

## CONCLUSION

This project in python language of Memory Puzzle is a simple console based GUI application. In this project,several icons are covered up by white boxes. There are two of each icon. The player can click on two boxes to see what icon is behind them. If the icons match, then those boxes remain uncovered. The player wins when all the boxes on the board are uncovered. To give the player a hint, the boxes are quickly uncovered once at the beginning of the game.

This project also gives us an idea of how we can use the random module and the pygame module to create GUI application with less effort and encourage us to create many more interesting projects.

## FUTURE WORK

Our project will implement in future after making some changes and modifications that can be done in our project are:

* We can add more options like time, number of attempts, score based on the time and number of attempts, background music.
* We can also include multi player in this and can calculate both the scores.

## REFERENCES

**Basic Python**

1. Course covered during 2nd semester by Dr. Ramesh Vassapanavara

Sir.

1. Ppts and handouts provided by the sir.
2. Python Programming - Using Problem Solving Approach First Edition by Reema Thareja.

**Pygame tutorials :** [**https://realpython.com/pygame-a-primer/**](https://realpython.com/pygame-a-primer/) [**https://www.youtube.com/watch?v=FfWpgLFMI7w**](https://www.youtube.com/watch?v=FfWpgLFMI7w)

[**Pygame in 90 Minutes - For Beginners**](https://www.youtube.com/watch?v=jO6qQDNa2UY)

Use cases and activity diagram

1. Hand-outs provided by DRL PRASANNA ma’am.
2. <https://www.youtube.com/watch?v=zid-MVo7M-E>
3. <https://www.youtube.com/watch?v=knM8BGY9yVI&t=161s>