

Maze Puzzle Game Based DFS

END TERM REPORT

by

NAME OF THE CANDIDATE(S)-

Name- Ram Mohan Mishra

Roll No.- 52

Registration No. -11804831

Section – K18TM

Name -Mudassir Raza Khan

Roll No.- 30

Registration No. -11804802

Section – K18TM



L OVELY
P ROFESSIONAL
U NIVERSITY

Department of Intelligent Systems

School of Computer Science Engineering

Lovely Professional University, Jalandhar

April 2020

Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Name- Ram Mohan Mishra

Roll No.- 52

Registration No.-11804831

Name- Mudassir Raza Khan

Roll No.- 30

Registration No.-11804802

Place – Lovely Professional university

Date- 12- April- 2020

TABLE OF CONTENTS

TITLE	PAGE NO.
1- Introduction of the Project.	5
2- Motivation Behind the Project.	5
3- Goal of the Project.	6
4- Description of the project.	6
4.1- Algorithm of the project.	6-7
4.2- Code of the project.	7-12
4.3- Screenshots of the project.	13-16
5- Description of Work Division in terms of Roles among Students.	16
6- Technologies and Framework to be used.	16-17
7- SWOT Analysis achieved in project.	17-18

BONAFIDE CERTIFICATE

Certified that this project report “ **Maze Puzzle Game based on DFS** ” is the bonafide work of “ **Ram Mohan Mishra(11804831) & Mudassir Raza Khan(11804802)** ” who carried out the project work under my supervision.

Dr. Dhanpratap Singh
Assistant professor
UID- 25706

School of Computer Science and Engineering.

Introduction

Labyrinth and maze puzzles are a famous structure that interests to all ages over each range of puzzle ability levels. For all intents and purposes everybody has had or played with some rendition of a smoothness puzzle sooner or later - you presumably got one of a Christmas wafers. Labyrinth puzzles have been a staple game structure all through mankind's history because of their simplicity of assembling, perpetual enjoyment and charming mazes.

Characterized by their befuddling many-sided system of winding pathways, aptitude puzzles characterize puzzles which have labyrinth and maze plans. Named because of their accentuation on consistent hands; mastery puzzles are interesting as they depend on persistence and tolerance instead of basic idea and sensible moves which are regular in puzzles. This is a reward too as there is no stunt to the riddle, when the riddle has been explained it can utilize over and over.

Puzzles are popular learning instruments for students' motivation and engagement. Furthermore, puzzle-based learning has many advantages for developing complex transversal skills.

Motivation

Puzzles are famous learning instruments for understudies' inspiration and commitment. Moreover, puzzle-based learning has numerous focal points for creating complex transversal abilities. The math and rationale bewilder in the game are made to assemble the information and learning capacity of an individual. It increments visual special mindfulness and builds up a more profound comprehension of subjects and points. everybody adapts diversely and puzzles might be their vehicle for getting a handle on a comprehension of specific topics. Finishing a puzzle, even the most least difficult of riddles defines a solitary objective to accomplish

The achievement of accomplishing an objective brings such a great amount of fulfilment to a person. Defeating the difficulties associated with explaining a puzzle truly gives them a feeling of accomplishment and pride inside themselves. It gives a lift to their fearlessness and confidence as this sets them up for different difficulties throughout everyday life.

puzzles are an enjoyment instructive toy that challenges youthful personalities, educating and setting them up from the get-go in life some significant fundamental abilities.

Goal

The goal of this project is to encourage the individuals to develop their mind by increasing their problem-solving skills and making them aware with the ongoing challenges that one needs to face in their lives. Once they complete the puzzle there is a sense of achievement which will keep them motivated all through their lives.

Failures will help them understand the value of success and success will motivate them further.

Description of project

- This is a maze generating puzzle.
- There is a maze built in 20x20 grid.
- There are various paths present to reach the goal from the starting point in any puzzle.
- While finding the path to reach the destination whenever you reach a dead-end you need to backtrack to next available memory space in the maze.
- The whole maze is scanned for finding the shortest path.
- This maze generating game will help you find the shortest path from all the available paths using the depth first traversal.

Algorithm

Mazes generated with a depth first search have a low branching factor and contain many long corridors, because the algorithm explores as far as possible along each branch before backtracking.

Recursive backtracker

the depth search algorithm of maze generation is frequently implemented using backtracking

1. make the initial cell as the current cell and mark it as visited.
2. while there is unvisited cells
 1. if the current cell has any neighbour which have not been visited
 - 1- Choose randomly one of the unvisited neighbours
 - 2- Push the current cell to the stack.
 - 3- Remove the wall between the current cell and the chosen cell.
 - 4- Mark the chosen cell the current cell and mark it as visited.
 2. Else if stack is not empty
 - 1- Pop a cell from the stack.
 - 2- Make it the current cell.

Code of the Project-

```
import pygame
```

```
import time
```

```
import random
```

```
WIDTH = 500
```

```
HEIGHT = 600
```

```
FPS = 30
```

```
WHITE = (255, 255, 255)
```

```
GREEN = (0, 255, 0,)
```

```
BLUE = (0, 0, 255)
```

```
YELLOW = (255 ,255 ,0)
```

```
pygame.init()
pygame.mixer.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Python Maze Generator")
clock = pygame.time.Clock()
```

```
x = 0
y = 0
w = 20
grid = []
visited = []
stack = []
solution = { }
```

```
def build_grid(x, y, w):
    for i in range(1,21):
        x = 20
        y = y + 20
        for j in range(1, 21):
            pygame.draw.line(screen, WHITE, [x, y], [x + w, y])
            pygame.draw.line(screen, WHITE, [x + w, y], [x + w, y + w])
            pygame.draw.line(screen, WHITE, [x + w, y + w], [x, y + w])
            pygame.draw.line(screen, WHITE, [x, y + w], [x, y])
            grid.append((x,y))
        x = x + 20
```



```
def push_up(x, y):  
    pygame.draw.rect(screen, BLUE, (x + 1, y - w + 1, 19, 39), 0)  
    pygame.display.update()
```

```
def push_down(x, y):  
    pygame.draw.rect(screen, BLUE, (x + 1, y + 1, 19, 39), 0)  
    pygame.display.update()
```

```
def push_left(x, y):  
    pygame.draw.rect(screen, BLUE, (x - w + 1, y + 1, 39, 19), 0)  
    pygame.display.update()
```

```
def push_right(x, y):  
    pygame.draw.rect(screen, BLUE, (x + 1, y + 1, 39, 19), 0)  
    pygame.display.update()
```

```
def single_cell( x, y):  
    pygame.draw.rect(screen, GREEN, (x + 1, y + 1, 18, 18), 0)  
    pygame.display.update()
```

```
def backtracking_cell(x, y):  
    pygame.draw.rect(screen, BLUE, (x + 1, y + 1, 18, 18), 0)  
    pygame.display.update()
```

```
def solution_cell(x,y):  
    pygame.draw.rect(screen, YELLOW, (x+8, y+8, 5, 5), 0)  
    pygame.display.update()
```

```
def carve_out_maze(x,y):  
    single_cell(x, y)  
    stack.append((x,y))  
    visited.append((x,y))  
    while len(stack) > 0:  
        time.sleep(.07)  
        cell = []  
        if (x + w, y) not in visited and (x + w, y) in grid:  
            cell.append("right")  
  
        if (x - w, y) not in visited and (x - w, y) in grid:  
            cell.append("left")  
  
        if (x , y + w) not in visited and (x , y + w) in grid:  
            cell.append("down")  
  
        if (x, y - w) not in visited and (x , y - w) in grid:  
            cell.append("up")  
  
        if len(cell) > 0:  
            cell_chosen = (random.choice(cell))  
  
            if cell_chosen == "right":  
                push_right(x, y)
```

```

        solution[(x + w, y)] = x, y

        x = x + w

        visited.append((x, y))

        stack.append((x, y))

    elif cell_chosen == "left":

        push_left(x, y)

        solution[(x - w, y)] = x, y

        x = x - w

        visited.append((x, y))

        stack.append((x, y))

    elif cell_chosen == "down":

        push_down(x, y)

        solution[(x , y + w)] = x, y

        y = y + w

        visited.append((x, y))

        stack.append((x, y))

    elif cell_chosen == "up":

        push_up(x, y)

        solution[(x , y - w)] = x, y

        y = y - w

        visited.append((x, y))

        stack.append((x, y))

    else:

        x, y = stack.pop()

        single_cell(x, y)

        time.sleep(.05)

        backtracking_cell(x, y)

```

```
def plot_route_back(x,y):  
    solution_cell(x, y)  
    while (x, y) != (20,20):  
        x, y = solution[x, y]  
        solution_cell(x, y)  
        time.sleep(.1)
```

```
x, y = 20, 20  
build_grid(40, 0, 20)  
carve_out_maze(x,y)  
plot_route_back(400, 400)
```

```
running = True  
while running:
```

```
    clock.tick(FPS)
```

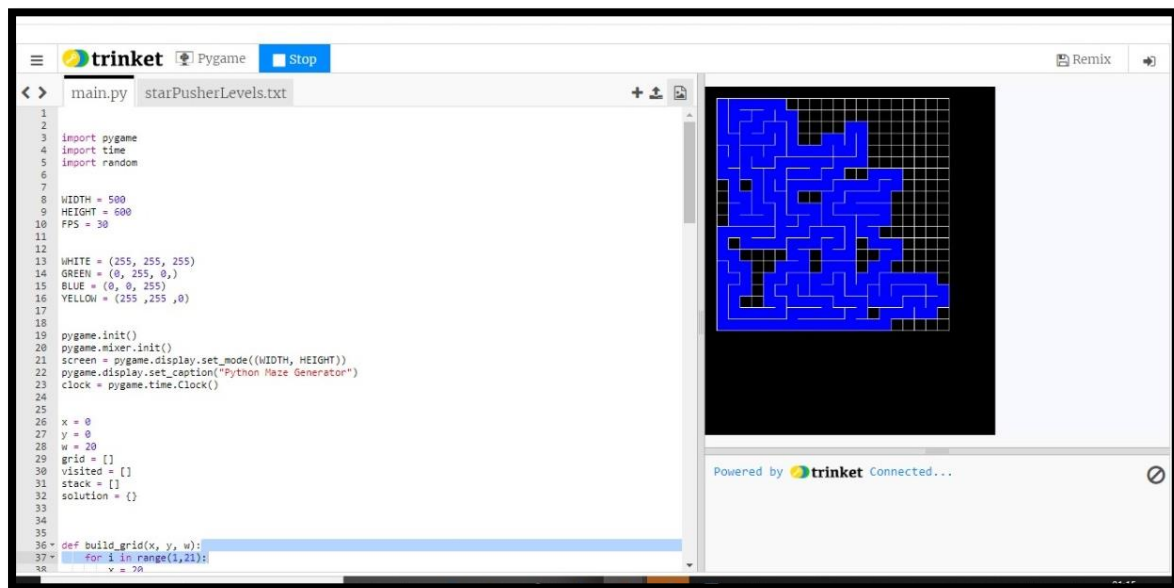
```
    for event in pygame.event.get():
```

```
        if event.type == pygame.QUIT:  
            running = False
```

Screenshot of the Project-

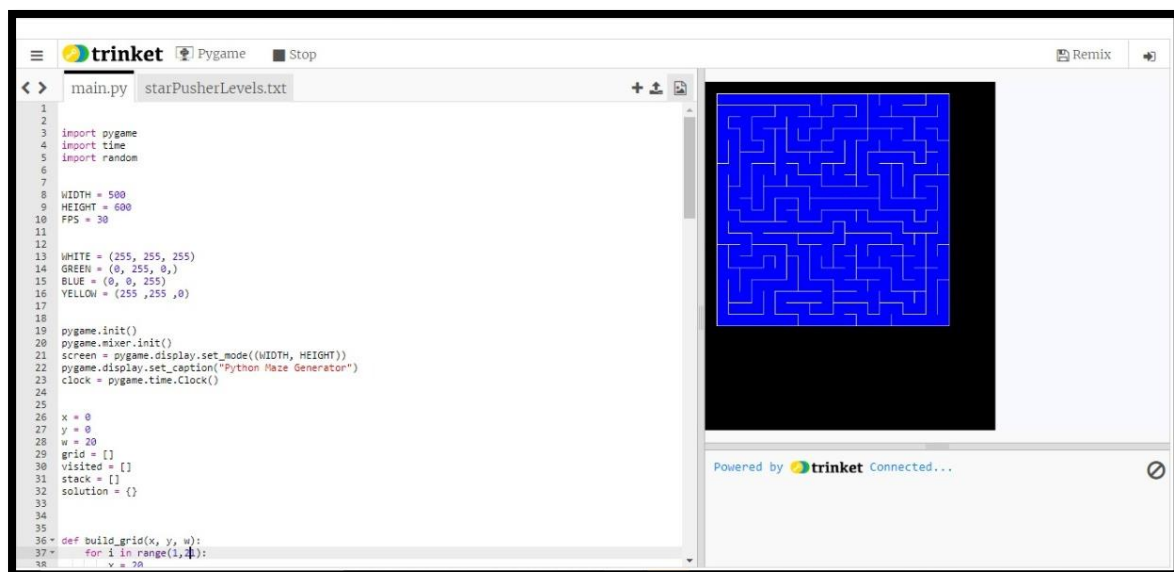
Sample run 1 of the programme-

1- Scanning the whole Maze-



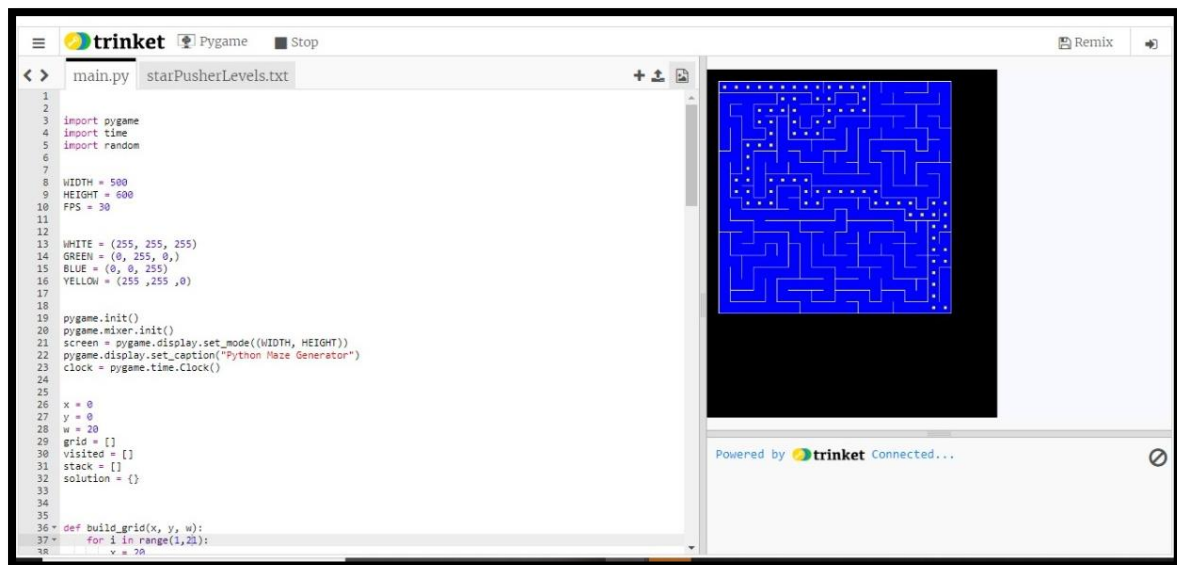
```
1
2
3 import pygame
4 import time
5 import random
6
7
8 WIDTH = 500
9 HEIGHT = 600
10 FPS = 30
11
12
13 WHITE = (255, 255, 255)
14 GREEN = (0, 255, 0)
15 BLUE = (0, 0, 255)
16 YELLOW = (255, 255, 0)
17
18
19 pygame.init()
20 pygame.mixer.init()
21 screen = pygame.display.set_mode((WIDTH, HEIGHT))
22 pygame.display.set_caption("Python Maze Generator")
23 clock = pygame.time.Clock()
24
25
26 x = 0
27 y = 0
28 w = 20
29 grid = []
30 visited = []
31 stack = []
32 solution = {}
33
34
35
36 def build_grid(x, y, w):
37     for i in range(1, 21):
38         x = 20
```

2- The Whole Maze is Scanned-



```
1
2
3 import pygame
4 import time
5 import random
6
7
8 WIDTH = 500
9 HEIGHT = 600
10 FPS = 30
11
12
13 WHITE = (255, 255, 255)
14 GREEN = (0, 255, 0)
15 BLUE = (0, 0, 255)
16 YELLOW = (255, 255, 0)
17
18
19 pygame.init()
20 pygame.mixer.init()
21 screen = pygame.display.set_mode((WIDTH, HEIGHT))
22 pygame.display.set_caption("Python Maze Generator")
23 clock = pygame.time.Clock()
24
25
26 x = 0
27 y = 0
28 w = 20
29 grid = []
30 visited = []
31 stack = []
32 solution = {}
33
34
35
36 def build_grid(x, y, w):
37     for i in range(1, 21):
38         x = 20
```

3- Finding the Shortest Path-

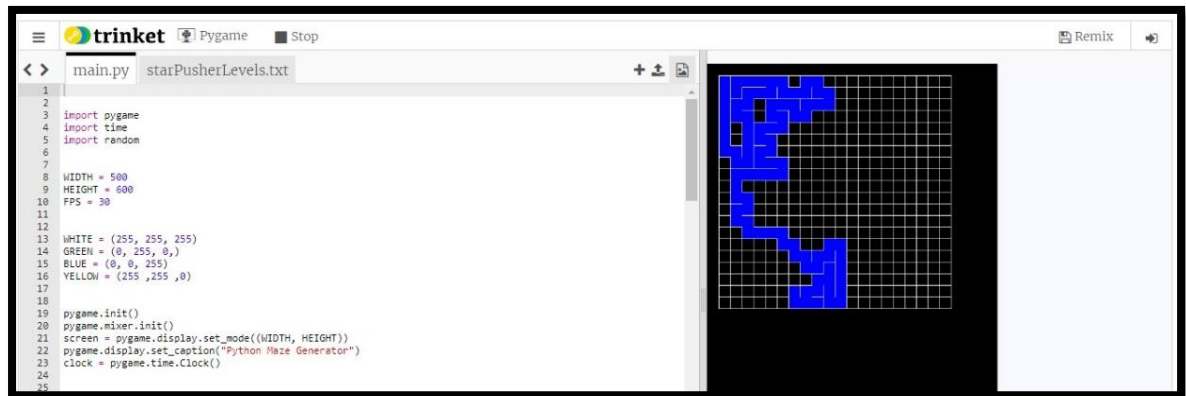


4 Shortest Path find using DFS-

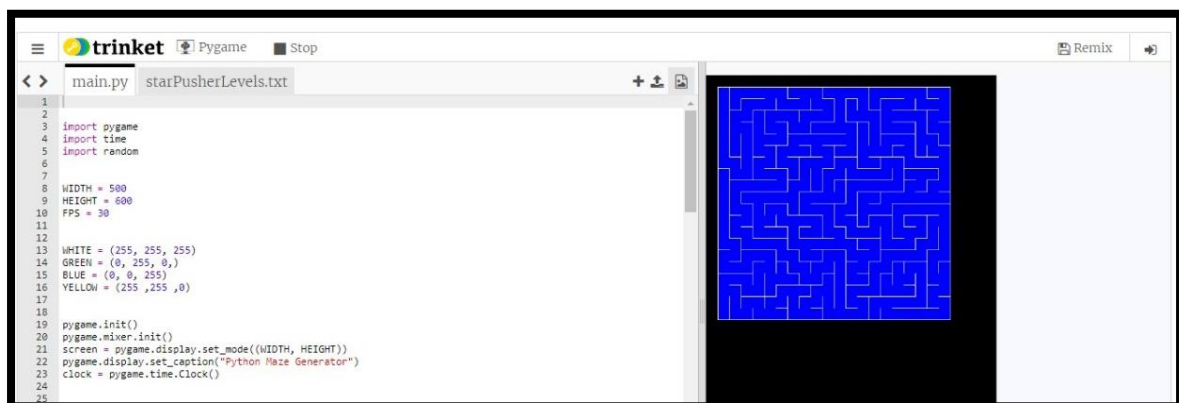


Sample run 2 of the programme -

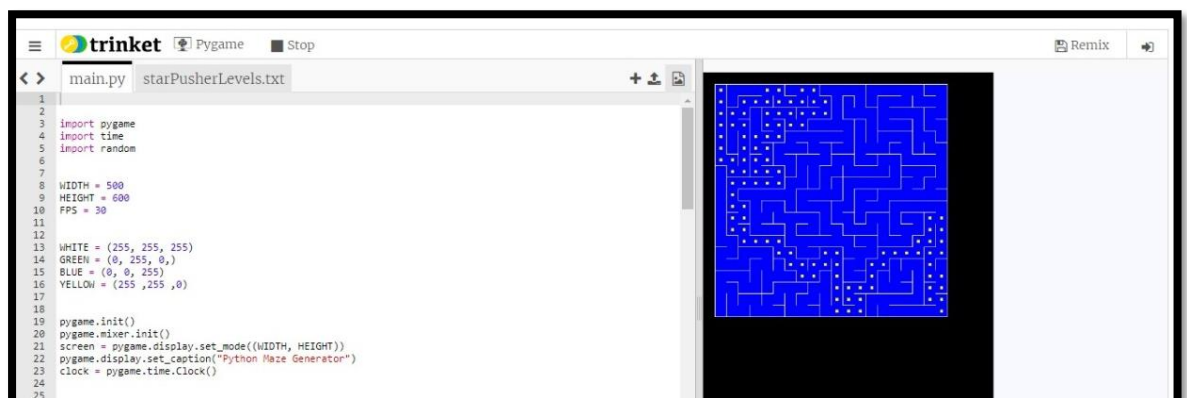
1- Scanning the whole Maze-



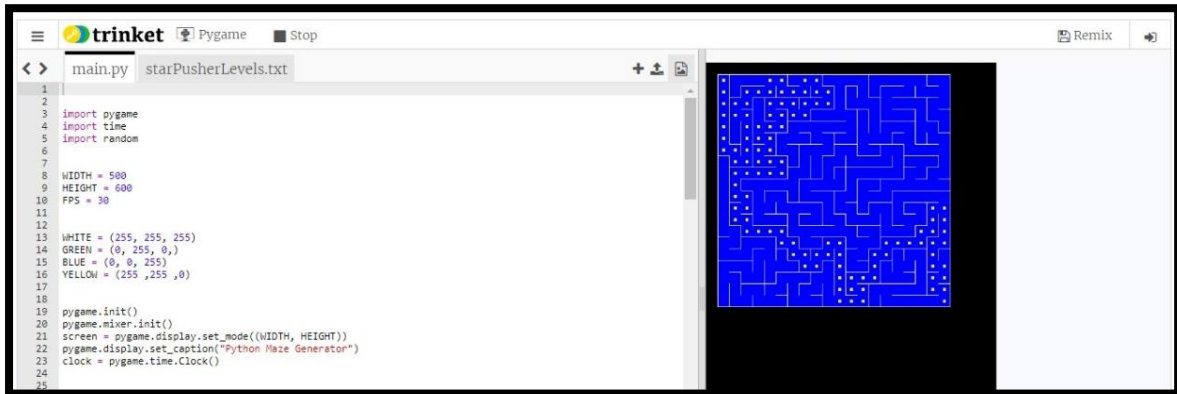
2- The Whole Maze is Scanned-



3- Finding the Shortest Path-



4- Shortest Path find using DFS-



Description of Work Division in terms of Roles among Students.

- This project is made by Ram Mohan Mishra and Mudassir Raza Khan.
- The work was divided equally among both of us.
- A full proof research on the project was done before MTE.
- We divided the coding between both of us half the coding was done by him and half by me. We did many dry runs before implementing the whole code together.
- We divided the report between us and compiled it together later.
- The preparation of final report took us almost a week.

Technologies and Framework to be used

The pygame library is an open-source module for the Python programming language specifically intended to help you make games and other multimedia applications. Built on top of the highly portable SDL (Simple Direct Media Layer) development library, pygame can run across many platforms and operating systems.

By using the pygame module, you can control the logic and graphics of your games without worrying about the backend complexities required for working with video and audio.

This tutorial will first go through installing pygame into your Python programming environment, and then walk you through creating a template to develop games with pygame and Python 3.

Prerequisites

To be able to use this tutorial, make sure you have Python 3 and a programming environment already installed on your local computer or server.

You should also be familiar with the following Python programming concepts:

Importing modules

Variables

while loops

for loops

Conditional statements

Boolean logical operators

With a programming environment set up and a familiarity with Python programming you are ready to begin working with pygame.

SWOT Analysis achieved in project.

Depth First Search has been utilized in taking care of the labyrinth puzzle issue.

Depth First Search (DFS) is a calculation for looking through a chart or tree information structure. The calculation begins at the root (top) hub of a tree and goes as far as possible down a given branch (way), at that point backtracks until it finds an unexplored way, and afterward investigates it. The calculation does this until the whole chart has been investigated. Numerous issues in software engineering can be thought of regarding diagrams. For instance, examining systems, mapping courses, booking, and finding traversing trees are diagram issues. To break down these issues, chart search calculations like profundity first hunt are valuable.

Depth First Search is a typical way that numerous individuals normally approach taking care of issues like labyrinths. Initially, we select a way in the labyrinth (for the model, we should

pick a way as per some standard we spread out early) and we tail it until we hit an impasse or arrive at the completing purpose of the labyrinth. In the event that a given way doesn't work, we backtrack and take an elective way from a past intersection, and attempt that way.

Strength-

- This is a very simple game that can be played by the children of any age group.
- Easy to handle and use.
- Shortest path is provided every time.

Weakness-

- This is a automatic game sometime the server can crash due to many players playing at the same time.
- This is less user friendly at the particular time.

Opportunities-

- With changing world new technologies will come in that will provide us a new libraries that can make the game more better.
- This game can further be made more user interactive in the future times.

Threat-

- The threat underlying in this game is that in future many more advanced game would come that will make this game less popular.
- With changing scenario it is possible that this game might not be compatible with the upcoming technologies and operating systems.