COMPUTER INVESTIGATORY PROJECT IN PYTHON PROGRAMMING LANGUAGE

The Adventure

Made By,

Mohit Mohandas

Class XI

Roll Number: 20

Acknowledgement

The success of the project depends on the persistent efforts of an individual projecting it as the sustained support received from a few others who are equally responsible for their precious appreciation of such endeavors. My strength is due to my honourable Principal Mrs. Shobha Rajiv, who has been an unending source of inspiration and support towards accomplishment of this project.

I also express my deepest gratitude to my Computer Science teacher, Mrs. Savreena Illango, who has been of utmost help at every stage of this project.

Last, but certainly not the least, I want to thank God, His Almighty for enlightening and guiding me to the completion of this project.

INDEX

|  |  |  |
| --- | --- | --- |
| Serial No. | Topic | Page No. |
| 1 | Synopsis | 5 |
| 2 | Hardware and Software Specifications | 6 |
| 3 | Source code | 7 |
| 4 | Output | 16 |
| 5 | Limitations | 21 |
| 6 | Bibliography | 22 |

SYNOPSIS

This is a game based on a tile map system where you control a player as he navigates through a maze, avoiding enemies and collecting coins on his way to the exit.

GAME PLAY:

The size of the grid is 25 \* 25.

Use (w, a, s, or d) followed by enter to navigate the player ‘P’ (up, left, down or right) respectively.

# represents walls. Players cannot go over walls.

C represents coins. They can be collected to increase your money.

X represents zombies. Being in the same tile as a zombie causes death. So avoid them!

Use ‘e’ followed by enter to end the game.

Use ‘m’ followed by enter three times consecutively to break all walls adjacent to you.

Note: Boundary wall cannot be destroyed.

HARDWARE AND SOFTWARE

SPECIFICATIONS

**Hardware:**

Processor: 3.80 GHz Processor

RAM: 8 Giga Bytes

Hard Disk Space: 1 TB

Visual Display Unit: LED

**Software:**

Operating system: Windows 10 Home

Python software: python 3.7 (64bit)

Source code

######################################

# #

# By: #

# Name: Mohit Mohandas #

# Class: XI #

# Roll no.: 20 #

# Version: 2.0 #

# #

######################################

# Note: The user interface has been modified to suit the python console.

# Double click on the file to run game.

# Certain features are disabled when running from the IDLE interpreter.

# Edits:

# 1) Added more information via comments.

# 2) Made game more user friendly.

import os

import random

import time

print("""By:

Name: Mohit Mohandas

Class: XI

Roll no.: 20

Version: 2.0""")

time.sleep(3)

# Clears screen

# Can cause errors in certain computers

os.system('cls' if os.name == 'nt' else 'clear')

# Welcome note

if input("Read instructions?(y for yes or any other key to skip): ") == "y":

# Clears screen

# Can cause errors in certain computers

os.system('cls' if os.name == 'nt' else 'clear')

print("Welcome to The Adventure.")

time.sleep(1)

print("Use w, a, s, or d followed by enter to navigate the player P " +

"up, left, down or right respectively.")

time.sleep(3)

print("# represents walls. Players cannot go over walls.")

time.sleep(2)

print("C represents coins. They can be collected to increase your money.")

time.sleep(2)

print("X represents zombies. Being in the same tile as a zombie " +

"causes death. So avoid them!")

time.sleep(3)

print("e followed by enter ends the game.")

time.sleep(2)

print("Use m followed by enter three times consecutively to break"+

" all walls surrounding you. \nNote: Boundary wall cannot be destroyed.")

time.sleep(3)

print("Press enter to start.")

input()

# Clears screen

# Can cause errors in certain computers

os.system('cls' if os.name == 'nt' else 'clear')

cmd = ""

# Initialises grid size and grid

grid\_size = (25, 15)

grid = []

for y in range(grid\_size[1]):

grid.append([])

for x in range(grid\_size[0]):

grid[y].append(" ")

# Represents the grid which is later converted to a list

grid\_str = """

#########################

# #

# #

# #

# #

# #

# #

# #

# #

# #

# #

# #

# #

# E#

#########################"""

# Converts grid\_str of string type to a grid of list type

x, y = 0, -1

for tile in grid\_str:

if tile != "\n":

grid[y][x] = tile

x += 1

else:

y += 1

x = 0

# Old method to create list. Above method allows editing of map beforehand.

# for i in range(grid\_size[0]):

# grid[0][i] = "#"

# grid[-1][i] = "#"

# for i in range(grid\_size[1]):

# grid[i][0] = "#"

# grid[i][-1] = "#"

# Adds extra walls

for i in range(100):

pos = [random.randint(1, grid\_size[0] - 1), random.randint(1, grid\_size[1] - 1)]

grid[pos[1]][pos[0]] = "#"

# Initialises player and money

player\_pos = [1, 1]

player\_pos\_old = list(player\_pos)

grid[player\_pos[1]][player\_pos[0]] = "P"

money = 0

coin\_value = 25

# Represents different game modes of the game

game\_modes = {

"easy": (2, 2),

"medium": (5, 5),

"hard": (10, 10)

}

# Get a game mode from the user

while True:

difficulty = input("Difficulty (easy, medium or hard): ")

try:

no\_of\_enemies, no\_of\_coins = game\_modes[difficulty]

break

except:

print("Invalid entry, try again.")

# Initialises enemies marked by a X

# no\_of\_enemies = int(input("Enter number of enemies: "))

random\_enemy = []

for i in range(no\_of\_enemies):

pos = [random.randint(0, grid\_size[0] - 1), random.randint(0, grid\_size[1] - 1)]

while grid[pos[1]][pos[0]] != " ":

pos = [random.randint(0, grid\_size[0] - 1), random.randint(0, grid\_size[1] - 1)]

random\_enemy.append({"pos": pos})

random\_enemy[-1]["pos\_old"] = pos

grid[pos[1]][pos[0]] = "X"

# Initialises coins marked by C

# no\_of\_coins = int(input("Enter number of coins: "))

for i in range(no\_of\_coins):

pos = [random.randint(0, grid\_size[0] - 1), random.randint(0, grid\_size[1] - 1)]

while grid[pos[1]][pos[0]] != " ":

pos = [random.randint(0, grid\_size[0] - 1), random.randint(0, grid\_size[1] - 1)]

grid[pos[1]][pos[0]] = "C"

# Mining

mine\_state = 0

# Main game loop

while True:

# Clears screen

# Can cause errors in certain computers

os.system('cls' if os.name == 'nt' else 'clear')

# Used for identifying commands

# wasd to move

if cmd == "w":

player\_pos[1] -= 1

if cmd == "a":

player\_pos[0] -= 1

if cmd == "s":

player\_pos[1] += 1

if cmd == "d":

player\_pos[0] += 1

# e to exit

if cmd == "e":

input("Exiting, press any key to quit.")

quit()

# m on three consequtive turns to destroy adjacent walls

if cmd == "m":

mine\_state += 1

if mine\_state == 3:

if grid[player\_pos[1] - 1][player\_pos[0]] == "#":

if player\_pos[1] - 1 != 0:

grid[player\_pos[1] - 1][player\_pos[0]] = " "

if grid[player\_pos[1] + 1][player\_pos[0]] == "#":

if player\_pos[1] + 1 != grid\_size[1] - 1:

grid[player\_pos[1] + 1][player\_pos[0]] = " "

if grid[player\_pos[1]][player\_pos[0] - 1] == "#":

if player\_pos[0] - 1 != 0:

grid[player\_pos[1]][player\_pos[0] - 1] = " "

if grid[player\_pos[1]][player\_pos[0] + 1] == "#":

if player\_pos[0] + 1 != grid\_size[0]:

grid[player\_pos[1]][player\_pos[0] + 1] = " "

else:

mine\_state = 0

# Checks if player collides with a wall

if grid[player\_pos[1]][player\_pos[0]] == "#":

player\_pos = list(player\_pos\_old)

# Checks if player gains a coin

if grid[player\_pos[1]][player\_pos[0]] == "C":

money += coin\_value

# Checks if the player reached the end

if grid[player\_pos[1]][player\_pos[0]] == "E":

print("You won!!!")

print("Your final score: ", money)

input("Press any key to exit.")

break

# Checks if player was killed by an enemy

if grid[player\_pos[1]][player\_pos[0]] == "X":

print("You lost.")

print("Your final score: ", money)

input("Press any key to exit.")

break

# Updates player location in grid

if player\_pos != player\_pos\_old:

grid[player\_pos\_old[1]][player\_pos\_old[0]] = " "

grid[player\_pos[1]][player\_pos[0]] = "P"

player\_pos\_old = list(player\_pos)

# Enemy's AI - currently random

for i\_ in range(len(random\_enemy)):

i = random\_enemy[i\_]

r = random.randint(1, 5)

if r == 1:

i["pos"][0] += 1

if r == 2:

i["pos"][0] -= 1

if r == 3:

i["pos"][1] += 1

if r == 4:

i["pos"][1] -= 1

if grid[i["pos"][1]][i["pos"][0]] in ("P", "#"):

i["pos"] = list(i["pos\_old"])

else:

grid[i["pos"][1]][i["pos"][0]] = "X"

grid[i["pos\_old"][1]][i["pos\_old"][0]] = " "

i["pos\_old"] = list(i["pos"])

# Displays grid

for y in grid:

for x in y:

print(x, end="")

print()

# Displays money

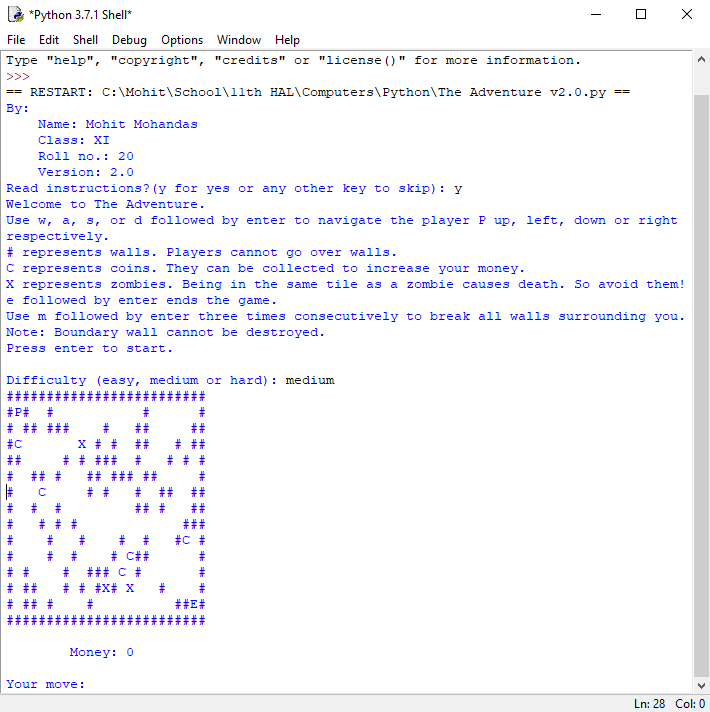
print("\n\tMoney:", money)

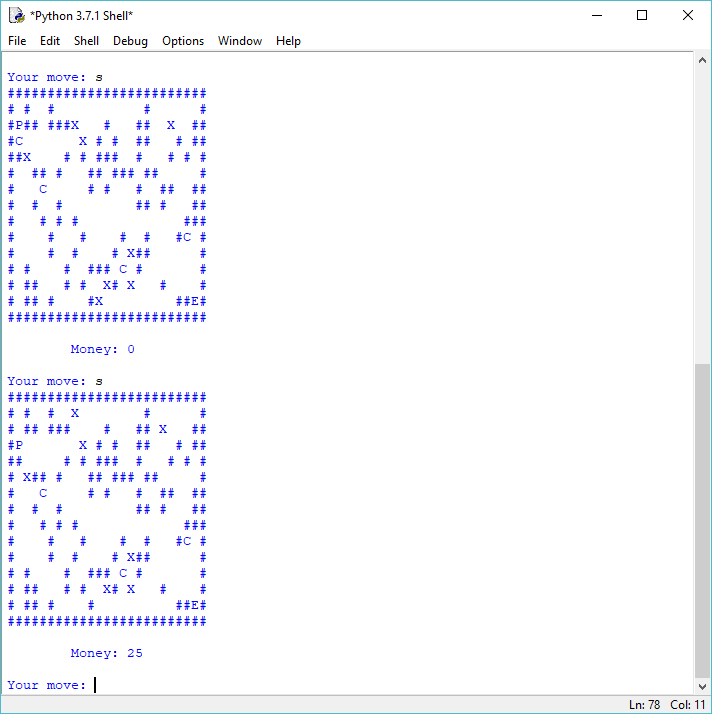
# Gets next command from user

cmd = input("\nYour move: ")

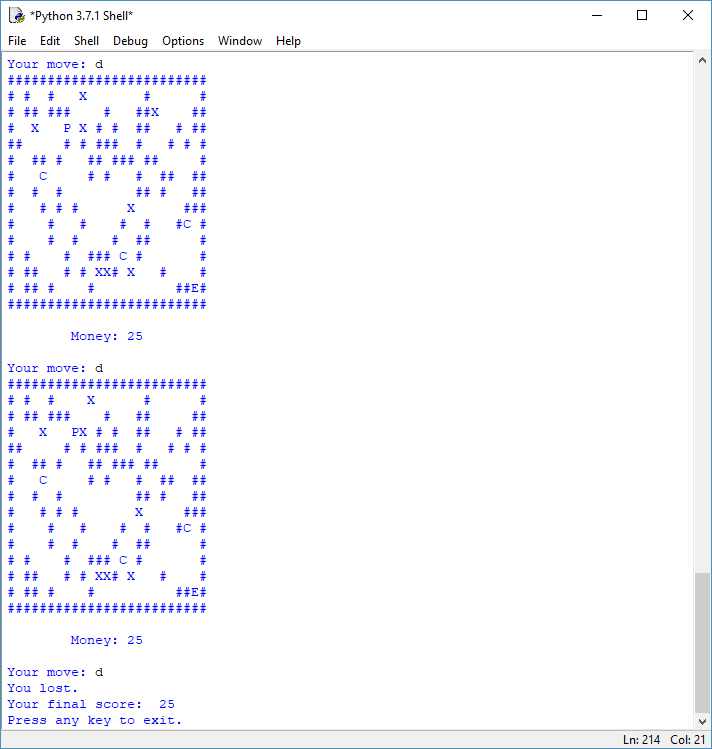
OUTPUT

Upon starting the program:

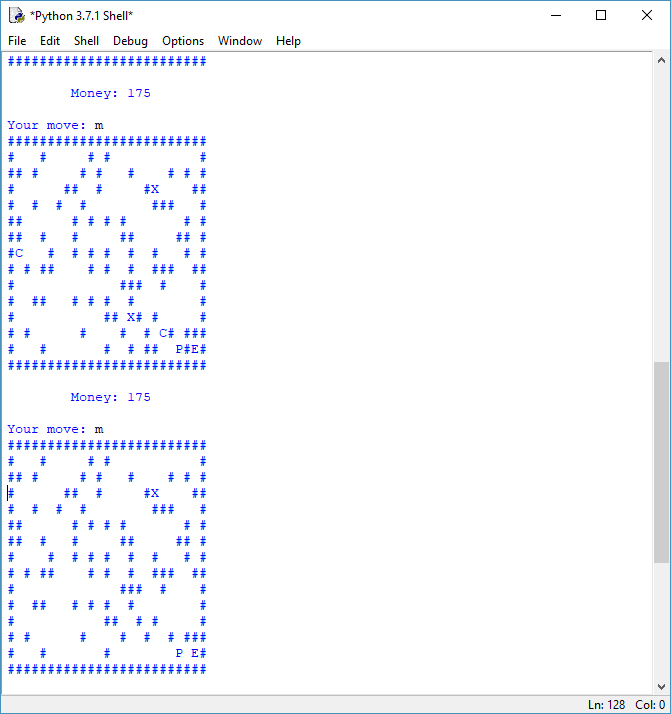


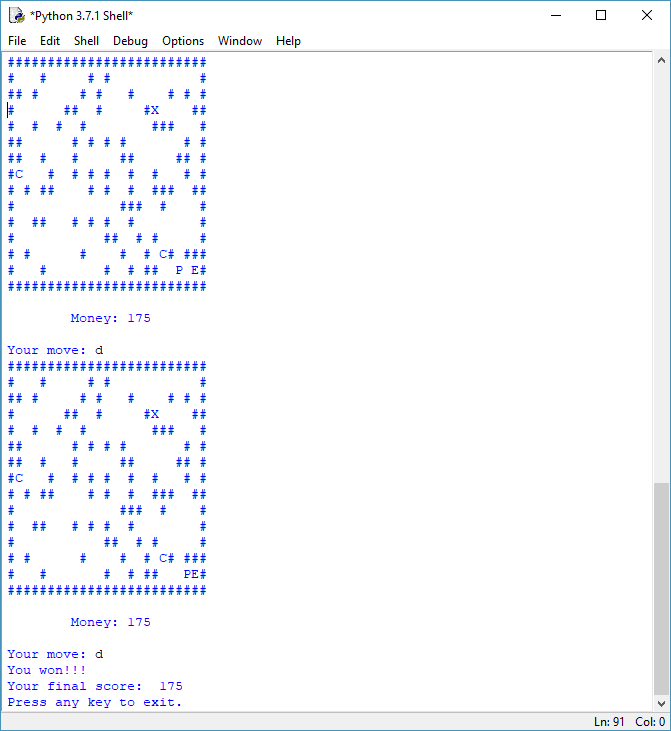
****

Collecting coins:

****Upon colliding with an enemy:

Destroying blocks with ‘m’:



Winning the game:****

LIMITATIONS

This project has a few limitations :

* A single mistake from your side may end the player.
* Currently the AI for enemies is random, there is no predictable pattern to their movement.
* More testing and refinement is required for a complete and entertaining game.
* There is no save feature for the game.
* As python uses an interpreter, a python interpreter has to be installed for the game to run.

BIBLIOGRAPHY

The sources of information for this project are :

* ‘Computer Science with Python’ for class 11 by Sumita Arora. Publication: Danpat Rai & co.
* Official python website: https://www.python.org/

Thank You.