PACMAN IN PYTHON

Mini-Project (OPEN SOURCE TECH LAB)

Submitted in partial fulfilment of the requirement of University of Mumbai
For the Degree of

Computer Engineering

By

1) Mitali Vyankat Mane (C-32) ID No: TUS3F181934
2) Bhushan Rajendra Patil (C-29) ID No: TUS3F181930
3) Tanuj Palaspagar (C-26) ID No: TUS3F181926

Under the Guidance of

Prof. Ramesh Shahabade

Department of Computer Engineering
TERNA ENGINEERING COLLEGE
Plot no.12, Sector-22, Opp. Nerul Railway station,
Phase-11, Nerul (w), Navi Mumbai 400706
UNIVERSITY OF MUMBAI

Terna Engineering College

NERUL, NAVI MUMBAI

CERTIFICATE

This is to certify that

1) Mitali Vyankat Mane (C-32)	**ID No: TUS3F181934**
2) Bhushan Rajendra Patil (C-29)	**ID No: TUS3F181930**
3) Tanuj Palaspagar (C-26)	**ID No: TUS3F181926**

*Has satisfactorily completed the requirements of the **Mini Project***

Of subject

Open Source Tech Lab

*As prescribed by the **University of Mumbai** Under the guidance of*

Prof. **Ramesh Shahabade**

Subject Incharge **APC** **HOD**

Index

Abstract

Pac-Man is a Japanese video game franchise published and owned by Bandai Namco Entertainment, formerly Namco. Entries have been developed by a wide array of other video game companies, including Midway Games, Atari and Mass Media, Inc.. The eponymous first entry was released in arcades in 1980 by Namco, and published by Midway Games in North America. Most Pac-Man games are maze chase games, however it has also delved into other genres, such as platformers, racing, and sports. Several games in the series have been released for a multitude of home consoles and are included in many Namco video game compilations.

Pac-Man is one of the longest-running, best-selling, and highest-grossing video game franchises in history; it has seen regular releases for nearly 40 years, has sold nearly 48 million copies across all platforms, and has grossed over US$14.107 billion, most of which has been from the original arcade game. The character of Pac-Man is the official mascot of Bandai Namco, and remains one of the most recognizable video game characters in history. The franchise is seen as important and influential, and is often used as a representation for 1980's popular culture and video games as a whole.

Creators Of the Original Game:

- Designers: Toru Iwatani, Shigeru Miyamoto, Alex Johnson, amongst others
- Developers: Nintendo, Namco, Atari, Inc., Midway Games, amongst others
- Publishers: Nintendo, Namco, Atari, Inc., Midway Games, Philips, amongst others

Pac-Man

Genre(s)	Maze
Developer(s)	Namco Bandai Namco Studios
Publisher(s)	Namco Bandai Namco Entertainment
Creator(s)	Toru Iwatani
Composer(s)	Toshio Kai
Platform(s)	**Ports** [show]
First release	*Pac-Man* July 1980
Latest release	*Pac-Man Party Royale* October 18, 2019

Introduction

Pac-Man, a single-player arcade maze-chase video game. In the game, a yellow, circular character navigates a maze, eating pellets and fruit, avoiding ghosts and occasionally eating them. The player controls the titular character through an enclosed maze; the objective of the game is to eat all of the dots placed in the maze while avoiding four ghosts. If Pac-Man makes contact with a ghost, he will lose a life; the game ends when all lives are lost.

Features:

- User friendly interface
- Easy to use
- Runs on various platforms

Design

The Code design is very reader friendly. We have used Three libraries: inbuilt random, and add on libraries : turtle and freegames.

Information of add-on libraries:

- **Turtle:**
  We have decided to use turtle library for the movement of the characters: the Pac-man as well as the ghosts. It is a very coder friendly library, intended for availing ease of movements in the code. Since our code is fairly simple, we decided to adapt our program to this library. It has made us focus on the design of the code more than just raw coding.
  Information from the official site of turtle:

  > "Python Turtle:An educational environment for learning Python, suitable for beginners and children. Inspired by LOGO.
  > Homepage: http://pythonturtle.org
  > An Appealing Environment to Learn Python
  > PythonTurtle strives to provide the lowest-threshold way to learn Python. Students command an interactive Python shell (similar to the IDLE development environment) and use Python functions to move a turtle displayed on the screen.
  > An illustrated help screen introduces the student to the basics of Python programming while demonstrating how to move the turtle. Simplicity and a colorful visual appearance makes the learning environment more appealing to students."

- **Freegames:**
  Freegames is an open-sourced library managed by many contributors. This library is originally intended for acting as a skeletal backbone which makes it easier for developers to create their own games. We have used this library to provide the map for this project. We went this route because otherwise this project would have had a very long development period and the code would have induced multiple complexities. We have used the 'floor' and 'vector' functions from this library to bring this design to life.
  Information from the core developers:

  > "Free Python Games is an Apache2 licensed collection of free Python games intended for education and fun. The games are written in simple Python code and designed for experimentation and changes."

The code is divided into various methods, so that it is easy to read and develop:

- **Main**
  The main method does most of the work in the code. Not all functions are given to the methods, hence the main function does more than just bringing value to the methods.

The characters of pacman and ghosts are set here. Also the values of the maps that are handled in the back hand are defined in this method. The main method is not defined in the code, because it functionally behaves more like a setup than an actual method. The commands in this method need to be executed only once hence, I feel this makes the reader understand the basic structure of the code as it does not break the code to call a method outside of its domain.
The Canvas is designed here, with proper positioning given to the score and the map.

- **Square**
  The square method draws a square depending on the path currently occupied by the character. This method uses the path function, and does not return any value to the main method. Rather it takes the responsibility to draw the said square directly to the graphic window. This saves on complexity of the program and results in a simple but effective way to map the squares.
- **Offset**
  This method adapts the calculation of point in tiles. This means, it rounds off the calculation of the value that is passed to it as a parameter and returns an offset so that it can be used in the program code effectively. This is a very simple but essential calculation of the values that are used in the program.
- **Valid**
  This method checks for the validity of the movement of the characters. This essentially forms a connection between the backhand and front-end of the program. It does a simple comparison of values given by the main as parameters, checks if the movement is valid and returns a boolean value accordingly.
- **World**
  This method draws on the game canvas. It provides visuals to the user and gives a sense of what the current world looks like. Here, we have set the background of the canvas black and the path marks the blue squares. White is given to dots that increase the score, and colors are given based on the original arcade game.
- **Move**
  This method controls the movement of all ghosts as well as pacman. The movement of ghosts is automated by random numbers. The movement of pacman in this method is purely dependent on the user's actions towards the program. This method only works for raw movement values and does not take into consideration any other factors.
- **Change**
  This method integrated the actual movement of the Pacman to the functions input by the user. The movement only takes place if the movement is valid by the map, that is there is no wall or obstacle in the path of the pacman. If the pacman is not able to move in the map according to the inputs received by the user, the pac man will stay in its previous square.

Implementation

Code in Python 3:

```
from random import choice
from turtle import *
from freegames import floor, vector

state = {'score': 0}
path = Turtle(visible=False)
writer = Turtle(visible=False)
aim = vector(5, 0)
pacman = vector(-40, -80)
ghosts = [
    [vector(-180, 160), vector(5, 0)],
    [vector(-180, -160), vector(0, 5)],
    [vector(100, 160), vector(0, -5)],
    [vector(100, -160), vector(-5, 0)],
]
tiles = [
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
]
```

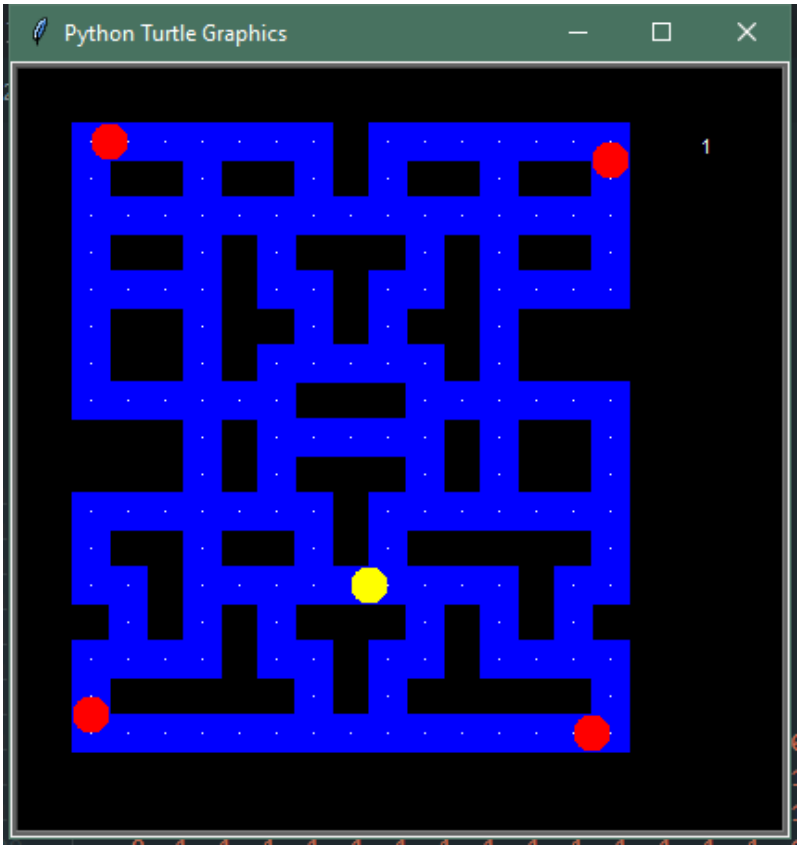
```
def square(x, y):
    # "Draw squares using the path at (x, y)."
    path.up()
    path.goto(x, y)
    path.down()
    path.begin_fill()

    for count in range(4):
        path.forward(20)
        path.left(90)

    path.end_fill()

def offset(point):
    # "Return offset of point in tiles."
    x = (floor(point.x, 20) + 200) / 20
    y = (180 - floor(point.y, 20)) / 20
    index = int(x + y * 20)
    return index

def valid(point):
    # "Return True if the point is valid in tiles."
    index = offset(point)

    if tiles[index] == 0:
        return False

    index = offset(point + 19)

    if tiles[index] == 0:
        return False

    return point.x % 20 == 0 or point.y % 20 == 0

def world():
    # "Draw the world using a path."
    bgcolor('black')
    path.color('blue')

    for index in range(len(tiles)):
```

```
        tile = tiles[index]

        if tile > 0:
            x = (index % 20) * 20 - 200
            y = 180 - (index // 20) * 20
            square(x, y)

            if tile == 1:
                path.up()
                path.goto(x + 10, y + 10)
                path.dot(2, 'white')

def move():
    # "Move pacman and all ghosts."
    writer.undo()
    writer.write(state['score'])

    clear()

    if valid(pacman + aim):
        pacman.move(aim)

    index = offset(pacman)

    if tiles[index] == 1:
        tiles[index] = 2
        state['score'] += 1
        x = (index % 20) * 20 - 200
        y = 180 - (index // 20) * 20
        square(x, y)

    up()
    goto(pacman.x + 10, pacman.y + 10)
    dot(20, 'yellow')

    for point, course in ghosts:
        if valid(point + course):
            point.move(course)
        else:
            options = [
                vector(5, 0),
```

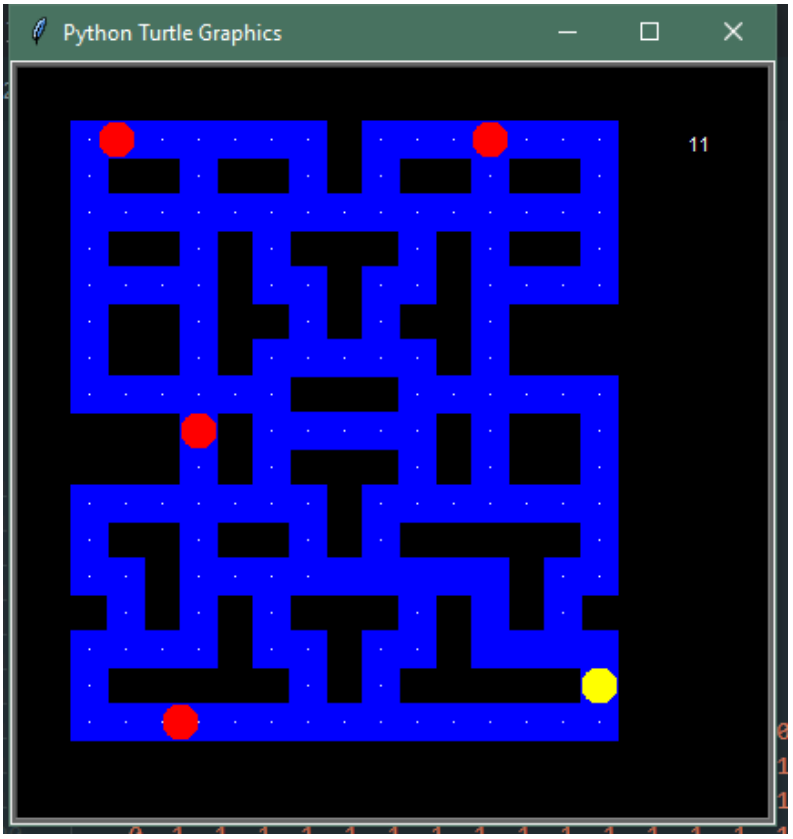
```
                vector(-5, 0),
                vector(0, 5),
                vector(0, -5),
            ]
            plan = choice(options)
            course.x = plan.x
            course.y = plan.y

        up()
        goto(point.x + 10, point.y + 10)
        dot(20, 'red')

    update()

    for point, course in ghosts:
        if abs(pacman - point) < 20:
            return

    ontimer(move, 100)

def change(x, y):
    # "Change pacman aim if valid."
    if valid(pacman + vector(x, y)):
        aim.x = x
        aim.y = y

setup(420, 420, 370, 0)
hideturtle()
tracer(False)
writer.goto(160, 160)
writer.color('white')
writer.write(state['score'])
listen()
onkey(lambda: change(5, 0), 'Right')
onkey(lambda: change(-5, 0), 'Left')
onkey(lambda: change(0, 5), 'Up')
onkey(lambda: change(0, -5), 'Down')
world()
move()
done()
```

Interface

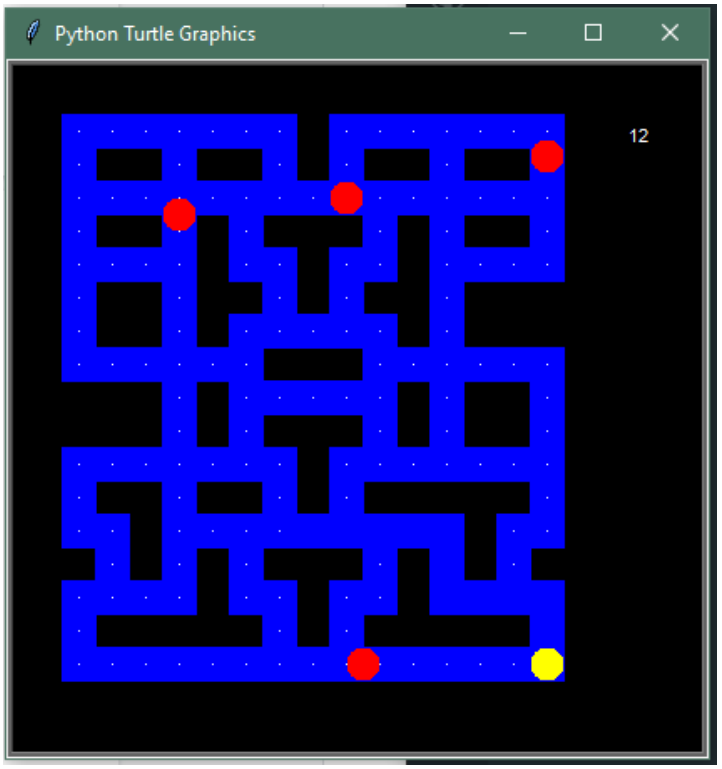
First Look:



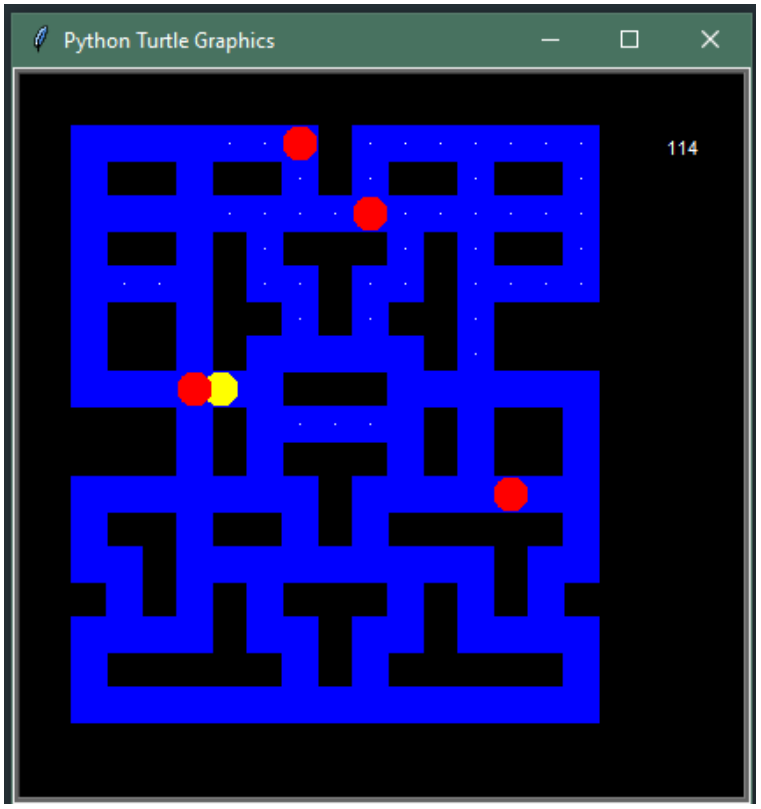
Earning Points:



The characters are constraint to the movable Blue path:



Game Ends when PacMan and Ghost share the same square:



Technical Considerations

- The code designed in Python 3.7.7
  - Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 11:52:54) [MSC v.1900 64 bit (AMD64)] on win32
- The development of this code is done in Microsoft's Visual Studio Code IDE:
  - Version: 1.44.0
  - Commit: 2aae1f26c72891c399f860409176fe435a154b13
  - Date: 2020-04-07T23:31:18.860Z
  - Electron: 7.1.11
  - Chrome: 78.0.3904.130
  - Node.js: 12.8.1
  - V8: 7.8.279.23-electron.0
  - OS: Windows_NT x64 10.0.18363
- This code is proven to run in the inbuilt python IDLE, but we have had some issues with running the code directly through CLI
- The code needs some libraries, that we have installed though the CLI based pip installer:
  Turtle: `pip install turtle`
  Freegames: `pip install freegames`
  The libraries are standard libraries and hence they are available through the installer. If there are issues with the said libraries, they can be manually created and imported through the official sites (linked earlier).
  We have not tested other installers like easy_install, so we cannot confirm the success of that method.
- This program was built and run on a Windows(NT) 10 based machine:

OS Name	Microsoft Windows 10 Home Single Language
Version	10.0.18363 Build 18363
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	MAVIS
System Manufacturer	Acer
System Model	Nitro AN515-52
System Type	x64-based PC
System SKU	0000000000000000
Processor	Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, 2304 Mhz, 4 Core(s), 8 Logical Processor(s)
BIOS Version/Date	Insyde Corp. V1.28, 05-Aug-19
SMBIOS Version	3.0
Embedded Controller Version	1.26
BIOS Mode	UEFI
BaseBoard Manufacturer	CFL
BaseBoard Product	Freed_CFS
BaseBoard Version	V1.28
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume2
Locale	United States
Hardware Abstraction Layer	Version = "10.0.18362.628"
User Name	MAVIS\tanuj
Time Zone	India Standard Time
Installed Physical Memory (RAM)	32.0 GB
Total Physical Memory	31.8 GB
Available Physical Memory	22.4 GB
Total Virtual Memory	36.6 GB
Available Virtual Memory	25.6 GB
Page File Space	4.75 GB
Page File	C:\pagefile.sys
Kernel DMA Protection	Off
Virtualization-based security	Not enabled
Device Encryption Support	Elevation Required to View
Hyper-V - VM Monitor Mode Extensions	Yes
Hyper-V - Second Level Address Translation Extensions	Yes
Hyper-V - Virtualization Enabled in Firmware	Yes
Hyper-V - Data Execution Protection	Yes

Conclusion

We have aimed at creating a simple game with minimal complexity. This code has no UI as such, only direct gameplay. This has resulted in a very barebones type of program that is complete in itself, but lacks customisation from the user. This is a naive approach in creating a game and introduces us to a variety of concepts that are more implementation-based. This code is an independent functioning software program with minimum bugs, if any, but it is in no sense complete. There is a huge headway to improvement and developing the game, more like the original game and studying about it has made us aware of the potential of the game.

Reference

- Project Ideas and information:
  - *https://en.wikipedia.org/wiki/List_of_Pac-Man_video_games*
  - *https://data-flair.training/blogs/python-project-ideas/*
  - Opensource Community on *Reddit.com* & *GitHub.com*
- Python Programming Guides:
  - *https://www.geeksforgeeks.org/*
  - *https://www.python.org/about/gettingstarted/*
  - *https://www.w3schools.com/python/*
  - Python: The Complete Reference Book