Project Report

On

PAC MAN

Submitted By:

RAJALAKSHMI D----------RA2211026040013

AKTHAR NM----------RA2211026040096

GOMATHI GANESAN K----------RA2211026040110

Under the guidance of

Dr Sridevi Sridhar

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING PRACTICE

in the

Department of Computer Science & Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and

Engineering SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY VADAPALANI CAMPUS

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Vadapalani Campus

BONAFIDE CERTIFICATE

Certified that this major project report for the course 21CSC203P ADVANCED PROGRAMMING PRACTICE entitled in "PAC MAN" is the bonafide work of RAJALAKSHMI D(RA2211026040013), AKTHAR NM(RA2211026040096), GOMATHI GANESAN K(RA2211026040110) who carried out the work under my supervision.

SIGNATURE

Dr Sridevi Sridhar

Assistant Professor (Sr. G)

Department of Computer Science & Engineering

SRM Institute of Science and Technology

Vadapalani campus

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy & Dr S Ramachandran Director (Academics),** for their encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. CV Jayakumar** for providing the support and infrastructure to perform the project.

We wish to express my sincere thanks to our Head of the Department **Dr S Prasanna Devi** for her constant encouragement and support.

We are highly thankful to our Course Faculty-in-Charge **Dr Sridevi Sridhar** for her assistance, timely suggestion and guidance throughout the duration of this project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings

on me to complete the project.

TABLE OF CONTENTS

ABSTRACT:

The Pac-Man Python project is an exciting, retro-inspired game created using Python. In this classic arcade-style game, players take control of Pac-Man, a yellow character on a mission to gobble up all the dots while avoiding pesky ghosts. The objective is to clear the maze of dots and fruit while staying one step ahead of the ghosts. The project offers a nostalgic yet user-friendly interface, randomized ghost behavior, score tracking, and serves as an engaging and entertaining way to relive the classic Pac-Man experience in Python. It's a fun and accessible game suitable for players of all ages, providing hours of thrilling gameplay and a trip down memory lane.

CHAPTER 1

INTRODUCTION

Introducing the Pac-Man Python project: A thrilling and nostalgic gaming adventure that will test your strategic skills! Control Pac-Man as he navigates through mazes, munching on dots and avoiding pesky ghosts. Your objective is to clear the maze of dots and collect fruits while staying one step ahead of the ghosts. This exciting project features a user-friendly interface, dynamic ghost behavior, and score tracking, providing hours of fun and a trip down memory lane. Suitable for players of all ages, it's a captivating Python game that combines entertainment and a touch of nostalgia in one exhilarating experience.

1.1 PROJECT AIMS AND OBJECTIVES:

.Dot-Munching Gameplay
.Strategic Challenge
.Maze Clearing Objective
.Fruit Bonuses
.User-Friendly Interface
.Dynamic Ghost Behavior
.Score Tracking
.Python Implementation

1.2 BACKGROUND OF PROJECT:

The Python Pac-Man project was born from the need to bring back the charm of classic arcade gaming in a digital era filled with distractions. The project's development team recognized the significance of preserving nostalgic experiences and decided to recreate the beloved Pac-Man game in Python. By doing so, they aimed to provide an entertaining and educational gaming experience that hones players' strategic skills.

Leveraging the versatility of Python, this project takes players on a journey through mazes, encouraging strategic thinking while offering a dose of nostalgia. With the objective of guiding Pac-Man through the maze, munching on dots, and avoiding ghosts, the project preserves the essence of the classic Pac-Man game while making it accessible and enjoyable to a broad audience.

Incorporating dynamic ghost behavior and fruit bonuses adds an element of strategy and excitement, further enhancing the project's appeal as both a

recreational and educational tool.

1.3 OPERATION ENVIRONMENT:

PROCESSOR	INTEL CORE PROCESSOR FOR BETTER PERFORMANCE
OPERATING SYSTEM	WINDOWS VISTA ,WINDOWS7, windows 10,windows 11
MEMORY	1GB RAM FOR MORE
HARD DISK SPACE	MINIMUM 3 GB FOR DATABASE USAGE FOR FUTURE

CHAPTER 2

SYSTEM ANALYSIS

In the system analysis phase of the Python Pac-Man project, the primary objectives revolve around defining essential requirements, designing the system architecture, and ensuring the seamless integration of various game components. This comprehensive process includes gathering requirements from stakeholders, designing the user interface to capture the essence of the classic Pac-Man experience, creating algorithms for managing Pac-Man's movements and ghost behaviors, and taking into consideration performance and security aspects.
To guarantee the game's longevity and adaptability, scalability and maintenance considerations are also addressed. This analysis phase serves as the solid foundation upon which the development and successful implementation of the Python Pac-Man project are built, ensuring it faithfully recreates the classic Pac-Man experience while providing a fun and educational gaming adventure for players of all ages.

2.1 SOFTWARE REQUIREMENT SPECIFICATION:

System Design: During this phase, the project's system architecture is meticulously planned. This involves outlining the overall game structure, creating a graphical user interface (GUI) that encapsulates the essence of the classic Pac-Man experience, and designing the data structures that will be used. The project also defines the algorithms responsible for Pac-Man's movement, ghost AI, and score tracking, while reaffirming the use of Python for implementation.

Data Flow Diagram (DFD): A DFD can be employed to visualize how data and control flow within the Pac-Man game. It illustrates player interactions with the game, how game data is processed, and how real-time feedback is provided to players.

User Interface Design: The project's user interface is meticulously designed to ensure an intuitive and engaging experience. This includes creating wireframes and mock-ups to capture the game's visual aesthetics and interactive elements, such as Pac-Man's movement, dot consumption, and score display.

Algorithm Design: The logic for Pac-Man's movements, ghost AI behavior, and scoring mechanisms are comprehensively defined. The use of appropriate data structures like grids or arrays for managing game data is

determined.

Testing and Validation: A robust testing strategy is formulated, encompassing unit testing to evaluate individual game components and integration testing to ensure the seamless functioning of various game modules. User testing is also vital to gather player feedback and make necessary improvements.

Performance Analysis: Performance metrics are established to assess the game's efficiency in terms of response time and resource utilization. This ensures that the Pac-Man game runs smoothly and does not impose unnecessary demands on system resources.

Security Considerations: Security measures, such as cheat prevention and protection against game manipulation, are integrated into the design to maintain a fair and enjoyable gaming experience.

Documentation: Extensive documentation is produced, including user manuals, developer guides, and any technical documentation required for understanding and working with the Pac-Man project.

Maintenance and Scalability: Future maintenance and scalability considerations are addressed to ensure that the Pac-Man game can be updated, expanded, or modified as needed to keep it engaging and relevant for players.

2.2 SOFTWARE TOOL

- Python
- Pygame
- Integrated Development Environment (IDE)
- Graphics Design Software
- Audio Editing Software
- Documentation Tools
- Testing and Debugging Tools
- Performance Profiling Tools
- Security Tools
- Text Editors

CHAPTER 3

SOURCE

CODE

pacman.py > Ghost > move_blinky

```
1
2   import copy
3   from board import boards
4   import pygame
5   import math
6
7   pygame.init()
8
9   WIDTH = 900
10  HEIGHT = 950
11  screen = pygame.display.set_mode([WIDTH, HEIGHT])
12  timer = pygame.time.Clock()
13  fps = 60
14  font = pygame.font.Font('freesansbold.ttf', 20)
15  level = copy.deepcopy(boards)
16  color = 'blue'
17  PI = math.pi
18  player_images = []
19  for i in range(1, 5):
20      player_images.append(pygame.transform.scale(pygame.image.load(f'assets/player_images/{i}.png'), (45,
21  blinky_img = pygame.transform.scale(pygame.image.load(f'assets/ghost_images/red.png'), (45, 45))
22  pinky_img = pygame.transform.scale(pygame.image.load(f'assets/ghost_images/pink.png'), (45, 45))
23  inky_img = pygame.transform.scale(pygame.image.load(f'assets/ghost_images/blue.png'), (45, 45))
24  clyde_img = pygame.transform.scale(pygame.image.load(f'assets/ghost_images/orange.png'), (45, 45))
25  spooked_img = pygame.transform.scale(pygame.image.load(f'assets/ghost_images/powerup.png'), (45, 45))
26  dead_img = pygame.transform.scale(pygame.image.load(f'assets/ghost_images/dead.png'), (45, 45))
27  player_x = 450
28  player_y = 663
29  direction = 0
30  blinky_x = 56
31  blinky_y = 58
32  blinky_direction = 0
33  inky_x = 440
34  inky_y = 388
```

pacman.py > Ghost > move_blinky

```
69  class Ghost:
70      def __init__(self, x_coord, y_coord, target, speed, img, direct, dead, box, id):
71          self.x_pos = x_coord
72          self.y_pos = y_coord
73          self.center_x = self.x_pos + 22
74          self.center_y = self.y_pos + 22
75          self.target = target
76          self.speed = speed
77          self.img = img
78          self.direction = direct
79          self.dead = dead
80          self.in_box = box
81          self.id = id
82          self.turns, self.in_box = self.check_collisions()
83          self.rect = self.draw()
84
85      def draw(self):
86          if (not powerup and not self.dead) or (eaten_ghost[self.id] and powerup and not self.dead):
87              screen.blit(self.img, (self.x_pos, self.y_pos))
88          elif powerup and not self.dead and not eaten_ghost[self.id]:
89              screen.blit(spooked_img, (self.x_pos, self.y_pos))
90          else:
91              screen.blit(dead_img, (self.x_pos, self.y_pos))
92          ghost_rect = pygame.rect.Rect((self.center_x - 18, self.center_y - 18), (36, 36))
93          return ghost_rect
94
95      def check_collisions(self):
96          # R, L, U, D
97          num1 = ((HEIGHT - 50) // 32)
98          num2 = (WIDTH // 30)
99          num3 = 15
100         self.turns = [False, False, False, False]
101         if 0 < self.center_x // 30 < 29:
```

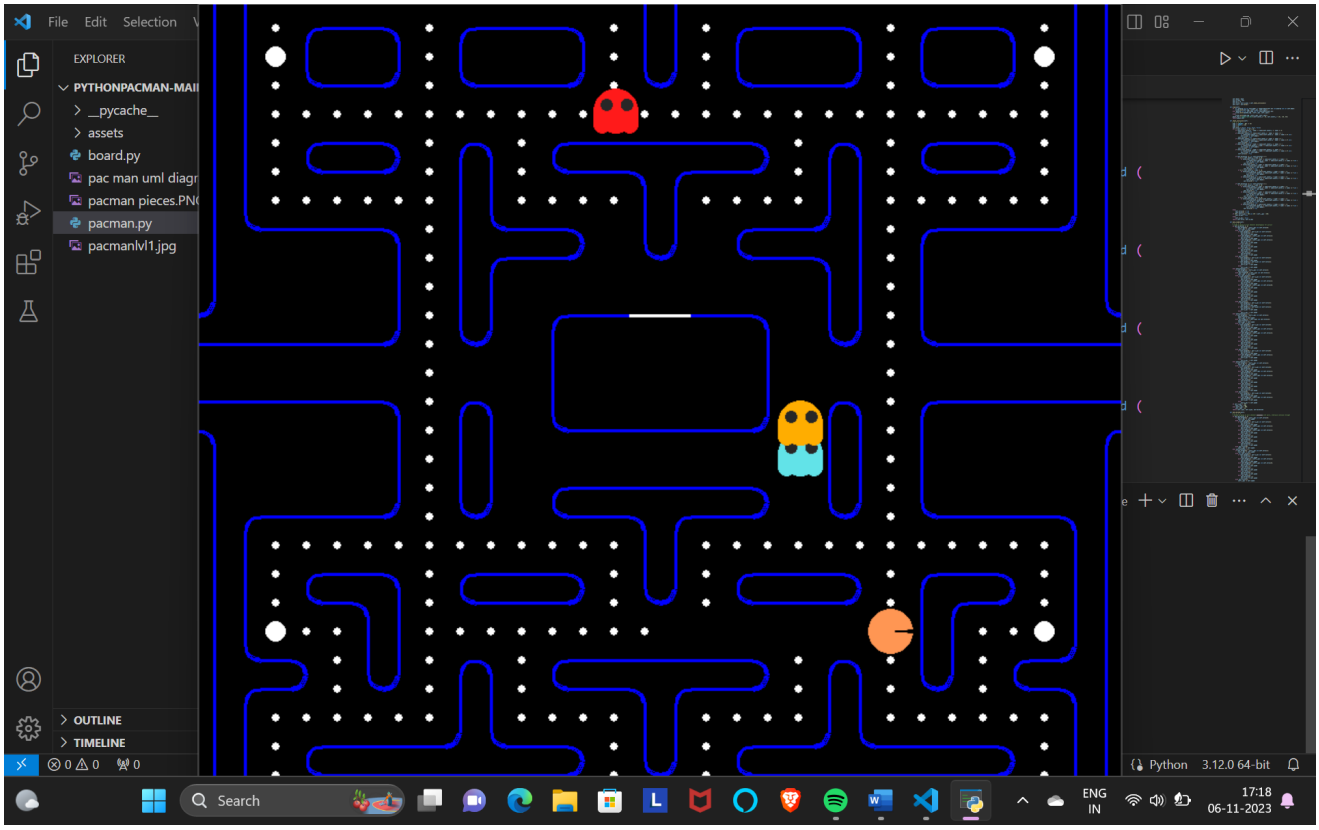
pacman.py > Ghost > move_blinky

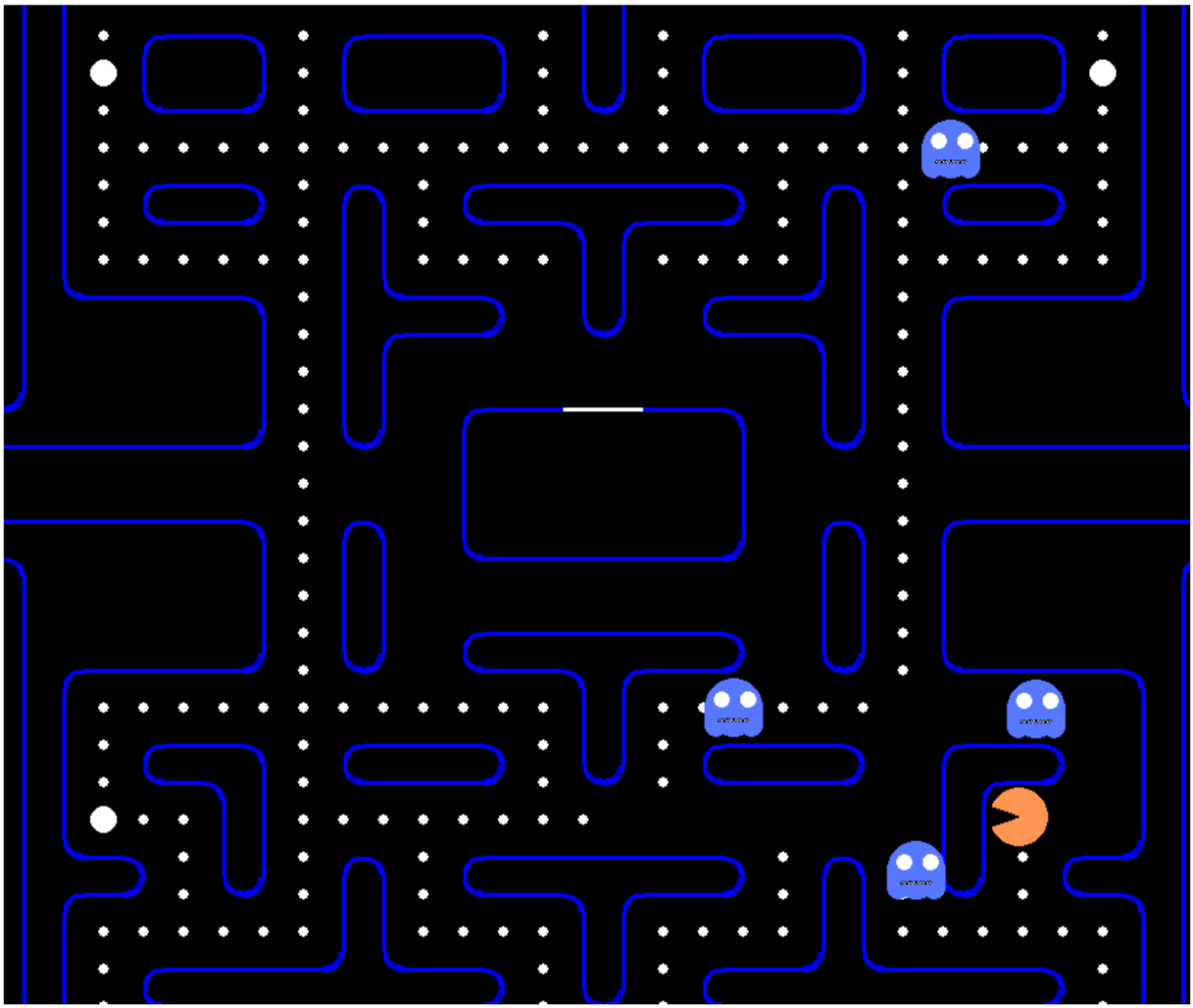
```
102                 if level[(self.center_y - num3) // num1][self.center_x // num2] == 9:
103                     self.turns[2] = True
104                 if level[self.center_y // num1][(self.center_x - num3) // num2] < 3 \
105                         or (level[self.center_y // num1][(self.center_x - num3) // num2] == 9 and (
106                         self.in_box or self.dead)):
107                     self.turns[1] = True
108                 if level[self.center_y // num1][(self.center_x + num3) // num2] < 3 \
109                         or (level[self.center_y // num1][(self.center_x + num3) // num2] == 9 and (
110                         self.in_box or self.dead)):
111                     self.turns[0] = True
112                 if level[(self.center_y + num3) // num1][self.center_x // num2] < 3 \
113                         or (level[(self.center_y + num3) // num1][self.center_x // num2] == 9 and (
114                         self.in_box or self.dead)):
115                     self.turns[3] = True
116                 if level[(self.center_y - num3) // num1][self.center_x // num2] < 3 \
117                         or (level[(self.center_y - num3) // num1][self.center_x // num2] == 9 and (
118                         self.in_box or self.dead)):
119                     self.turns[2] = True
120 
121                 if self.direction == 2 or self.direction == 3:
122                     if 12 <= self.center_x % num2 <= 18:
123                         if level[(self.center_y + num3) // num1][self.center_x // num2] < 3 \
124                                 or (level[(self.center_y + num3) // num1][self.center_x // num2] == 9 and (
125                                 self.in_box or self.dead)):
126                             self.turns[3] = True
127                         if level[(self.center_y - num3) // num1][self.center_x // num2] < 3 \
128                                 or (level[(self.center_y - num3) // num1][self.center_x // num2] == 9 and (
129                                 self.in_box or self.dead)):
130                             self.turns[2] = True
131                     if 12 <= self.center_y % num1 <= 18:
132                         if level[self.center_y // num1][(self.center_x - num2) // num2] < 3 \
133                                 or (level[self.center_y // num1][(self.center_x - num2) // num2] == 9 and (
134                                 self.in_box or self.dead)):
```

CHAPTER 4

SYSTEM IMPLEMENTATION

Screenshots of the pages





CHAPTER 5

SYSTEM TESTING

In the context of a Pac-Man Python project, the aim of the testing process is to identify and rectify any defects or issues within the game. The game is exposed to a range of test scenarios and inputs, and observations are made based on these scenarios to determine if the Pac-Man game functions as expected. The project typically undergoes two levels of testing:

Unit Testing:

In the unit testing phase of the Pac-Man Python project, individual components or modules of the game are examined in isolation. Each component, such as Pac-Man movement, ghost behavior, scoring system, and power-ups, is tested independently to ensure that it functions correctly. This testing phase ensures that each piece of the game operates as intended before they are integrated into the complete game.

Integration Testing:

In the integration testing phase, the various components and modules of the Pac-Man game are combined and tested as a whole to evaluate how they interact with each other. This phase ensures that the integration of different elements, such as Pac-Man, ghosts, mazes, and game logic, does not introduce unexpected issues or conflicts. It also verifies that the game's overall functionality, scoring, and behavior are consistent with the game's design.

By undergoing both unit testing and integration testing, the Pac-Man Python project can be thoroughly evaluated and fine-tuned to ensure that it behaves as expected and provides an enjoyable gaming experience for players.

For a Pac-Man Python project, the integration testing scenarios should focus on ensuring that the game's components and features work together correctly to create a cohesive and enjoyable gaming experience. Here are some integration test scenarios tailored to the Pac-Man project:

Testing User Interface and Game Logic Interaction:

Verify that Pac-Man's movement is responsive to player input.
Confirm that Pac-Man consumes pellets and power pellets, updating the score accordingly.
Test the interaction between Pac-Man and ghosts, including collision detection and game over conditions.
Ensure that the game accurately displays the number of lives remaining and the current score on the user interface.

->Testing Level Progression:

Verify that the game transitions to the next level when all pellets are consumed.
Confirm that the maze layout, number of ghosts, and pellet distribution change with each new level.
Test that the game resets Pac-Man's position and the ghosts' behavior at the start of each level.
->Testing Ghost AI and Movement:

Verify that the ghosts exhibit appropriate behaviors, such as chasing Pac-Man, using a scatter mode, and avoiding collisions with walls.
Confirm that ghost movement is random when they are in frightened mode.
Test that ghosts return to their base after being eaten by Pac-Man and are subsequently released.
->Testing Power-ups and Special Items:

Verify that consuming a power pellet causes the ghosts to enter frightened mode and Pac-Man can eat them.
Confirm that consuming fruit or other special items awards bonus points and updates the score accordingly.
Test that power-up effects, such as ghost vulnerability, have a limited duration.
->Testing Game Over and Restart:

Verify that the game ends when Pac-Man loses all lives and displays a game over screen.
Confirm that the player has the option to restart the game from the beginning or at the current level.
Test that the high score is updated and saved if it is achieved.
->Testing Audio and Visual Feedback:

Verify that the game provides audio feedback, such as eating sounds and background music.
Confirm that visual feedback, like animations, indicates significant events, such as Pac-Man death or power-up activation.
->Testing Performance and Resource Usage:

Ensure that the game runs smoothly and without lag on various hardware configurations.
Test the resource usage, including CPU and memory, to prevent excessive consumption.

CHAPTER 6 - CONCLUSION AND FUTURE SCOPE

Conclusion:

The pac-man python project represents a captivating addition to the world of python-based games. With its iconic gameplay and the perfect balance of challenge and fun, this project delivers an immersive gaming experience suitable for players of all ages. By guiding pac-man through mazes, avoiding ghosts, and gobbling up pellets, the game not only provides entertainment but also engages players in strategic thinking.

Through project analysis, the game core objectives were meticulously defined, including smooth pac-man movement, intelligent ghost, visually appealing mazes, and optimal performance. The projects implementation harnessed the power of python and pygame, resulting in a dynamic and interactive gaming environment.

In the realm of software tools, a range of resources, from code editors to graphic design software, were utilized to bring the pac-man project to life. Unit testing rigorously verified the correctness of individual game components, while integration testing ensured the seamless interaction between these elements, guaranteeing a polished and bug-free gaming experience.

The pac-man python project underscores the important of creating games that not only entertain but also challenge player's strategic thinking and reflexes. Its captivating gameplay, along with its potential for further expansion and development, positions it as a relevant and valuable addition to the ever-evolving gaming landscape. In summary, this project offers an exhilarating blend of entertainment and cognitive engagement, making it a valuable asset for personal enjoyment and educational purposes alike.

Future Scope for the Pac-Man Python Project:

Enhanced Levels: Implement different mazes with varying complexity and challenges to provide players with a more diverse gaming experience.

Multiplayer Mode: Introduce a multiplayer feature that allows players to compete in real-time matches or cooperatively play with others. Add leaderboards to encourage competition and social interactions within the Pac-Man community.

Customization Options: Allow players to personalize their Pac-Man experience by choosing different maze themes, Pac-Man skins, and ghost designs. Provide adjustable difficulty settings to cater to players of all skill levels.

Power-Ups and Boosters: Enhance gameplay by introducing power-ups or boosters that provide Pac-Man with temporary abilities or advantages, adding strategic depth to the game.

High Score Tracking: Expand the high score tracking system to record and display historical data for player progress, encouraging friendly competition among players.

Educational Elements: Include informative pop-ups or mini-games that provide insights into the history of Pac-Man, its impact on gaming, or other educational content related to the game.

Mobile Adaptation: Develop a mobile application version of the Pac-Man game, optimized for smartphones and tablets, to reach a broader audience.

Cross-Platform Compatibility: Ensure that the game is compatible with various devices and operating systems, allowing players to enjoy Pac-Man on their platform of choice.

Accessibility Features: Implement accessibility options to make the game playable for individuals with disabilities, such as screen reader support, colorblind modes, and alternative control options.

Player Feedback Mechanisms: Establish feedback channels to collect user suggestions and bug reports, fostering continuous improvement and user engagement.

Internationalization: Localize the game by offering it in multiple languages, making it accessible and enjoyable for a global audience.

Educational Partnerships: Collaborate with educational institutions to use Pac-Man as a fun and engaging tool for teaching concepts like game design, computer

programming, and problem-solving.

Marketplace Integration: Publish the Pac-Man game on various app marketplaces to increase its visibility and potentially generate revenue through in-app purchases or ads, if applicable.

REFERENCE

*google for the knowledge about the topic and the project

*chatgpt for text references

*Python Documentation

*Pygame Documentation.

*Online Tutorials and

*Educational Resources

*Game Design Books

*Open Source Game Projects

****