P.G.D.A.V. College

University of Delhi

Nehru Nagar, Ring Road, New Delhi – 110065

Website: http://pgdavcollege.in

Email: pgdavcollege.edu@gmail.com

Supporting document

for

Annual Quality Assurance Report, 2022-23

Criteria 1.3.2

Number of courses that include experiential learning through project work/field work/internship during the year

Table of Content

EDUCATIONAL VISIT TO SECURITIES EXCHANGE BOARD OF INDIA (SEBI)

7th September 2022

On September 7th, 2022, COMMERCIUM, the Commerce Society of PGDAV College, under the supervision of Mr. Varun Gautam, Convenor, organized an educational visit to SEBI, with 40 students from B. Com. (Hons) and B. Com. of the third and second year. They were accompanied by four faculty members: Dr Rakesh Kumar, Teacher-in-charge; Dr Rajni Jagota, Ms Gitu Nijhawan and Mr Saurabh. The objective of this visit is to provide valuable insights to our students into the functioning of SEBI and its role as a regulator of the capital market. The group reached its Northern Regional Office, NBCC Complex, East Kidwai Nagar, New Delhi – 110023, at 11 am by bus. The resource persons guided the students to build their capacity as investors and develop awareness to enable them to take an informed investment decision. They provided information about products, players, and participants in the securities market; the concept of mutual funds; regulators in the financial sector; grievance redressal system- SCORES; etc. It was emphasized

that SEBI is “Har Investor ki Taaqat”. The interaction was followed by a Q&A round, which ended at about 1 pm. They arranged delicious Begali sweet boxes. We thanked them for their hospitality. We departed at 1:30 pm for the college. On a whole, the visit was beneficial for students and we can look forward to other such educational and informative trips.

PROJECT REPORT

Group 8

MEMORY GAME

SESSION: MAY-JUNE 2023

दिल्ली विश्वविद्यालय
University of Delhi

62347628 : DISSERTATION / PROJECT WORK
Semester 6
B.A. Programme

Submitted by:

Akshita Dabral	**20053501010**
Shreya Mathur	**20053501020**

Under the Guidance of:

Dr. Geeta Aggarwal
Professor

Department of Computer Science
P. G. D. A. V. College
Nehru Nagar, New Delhi- 110065

P. G. D. A. V. COLLEGE
(UNIVERSITY OF DELHI)
DEPARTMENT OF COMPUTER SCIENCE

Certificate

This is to certify that the project entitled "MEMORY GAME" is a bonafide work carried out by
AKSHITA DABRAL (20053501010) ,
SHREYA MATHUR (20053501020),
as a part of 62347628 : DISSERTATION / PROJECT WORK
in Computer Application during the session of May-June 2023.

DATE

DR. GEETA AGGARWAL
DEPARTMENT OF COMPUTER SCIENCE

Acknowledgement

We would like to thank Prof. Krishna Sharma, Principal P.G.D.A.V College(M).

We are deeply indebted to our mentor Dr. Geeta Aggarwal.

We further want to give a personal Thank You to all the teachers and staff members of the Department of Computer Science & BA Programme. We owe our sincere gratitude towards P.G.D.A.V College(M), University of Delhi.

We also express our deepest gratitude to our parents.

Finally, we would like to wind up by paying our heartfelt thanks to all our near and dear ones.

Akshita Dabral

Shreya Mathur

CONENTS:

INTRODUCTION

The Memory Game is a classic game that has been enjoyed by people of all ages for decades. It is a popular game that game consists of an even number of tiles with images on one side and a generic back. Each image appears on precisely two tiles.

When the game starts, all tiles are turned face down.

The player then flips over two cards, selecting them by clicking on them. If the two tiles have the same image, they remain face up. Otherwise, the tiles flip back over after a small period of time.

The goal of the game is to get all the tiles flipped face up (i.e., find all the matching image pairs) in the least number of tries. That means that lower number of tries are better scores.

The Memory Game project was undertaken as a computer applications project by Vipul jain during their final semester, with the objective of providing an opportunity to learn game development and programming.

OVERVIEW

- The game starts with the set of 4 cards with 2 rows and 4 columns.
- The number of cards will be increased with each new level while the columns reamain same, number of rows increases by 1.
- In the game, user needs to find the similar set of cards.
- Demo and rules for the game are provided on the top left corner of the window.
- After the completion of level 1 user needs to press space to move to the next level.

SOFTWARES USED:

- Visual Studio Code: VS code is streamlined code editor with support of development operations like debugging, task running etc.

PYTHON LIBRARIES USED :

- pygame: a set of Python modules designed for writing video games.
- random: a module that implements pseudo-random number generators for various uses.
- sys: a module that provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

FUCTIONS USED IN CODE:

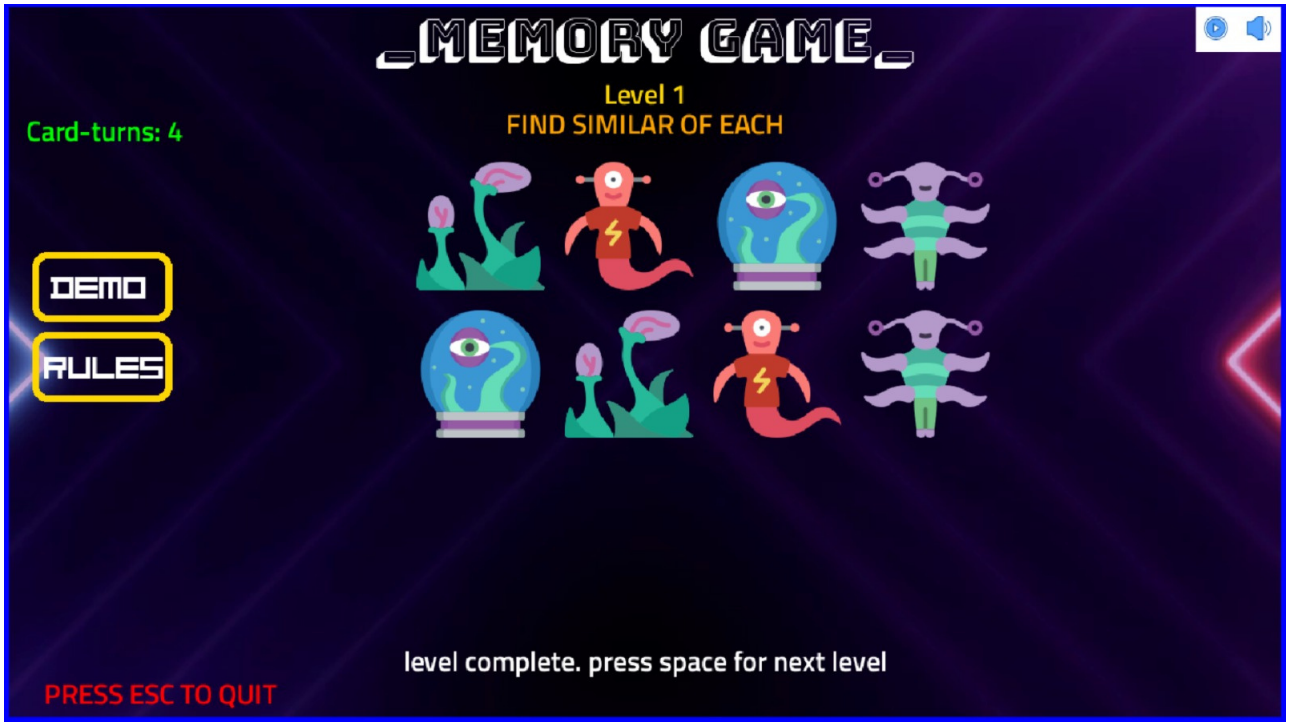
- pygame.init(): This function initializes all the imported pygame modules.
- pygame.display.set_mode(): This function sets the display mode and returns a surface object representing the game window.
- pygame.time.Clock(): This function creates a Clock object that can be used to control the frame rate of the game.
- pygame.font.Font(): This function creates a Font object that can be used to render text in a particular font and size.
- pygame.mixer.Sound(): This function loads a sound file and returns a Sound object that can be used to play the sound.
- pygame.image.load(): This function loads an image file.
- random(): This function chooses a random cards from a sequence.
- Menubar(): This fuction is called when the user clicks on rules button to show rules tab.

- pygame.Rect(): This function creates a new Rect object with the specified coordinates and dimensions.
- pygame.sprite.Sprite(): This function creates a new Sprite object for tiles.
- Font.render(): This function renders the specified text using the font and size of the Font object and returns a Surface object representing the rendered text.
- Rect.collidepoint(): This function checks if the given point is inside the Rect object.
- pygame.draw.rect(): This function draws a rectangle on the surface object.
- Screen.blit(): This function copies the contents of one surface to another surface.
- Button.collision(): This function checks if the mouse cursor is over the button and returns True if the button is clicked.
- Demo(): This fuction is called when the user clicks on demo button to show demo tab.

OUTPUT SCREENSHOTS:

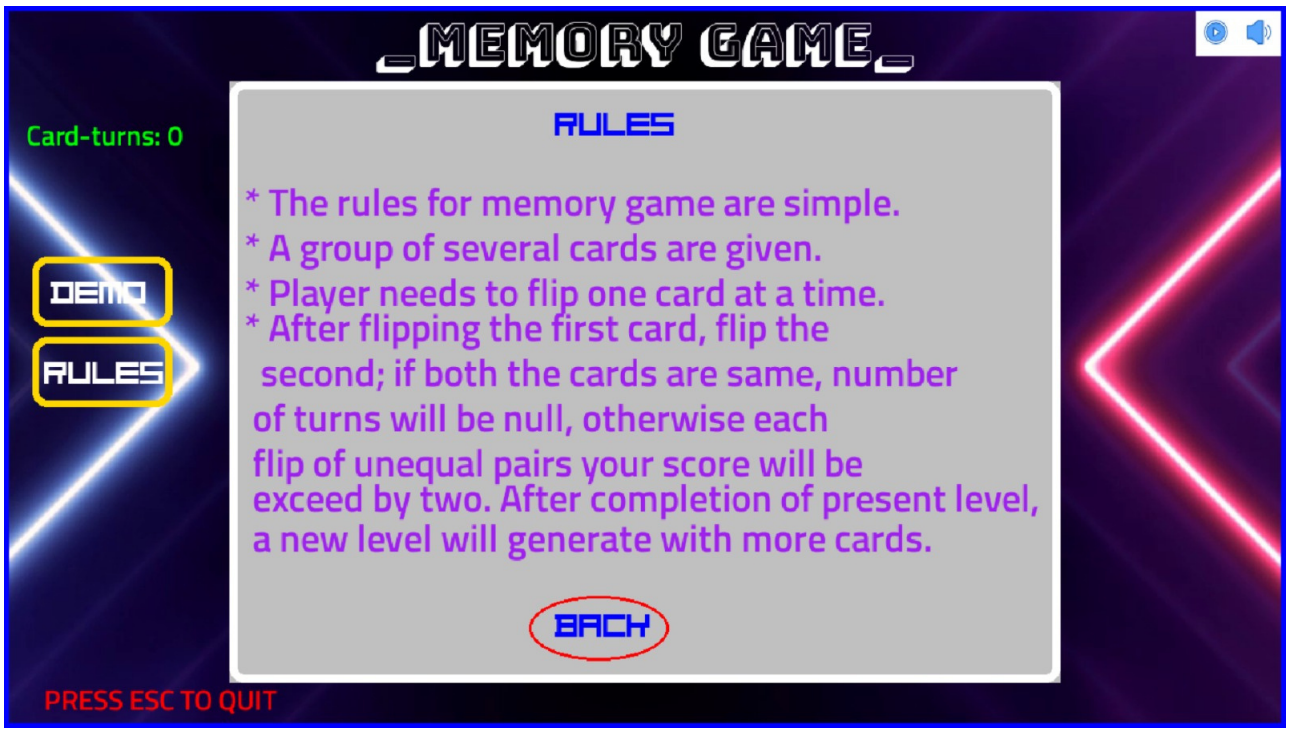
MAIN WINDOW

LEVEl COMPLETE



OUTPUT SCREENSHOTS:

RULES



DEMO

BIBLIOGRAPHY:

- Programming and Problem Solving with PYTHON by Ashok Namdev Kamthane and Amit Ashok Kamthane
- Python for Everybody by Charles R Severance

REFERENCES:

- Youtube channel "CodeWithHarry"
- https://youtu.be/Xpvf9lwRERU
- Website stackoverflow
- Website w3schools
- Website geekforgeeks

असतो मा सद्गमय

PGDAV COLLEGE
UNIVERSITY OF DELHI

PROJECT REPORT

In

Computer Applications

Submitted By:

Aman (20053501122)

Jatin Jangra (20053501014)

Under the Supervision of:

Dr. Geeta Aggarwal

P.G.D.A.V. College, University of Delhi

Certificate of Completion

This certifies that **Aman, Jatin Jangra** have successfully completed the project **Pacman Game** using Python and Pygame. The project was completed on **25 April 2023**, under my guidance.

Dr. Geeta Aggarwal
(Associate Professor)
P.G.D.A.V. College.

Acknowledgements

I would like to express my sincere gratitude to the following individuals who have made valuable contributions to the successful completion of this project (**Pacman Game**).

I am deeply grateful to my project supervisor, **Dr. Geeta Aggarwal**, for her constant guidance, support, and valuable feedback throughout the project. Her insights and expertise in the field of **Game Development & Computer Programming** have been instrumental in shaping the direction and scope of the project.

I would also like to express my appreciation to **P.G.D.A.V. College, University of Delhi**, for providing the necessary resources and facilities to carry out this project.

Without the help and support of these individuals and institutions, this project would not have been possible. Thank you all for your contributions.

Aman, Jatin Jangra
(P.G.D.A.V. College)

Table of Contents

Introduction

The Pacman Game is a classic arcade game that has been enjoyed by people of all ages for decades. It is a popular game that involves navigating a yellow, circular character through a maze-like board, collecting pellets, fruits, and power-ups while avoiding enemy ghosts.

The Pacman Game project was undertaken as a computer applications project by Aman and Jatin Jangra during their final semester. The aim of the project was to develop a game/application using Python and Pygame, with the objective of providing an opportunity to learn game development and programming.

The project involved creating a tile-based board, designing a Pacman player, and developing four ghosts named Inky, Blinky, Pinky, and Clyde, each with their specific chase patterns. The game was designed with two levels, and the player had six lives, with power-ups to assist them in advancing through the levels. The target audience for the game was children and students who are interested in learning game development and programming.

This report presents a detailed overview of the design, implementation, and challenges faced during the development of the Pacman Game project. The report also highlights the various features and functionalities of the game, and its potential for learning and entertainment.

Design and Implementation

The Pacman Game project was developed using Python language and the Pygame library. The project includes a tile-based game board that was designed from scratch. The board includes walls, pellets, power pellets, and obstacles that Pacman and ghosts must navigate. The game also includes a Pacman player and four ghosts, each with a unique chase pattern: Inky, Blinky, Pinky, and Clyde. The game has two game boards, each with a different level of difficulty. The game has a joystick-like control system with six player lives and power-ups that can be collected by the player.

The project's target audience is children and students interested in learning programming and game development. The main goal of the project was to develop an entertaining and engaging game that could also serve as an educational tool for beginner programmers. The game's design and implementation were focused on creating a challenging and interactive experience for players.

During the design and implementation process, several challenges were encountered. The board design was particularly challenging due to the complexity of the layout, which included walls and obstacles in various shapes and sizes. The chase patterns of the ghosts were also a challenge, as each ghost had to follow a unique path based on Pacman's location on the board. Additionally, the implementation of power-ups required careful consideration to ensure that they provided a challenging and balanced gameplay experience.

To overcome these challenges, the project team spent considerable time researching and experimenting with different design and implementation approaches. The team also utilized iterative development techniques, which involved frequent testing and feedback to refine the game's design and implementation.

In conclusion, the Pacman Game project was a challenging and rewarding experience for the project team. The game's design and implementation required a significant amount of time and effort, but the end result was a fun and engaging game that can serve as an educational tool for beginner programmers. The project team hopes that the game will inspire and motivate others to explore the world of programming and game development

Player

The Pacman player is the central character in the Pacman Game project. It is a yellow, circular character that moves around the game board, collecting pellets and other objects while avoiding the enemy ghosts.

In the Pacman Game project, the Pacman player was designed using Python and Pygame. It was developed to move around the board using joystick type controls, with the arrow keys on the keyboard used to change the direction of movement. The player had six lives, and each time it collided with a ghost, it lost one life.

The Pacman player also had the ability to collect power-ups, which provided temporary invincibility and the ability to eat the ghosts. The power-ups were represented by large dots located at various points on the game board.

Overall, the design and implementation of the Pacman player was a critical component of the project, and it added to the game's fun and entertaining nature. The player's movement, lives, and power-ups were all carefully crafted to ensure that the game was challenging yet enjoyable for the target audience.

Ghosts

The ghosts are the antagonists in the Pacman Game project. They are four in number and have unique personalities and chase patterns. The four ghosts are named Inky, Blinky, Pinky, and Clyde.

Inky is the blue ghost who always tries to ambush the player by taking a position behind them. Blinky, the red ghost, is the most aggressive and always tries to move towards the player's location. Pinky, the pink ghost, always tries to get ahead of the player and ambush them from the front. Lastly, Clyde, the orange ghost, behaves somewhat erratically and changes direction randomly.

The chase patterns of the ghosts were carefully designed to make the game more challenging and exciting. Each ghost had its specific behaviour and followed the player in a different way. The ghost’s movement was dependent on the distance between themselves and the player's location. For example, Blinky always tried

to move closer to the player, while Inky would try to predict where the player would move next and head towards that location.

Overall, the implementation of the ghosts and their chase patterns added to the game's excitement and difficulty level, providing players with a challenging and thrilling gaming experience.

Features

The Pacman Game project was developed using the Python programming language and the Pygame library. The game included various features and components, including:

Tile-based board: The game featured a tile-based board that was designed and implemented entirely using Pygame. The board served as the backdrop for the game and provided a structured environment for the player and ghosts to move around.

Pacman player: The player was a yellow circular character that the user controlled using joystick-type controls. The player could move horizontally or vertically across the board, eating pellets and power-ups while avoiding the ghosts.

- **Ghosts**: The game had four ghosts named Inky, Blinky, Pinky, and Clyde. Each ghost had a unique personality and behaviour, making the game more challenging and exciting.
- **Power-ups**: The game featured power-ups that the player could collect to gain temporary invincibility and the ability to eat the ghosts.
- **Joystick controls**: The game was controlled using joystick-like controls, which made it more intuitive and fun to play.
- **Game boards**: The game had two game boards that worked at different levels. Each board was more challenging than the previous one, making the game more exciting as the player progressed.
- **Score tracking**: The game kept track of the player's score as they collected pellets, power-ups, and other items on the board. The score was displayed in real-time on the game screen, adding to the game's excitement.

- **Game Over screen**: If the player lost all their lives or failed to complete the game objective, the game displayed a Game Over screen with the player's final score. The screen also included options to restart the game or exit the game altogether.
- **Two game levels**: The Pacman Game project included two levels, each with a unique game board design and set of challenges. The second level was unlocked only after the player successfully completed the first level, adding to the game's replay value and providing a sense of achievement for the player. The two levels were designed to be progressively more challenging, providing a more engaging and rewarding gaming experience for the player.

The implementation of these features made the game more challenging, exciting, and fun to play.

Software Used

- Python Programming Language (Version 3.x)
- Pygame Library (Version 2.0.1)
- Visual Studio Code (Version 1.55.0)
- Canva for designing the start page

Hardware Requirements

- Processor: A 1.6 GHz or faster processor is recommended to ensure smooth performance of the game.
- RAM: A minimum of 2 GB of RAM is recommended to run the game without any lag or crashes.
- Graphics Card: A dedicated graphics card is not mandatory, but it is recommended for better graphics performance.
- Sound Card: A sound card is recommended to enjoy the game with its sound effects and background music.

- Display: The game can be played on any display device, but a higher resolution display is recommended for a better visual experience.

Please note that these requirements may vary depending on the complexity of the game board and the level of graphics used in the game.

Installation Instructions

- Install Python 3.x on your computer from the official Python website (https://www.python.org/downloads/)
- Install Pygame Library 2.0.1 using pip command in the command prompt: pip install pygame==2.0.1
- Install Visual Studio Code on your computer from the official Visual Studio Code website (https://code.visualstudio.com/)

Once the above software and requirements are installed, you can run the Pacman Game project by opening the project directory in Visual Studio Code and executing the "main.py" file. The game can be played using the arrow keys on your keyboard.

Platform

The project was developed and tested on a **Windows 11** and **Mac** operating system.

- **Python Version:** The project was developed using Python 3.x.
- **Pygame Version**: The project was developed using Pygame library version 2.0.1.
- **Modules Used**: The project used several built-in Python modules such as **time**, and os modules. Additionally, it used Pygame modules such as **pygame.sprite**, **pygame.image**, and **pygame.mixer**.

Package	Version
Autopep8	2.0.1
Freegames	2.5.3
Pip	22.3.1
Pycodestyle	2.10.0
Pygame	2.1.3.dev8
Setuptools	65.5.0

Coding

Startpage.py

```
import pygame

# Initialize Pygame
pygame.init()

# Set window dimensions
window_width = 900
window_height = 950

back_music = pygame.mixer.Sound('resources/sounds/pacman_beginning.wav')
back_music.set_volume(0.5)

# Set colors
black = (0, 0, 0)
yellow = (255, 255, 0)

# Load the background image
background_image = pygame.image.load('resources/images/newbck.png')

# Create the game window
window = pygame.display.set_mode((window_width, window_height))
pygame.display.set_caption("Pacman")

# Define the start button
```

```
start_button = pygame.Rect((window_width/3), (window_height/1.3), 300, 100)

# Set the blinking parameters
blink_speed = 500  # milliseconds
last_blink_time = 0
blink_on = True

# Set the cursor shapes
cursor_normal = pygame.mouse.get_cursor()
cursor_hover = pygame.cursors.tri_left

# Create a font object for the button text
font = pygame.font.SysFont('Arial', 20)

# Main game loop
while True:
    # Check for events
    back_music.play()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        elif event.type == pygame.MOUSEBUTTONDOWN:
            # Check if the start button was clicked
            if start_button.collidepoint(event.pos):
                import maingame
                back_music.stop()

    # Blit the background image onto the screen
    window.blit(background_image, (0, 0))
```

```
    # Draw the start button
    pygame.draw.rect(window, black, start_button)

    # Draw the button text
    current_time = pygame.time.get_ticks()
    if current_time - last_blink_time > blink_speed:
        last_blink_time = current_time
        blink_on = not blink_on
    text_surface = font.render('Click here to Start', True, black if blink_on else yellow)
    text_rect = text_surface.get_rect(center=start_button.center)
    window.blit(text_surface, text_rect)

    # Change cursor shape when hovering over button
    if start_button.collidepoint(pygame.mouse.get_pos()):
        pygame.mouse.set_cursor(*cursor_hover)
    else:
        pygame.mouse.set_cursor(*cursor_normal)

    # Update the screen
    pygame.display.update()
```

Board.py

```
# 0 = empty black rectangle, 1 = dot, 2 = big dot, 3 = vertical line,
# 4 = horizontal line, 5 = top right, 6 = top left, 7 = bot left, 8 = bot right
# 9 = gate
boards2 = [
[6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5],
[3, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3],
[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],
[3, 3, 1, 6, 4, 4, 5, 1, 6, 4, 4, 4, 5, 1, 3, 3, 1, 6, 4, 4, 4, 5, 1, 6, 4, 4, 5, 1, 3, 3],
```

```
[3, 3, 2, 3, 0, 0, 3, 1, 3, 0, 0, 0, 3, 1, 3, 3, 1, 3, 0, 0, 0, 3, 1, 3, 0, 0, 3, 2, 3, 3],
[3, 3, 1, 7, 4, 4, 8, 1, 7, 4, 4, 4, 8, 1, 7, 8, 1, 7, 4, 4, 4, 8, 1, 7, 4, 4, 8, 1, 3, 3],
[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],
[3, 3, 1, 6, 4, 4, 5, 1, 6, 5, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 6, 5, 1, 6, 4, 4, 5, 1, 3, 3],
[3, 3, 1, 7, 4, 4, 8, 1, 3, 3, 1, 7, 4, 4, 5, 6, 4, 4, 8, 1, 3, 3, 1, 7, 4, 4, 8, 1, 3, 3],
[3, 3, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 3, 3],
[3, 7, 4, 4, 4, 4, 5, 1, 3, 7, 4, 4, 5, 1, 3, 3, 1, 6, 4, 4, 8, 3, 1, 6, 4, 4, 4, 4, 8, 3],
[3, 0, 0, 0, 0, 0, 3, 1, 3, 6, 4, 4, 8, 1, 7, 8, 1, 7, 4, 4, 5, 3, 1, 3, 0, 0, 0, 0, 0, 3],
[3, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 3],
[8, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 6, 4, 4, 9, 9, 4, 4, 5, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 7],
[4, 4, 4, 4, 4, 4, 8, 1, 7, 8, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 7, 8, 1, 7, 4, 4, 4, 4, 4, 4],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[4, 4, 4, 4, 4, 4, 5, 1, 6, 5, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 6, 5, 1, 6, 4, 4, 4, 4, 4, 4],
[5, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 7, 4, 4, 4, 4, 4, 4, 8, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 6],
[3, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 3],
[3, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 3],
[3, 6, 4, 4, 4, 4, 8, 1, 7, 8, 1, 7, 4, 4, 5, 6, 4, 4, 8, 1, 7, 8, 1, 7, 4, 4, 4, 4, 5, 3],
[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],
[3, 3, 1, 6, 4, 4, 5, 1, 6, 4, 4, 4, 5, 1, 3, 3, 1, 6, 4, 4, 4, 5, 1, 6, 4, 4, 5, 1, 3, 3],
[3, 3, 1, 7, 4, 5, 3, 1, 7, 4, 4, 4, 8, 1, 7, 8, 1, 7, 4, 4, 4, 8, 1, 3, 6, 4, 8, 1, 3, 3],
[3, 3, 2, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 2, 3, 3],
[3, 7, 4, 5, 1, 3, 3, 1, 6, 5, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 6, 5, 1, 3, 3, 1, 6, 4, 8, 3],
[3, 6, 4, 8, 1, 7, 8, 1, 3, 3, 1, 7, 4, 4, 5, 6, 4, 4, 8, 1, 3, 3, 1, 7, 8, 1, 7, 4, 5, 3],
[3, 3, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 3, 3],
[3, 3, 1, 6, 4, 4, 4, 4, 8, 7, 4, 4, 5, 1, 3, 3, 1, 6, 4, 4, 8, 7, 4, 4, 4, 4, 5, 1, 3, 3],
[3, 3, 1, 7, 4, 4, 4, 4, 4, 4, 4, 4, 8, 1, 7, 8, 1, 7, 4, 4, 4, 4, 4, 4, 4, 4, 8, 1, 3, 3],
[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],
[3, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8, 3],
[7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8]
    ]
boards1 =[
```

[6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5],

[3, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3],

[3, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3],

[3, 3, 1, 6, 4, 4, 4, 4, 5, 1, 6, 4, 4, 5, 1, 1, 6, 4, 4, 5, 1, 6, 4, 4, 4, 4, 5, 1, 3, 3],

[3, 3, 1, 3, 0, 0, 0, 0, 3, 1, 7, 4, 5, 3, 1, 1, 3, 6, 4, 8, 1, 3, 0, 0, 0, 0, 3, 1, 3, 3],

[3, 3, 1, 7, 4, 4, 4, 4, 8, 1, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 1, 7, 4, 4, 4, 4, 8, 1, 3, 3],

[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],

[3, 3, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 3, 3, 1, 1, 3, 3, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 3, 3],

[3, 3, 1, 7, 4, 4, 4, 4, 4, 4, 8, 1, 3, 3, 1, 1, 3, 3, 1, 7, 4, 4, 4, 4, 4, 4, 8, 1, 3, 3],

[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],

[3, 3, 1, 1, 6, 4, 4, 4, 4, 4, 4, 4, 8, 3, 1, 1, 3, 7, 4, 4, 4, 4, 4, 4, 4, 5, 1, 1, 3, 3],

[3, 3, 1, 1, 7, 4, 4, 4, 4, 4, 4, 4, 4, 8, 1, 1, 7, 4, 4, 4, 4, 4, 4, 4, 4, 8, 1, 1, 3, 3],

[3, 7, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 8, 3],

[8, 0, 3, 1, 6, 4, 4, 4, 4, 5, 1, 6, 4, 4, 9, 9, 4, 4, 5, 1, 6, 4, 4, 4, 4, 5, 1, 3, 0, 7],

[4, 4, 8, 1, 3, 6, 4, 4, 5, 3, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 3, 6, 4, 4, 5, 3, 1, 7, 4, 4],

[0, 0, 0, 1, 3, 3, 1, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 1, 3, 3, 1, 0, 0, 0],

[4, 4, 5, 1, 3, 3, 1, 1, 3, 3, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 3, 3, 1, 1, 3, 3, 1, 6, 4, 4],

[5 ,0, 3, 1, 7, 8, 1, 1, 7, 8, 1, 3, 0, 0, 0, 0, 0, 0, 3, 1, 7, 8, 1, 1, 7, 8, 1, 3, 0, 6],

[3, 6, 8, 1, 1, 1, 1, 1, 1, 1, 7, 4, 4, 4, 4, 4, 4, 8, 1, 1, 1, 1, 1, 1, 1, 1, 7, 5, 3],

[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],

[3, 3, 1, 1, 6, 4, 4, 4, 4, 4, 4, 4, 4, 5, 1, 1, 6, 4, 4, 4, 4, 4, 4, 4, 4, 5, 1, 1, 3, 3],

[3, 3, 1, 1, 7, 4, 4, 4, 4, 4, 4, 4, 5, 3, 1, 1, 3, 6, 4, 4, 4, 4, 4, 4, 4, 8, 1, 1, 3, 3],

[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],

[3, 3, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 3, 3, 1, 1, 7, 8, 1, 6, 4, 4, 4, 4, 4, 4, 5, 1, 3, 3],

[3, 3, 1, 7, 4, 4, 4, 4, 4, 4, 8, 1, 3, 3, 0, 0, 1, 1, 1, 7, 4, 4, 4, 4, 4, 4, 8, 1, 3, 3],

[3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3],

[3, 3, 1, 6, 4, 4, 4, 4, 5, 1, 1, 1, 3, 3, 1, 1, 6, 5, 1, 1, 1, 6, 4, 4, 4, 4, 5, 1, 3, 3],

[3, 3, 1, 3, 0, 0, 0, 0, 3, 1, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 1, 3, 0, 0, 0, 0, 3, 1, 3, 3],

[3, 3, 1, 3, 0, 0, 0, 0, 3, 1, 6, 4, 8, 3, 1, 1, 3, 7, 4, 5, 1, 3, 0, 0, 0, 0, 3, 1, 3, 3],

[3, 3, 1, 7, 4, 4, 4, 4, 8, 1, 7, 4, 4, 8, 1, 1, 7, 4, 4, 8, 1, 7, 4, 4, 4, 4, 8, 1, 3, 3],

[3, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3],

```
[3, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8, 3],
[7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8]
    ]
```

Maingame.py

```
# Build Pac-Man from Scratch in Python with PyGame!!
import copy
import board

import pygame
import math
boards = [[board.boards1,board.boards2]]
boards1 = 0
boards2 = 0
pygame.init()

WIDTH = 900
HEIGHT = 950
screen = pygame.display.set_mode([WIDTH, HEIGHT])
timer = pygame.time.Clock()
fps = 60
font = pygame.font.Font('freesansbold.ttf', 20)
level = copy.deepcopy(boards[boards1][boards2])
color = 'blue'
PI = math.pi
player_images = []
for i in range(1, 5):

player_images.append(pygame.transform.scale(pygame.image.load(f'resources/images/player/{i}.png'), (45,45)))
```

```
blinky_img = pygame.transform.scale(pygame.image.load(f'resources/images/ghosts/red.png'), (45,45))

inky_img = pygame.transform.scale(pygame.image.load(f'resources/images/ghosts/blue1.png'), (45,45))

pinky_img = pygame.transform.scale(pygame.image.load(f'resources/images/ghosts/pink.png'), (45,45))

clyde_img = pygame.transform.scale(pygame.image.load(f'resources/images/ghosts/orange.png'), (45,45))

spooked_img = pygame.transform.scale(pygame.image.load(f'resources/images/ghosts/powerup.png'), (45,45))

dead_img = pygame.transform.scale(pygame.image.load(f'resources/images/ghosts/dead.png'), (45,45))

eat_sound = pygame.mixer.Sound("resources\sounds\pacman_death.wav")

eat_sound.set_volume(0.1)

eat_fruit = pygame.mixer.Sound("resources\sounds\pacman_eatfruit.wav")

eat_fruit.set_volume(0.1)



player_x = 450

player_y = 663

direction = 0

blinky_x = 56

blinky_y = 58

blinky_direction = 0

inky_x = 440

inky_y = 388

inky_direction = 2

pinky_x = 440

pinky_y = 438

pinky_direction = 2

clyde_x = 420
```

```
clyde_y = 438
clyde_direction = 2
counter = 0
flicker = False
# R, L, U, D
turns_allowed = [False, False, False, False]
direction_command = 0
player_speed = 2
score = 0
powerup = False
power_counter = 0
eaten_ghost = [False, False, False, False]
targets = [(player_x, player_y), (player_x, player_y), (player_x, player_y), (player_x, player_y)]
blinky_dead = False
inky_dead = False
clyde_dead = False
pinky_dead = False
blinky_box = False
inky_box = False
clyde_box = False
pinky_box = False
moving = False
ghost_speeds = [2, 2, 2, 2]
startup_counter = 0
lives = 6
game_over = False
game_won = False



class Ghost:
    def __init__(self, x_coord, y_coord, target, speed, img, direct, dead, box, id):
```

```
    self.x_pos = x_coord
    self.y_pos = y_coord
    self.center_x = self.x_pos + 22
    self.center_y = self.y_pos + 22
    self.target = target
    self.speed = speed
    self.img = img
    self.direction = direct
    self.dead = dead
    self.in_box = box
    self.id = id
    self.turns, self.in_box = self.check_collisions()
    self.rect = self.draw()

def draw(self):
    if (not powerup and not self.dead) or (eaten_ghost[self.id] and powerup and not self.dead):
        screen.blit(self.img, (self.x_pos, self.y_pos))
    elif powerup and not self.dead and not eaten_ghost[self.id]:
        screen.blit(spooked_img, (self.x_pos, self.y_pos))
    else:
        screen.blit(dead_img, (self.x_pos, self.y_pos))
    ghost_rect = pygame.rect.Rect((self.center_x - 18, self.center_y - 18), (36, 36))
    return ghost_rect

def check_collisions(self):
    # R, L, U, D
    num1 = ((HEIGHT - 50) // 32)
    num2 = (WIDTH // 30)
    num3 = 15
    self.turns = [False, False, False, False]
    if 0 < self.center_x // 30 < 29:
```

```
if level[(self.center_y - num3) // num1][self.center_x // num2] == 9:
    self.turns[2] = True
if level[self.center_y // num1][(self.center_x - num3) // num2] < 3 \
        or (level[self.center_y // num1][(self.center_x - num3) // num2] == 9 and (
        self.in_box or self.dead)):
    self.turns[1] = True
if level[self.center_y // num1][(self.center_x + num3) // num2] < 3 \
        or (level[self.center_y // num1][(self.center_x + num3) // num2] == 9 and (
        self.in_box or self.dead)):
    self.turns[0] = True
if level[(self.center_y + num3) // num1][self.center_x // num2] < 3 \
        or (level[(self.center_y + num3) // num1][self.center_x // num2] == 9 and (
        self.in_box or self.dead)):
    self.turns[3] = True
if level[(self.center_y - num3) // num1][self.center_x // num2] < 3 \
        or (level[(self.center_y - num3) // num1][self.center_x // num2] == 9 and (
        self.in_box or self.dead)):
    self.turns[2] = True

if self.direction == 2 or self.direction == 3:
    if 12 <= self.center_x % num2 <= 18:
        if level[(self.center_y + num3) // num1][self.center_x // num2] < 3 \
                or (level[(self.center_y + num3) // num1][self.center_x // num2] == 9 and (
                self.in_box or self.dead)):
            self.turns[3] = True
        if level[(self.center_y - num3) // num1][self.center_x // num2] < 3 \
                or (level[(self.center_y - num3) // num1][self.center_x // num2] == 9 and (
                self.in_box or self.dead)):
            self.turns[2] = True
    if 12 <= self.center_y % num1 <= 18:
        if level[self.center_y // num1][(self.center_x - num2) // num2] < 3 \
```

```
                or (level[self.center_y // num1][(self.center_x - num2) // num2] == 9 and (
                self.in_box or self.dead)):
            self.turns[1] = True
        if level[self.center_y // num1][(self.center_x + num2) // num2] < 3 \
                or (level[self.center_y // num1][(self.center_x + num2) // num2] == 9 and (
                self.in_box or self.dead)):
            self.turns[0] = True

    if self.direction == 0 or self.direction == 1:
        if 12 <= self.center_x % num2 <= 18:
            if level[(self.center_y + num3) // num1][self.center_x // num2] < 3 \
                    or (level[(self.center_y + num3) // num1][self.center_x // num2] == 9 and (
                    self.in_box or self.dead)):
                self.turns[3] = True
            if level[(self.center_y - num3) // num1][self.center_x // num2] < 3 \
                    or (level[(self.center_y - num3) // num1][self.center_x // num2] == 9 and (
                    self.in_box or self.dead)):
                self.turns[2] = True
        if 12 <= self.center_y % num1 <= 18:
            if level[self.center_y // num1][(self.center_x - num3) // num2] < 3 \
                    or (level[self.center_y // num1][(self.center_x - num3) // num2] == 9 and (
                    self.in_box or self.dead)):
                self.turns[1] = True
            if level[self.center_y // num1][(self.center_x + num3) // num2] < 3 \
                    or (level[self.center_y // num1][(self.center_x + num3) // num2] == 9 and (
                    self.in_box or self.dead)):
                self.turns[0] = True
else:
    self.turns[0] = True
    self.turns[1] = True
if 350 < self.x_pos < 550 and 370 < self.y_pos < 480:
```

```
            self.in_box = True
        else:
            self.in_box = False
        return self.turns, self.in_box


    def move_clyde(self):
        # r, l, u, d
        # clyde is going to turn whenever advantageous for pursuit
        if self.direction == 0:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.x_pos += self.speed
            elif not self.turns[0]:
                if self.target[1] > self.y_pos and self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.target[1] < self.y_pos and self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.target[0] < self.x_pos and self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
            elif self.turns[0]:
```

```
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            if self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            else:
                self.x_pos += self.speed
        elif self.direction == 1:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.x_pos -= self.speed
            elif not self.turns[1]:
                if self.target[1] > self.y_pos and self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.target[1] < self.y_pos and self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.target[0] > self.x_pos and self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
                elif self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.turns[0]:
                    self.direction = 0
```

```
                self.x_pos += self.speed
            elif self.turns[1]:
                if self.target[1] > self.y_pos and self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                if self.target[1] < self.y_pos and self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                else:
                    self.x_pos -= self.speed
        elif self.direction == 2:
            if self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif not self.turns[2]:
                if self.target[0] > self.x_pos and self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
                elif self.target[0] < self.x_pos and self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.target[1] > self.y_pos and self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.turns[3]:
```

```
                self.direction = 3
                self.y_pos += self.speed
            elif self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
        elif self.turns[2]:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            else:
                self.y_pos -= self.speed
    elif self.direction == 3:
        if self.target[1] > self.y_pos and self.turns[3]:
            self.y_pos += self.speed
        elif not self.turns[3]:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[1]:
```

```
                self.direction = 1
                self.x_pos -= self.speed
            elif self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
        elif self.turns[3]:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            else:
                self.y_pos += self.speed
    if self.x_pos < -30:
        self.x_pos = 900
    elif self.x_pos > 900:
        self.x_pos - 30
    return self.x_pos, self.y_pos, self.direction

def move_blinky(self):
    # r, l, u, d
    # blinky is going to turn whenever colliding with walls, otherwise continue straight
    if self.direction == 0:
        if self.target[0] > self.x_pos and self.turns[0]:
            self.x_pos += self.speed
        elif not self.turns[0]:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
```

```
            self.direction = 2
            self.y_pos -= self.speed
        elif self.target[0] < self.x_pos and self.turns[1]:
            self.direction = 1
            self.x_pos -= self.speed
        elif self.turns[3]:
            self.direction = 3
            self.y_pos += self.speed
        elif self.turns[2]:
            self.direction = 2
            self.y_pos -= self.speed
        elif self.turns[1]:
            self.direction = 1
            self.x_pos -= self.speed
    elif self.turns[0]:
        self.x_pos += self.speed
elif self.direction == 1:
    if self.target[0] < self.x_pos and self.turns[1]:
        self.x_pos -= self.speed
    elif not self.turns[1]:
        if self.target[1] > self.y_pos and self.turns[3]:
            self.direction = 3
            self.y_pos += self.speed
        elif self.target[1] < self.y_pos and self.turns[2]:
            self.direction = 2
            self.y_pos -= self.speed
        elif self.target[0] > self.x_pos and self.turns[0]:
            self.direction = 0
            self.x_pos += self.speed
        elif self.turns[3]:
            self.direction = 3
```

```
            self.y_pos += self.speed
        elif self.turns[2]:
            self.direction = 2
            self.y_pos -= self.speed
        elif self.turns[0]:
            self.direction = 0
            self.x_pos += self.speed
    elif self.turns[1]:
        self.x_pos -= self.speed
elif self.direction == 2:
    if self.target[1] < self.y_pos and self.turns[2]:
        self.direction = 2
        self.y_pos -= self.speed
    elif not self.turns[2]:
        if self.target[0] > self.x_pos and self.turns[0]:
            self.direction = 0
            self.x_pos += self.speed
        elif self.target[0] < self.x_pos and self.turns[1]:
            self.direction = 1
            self.x_pos -= self.speed
        elif self.target[1] > self.y_pos and self.turns[3]:
            self.direction = 3
            self.y_pos += self.speed
        elif self.turns[3]:
            self.direction = 3
            self.y_pos += self.speed
        elif self.turns[0]:
            self.direction = 0
            self.x_pos += self.speed
        elif self.turns[1]:
            self.direction = 1
```

```
            self.x_pos -= self.speed
        elif self.turns[2]:
            self.y_pos -= self.speed
    elif self.direction == 3:
        if self.target[1] > self.y_pos and self.turns[3]:
            self.y_pos += self.speed
        elif not self.turns[3]:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
        elif self.turns[3]:
            self.y_pos += self.speed
    if self.x_pos < -30:
        self.x_pos = 900
    elif self.x_pos > 900:
        self.x_pos - 30
```

```
    return self.x_pos, self.y_pos, self.direction


def move_inky(self):
    # r, l, u, d
    # inky turns up or down at any point to pursue, but left and right only on collision
    if self.direction == 0:
        if self.target[0] > self.x_pos and self.turns[0]:
            self.x_pos += self.speed
        elif not self.turns[0]:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
        elif self.turns[0]:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
```

```
            if self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            else:
                self.x_pos += self.speed
        elif self.direction == 1:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.x_pos -= self.speed
            elif not self.turns[1]:
                if self.target[1] > self.y_pos and self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.target[1] < self.y_pos and self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.target[0] > self.x_pos and self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
                elif self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
            elif self.turns[1]:
                if self.target[1] > self.y_pos and self.turns[3]:
```

```
                self.direction = 3
                self.y_pos += self.speed
            if self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            else:
                self.x_pos -= self.speed
        elif self.direction == 2:
            if self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif not self.turns[2]:
                if self.target[0] > self.x_pos and self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
                elif self.target[0] < self.x_pos and self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.target[1] > self.y_pos and self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.turns[3]:
                    self.direction = 3
                    self.y_pos += self.speed
                elif self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
            elif self.turns[2]:
```

```
            self.y_pos -= self.speed
        elif self.direction == 3:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.y_pos += self.speed
            elif not self.turns[3]:
                if self.target[0] > self.x_pos and self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
                elif self.target[0] < self.x_pos and self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.target[1] < self.y_pos and self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.turns[2]:
                    self.direction = 2
                    self.y_pos -= self.speed
                elif self.turns[1]:
                    self.direction = 1
                    self.x_pos -= self.speed
                elif self.turns[0]:
                    self.direction = 0
                    self.x_pos += self.speed
            elif self.turns[3]:
                self.y_pos += self.speed
        if self.x_pos < -30:
            self.x_pos = 900
        elif self.x_pos > 900:
            self.x_pos - 30
        return self.x_pos, self.y_pos, self.direction
```

```
def move_pinky(self):
    # r, l, u, d
    # inky is going to turn left or right whenever advantageous, but only up or down on collision
    if self.direction == 0:
        if self.target[0] > self.x_pos and self.turns[0]:
            self.x_pos += self.speed
        elif not self.turns[0]:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
        elif self.turns[0]:
            self.x_pos += self.speed
    elif self.direction == 1:
        if self.target[1] > self.y_pos and self.turns[3]:
            self.direction = 3
        elif self.target[0] < self.x_pos and self.turns[1]:
```

```
            self.x_pos -= self.speed
        elif not self.turns[1]:
            if self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
        elif self.turns[1]:
            self.x_pos -= self.speed
    elif self.direction == 2:
        if self.target[0] < self.x_pos and self.turns[1]:
            self.direction = 1
            self.x_pos -= self.speed
        elif self.target[1] < self.y_pos and self.turns[2]:
            self.direction = 2
            self.y_pos -= self.speed
        elif not self.turns[2]:
            if self.target[0] > self.x_pos and self.turns[0]:
```

```
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.target[1] > self.y_pos and self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.turns[3]:
                self.direction = 3
                self.y_pos += self.speed
            elif self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
        elif self.turns[2]:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            else:
                self.y_pos -= self.speed
    elif self.direction == 3:
        if self.target[1] > self.y_pos and self.turns[3]:
            self.y_pos += self.speed
        elif not self.turns[3]:
            if self.target[0] > self.x_pos and self.turns[0]:
```

```
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.target[1] < self.y_pos and self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[2]:
                self.direction = 2
                self.y_pos -= self.speed
            elif self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            elif self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
        elif self.turns[3]:
            if self.target[0] > self.x_pos and self.turns[0]:
                self.direction = 0
                self.x_pos += self.speed
            elif self.target[0] < self.x_pos and self.turns[1]:
                self.direction = 1
                self.x_pos -= self.speed
            else:
                self.y_pos += self.speed
    if self.x_pos < -30:
        self.x_pos = 900
    elif self.x_pos > 900:
        self.x_pos - 30
    return self.x_pos, self.y_pos, self.direction
```

```
def draw_misc():
    score_text = font.render(f'Score: {score}', True, 'white')
    screen.blit(score_text, (10, 920))
    if powerup:
        eat_fruit.play()
        pygame.draw.circle(screen, 'blue', (140, 930), 15)
    for i in range(lives):
        screen.blit(pygame.transform.scale(player_images[0], (30, 30)), (650 + i * 40, 915))
    if game_over:
        pygame.draw.rect(screen, 'white', [50, 200, 800, 300],0, 10)
        pygame.draw.rect(screen, 'dark gray', [70, 220, 760, 260], 0, 10)
        gameover_text = font.render('Game over! Space bar to restart!', True, 'red')
        screen.blit(gameover_text, (100, 300))
    if game_won:
        pygame.draw.rect(screen, 'white', [50, 200, 800, 300],0, 10)
        pygame.draw.rect(screen, 'dark gray', [70, 220, 760, 260], 0, 10)
        gameover_text = font.render('Victory! Space bar to restart!', True, 'green')
        screen.blit(gameover_text, (100, 300))


def check_collisions(scor, power, power_count, eaten_ghosts):
    num1 = (HEIGHT - 50) // 32
    num2 = WIDTH // 30
    if 0 < player_x < 870:
        if level[center_y // num1][center_x // num2] == 1:
            level[center_y // num1][center_x // num2] = 0
            scor += 10
        if level[center_y // num1][center_x // num2] == 2:
            level[center_y // num1][center_x // num2] = 0
```

```
            scor += 50
            power = True
            power_count = 0
            eaten_ghosts = [False, False, False, False]
    return scor, power, power_count, eaten_ghosts


def draw_board():
    num1 = ((HEIGHT - 50) // 32)
    num2 = (WIDTH // 30)
    for i in range(len(level)):
        for j in range(len(level[i])):
            if level[i][j] == 1:
                pygame.draw.circle(screen, 'white', (j * num2 + (0.5 * num2), i * num1 + (0.5 * num1)), 4)
            if level[i][j] == 2 and not flicker:
                pygame.draw.circle(screen, 'white', (j * num2 + (0.5 * num2), i * num1 + (0.5 * num1)), 10)
            if level[i][j] == 3:
                pygame.draw.line(screen, color, (j * num2 + (0.5 * num2), i * num1),
                                 (j * num2 + (0.5 * num2), i * num1 + num1), 3)
            if level[i][j] == 4:
                pygame.draw.line(screen, color, (j * num2, i * num1 + (0.5 * num1)),
                                 (j * num2 + num2, i * num1 + (0.5 * num1)), 3)
            if level[i][j] == 5:
                pygame.draw.arc(screen, color, [(j * num2 - (num2 * 0.4)) - 2, (i * num1 + (0.5 * num1)),
num2, num1],
                                0, PI / 2, 3)
            if level[i][j] == 6:
                pygame.draw.arc(screen, color,
                                [(j * num2 + (num2 * 0.5)), (i * num1 + (0.5 * num1)), num2, num1], PI / 2, PI, 3)
            if level[i][j] == 7:
                pygame.draw.arc(screen, color, [(j * num2 + (num2 * 0.5)), (i * num1 - (0.4 * num1)), num2,
num1], PI,
```

```
                3 * PI / 2, 3)
        if level[i][j] == 8:
            pygame.draw.arc(screen, color,
                [(j * num2 - (num2 * 0.4)) - 2, (i * num1 - (0.4 * num1)), num2, num1], 3 * PI / 2,
                2 * PI, 3)
        if level[i][j] == 9:
            pygame.draw.line(screen, 'white', (j * num2, i * num1 + (0.5 * num1)),
                (j * num2 + num2, i * num1 + (0.5 * num1)), 3)


def draw_player():
    # 0-RIGHT, 1-LEFT, 2-UP, 3-DOWN
    if direction == 0:
        screen.blit(player_images[counter // 5], (player_x, player_y))
    elif direction == 1:
        screen.blit(pygame.transform.flip(player_images[counter // 5], True, False), (player_x, player_y))
    elif direction == 2:
        screen.blit(pygame.transform.rotate(player_images[counter // 5], 90), (player_x, player_y))
    elif direction == 3:
        screen.blit(pygame.transform.rotate(player_images[counter // 5], 270), (player_x, player_y))


def check_position(centerx, centery):
    turns = [False, False, False, False]
    num1 = (HEIGHT - 50) // 32
    num2 = (WIDTH // 30)
    num3 = 15
    # check collisions based on center x and center y of player +/- fudge number
    if centerx // 30 < 29:
        if direction == 0:
            if level[centery // num1][(centerx - num3) // num2] < 3:
```

```
        turns[1] = True
if direction == 1:
    if level[centery // num1][(centerx + num3) // num2] < 3:
        turns[0] = True
if direction == 2:
    if level[(centery + num3) // num1][centerx // num2] < 3:
        turns[3] = True
if direction == 3:
    if level[(centery - num3) // num1][centerx // num2] < 3:
        turns[2] = True


if direction == 2 or direction == 3:
    if 12 <= centerx % num2 <= 18:
        if level[(centery + num3) // num1][centerx // num2] < 3:
            turns[3] = True
        if level[(centery - num3) // num1][centerx // num2] < 3:
            turns[2] = True
    if 12 <= centery % num1 <= 18:
        if level[centery // num1][(centerx - num2) // num2] < 3:
            turns[1] = True
        if level[centery // num1][(centerx + num2) // num2] < 3:
            turns[0] = True
if direction == 0 or direction == 1:
    if 12 <= centerx % num2 <= 18:
        if level[(centery + num1) // num1][centerx // num2] < 3:
            turns[3] = True
        if level[(centery - num1) // num1][centerx // num2] < 3:
            turns[2] = True
    if 12 <= centery % num1 <= 18:
        if level[centery // num1][(centerx - num3) // num2] < 3:
            turns[1] = True
```

```
            if level[centery // num1][(centerx + num3) // num2] < 3:
                turns[0] = True
    else:
        turns[0] = True
        turns[1] = True

    return turns


def move_player(play_x, play_y):
    # r, l, u, d
    if direction == 0 and turns_allowed[0]:
        play_x += player_speed
    elif direction == 1 and turns_allowed[1]:
        play_x -= player_speed
    if direction == 2 and turns_allowed[2]:
        play_y -= player_speed
    elif direction == 3 and turns_allowed[3]:
        play_y += player_speed
    return play_x, play_y


def get_targets(blink_x, blink_y, ink_x, ink_y, pink_x, pink_y, clyd_x, clyd_y):
    if player_x < 450:
        runaway_x = 900
    else:
        runaway_x = 0
    if player_y < 450:
        runaway_y = 900
    else:
        runaway_y = 0
```

```
return_target = (380, 400)
if powerup:
    if not blinky.dead and not eaten_ghost[0]:
        blink_target = (runaway_x, runaway_y)
    elif not blinky.dead and eaten_ghost[0]:
        if 340 < blink_x < 560 and 340 < blink_y < 500:
            blink_target = (400, 100)
        else:
            blink_target = (player_x, player_y)
    else:
        blink_target = return_target
    if not inky.dead and not eaten_ghost[1]:
        ink_target = (runaway_x, player_y)
    elif not inky.dead and eaten_ghost[1]:
        if 340 < ink_x < 560 and 340 < ink_y < 500:
            ink_target = (400, 100)
        else:
            ink_target = (player_x, player_y)
    else:
        ink_target = return_target
    if not pinky.dead:
        pink_target = (player_x, runaway_y)
    elif not pinky.dead and eaten_ghost[2]:
        if 340 < pink_x < 560 and 340 < pink_y < 500:
            pink_target = (400, 100)
        else:
            pink_target = (player_x, player_y)
    else:
        pink_target = return_target
    if not clyde.dead and not eaten_ghost[3]:
        clyd_target = (450, 450)
```

```
    elif not clyde.dead and eaten_ghost[3]:
        if 340 < clyd_x < 560 and 340 < clyd_y < 500:
            clyd_target = (400, 100)
        else:
            clyd_target = (player_x, player_y)
    else:
        clyd_target = return_target
else:
    if not blinky.dead:
        if 340 < blink_x < 560 and 340 < blink_y < 500:
            blink_target = (400, 100)
        else:
            blink_target = (player_x, player_y)
    else:
        blink_target = return_target
    if not inky.dead:
        if 340 < ink_x < 560 and 340 < ink_y < 500:
            ink_target = (400, 100)
        else:
            ink_target = (player_x, player_y)
    else:
        ink_target = return_target
    if not pinky.dead:
        if 340 < pink_x < 560 and 340 < pink_y < 500 and score >300:
            pink_target = (400, 100)
        else:
            pink_target = (player_x, player_y)
    else:
        pink_target = return_target
    if not clyde.dead:
        if 340 < clyd_x < 560 and 340 < clyd_y < 500 and score > 700:
```

```
            clyd_target = (400, 100)
        else:
            clyd_target = (player_x, player_y)
    else:
        clyd_target = return_target
return [blink_target, ink_target, pink_target, clyd_target]



run = True
while run:
    timer.tick(fps)
    if counter < 19:
        counter += 1
        if counter > 3:
            flicker = False
    else:
        counter = 0
        flicker = True
    if powerup and power_counter < 600:
        power_counter += 1
    elif powerup and power_counter >= 600:
        power_counter = 0
        powerup = False
        eaten_ghost = [False, False, False, False]
    if startup_counter < 180 and not game_over and not game_won:
        moving = False
        startup_counter += 1
    else:
        moving = True

    screen.fill('black')
```

```
draw_board()
center_x = player_x + 23
center_y = player_y + 24
if powerup:
    ghost_speeds = [1, 1, 1, 1]
else:
    ghost_speeds = [2, 2, 2, 2]
if eaten_ghost[0]:
    ghost_speeds[0] = 2
if eaten_ghost[1]:
    ghost_speeds[1] = 2
if eaten_ghost[2]:
    ghost_speeds[2] = 2
if eaten_ghost[3]:
    ghost_speeds[3] = 2
if blinky_dead:
    ghost_speeds[0] = 4
if inky_dead:
    ghost_speeds[1] = 4
if pinky_dead:
    ghost_speeds[2] = 4
if clyde_dead:
    ghost_speeds[3] = 4

game_won = True
for i in range(len(level)):
    if 1 in level[i] or 2 in level[i]:
        game_won = False

player_circle = pygame.draw.circle(screen, 'black', (center_x, center_y), 20, 2)
draw_player()
```

```
    blinky = Ghost(blinky_x, blinky_y, targets[0], ghost_speeds[0], blinky_img, blinky_direction,
blinky_dead,
                   blinky_box, 0)
    inky = Ghost(inky_x, inky_y, targets[1], ghost_speeds[1], inky_img, inky_direction, inky_dead,
                  inky_box, 1)
    pinky = Ghost(pinky_x, pinky_y, targets[2], ghost_speeds[2], pinky_img, pinky_direction,
pinky_dead,
                   pinky_box, 2)
    clyde = Ghost(clyde_x, clyde_y, targets[3], ghost_speeds[3], clyde_img, clyde_direction,
clyde_dead,
                   clyde_box, 3)
    draw_misc()
    targets = get_targets(blinky_x, blinky_y, inky_x, inky_y, pinky_x, pinky_y, clyde_x, clyde_y)


    turns_allowed = check_position(center_x, center_y)
    if moving:
        player_x, player_y = move_player(player_x, player_y)
        if not blinky_dead and not blinky.in_box:
            blinky_x, blinky_y, blinky_direction = blinky.move_blinky()
        else:
            blinky_x, blinky_y, blinky_direction = blinky.move_clyde()
        if not pinky_dead and not pinky.in_box:
            pinky_x, pinky_y, pinky_direction = pinky.move_pinky()
        else:
            pinky_x, pinky_y, pinky_direction = pinky.move_clyde()
        if not inky_dead and not inky.in_box:
            inky_x, inky_y, inky_direction = inky.move_inky()
        else:
            inky_x, inky_y, inky_direction = inky.move_clyde()
        clyde_x, clyde_y, clyde_direction = clyde.move_clyde()
    score, powerup, power_counter, eaten_ghost = check_collisions(score, powerup, power_counter,
eaten_ghost)
```

```
# add to if not powerup to check if eaten ghosts
if not powerup:
    if (player_circle.colliderect(blinky.rect) and not blinky.dead) or \
            (player_circle.colliderect(inky.rect) and not inky.dead) or \
            (player_circle.colliderect(pinky.rect) and not pinky.dead) or \
            (player_circle.colliderect(clyde.rect) and not clyde.dead):
        if lives > 0:
            lives -= 1
            eat_sound.play()
            startup_counter = 0
            powerup = False
            power_counter = 0
            player_x = 450
            player_y = 663
            direction = 0
            direction_command = 0
            blinky_x = 56
            blinky_y = 58
            blinky_direction = 0
            inky_x = 440
            inky_y = 388
            inky_direction = 2
            pinky_x = 440
            pinky_y = 438
            pinky_direction = 2
            clyde_x = 420
            clyde_y = 438
            clyde_direction = 2
            eaten_ghost = [False, False, False, False]
            blinky_dead = False
            inky_dead = False
```

```
            clyde_dead = False
            pinky_dead = False
        else:
            game_over = True
            moving = False
            startup_counter = 0
    if powerup and player_circle.colliderect(blinky.rect) and eaten_ghost[0] and not blinky.dead:
        if lives > 0:
            powerup = False
            power_counter = 0
            lives -= 1
            startup_counter = 0
            player_x = 450
            player_y = 663
            direction = 0
            direction_command = 0
            blinky_x = 56
            blinky_y = 58
            blinky_direction = 0
            inky_x = 440
            inky_y = 388
            inky_direction = 2
            pinky_x = 440
            pinky_y = 438
            pinky_direction = 2
            clyde_x = 420
            clyde_y = 438
            clyde_direction = 2
            eaten_ghost = [False, False, False, False]
            blinky_dead = False
            inky_dead = False
```

```
            clyde_dead = False
            pinky_dead = False
        else:
            game_over = True
            moving = False
            startup_counter = 0
    if powerup and player_circle.colliderect(inky.rect) and eaten_ghost[1] and not inky.dead:
        if lives > 0:
            powerup = False
            power_counter = 0
            lives -= 1
            startup_counter = 0
            player_x = 450
            player_y = 663
            direction = 0
            direction_command = 0
            blinky_x = 56
            blinky_y = 58
            blinky_direction = 0
            inky_x = 440
            inky_y = 388
            inky_direction = 2
            pinky_x = 440
            pinky_y = 438
            pinky_direction = 2
            clyde_x = 420
            clyde_y = 438
            clyde_direction = 2
            eaten_ghost = [False, False, False, False]
            blinky_dead = False
            inky_dead = False
```

```
            clyde_dead = False
            pinky_dead = False
        else:
            game_over = True
            moving = False
            startup_counter = 0
    if powerup and player_circle.colliderect(pinky.rect) and eaten_ghost[2] and not pinky.dead:
        if lives > 0:
            powerup = False
            power_counter = 0
            lives -= 1
            startup_counter = 0
            player_x = 450
            player_y = 663
            direction = 0
            direction_command = 0
            blinky_x = 56
            blinky_y = 58
            blinky_direction = 0
            inky_x = 440
            inky_y = 388
            inky_direction = 2
            pinky_x = 440
            pinky_y = 438
            pinky_direction = 2
            clyde_x = 420
            clyde_y = 438
            clyde_direction = 2
            eaten_ghost = [False, False, False, False]
            blinky_dead = False
            inky_dead = False
```

```
            clyde_dead = False
            pinky_dead = False
        else:
            game_over = True
            moving = False
            startup_counter = 0
    if powerup and player_circle.colliderect(clyde.rect) and eaten_ghost[3] and not clyde.dead:
        if lives > 0:
            powerup = False
            power_counter = 0
            lives -= 1
            startup_counter = 0
            player_x = 450
            player_y = 663
            direction = 0
            direction_command = 0
            blinky_x = 56
            blinky_y = 58
            blinky_direction = 0
            inky_x = 440
            inky_y = 388
            inky_direction = 2
            pinky_x = 440
            pinky_y = 438
            pinky_direction = 2
            clyde_x = 420
            clyde_y = 438
            clyde_direction = 2
            eaten_ghost = [False, False, False, False]
            blinky_dead = False
            inky_dead = False
```

```
            clyde_dead = False
            pinky_dead = False
        else:
            game_over = True
            moving = False
            startup_counter = 0
    if powerup and player_circle.colliderect(blinky.rect) and not blinky.dead and not eaten_ghost[0]:
        blinky_dead = True
        eaten_ghost[0] = True
        score += (2 ** eaten_ghost.count(True)) * 100
    if powerup and player_circle.colliderect(inky.rect) and not inky.dead and not eaten_ghost[1]:
        inky_dead = True
        eaten_ghost[1] = True
        score += (2 ** eaten_ghost.count(True)) * 100
    if powerup and player_circle.colliderect(pinky.rect) and not pinky.dead and not eaten_ghost[2]:
        pinky_dead = True
        eaten_ghost[2] = True
        score += (2 ** eaten_ghost.count(True)) * 100
    if powerup and player_circle.colliderect(clyde.rect) and not clyde.dead and not eaten_ghost[3]:
        clyde_dead = True
        eaten_ghost[3] = True
        score += (2 ** eaten_ghost.count(True)) * 100

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                direction_command = 0
            if event.key == pygame.K_LEFT:
                direction_command = 1
```

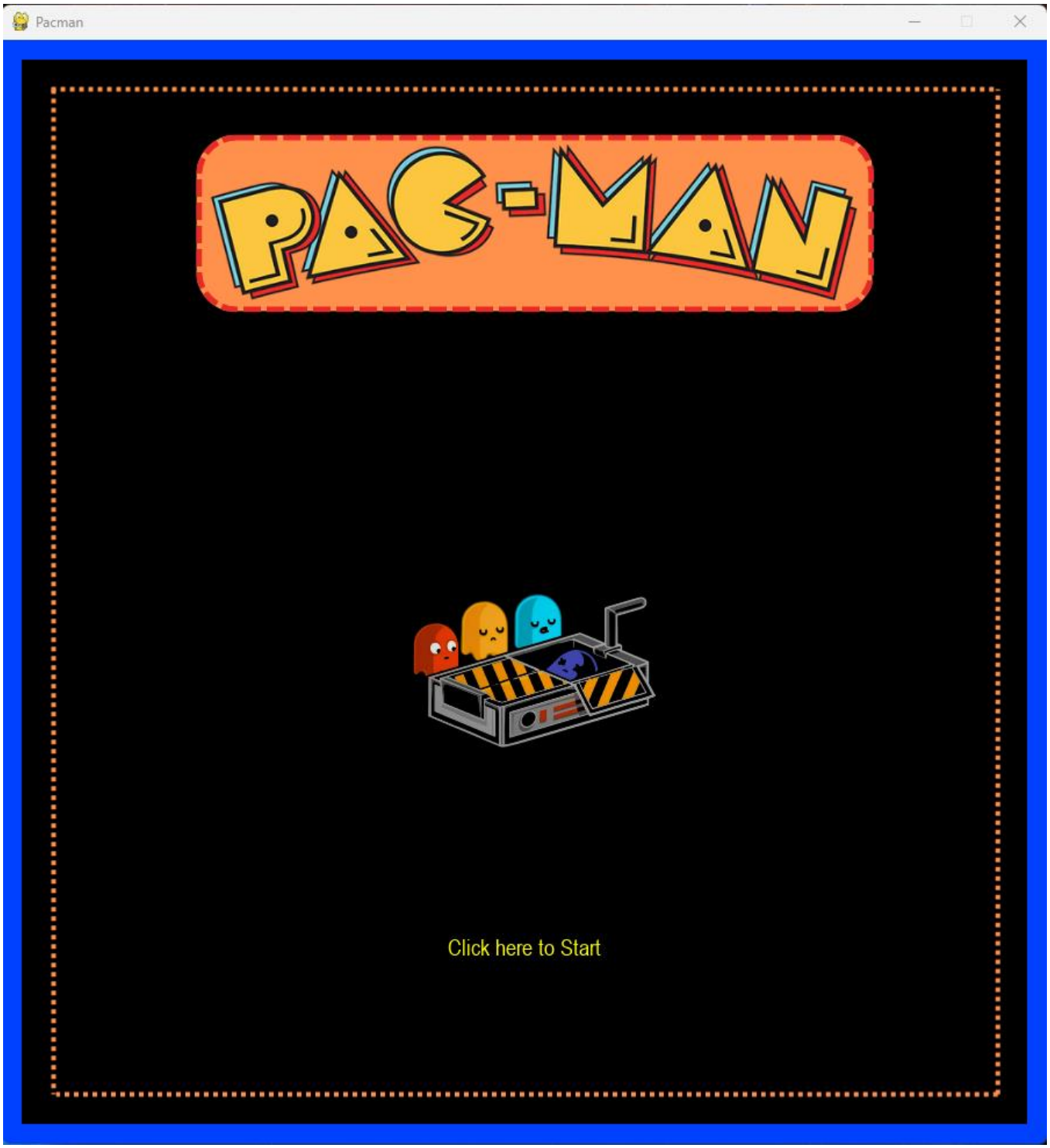
```
if event.key == pygame.K_UP:
    direction_command = 2
if event.key == pygame.K_DOWN:
    direction_command = 3
if event.key == pygame.K_SPACE and (game_over or game_won):
    powerup = False
    power_counter = 0
    lives -= 1
    startup_counter = 0
    player_x = 450
    player_y = 663
    direction = 0
    direction_command = 0
    blinky_x = 56
    blinky_y = 58
    blinky_direction = 0
    inky_x = 440
    inky_y = 388
    inky_direction = 2
    pinky_x = 440
    pinky_y = 438
    pinky_direction = 2
    clyde_x = 420
    clyde_y = 438
    clyde_direction = 2
    eaten_ghost = [False, False, False, False]
    blinky_dead = False
    inky_dead = False
    clyde_dead = False
    pinky_dead = False
    score = 0
```

```
            lives = 6
            boards2 += 1
            if boards2 >= len(boards[boards1]):
                boards2 = 0
                boards1 += 1
                if boards1 >= len(boards):
                    break
            level = copy.deepcopy(boards[boards1][boards2])
            game_over = False
            game_won = False

    if event.type == pygame.KEYUP:
        if event.key == pygame.K_RIGHT and direction_command == 0:
            direction_command = direction
        if event.key == pygame.K_LEFT and direction_command == 1:
            direction_command = direction
        if event.key == pygame.K_UP and direction_command == 2:
            direction_command = direction
        if event.key == pygame.K_DOWN and direction_command == 3:
            direction_command = direction

if direction_command == 0 and turns_allowed[0]:
    direction = 0
if direction_command == 1 and turns_allowed[1]:
    direction = 1
if direction_command == 2 and turns_allowed[2]:
    direction = 2
if direction_command == 3 and turns_allowed[3]:
    direction = 3

if player_x > 900:
```

```
        player_x = -47
    elif player_x < -50:
        player_x = 897

    if blinky.in_box and blinky_dead:
        blinky_dead = False
    if inky.in_box and inky_dead:
        inky_dead = False
    if pinky.in_box and pinky_dead:
        pinky_dead = False
    if clyde.in_box and clyde_dead:
        clyde_dead = False

    pygame.display.flip()

pygame.quit()

# sound effects, restart and winning messages
```

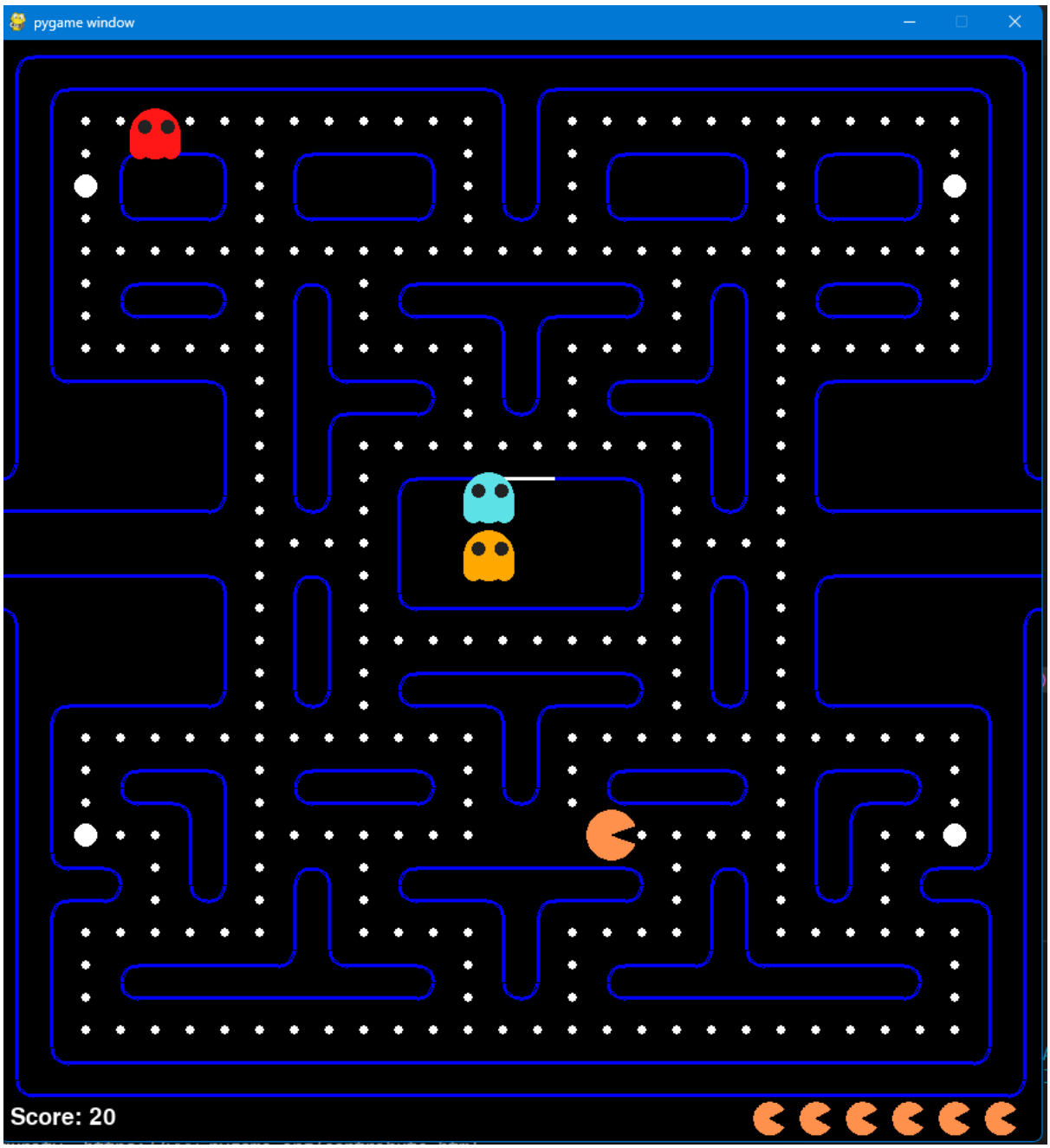
Output of the Code

Start Page of the Game

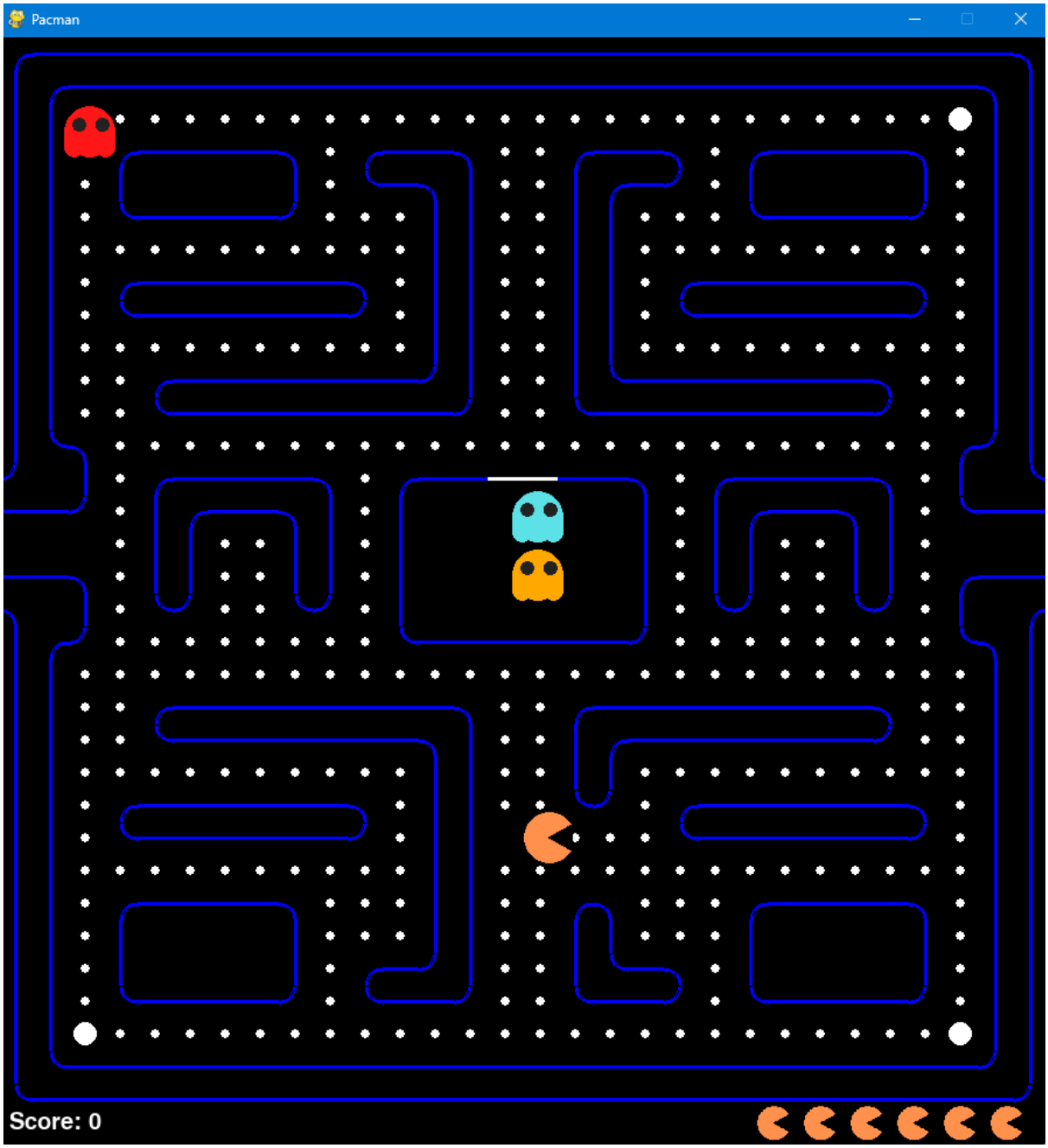


After Clicking ‘Click here to Start’

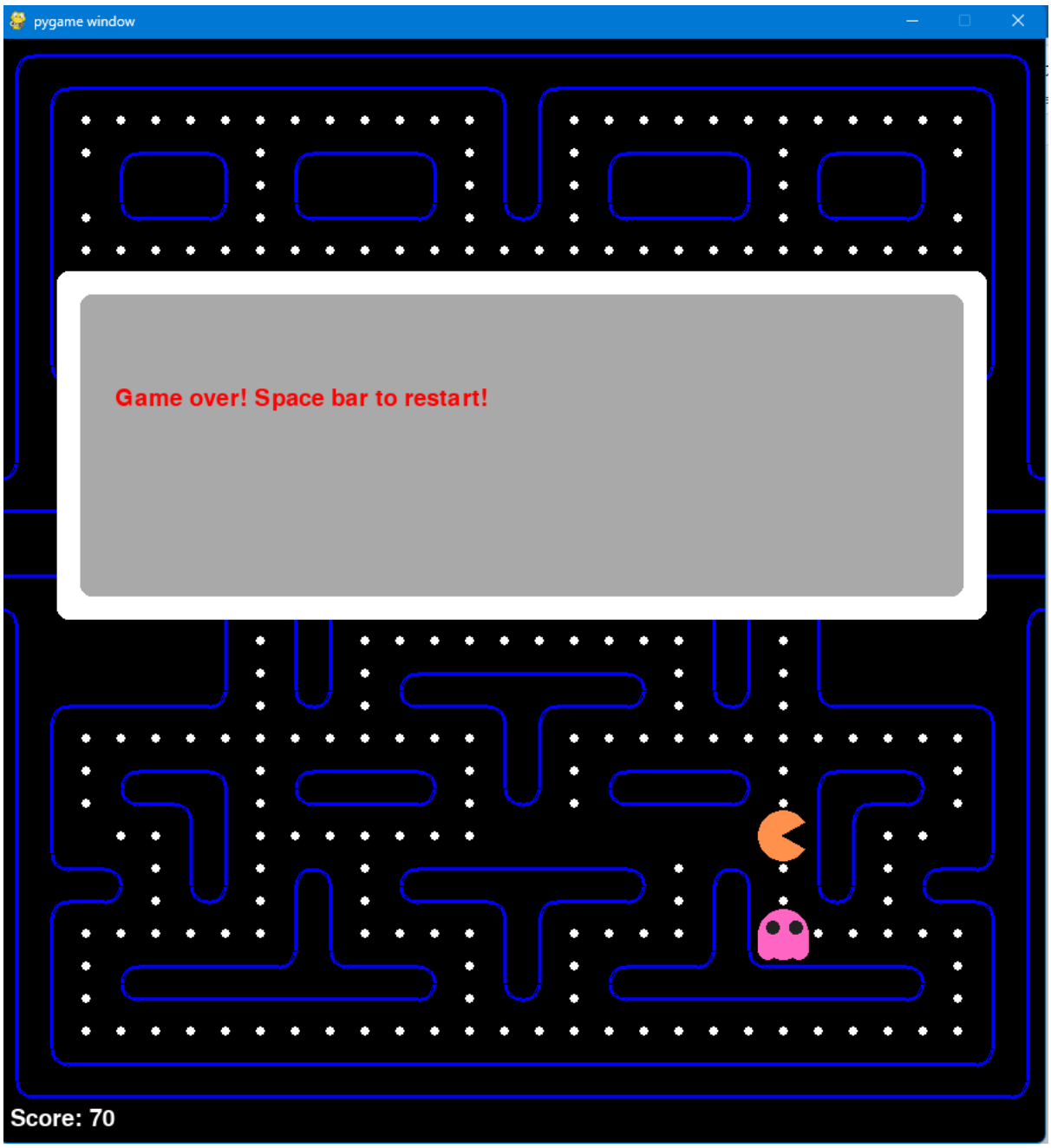
Board1



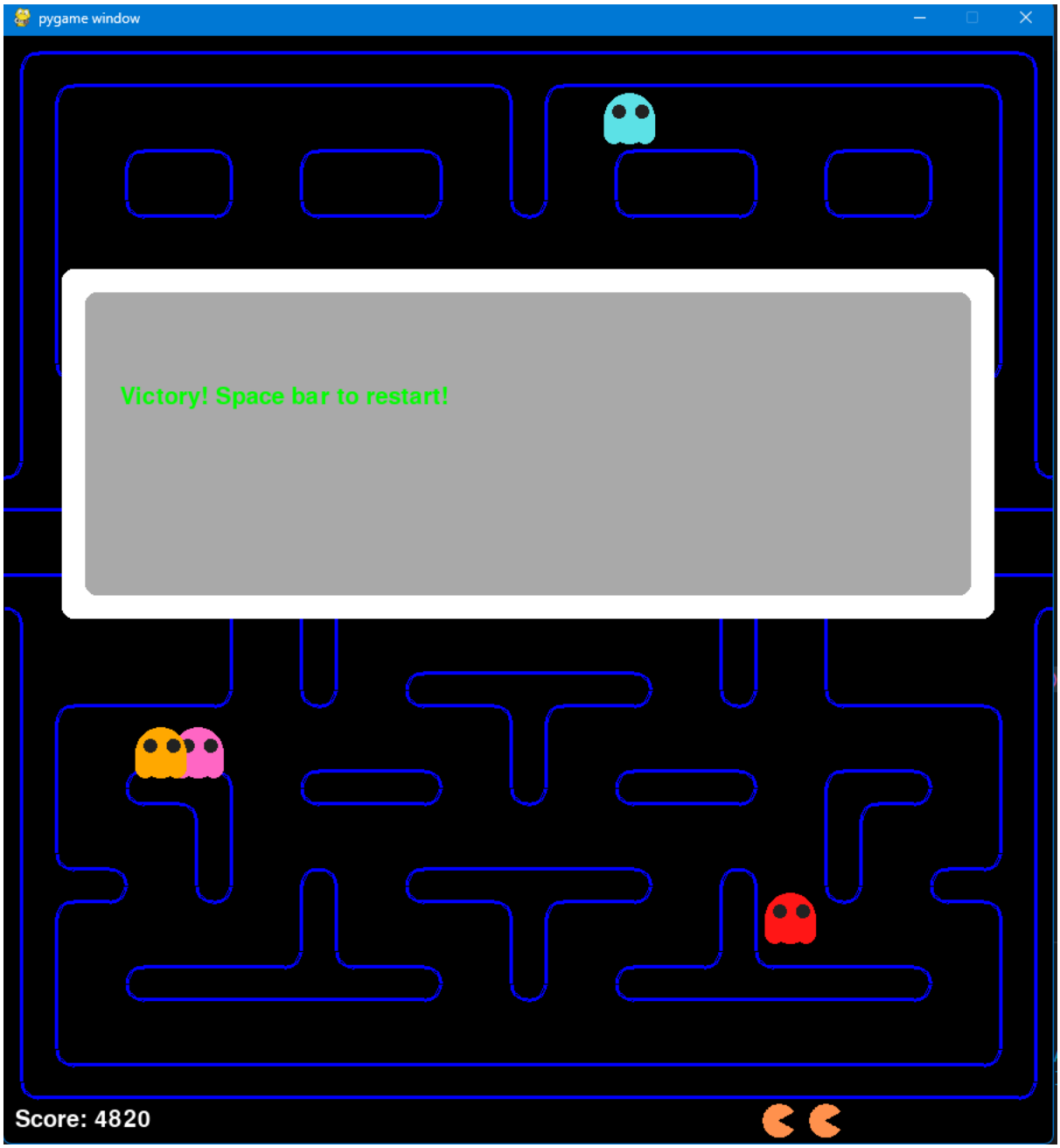
Board 2



Game Over Window



Game Win Window



Conclusion

In conclusion, the Pacman game developed using Python and Pygame is an excellent example of game development and programming. The project provided a great opportunity to explore the world of game development and to apply programming concepts in a fun and engaging way.

The game includes a tile-based board, Pacman player, four ghosts with unique chase patterns, power-ups, and two levels, making it a challenging and entertaining experience for players. The game is also designed to target children and students who want to learn programming concepts while enjoying a classic game.

The project successfully achieved its objectives and provided a valuable learning experience for the development team. The team faced several challenges during the project, including designing the game board and implementing the chase patterns for the ghosts. However, these challenges were successfully overcome, and the final product is a fun and engaging game that can be played on basic computers.

Overall, the Pacman game project is a great example of how programming concepts can be applied to develop a fun and engaging game. The project has provided valuable insights into the world of game development and programming, and the team hopes that this report inspires others to explore this exciting field.

Bibliography

- Python Software Foundation. Python Language Reference, version 3.9. Available at https://docs.python.org/3/reference/
- Pygame community. Pygame documentation, version 2.0.1. Available at https://www.pygame.org/docs/
- Visual Studio Code. Version 1.56.2. Available at https://code.visualstudio.com/
- Microsoft Corporation. Microsoft Windows operating system. Available at https://www.microsoft.com/windows/
- Intel Corporation. Intel Core i5 processor. Available at https://www.intel.com/content/www/us/en/products/processors/core/i5-processors.html
- Stack Overflow. "How to implement collision detection between images in Pygame?" Stack Overflow, 16 July 2018, https://stackoverflow.com/questions/51434143/how-to-implement-collision-detection-between-images-in-pygame
- GeeksforGeeks. (n.d.). Python - Pygame. Retrieved from https://www.geeksforgeeks.org/python-pygame/
- Pac-Man. (n.d.). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Pac-Man
- ChatGPT. (2023). Personal conversation with the project developer.
- Images from Google search: https://www.google.com/search?q=pacman+game+images&rlz=1C1GCEA_enUS832US832&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiJkM2arK_iAhVJ1qwKHcYvCh0Q_AUIDigB&biw=1366&bih=657

Further Improvement & Future Works

- Adding more levels: Currently, the game has two levels, but more levels could be added to make the game more challenging and interesting for the players.
- Enhancing the graphics: The current game graphics are simple, and adding more animations and effects could enhance the game's visual appeal.
- Adding multiplayer support: Adding multiplayer support could make the game more engaging and fun for players who enjoy competing with others.
- Implementing different game modes: Different game modes such as time trials, endless mode, and others could be implemented to provide players with more variety and different challenges.
- Adding more power-ups and challenges: Currently, the game has some power-ups and challenges, but more could be added to make the game more exciting and engaging for the players.
- Adding different characters: Besides the Pacman and Ghosts characters, adding more characters to the game could provide players with more variety and options.
- Making the game open-source: Making the game open-source could encourage other developers to contribute to the project and improve it further, making it more robust and engaging.

Thank You

We would like to express our sincere gratitude to everyone who has supported and contributed to the completion of this project. We would like to thank the following:

- Our project supervisor for providing guidance, support, and constructive feedback throughout the project
- Our fellow classmates for their assistance and valuable insights
- Our families and friends for their encouragement and understanding throughout this journey
- The authors and developers of the various tools, libraries, and resources used in this project
- The participants who helped us gather data and feedback for the project
- We would also like to thank you, the reader, for taking the time to read our project report. We hope that this project has been informative and useful to you.

Thank you again to everyone who has contributed to this project in one way or another. We appreciate all your help and support.

Sincerely,

Aman, Jatin Jangra