VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

NAME:VIJAY.R

REG NO:22BIT0687

SUBJECT:ARTIFICIAL INTELLIGENCE LAB

LAB SLOT : L11+L12

WUMPUS WORLD : GREEN RUSH

Code:

```
import pygame
import random
import time
# Initialize Pygame
pygame.init()
# Constants
GRID_SIZE = 12  # Grid size
CELL_SIZE = 60
WIDTH, HEIGHT = GRID_SIZE * CELL_SIZE, GRID_SIZE * CELL_SIZE
INFO_PANEL_WIDTH = 200  # Width for the score display
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)
GRAY = (169, 169, 169)
GOLD_COLOR = (255, 215, 0)
SILVER_COLOR = (192, 192, 192)
DARK_GREEN = (0, 100, 0)  # Color for bushes
```

```
DOOR_COLOR = (150, 75, 0)  # Brown color for the door
DIAMOND_COLOR = (0, 255, 255)  # Light blue color for the diamond
# Create the display window with additional space for the info panel
screen = pygame.display.set_mode((WIDTH + INFO_PANEL_WIDTH, HEIGHT))
pygame.display.set_caption("Green Rush")
# Game state variables
level = 1
cumulative_score = 0
time_limit = 15  # Timer for each level (in seconds)
start_time = None
door_position = None
game_over = False
diamond = None  # For the diamond on every 5th level
coins = []  # Silver coins
game_started = False
instruction_duration = 7  # 7 seconds countdown for instructions page
instruction_start_time = time.time()
# Unique Game Name
GAME_NAME = "GREEN RUSH"
# Function to display instructions and game conditions with countdown
def display_instructions(remaining_time):
    screen.fill(WHITE)
    font_title = pygame.font.SysFont('Arial', 65, bold=True)
    font_text = pygame.font.SysFont('Arial', 40)
    title_text = font_title.render(f"Welcome to {GAME_NAME}", True, DARK_GREEN)
    instructions = [
        "Instructions:",
        "- Use arrow keys to move the agent.",
        "- Avoid hitting green bushes (obstacles).",
        "- Find the door to complete each level.",
        f" -Each level lasts for {time_limit} seconds.",
    ]
```

```
    screen.blit(title_text, (WIDTH // 6, HEIGHT // 6))
    for idx, line in enumerate(instructions):
        instruction_text = font_text.render(line, True, BLACK)
        screen.blit(instruction_text, (WIDTH // 8, HEIGHT // 3 + (idx * 50)))
    countdown_text = font_text.render(f"Game starting in: {remaining_time} seconds", True, RED)
    screen.blit(countdown_text, (WIDTH // 4, HEIGHT // 2 + 150))
    pygame.display.flip()
# Function to initialize the game state
def init_game():
    global agent, goals, obstacles, coins, door_position, start_time, game_over, diamond, time_limit
    agent = {"position": [0, 0], "score": cumulative_score}
    goals = []
    obstacles = []
    coins = []
    diamond = None
    game_over = False  # Reset game_over when the game initializes
    # Create a safe zone around the agent's starting position (3x3 area)
    safe_zone = [(agent["position"][0] + dx, agent["position"][1] + dy)
                 for dx in range(-1, 2) for dy in range(-1, 2)]
    # Check if it's a special level (every 5th level)
    if level % 5 == 0:
        # Place a diamond and adjust the timer for special levels
        time_limit = 7
        while True:
            diamond_position = [random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)]
            distance = abs(diamond_position[0] - agent["position"][0]) + abs(diamond_position[1] -
agent["position"][1])
            if distance > 6 and diamond_position not in safe_zone:  # Ensure the diamond is far from the agent and in a
safe position
                diamond = diamond_position
                goals.append({"type": "diamond", "position": diamond_position, "color": DIAMOND_COLOR})
```

```
            break
    else:
        # Normal levels with gold and silver coins, reset the timer
        time_limit = 15
        # Ensure coins and obstacles don't overlap
        for _ in range(12):  # Create obstacles (bushes)
            while True:
                pos = [random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)]
                if pos not in obstacles and pos not in safe_zone and pos != agent["position"] and pos != door_position and pos not in coins and pos != diamond:
                    hidden = random.choice([True, False])  # 50% chance the obstacle will be hidden
                    obstacles.append({"position": pos, "hidden": hidden})
                    break
        # Place up to 3 gold coins
        for _ in range(3):
            while True:
                pos = [random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)]
                if pos not in obstacles and pos not in safe_zone and pos != agent["position"] and pos != door_position and pos != diamond:
                    goals.append({"type": "gold", "position": pos, "color": GOLD_COLOR})
                    break
        # Place up to 6 silver coins
        for _ in range(6):
            while True:
                pos = [random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)]
                if pos not in obstacles and pos not in safe_zone and pos != agent["position"] and pos != door_position and pos != diamond:
                    coins.append(pos)
                    break
    # Place the door far from the agent and without overlapping with obstacles, coins, or diamond
    max_distance = 0
    for _ in range(100):
        door_candidate = [random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)]
```

```
        distance = abs(door_candidate[0] - agent["position"][0]) + abs(door_candidate[1] - agent["position"][1])
        if distance > max_distance and door_candidate not in obstacles and door_candidate not in coins and door_candidate != diamond:
            max_distance = distance
            door_position = door_candidate
    start_time = time.time()  # Set the start time for the level
# Function to draw the grid
def draw_grid():
    for x in range(0, WIDTH, CELL_SIZE):
        for y in range(0, HEIGHT, CELL_SIZE):
            rect = pygame.Rect(x, y, CELL_SIZE, CELL_SIZE)
            pygame.draw.rect(screen, GRAY, rect, 1)
# Function to draw the agent (as a triangle)
def draw_agent():
    x, y = agent["position"]
    points = [(x * CELL_SIZE + CELL_SIZE // 2, y * CELL_SIZE + 10),  # Top point
              (x * CELL_SIZE + 10, y * CELL_SIZE + CELL_SIZE - 10),  # Bottom-left point
              (x * CELL_SIZE + CELL_SIZE - 10, y * CELL_SIZE + CELL_SIZE - 10)]  # Bottom-right point
    pygame.draw.polygon(screen, GREEN, points)
# Function to draw goals (gold coins, silver coins, and diamond)
def draw_goals():
    for goal in goals:
        x, y = goal["position"]
        pygame.draw.circle(screen, goal["color"], (x * CELL_SIZE + CELL_SIZE // 2, y * CELL_SIZE + CELL_SIZE // 2), CELL_SIZE // 3)
# Function to draw silver coins
def draw_silver_coins():
    for cx, cy in coins:
        pygame.draw.circle(screen, SILVER_COLOR, (cx * CELL_SIZE + CELL_SIZE // 2, cy * CELL_SIZE + CELL_SIZE // 2), CELL_SIZE // 4)
# Function to draw bushes using polygons (for irregular shapes) and handle hidden obstacles
def draw_obstacles():
    for obstacle in obstacles:
```

```
        ox, oy = obstacle["position"]
        if not obstacle["hidden"]:  # Only draw if the obstacle is not hidden
            base_x, base_y = ox * CELL_SIZE + CELL_SIZE // 2, oy * CELL_SIZE + CELL_SIZE // 2
            points = [
                (base_x - 20, base_y),  # Left
                (base_x - 10, base_y - 15),  # Top-left
                (base_x + 10, base_y - 15),  # Top-right
                (base_x + 20, base_y),  # Right
                (base_x + 10, base_y + 15),  # Bottom-right
                (base_x - 10, base_y + 15)   # Bottom-left
            ]
            pygame.draw.polygon(screen, DARK_GREEN, points)
# Function to draw the door
def draw_door():
    x, y = door_position
    pygame.draw.arc(screen, DOOR_COLOR, (x * CELL_SIZE, y * CELL_SIZE, CELL_SIZE, CELL_SIZE), 3.14, 6.28, 5)  # Draw dome shape
    pygame.draw.rect(screen, DOOR_COLOR, (x * CELL_SIZE + CELL_SIZE // 4, y * CELL_SIZE + CELL_SIZE // 2, CELL_SIZE // 2, CELL_SIZE // 2))  # Draw the door
# Function to draw the diamond (as a light blue circle on every 5th level)
def draw_diamond():
    if diamond:
        x, y = diamond
        pygame.draw.circle(screen, DIAMOND_COLOR, (x * CELL_SIZE + CELL_SIZE // 2, y * CELL_SIZE + CELL_SIZE // 2), CELL_SIZE // 3)
# Function to move the agent smoothly and trigger game over if an obstacle is hit
def move_agent(direction):
    x, y = agent["position"]
    new_x, new_y = x, y  # Default to current position
    if direction == "up":
        new_y = max(0, y - 1)
    elif direction == "down":
        new_y = min(GRID_SIZE - 1, y + 1)
```

```
    elif direction == "left":
        new_x = max(0, x - 1)
    elif direction == "right":
        new_x = min(GRID_SIZE - 1, x + 1)
    # Check if the new position is an obstacle
    if any(obstacle["position"] == [new_x, new_y] for obstacle in obstacles):
        print("You hit an obstacle! Game Over!")
        global game_over
        game_over = True  # Set the game over state
    else:
        agent["position"] = [new_x, new_y]
# Function to check if the agent reaches a goal
def check_goal():
    global cumulative_score, diamond
    for goal in goals[:]:
        if agent["position"] == goal["position"]:
            if goal["type"] == "diamond":
                agent["score"] += 100  # Diamond gives 50 points
                diamond = None  # Remove the diamond after collection
            else:
                agent["score"] += 20
            cumulative_score = agent["score"]
            goals.remove(goal)
            print(f"Collected {goal['type']}! Score: {agent['score']}")
# Function to check if the agent encounters an obstacle or coin
def check_obstacle_or_coin():
    global coins
    if any(obstacle["position"] == agent["position"] for obstacle in obstacles):
        print("You hit an obstacle (green bush)! Game Over!")
        global game_over
        game_over = True
    if agent["position"] in coins:
```

```
        agent["score"] += 10
        coins.remove(agent["position"])
        print(f"Collected a silver coin! Score: {agent['score']}")
        cumulative_score = agent["score"]
# Function to check if the agent is near the door to progress to the next level
def check_door():
    if agent["position"] == door_position:
        print("Reached the door! Proceeding to the next level...")
        next_level()
# Function to display the score, level, timer, and buttons on the side panel
def display_info_panel():
    font = pygame.font.SysFont(None, 35)
    score_text = font.render(f"Score: {agent['score']}", True, BLACK)
    level_text = font.render(f"Level: {level}", True, BLACK)
    elapsed_time = int(time.time() - start_time)
    time_left = max(time_limit - elapsed_time, 0)
    if not game_over:
        timer_text = font.render(f"Time: {time_left}", True, BLACK)
    else:
        timer_text = font.render(f"Time: 0", True, BLACK)  # Stop timer when game over
    screen.fill(WHITE, pygame.Rect(WIDTH, 0, INFO_PANEL_WIDTH, HEIGHT))
    screen.blit(score_text, [WIDTH + 10, 70])
    screen.blit(level_text, [WIDTH + 10, 110])
    screen.blit(timer_text, [WIDTH + 10, 150])
    # Draw Exit and Retry buttons
    pygame.draw.rect(screen, RED, pygame.Rect(WIDTH + 10, 10, 180, 40))
    pygame.draw.rect(screen, GREEN, pygame.Rect(WIDTH + 10, 200, 180, 40))
    exit_text = font.render("Exit", True, WHITE)
    retry_text = font.render("Retry", True, WHITE)
    screen.blit(exit_text, [WIDTH + 70, 20])
    screen.blit(retry_text, [WIDTH + 60, 210])
```

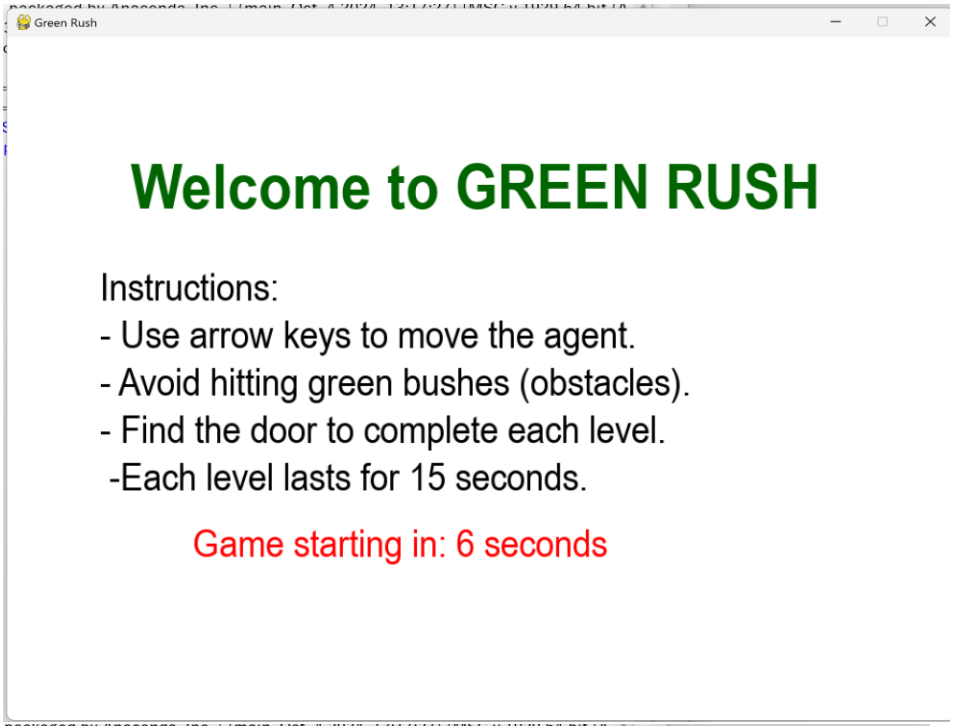
```
# Function to reset the game for a new level
def next_level():
    global level, time_limit, game_over
    level += 1
    game_over = False
    init_game()
# Main game loop with instructions and timed start
while True:
    if not game_started:
        remaining_time = int(instruction_duration - (time.time() - instruction_start_time))
        display_instructions(remaining_time)
        if remaining_time <= 0:
            game_started = True
            init_game()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        if game_started and not game_over:
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP:
                    move_agent("up")
                elif event.key == pygame.K_DOWN:
                    move_agent("down")
                elif event.key == pygame.K_LEFT:
                    move_agent("left")
                elif event.key == pygame.K_RIGHT:
                    move_agent("right")
                if not game_over:  # Prevent further actions after game over
                    check_goal()
                    check_obstacle_or_coin()
                    check_door()
```

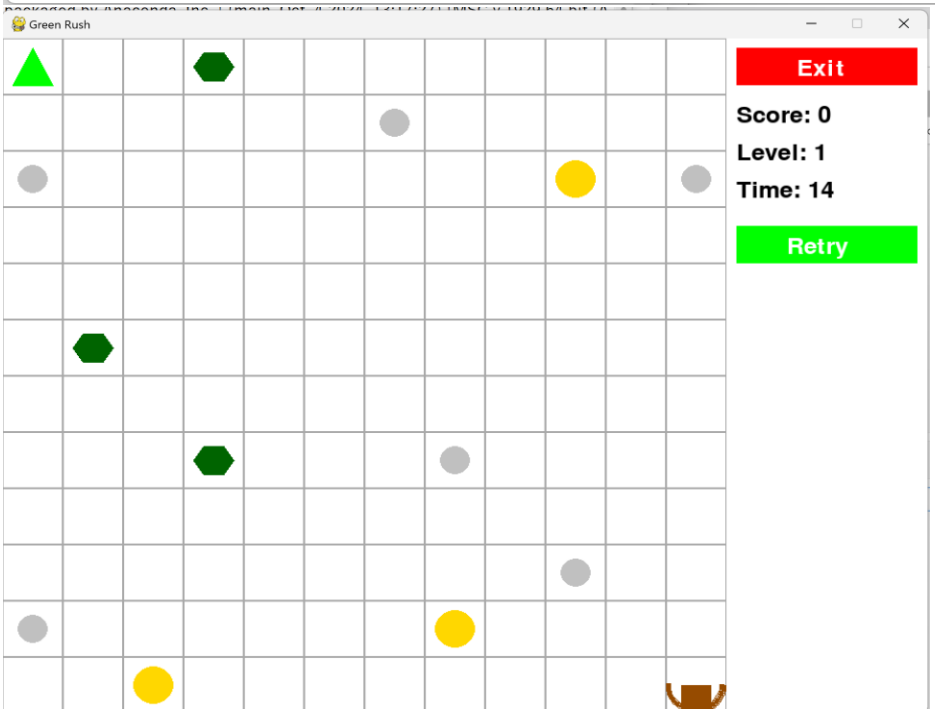
```
            if not coins and not diamond:  # If all coins/diamond are collected, advance to the next level
                print("All coins collected! Advancing to the next level...")
                next_level()
            elapsed_time = time.time() - start_time
            if elapsed_time > time_limit:
                print("Time's up! Game Over!")
                game_over = True
    if event.type == pygame.KEYDOWN and event.key == pygame.K_r and game_over:  # Retry handling
        level = 1
        cumulative_score = 0
        time_limit = 20
        init_game()
        game_over = False
if game_started and not game_over:
    screen.fill(WHITE)
    draw_grid()
    draw_agent()
    draw_goals()
    draw_silver_coins()
    draw_obstacles()
    draw_door()
    draw_diamond()
    display_info_panel()
if game_over:
    font = pygame.font.SysFont(None, 55)
    game_over_text = font.render("Game Over ! Press 'R' to Retry", True, RED)
    screen.blit(game_over_text, [WIDTH // 4, HEIGHT // 2])
pygame.display.flip()
```

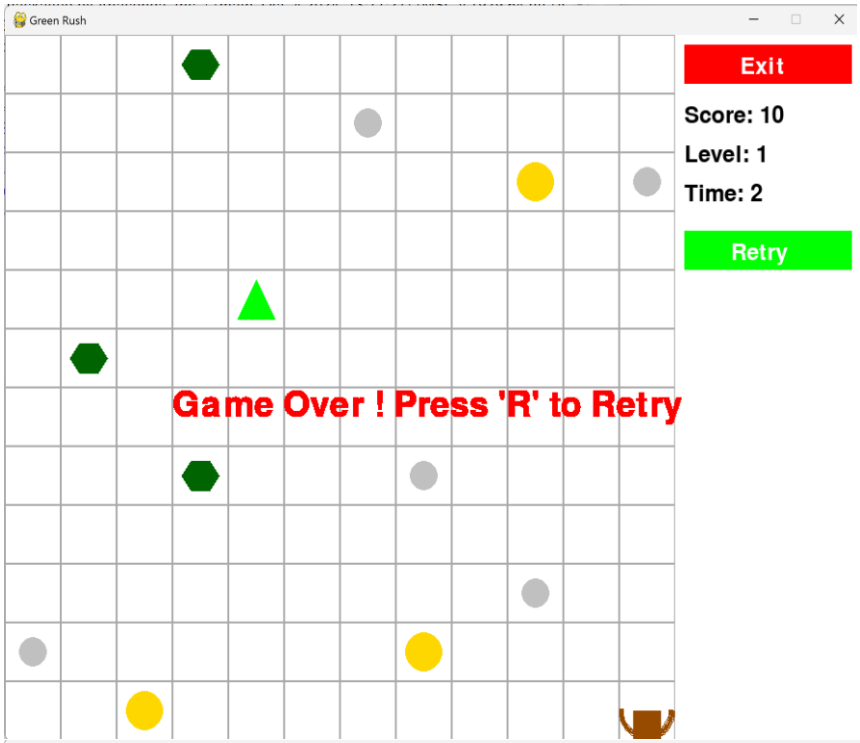
SCREENSHOTS:

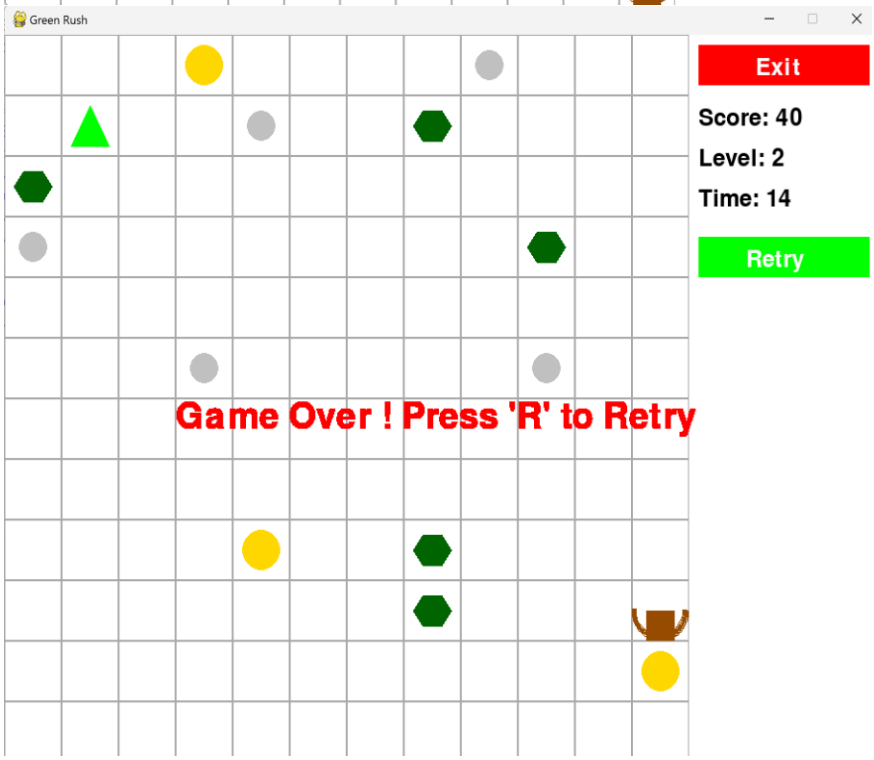
I. For every level there are obstacles which are hidden and some are revealed and there will be checkpoint like a door(checkpoint) if you want to go to next level , or else you can comllect all the coins to got to the next level

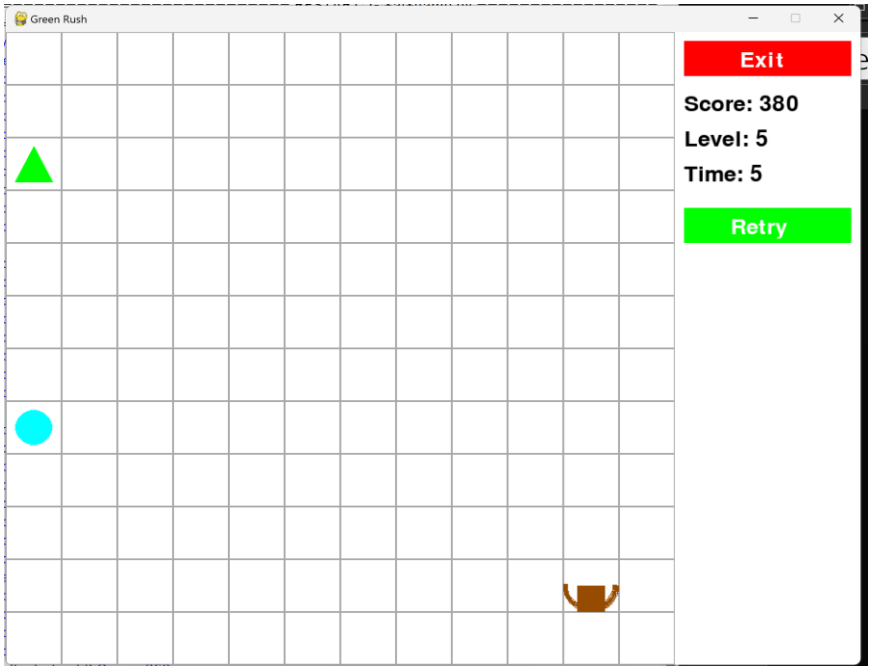
II. For every 5 levels there is a bonus level without any obstacles



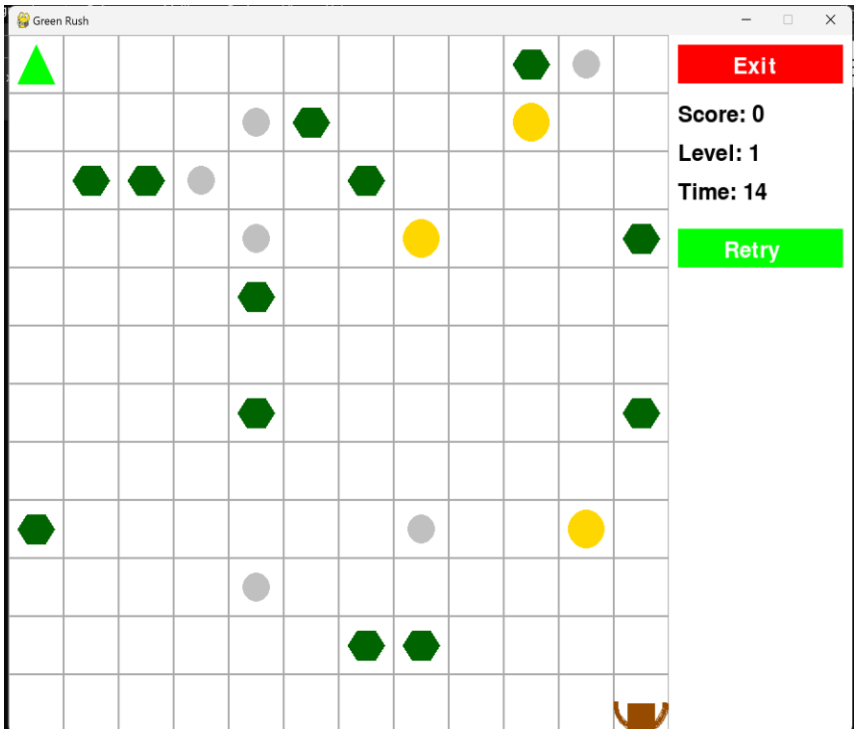


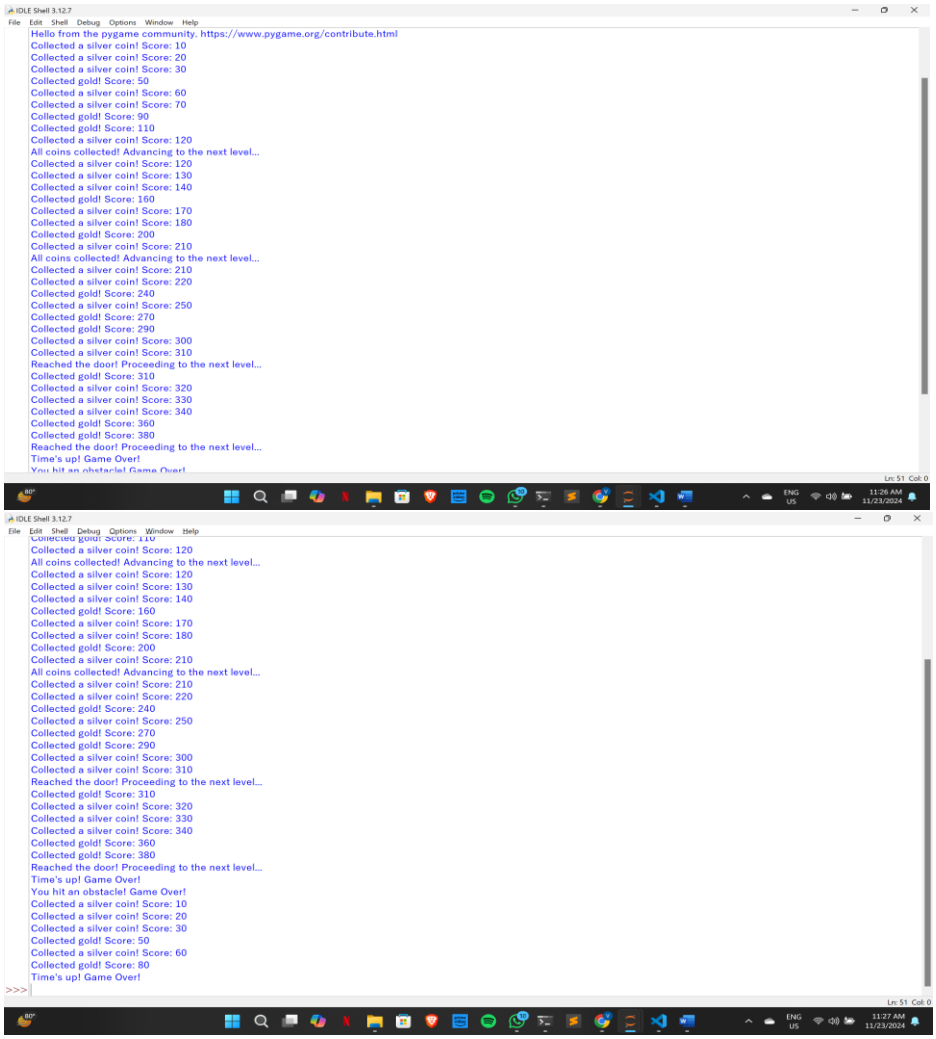

Green Rush
Exit
Score: 10
Level: 1
Time: 2
Retry
Game Over ! Press 'R' to Retry


Green Rush
Exit
Score: 40
Level: 2
Time: 14
Retry
Game Over ! Press 'R' to Retry



If obstacles are not hidden then:





Google drive link:

https://drive.google.com/file/d/1Wgc6g8CukYUxY0lSitLphDIp6lIr-Rjz/view?usp=sharing