```
import cv2
import numpy as np
import pygame
import random
import mediapipe as mp
from tkinter import *

# Initialize Pygame
pygame.init()

# Set screen dimensions
screen_width = 1080
screen_height = 720

# Create Pygame screen
screen = pygame.display.
set_mode((screen_width,
screen_height))
pygame.display.set_caption("
Gesture Racing Game")

# Load background image
background_image = pygame.
image.load(r"C:\Users\rkssp\
Desktop\virtual envi\republic\
patriatic\road (1).jpg").
convert()
```

```python

# Load car image
car_image = pygame.image.load(
r"C:\Users\rkssp\Downloads\
car_top (3).jpg").
convert_alpha()

# Load coin image
coin_image = pygame.image.load
(r"C:\Users\rkssp\Downloads\
coin_ (1).jpg").convert_alpha
()

# Colors
WHITE = (255, 255, 255)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)

# Car attributes
car_width = car_image.
get_width()
car_height = car_image.
get_height()
car_x = screen_width // 2 -
car_width // 2
```

```python
car_y = screen_height // 2 - car_height // 2
car_speed = 5

# Ball attributes
ball_radius = 20
ball_speed = 3
blue_balls = []
red_balls = []
coins = []

# Score
score = 0
font = pygame.font.Font(None, 36)

# OpenCV settings
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1)
mp_drawing = mp.solutions.drawing_utils
cap = cv2.VideoCapture(0)  # Use the default camera

# Initialize face detection
```

```
mp_face_detection = mp.solutions.face_detection
face_detection = mp_face_detection.FaceDetection(min_detection_confidence=0.5)

# Define hand positions
HAND_LEFT = 'left'
HAND_RIGHT = 'right'
HAND_UP = 'up'
HAND_DOWN = 'down'

# Define head positions
HEAD_LEFT = 'left'
HEAD_RIGHT = 'right'
HEAD_UP = 'up'
HEAD_DOWN = 'down'

# Initialize webcam position and dimensions
webcam_x = 20
webcam_y = 20
webcam_width = 160
webcam_height = 120

# Variable to track mouse drag
```

```python
 state
is_dragging = False


# Function to check hand position
def check_hand_position(hand_landmarks):
    if hand_landmarks:
        for hand_landmark in hand_landmarks:
            landmarks = hand_landmark.landmark
            x_values = [landmark.x for landmark in landmarks]
            y_values = [landmark.y for landmark in landmarks]
            center_x = (max(x_values) + min(x_values)) / 2
            center_y = (max(y_values) + min(y_values)) / 2

            if center_x < 0.4
```

```python
:
                return HAND_LEFT
            elif center_x > 0.6:
                return HAND_RIGHT
            elif center_y < 0.4:
                return HAND_UP
            elif center_y > 0.6:
                return HAND_DOWN
    return None


# Function to check head position
def check_head_position(face_detections):
    if face_detections.detections:
        for detection in face_detections.detections:
            bboxC = detection
```

```
.location_data.relative_bounding_box
            cx = int(bboxC.xmin * screen_width + bboxC.width * screen_width / 2)
            cy = int(bboxC.ymin * screen_height + bboxC.height * screen_height / 2)

            if cx < 0.4 * screen_width:
                return HEAD_LEFT
            elif cx > 0.6 * screen_width:
                return HEAD_RIGHT
            elif cy < 0.4 * screen_height:
                return HEAD_UP
            elif cy > 0.6 * screen_height:
                return HEAD_DOWN
    return None

```

```

# Function to create a new blue ball
def create_blue_ball():
    ball_x = screen_width
    ball_y = random.randint(ball_radius, screen_height - ball_radius)
    return {'x': ball_x, 'y': ball_y}


# Function to create a new red ball
def create_red_ball():
    ball_x = screen_width
    ball_y = random.randint(ball_radius, screen_height - ball_radius)
    return {'x': ball_x, 'y': ball_y}


# Function to create a new coin
def create_coin():
    coin_x = screen_width
```

```python
    coin_y = random.randint(ball_radius, screen_height - ball_radius)
    return {'x': coin_x, 'y': coin_y}


# Function to start the selected game mode
def start_game(mode):
    if mode == "Hand Gesture Control":
        hand_gesture_control()
    elif mode == "Head Gesture Control":
        head_gesture_control()


# Function for hand gesture control
def hand_gesture_control():
    global car_x, car_y, score, blue_balls, red_balls, coins, is_dragging, webcam_x, webcam_y

    running = True
    clock = pygame.time.Clock()

    while running:
        screen.blit(background_image, (0, 0))

        ret, frame = cap.read()
        if not ret:
            break

        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = hands.process(rgb_frame)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            elif event.type
```

```
== pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    running = False

        hand_position = check_hand_position(results.multi_hand_landmarks)

        if hand_position == HAND_LEFT:
            car_x -= car_speed
            if car_x < 0:
                car_x = 0
        elif hand_position == HAND_RIGHT:
            car_x += car_speed
            if car_x > screen_width - car_width:
                car_x = screen_width - car_width
        elif hand_position == HAND_UP:
            car_y -=
car_speed
            if car_y < 0:
                car_y = 0
        elif hand_position == HAND_DOWN:
            car_y += car_speed
            if car_y > screen_height - car_height:
                car_y = screen_height - car_height

        if random.randint(0, 100) < 7:
            blue_balls.append(create_blue_ball())
        if random.randint(0, 100) < 5:
            red_balls.append(create_red_ball())
        if random.randint(0, 1000) < 1:
            coins.append(create_coin())

        for ball in blue_balls:
```

```python
            ball['x'] -= ball_speed
            pygame.draw.circle(screen, BLUE, (ball['x'], ball['y']), ball_radius)

        for ball in red_balls:
            ball['x'] -= ball_speed
            pygame.draw.circle(screen, RED, (ball['x'], ball['y']), ball_radius)

        for coin in coins:
            coin['x'] -= ball_speed
            screen.blit(coin_image, (coin['x'], coin['y']))

        for ball in blue_balls:
            if car_x < ball['x'] < car_x + car_width and car_y < ball['y'] < car_y + car_height:
                score += 5
                blue_balls.remove(ball)
        for ball in red_balls:
            if car_x < ball['x'] < car_x + car_width and car_y < ball['y'] < car_y + car_height:
                score -= 10
                red_balls.remove(ball)
        for coin in coins:
            if car_x < coin['x'] < car_x + car_width and car_y < coin['y'] < car_y + car_height:
                score += 20
                coins.remove(coin)

        screen.blit(car_image, (car_x, car_y))

        score_text = font.render("Score: " + str(score), True, BLACK)
        screen.blit(score_text, (10, 10))

        # Blit webcam frame onto the screen
        display_webcam_feed()

        pygame.display.flip()
        clock.tick(30)

    cap.release()
    cv2.destroyAllWindows()


# Function for head gesture control
def head_gesture_control():
    global car_x, car_y, score, blue_balls, red_balls, coins, is_dragging, webcam_x, webcam_y

    running = True
    clock = pygame.time.Clock()

    while running:
        screen.blit(background_image, (0, 0))

        ret, frame = cap.read()
        if not ret:
            break

        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        face_detections = face_detection.process(rgb_frame)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    running = False
```

```python

        head_position = check_head_position(face_detections)

        if head_position == HEAD_LEFT:
            car_x -= car_speed
            if car_x < 0:
                car_x = 0
        elif head_position == HEAD_RIGHT:
            car_x += car_speed
            if car_x > screen_width - car_width:
                car_x = screen_width - car_width
        elif head_position == HEAD_UP:
            car_y -= car_speed
            if car_y < 0:
                car_y = 0
        elif head_position == HEAD_DOWN:
            car_y += car_speed
            if car_y > screen_height - car_height:
                car_y = screen_height - car_height

        if random.randint(0, 100) < 7:
            blue_balls.append(create_blue_ball())
        if random.randint(0, 100) < 5:
            red_balls.append(create_red_ball())
        if random.randint(0, 1000) < 1:
            coins.append(create_coin())

        for ball in blue_balls:
            ball['x'] -= ball_speed
            pygame.draw.circle(screen, BLUE, (ball['x'], ball['y']), ball_radius)
```

```python

        for ball in red_balls:
            ball['x'] -= ball_speed
            pygame.draw.circle(screen, RED, (ball['x'], ball['y']), ball_radius)

        for coin in coins:
            coin['x'] -= ball_speed
            screen.blit(coin_image, (coin['x'], coin['y']))

        for ball in blue_balls:
            if car_x < ball['x'] < car_x + car_width and car_y < ball['y'] < car_y + car_height:
                score += 5
                blue_balls.remove(ball)
        for ball in red_balls:
            if car_x < ball['x'] < car_x + car_width and car_y < ball['y'] < car_y + car_height:
                score -= 10
                red_balls.remove(ball)
        for coin in coins:
            if car_x < coin['x'] < car_x + car_width and car_y < coin['y'] < car_y + car_height:
                score += 20
                coins.remove(coin)

        screen.blit(car_image, (car_x, car_y))

        score_text = font.render("Score: " + str(score), True, BLACK)
        screen.blit(score_text, (10, 10))

        # Blit webcam frame onto the screen
        display_webcam_feed()

        pygame.display.flip()
        clock.tick(30)

    cap.release()
    cv2.destroyAllWindows()


# Function to capture webcam frame and display in Pygame window
def display_webcam_feed():
    global is_dragging, webcam_x, webcam_y

    # Capture a frame from the webcam
    ret, frame = cap.read()
    if ret:
        # Rotate the frame by 90 degrees counterclockwise
        rotated_frame = cv2.rotate(frame, cv2.ROTATE_90_COUNTERCLOCKWISE)

        # Convert the frame
```

```python
to RGB format
        rgb_frame = cv2.cvtColor(rotated_frame, cv2.COLOR_BGR2RGB)

        # Resize the frame to match the dimensions of the webcam surface
        resized_frame = cv2.resize(rgb_frame, (webcam_width, webcam_height))

        # Convert the resized frame to a Pygame surface
        webcam_surface = pygame.surfarray.make_surface(resized_frame)

        # Blit the webcam surface onto the screen
        screen.blit(webcam_surface, (webcam_x, webcam_y))

        # Handle mouse events for dragging the webcam feed
        mouse_x, mouse_y =
pygame.mouse.get_pos()
        mouse_click = pygame.mouse.get_pressed()

        if webcam_x < mouse_x < webcam_x + webcam_width and webcam_y < mouse_y < webcam_y + webcam_height:
            if mouse_click[0]:
                is_dragging = True
        elif not mouse_click[0]:
            is_dragging = False

        if is_dragging:
            webcam_x, webcam_y = mouse_x - webcam_width // 2, mouse_y - webcam_height // 2
        else:
            is_dragging = False

    else:
        print("Error: Unable to capture frame from the webcam")

# GUI Popup to select game mode
def select_game_mode():
    root = Tk()
    root.title("Select Game Mode")

    def start_hand_gesture():
        root.destroy()
        start_game("Hand Gesture Control")

    def start_head_gesture():
        root.destroy()
        start_game("Head Gesture Control")

    hand_button = Button(root, text="Hand Gesture Control", command=start_hand_gesture)
    hand_button.pack()

    head_button = Button(root
, text="Head Gesture Control", command=start_head_gesture)
    head_button.pack()

    root.mainloop()

# Run the game mode selection GUI
select_game_mode()
```