Documentación para el Desarrollo del Backend Checogame

(IA- reconocimiento facial)

Desarrollo de la lógica de negocio y algoritmos:

1. **Inicialización de Facemesh**: Se inicializa la detección de rostros utilizando la biblioteca MediaPipe. Esto implica configurar el modelo de FaceMesh con los parámetros necesarios.
2. **Procesamiento de la cámara**: En el método **process_camera()**, se captura una imagen de la cámara, se la prepara y se la pasa al modelo de FaceMesh para detectar los landmarks faciales. Luego, se calculan las coordenadas de la cara y se determina el movimiento de la cabeza basado en el ángulo entre dos puntos en el rostro.
3. **Detección de movimiento de la cabeza**: Utilizando las coordenadas de los landmarks faciales, se calcula el ángulo de inclinación de la cabeza. Este ángulo se normaliza en un rango de -1 a 1, donde -1 representa una inclinación hacia abajo y 1 una inclinación hacia arriba.
4. **Renderización de la cámara**: En el método **render_camera()**, se dibuja un rectángulo alrededor de la cara detectada y se escala para que tenga un tamaño consistente. Luego, se muestra en la pantalla.

Implementación de las Funciones de Procesamiento de Datos

En el backend del juego, las funciones de procesamiento de datos se encargan de manejar la entrada de datos proveniente de la cámara web, procesar estas imágenes para detectar landmarks faciales utilizando el modelo FaceMesh de MediaPipe, y luego utilizar esta información para realizar acciones específicas dentro del juego. Aquí se detalla cómo se han implementado estas funciones:

1. **Inicialización de Facemesh:**
    - Al iniciar el juego, se inicializa el modelo de FaceMesh de MediaPipe. Esto se realiza en el método **_init_** de la clase **Game**.
    - Se asegura de que los parámetros del modelo estén configurados adecuadamente para el seguimiento facial en tiempo real.
2. **Procesamiento de Datos de la Cámara:**
    - El método **process_camera** se encarga de capturar imágenes de la cámara web y procesarlas para detectar landmarks faciales.
    - Se utiliza la biblioteca OpenCV (**cv2**) para capturar imágenes de la cámara web y realizar operaciones de manipulación de imágenes.
    - Las imágenes se convierten de BGR a RGB, ya que MediaPipe espera imágenes en este formato para la detección facial.

    - Se utiliza el modelo FaceMesh previamente inicializado para procesar la imagen y detectar los landmarks faciales.
    - Si se detectan landmarks faciales en la imagen, se almacenan las coordenadas relevantes para su posterior uso en el juego.
3. **Integración con la Lógica del Juego:**
    - Una vez que se han detectado los landmarks faciales, la información se utiliza en la lógica del juego para controlar el movimiento del jugador u otras acciones.
    - Por ejemplo, la detección del movimiento de la cabeza puede traducirse en movimientos del jugador en la pantalla del juego.
    - La implementación específica de esta integración depende de cómo se haya diseñado la lógica del juego y cómo se utilicen los datos de los landmarks faciales en ella.

Manejo de las Solicitudes y Respuestas del Cliente:

En el backend del juego, se ha implementado un sistema de detección de movimiento facial para permitir que el jugador controle el movimiento del carro dentro del juego. Aquí se describe cómo se ha implementado este proceso:

1. **Solicitudes del Cliente:**
    - El cliente envía solicitudes al backend del juego para controlar el movimiento del carro utilizando la detección de movimientos faciales.
    - La solicitud del cliente consiste en identificar el movimiento de la cabeza del jugador a través de la cámara web y reflejar este movimiento en el juego.
2. **Respuestas del Cliente:**
    - El backend procesa las imágenes capturadas por la cámara web del jugador utilizando el modelo de detección facial FaceMesh de MediaPipe.
    - Se identifican los landmarks faciales relevantes para determinar la posición y el movimiento de la cabeza del jugador.
    - El movimiento detectado se traduce en acciones de movimiento del carro en el juego.
3. **Implementación en el Código:**
    - En el método **process_camera**, se procesan las imágenes capturadas por la cámara web para detectar los landmarks faciales del jugador.
    - Se identifican los landmarks específicos que representan la posición de la cabeza, como el centro de la frente y la barbilla.
    - Utilizando la posición de estos landmarks, se calcula el ángulo de inclinación de la cabeza del jugador.

- El ángulo de inclinación se traduce en acciones de movimiento del carro en el juego, como moverse hacia la izquierda, hacia la derecha o mantenerse en la misma posición.
- Las acciones de movimiento del carro se actualizan en tiempo real según el ángulo de inclinación detectado, lo que permite al jugador controlar el carro simplemente moviendo la cabeza.

Pruebas Unitarias:

1. **Pruebas del Método detect_head_movement:**
   - Se han diseñado casos de prueba para verificar la precisión del cálculo del ángulo de inclinación de la cabeza a partir de los landmarks faciales detectados.
   - Se han creado casos de prueba para cubrir diversos escenarios, como movimientos de cabeza hacia la izquierda, hacia la derecha y movimientos neutros.
2. **Pruebas del Método process_camera:**
   - Se han desarrollado casos de prueba para evaluar el correcto funcionamiento del procesamiento de imágenes de la cámara web.
   - Se han simulado diferentes condiciones de iluminación y posiciones de la cabeza para verificar la robustez del sistema de detección facial.

Pruebas de Integración:

1. **Integración con el Juego:**
   - Se ha integrado el sistema de detección de movimiento facial con la lógica del juego para permitir el control del carro.
   - Se han realizado pruebas para verificar la sincronización adecuada entre los movimientos detectados y las acciones del carro en el juego.
   - Se ha comprobado que el sistema responde de manera coherente y precisa a los movimientos faciales del jugador en tiempo real.
2. **Integración con el Backend:**
   - Se ha asegurado que el sistema de detección de movimiento facial esté correctamente integrado con el resto del backend del juego.
   - Se han verificado las interfaces de comunicación entre los diferentes componentes del backend para garantizar una interacción fluida y eficiente.

Resultados de las Pruebas:

- Las pruebas unitarias han confirmado que el método **detect_head_movement** calcula con precisión el ángulo de inclinación de la cabeza en una variedad de situaciones.
- Las pruebas de integración han demostrado que el sistema de detección de movimiento facial se integra perfectamente con el juego y responde de manera adecuada a los movimientos faciales del jugador.
- Se ha validado que el sistema funciona de manera consistente y confiable, proporcionando una experiencia de juego inmersiva y satisfactoria.

Estas pruebas han sido fundamentales para garantizar la calidad y la fiabilidad del sistema de detección de movimiento facial, asegurando que el juego pueda ser disfrutado sin problemas por los usuarios finales.

Código fuente:

```
import cv2

import mediapipe as mp

import math

class Game:

    def _init_(self):

        ...

        # Inicialización de Facemesh(IA)

        self.mp_face_mesh = mp.solutions.face_mesh

        self.mp_drawing = mp.solutions.drawing_utils

        self.mp_drawing_styles = mp.solutions.drawing_styles

        ...

        # Inicializa las variables con el seguimiento facial para detectar movimientos faciales del jugador

        self.max_face_surf_height = 0

        self.face_left_x = 0

        self.face_right_x = 0

        self.face_top_y = 0
```

```
    self.face_bottom_y = 0
    ...

def loop(self):
    ...
    # Inicializa el modelo FaceMesh
    with self.mp_face_mesh.FaceMesh(
            static_image_mode=False,
            max_num_faces=1,
            min_detection_confidence=0.5,
            refine_landmarks=True
    ) as self.face_mesh:
        while self.running:
            # Verifica si el jugador aún no ha perdido
            if not self.lost:
                # Verifica si la cámara web está lista para ser utilizada
                if not self.webcam.ready():
                    continue
                # Seguimiento facial
                self.process_camera()
            else:
                # Detener la música cuando el jugador pierda
                self.background_music.stop()
            ...
def process_camera(self):
    image = self.webcam.read()
    if image is not None:
        image.flags.writeable = False
```

```
            image = cv2.flip(image, 1)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            image.flags.writeable = True
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
            results = self.face_mesh.process(image)
            self.webcam_image = image
            if results.multi_face_landmarks is not None:
                for face_landmarks in results.multi_face_landmarks:

                    # Coordenadas de la cara (arriba y abajo)
                    top = (face_landmarks.landmark[10].x, face_landmarks.landmark[10].y)
                    bottom = (face_landmarks.landmark[152].x,
face_landmarks.landmark[152].y)

                    # Obtener coordenadas del 'cuadrado' de la cara para poder mostrarlo en la
pantalla después
                    self.face_left_x = face_landmarks.landmark[234].x
                    self.face_right_x = face_landmarks.landmark[454].x
                    self.face_top_y = face_landmarks.landmark[10].y
                    self.face_bottom_y = face_landmarks.landmark[152].y

                    # Dejar algo de espacio alrededor
                    self.face_left_x = self.face_left_x - .1
                    self.face_right_x = self.face_right_x + .1
                    self.face_top_y = self.face_top_y - .1
                    self.face_bottom_y = self.face_bottom_y + .1

                    # Detección de ángulo
                    self.detect_head_movement(top, bottom)
```

```
        k = cv2.waitKey(1) & 0xFF

    def detect_head_movement(self, top, bottom):
        radians = math.atan2(bottom[1] - top[1], bottom[0] - top[0])
        degrees = math.degrees(radians)

        # Ángulo de detección de 70 a 110 (-1 a 1)
        min_degrees = 70
        max_degrees = 110
        degree_range = max_degrees - min_degrees

        if degrees < min_degrees: degrees = min_degrees
        if degrees > max_degrees: degrees = max_degrees

        self.movement = (((degrees - min_degrees) / degree_range) * 2) - 1

    def render_camera(self):
        # Limpiar coordenadas del cuadro de la cara
        if self.face_left_x < 0: self.face_left_x = 0
        if self.face_right_x > 1: self.face_right_x = 1
        if self.face_top_y < 0: self.face_top_y = 0
        if self.face_bottom_y > 1: self.face_bottom_y = 1

        face_surf = pygame.image.frombuffer(self.webcam_image,
(int(self.webcam.width()), int(self.webcam.height())), "BGR")

        face_rect = pygame.Rect(
```

```
            int(self.face_left_x * self.webcam.width()),
            int(self.face_top_y * self.webcam.height()),
            int(self.face_right_x * self.webcam.width()) - int(self.face_left_x *
self.webcam.width()),
            int(self.face_bottom_y * self.webcam.height()) - int(self.face_top_y *
self.webcam.height())
        )

        only_face_surf = pygame.Surface((
            int(self.face_right_x * self.webcam.width()) - int(self.face_left_x *
self.webcam.width()),
            int(self.face_bottom_y * self.webcam.height()) - int(self.face_top_y *
self.webcam.height())
        ))
        only_face_surf.blit(face_surf, (0, 0), face_rect)

        height = only_face_surf.get_rect().height
        width = only_face_surf.get_rect().width
        if width == 0:
            width = 1
        face_ratio = height / width
        face_area_width = 130
        face_area_height = face_area_width * face_ratio
        if (face_area_height > self.max_face_surf_height):
            self.max_face_surf_height = face_area_height
        only_face_surf = pygame.transform.scale(only_face_surf, (int(face_area_width),
int(self.max_face_surf_height)))
        self.screen.blit(only_face_surf, only_face_surf.get_rect())
```

Requisitos del Sistema:

- **Dependencias:**
  - Python 3.11
  - OpenCV (cv2)
  - Mediapipe
  - Numpy

Requisitos Previos:

1. Asegúrese de tener instalado Python en su sistema. Puede descargarlo desde python.org.
2. Instale las bibliotecas necesarias ejecutando el siguiente comando en su terminal:

Copiar código

pip install mediapipe opencv-python

3. Descargue el código proporcionado del sistema de detección de movimiento facial.

Instrucciones de Ejecución:

1. Navegue hasta el directorio donde ha guardado el código del sistema de detección de movimiento facial.
2. Abra un terminal en este directorio.
3. Ejecute el siguiente comando para iniciar el sistema:

Copiar código

python nombre_del_archivo.py

Reemplace **nombre_del_archivo.py** con el nombre del archivo principal del código proporcionado.

4. Una vez que el sistema esté en funcionamiento, asegúrese de tener una cámara web conectada a su computadora.

Posicione su rostro frente a la cámara web para que el sistema pueda detectar sus movimientos faciales.