Ingeniería Informática en Ingeniería del Software

Computer Vision con Python y OpenCV

IMAGEN DIGITAL

Víctor Andrés Navareño Moza

ÍNDICE

1. Objetivos del proyecto

Este proyecto en **Python con OpenCV tiene como objetivo principal desarrollar una aplicación interactiva de procesamiento de imágenes que combine varios aspectos tecnológicos y prácticos**. El proyecto busca **profundizar en los fundamentos de Python aplicados al procesamiento de imágenes**, al tiempo que se exploran conceptos y técnicas avanzadas en este campo.

Un objetivo clave es demostrar el dominio de ***OpenCV*** para implementar funcionalidades de seguimiento de movimiento en tiempo real, complementado con el uso de *MediaPipe* para el reconocimiento y **rastreo de puntos clave faciales, específicamente la nariz.**

Objetivos principales del proyecto:

- ☐ **Aprender Python**
- ☐ **Entender procesamiento de imágenes**
- ☐ **Demostrar conocimientos OpenCV**
- ☐ **Aprender a utilizar MediaPipe**
- ☐ **Aplicar una solución en el mundo real**

2. Presentación del proyecto a realizar

El proyecto consiste en el **desarrollo de un juego sencillo en Python, utilizando las bibliotecas OpenCV y *MediaPipe* para el reconocimiento y seguimiento de movimientos faciales en tiempo real.**

Librería MediaPipe

En este reto, el jugador es un planeta que debe esquivar asteroides que aparecen en pantalla. **La posición del planeta se controla mediante el rastreo de la nariz del jugador, detectada por la cámara del dispositivo.** El objetivo del juego es evitar colisiones con los asteroides para sobrevivir el mayor número de rondas posible.

Captura del Juego

Objetivo del Juego: El propósito del juego es **esquivar los asteroides** durante el mayor tiempo posible, promoviendo la coordinación entre el movimiento de la cabeza y el control visual. Este proyecto representa una aplicación real de tecnologías de procesamiento de imágenes y reconocimiento facial en una experiencia interactiva y divertida.

Funcionalidades del Proyecto:

- ★ **Detección y Seguimiento Facial:**
  - Utilización de *MediaPipe* para identificar y rastrear puntos clave faciales, en nuestro caso la nariz.
  - Mapeo de la posición de la nariz en la pantalla para mover el planeta de forma dinámica en el eje horizontal y vertical.
- ★ **Interacción en Tiempo Real:**
  - Actualización en tiempo real de la posición del planeta, lo que permite al usuario mover la cabeza para esquivar los asteroides.
- ★ **Generación y Movimiento de Asteroides:**
  - Creación de asteroides que aparecen de forma aleatoria en diferentes puntos de la pantalla y se desplazan en direcciones aleatorias.
- ★ **Detección de Colisiones:**
  - Implementación de un sistema que detecta colisiones entre el planeta y los asteroides.Cuando ocurre una colisión, finaliza el juego.
- ★ **Sistema de Rondas:**
  - Registro del tiempo que el jugador permanece en el juego sin chocar con los asteroides. Cada 15 segundos se avanza a la siguiente ronda, añadiendo 2 asteroides a la pantalla.

3. Aspectos técnicos del proyecto

Estas son algunas **explicaciones de trozos de código importantes** en la realización del juego. Todo el código fuente completo, cedido bajo licencia *MIT,* puede ser revisado, descargado y modificado añadiendo una nueva rama en **mi perfil personal de GitHub** [victornavareno](https://github.com/victornavareno) *(doktor)* de forma pública:

*Link al proyecto en **GitHub**:*
https://github.com/victornavareno/FaceTrackingGame

En el link anterior se puede encontrar el **sistema de control de versiones**, con todo el **listado de commits y funcionalidades añadidas** desde la creación de la versión base.

Importando las librerías que usaremos:

Lo primero es importar las librerías que usaré en el *main.py* de este proyecto:

```
import pygame
import sys
import cv2
import numpy as np
import mediapipe as mp
```

- **pygame:** Biblioteca para crear juegos y aplicaciones multimedia en Python.
- **sys:** Proporciona acceso a algunas variables y funciones utilizadas o mantenidas por el intérprete de Python.
- **cv2:** Interfaz de Python para OpenCV, utilizada para procesamiento de imágenes y visión por computadora.
- **numpy:** Biblioteca para computación numérica en Python, especialmente útil para operaciones con matrices y arrays.
- **mediapipe:** Framework de Google para construir aplicaciones de machine learning para procesamiento de multimedia.

Captura de la webcam:

Snippet para realizar la captura de input desde la webcam de nuestro equipo:

```
cap = cv2.VideoCapture(0)  # 0 se refiere a la webcam
```

Inicializo el modelo de detección de rostros con *MediaPipe*:
La siguiente línea define de forma simple una variable que almacenará parámetros de un modelo entrenado para la **detección de rostros a partir del input de la webcam**.

```
    mp_face_mesh = mp.solutions.face_mesh
    face_mesh = mp_face_mesh.FaceMesh(static_image_mode=False,
    max_num_faces=1, min_detection_confidence=0.5)
```

Definición de la clase base Asteroide
Esta clase **representa los enemigos (Asteroides)** que aparecen en la pantalla cada ronda. Su velocidad y posición es asignada de forma aleatoria al instanciarlos en el constructor.

```
class Asteroide:
    def __init__(self):
        # Posicion y velocidad inicial random
        self.x = random.randint(0, WIDTH - ENEMY_SIZE)
        self.y = random.randint(0, HEIGHT - ENEMY_SIZE)
        self.vx = random.choice([-3, -2, 2, 3])
        self.vy = random.choice([-3, -2, 2, 3])

    def move(self):
        self.x += self.vx
        self.y += self.vy
        # Colision con los muros para rebotar
        if self.x <= 0 or self.x >= WIDTH - ENEMY_SIZE:
            self.vx = -self.vx
        if self.y <= 0 or self.y >= HEIGHT - ENEMY_SIZE:
           self.vy = -self.vy
```

Definición del collider de los asteroides:
Aquí defino un sistema simple de detección de colisiones entre el planeta trackeado por *MediaPipe* y los **asteroides enemigos, generados de forma aleatoria:**

```
def check_collisions(player_pos, asteroids):
    for asteroid in asteroids:
        if detect_collision(player_pos, asteroid.get_position()):
            return True
    return False
```

Implementación del sistema de rondas:

El siguiente método se encuentra en el flujo principal del programa. Tras mostrar la pantalla de carga del juego, inicializa el número de ronda a 1, y comienza el juego dentro de un bucle de rondas.

```
display_loading_screen()
round_number = 1

# Comienzan las rondas hasta que el jugador muera
while True:
    survived = run_round(round_number)
    if not survived:
        break
    round_number += 1
```

Trackeo de la nariz utilizando *MediaPipe*

Las siguientes líneas de código actualizan la posición *x* e *y* del jugador, utilizando el acceso a landmarks de *MediaPipe* definido utilizando *face_mesh* de la librería.

```
def handle_player_position():
   nose_tip = face_landmarks.landmark[1]
   nose_x = int((1 - nose_tip.x) * WIDTH)
   nose_y = int(nose_tip.y * HEIGHT)
```

Reescalado de la imagen captado por la webcam

Una vez captado el input de la webcam utilizando ***cv2.VideoCapture(0)***, procedemos a **reescalar esta imagen para poder procesarla** y mostrarla correctamente en la screen de nuestro juego**, utilizando el método *resize()*.**

```
# Reescalado de la webcam:
        frame = cv2.resize(frame, (WIDTH, HEIGHT))
```

4. Conclusión

En conclusión, creo que este proyecto desarrollado para la asignatura de **Imagen Digital** representa una **síntesis innovadora de tecnologías de visión por computadora y programación interactiva**. La aplicación creada no solo me ha ayudado a adquirir un mayor dominio técnico de herramientas como OpenCV y *MediaPipe*, sino que también me ha ayudado a desarrollar la capacidad de aplicar conceptos teóricos de procesamiento de imágenes en un contexto práctico y lúdico, **además de aprender muchísimo sobre Python, que es un lenguaje que llevaba tiempo con ganas de aprender y extremadamente solicitado en el panorama laboral hoy en día.**

El resultado final, un juego interactivo controlado por movimientos faciales, ilustra de manera efectiva cómo las tecnologías de imagen digital pueden ser utilizadas para crear experiencias de usuario innovadoras y atractivas. **Este proyecto desafía a los estudiantes a pensar de manera creativa sobre las aplicaciones prácticas del computer vision** en entornos algo más desenfadados y con aplicaciones reales a nuestro entorno. **Además, me lo he pasado muy bien desarrollándolo.**