**Nombre: Miguel Esteban Pinilla Leal**

**Código: 20191005036**

<!DOCTYPE html>

<html lang="en">

<!-- Entrega primer actividad: Miguel Esteban Pinilla Leal Codigo: 20191005036 -->

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Detección de Mano</title>

  <!-- Importación de MediaPipe y Utilidades para Dibujo -->

  <script src="https://cdn.jsdelivr.net/npm/@mediapipe/drawing\_utils/drawing\_utils.js" crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/@mediapipe/hands/hands.js" crossorigin="anonymous"></script>

  <style>

    /\* Estilo para centrar el video y el canvas en la página \*/

    body { display: flex; justify-content: center; align-items: center; height: 100vh; margin: 0; }

    video, canvas { position: absolute; transform: rotateY(180deg); } /\* Voltear video y canvas \*/

  </style>

</head>

<body>

  <!-- Elemento de video con el video pre-cargado -->

  <video id="videoElemento" autoplay playsinline loop>

    <source src="mano\_bien\_4.mp4" type="video/mp4">  <!-- Reemplaza con tu video -->

    Tu navegador no soporta el formato de video.

  </video>

  <!-- Canvas para dibujar los resultados -->

  <canvas id="canvasResultado"></canvas>

  <script type="module">

    import { HandLandmarker, FilesetResolver } from "https://cdn.jsdelivr.net/npm/@mediapipe/tasks-vision@0.10.0";

    let marcadorMano;

    let modoEjecucion = "VIDEO";

    let tiempoUltimoVideo = -1;

    // Función asincrónica para inicializar el marcador de manos

    const inicializarMarcadorMano = async () => {

      const vision = await FilesetResolver.forVisionTasks("https://cdn.jsdelivr.net/npm/@mediapipe/tasks-vision@0.10.0/wasm");

      marcadorMano = await HandLandmarker.createFromOptions(vision, {

        baseOptions: {

          modelAssetPath: "https://storage.googleapis.com/mediapipe-models/hand\_landmarker/hand\_landmarker/float16/1/hand\_landmarker.task",

          delegate: "GPU"

        },

        runningMode: modoEjecucion,

        numHands: 2

      });

    };

    inicializarMarcadorMano(); // Inicializar el marcador de manos

    const videoElemento = document.getElementById("videoElemento");

    const canvasResultado = document.getElementById("canvasResultado");

    const contextoCanvas = canvasResultado.getContext("2d");

    // Calculo del coseno entre el rango de las falanges configuradas

    function calcularCoseno(vect1, vect2) {

      const productoPunto = vect1.x \* vect2.x + vect1.y \* vect2.y;

      const magnitud1 = Math.sqrt(vect1.x \* vect1.x + vect1.y \* vect1.y);

      const magnitud2 = Math.sqrt(vect2.x \* vect2.x + vect2.y \* vect2.y);

      return productoPunto / (magnitud1 \* magnitud2);

    }

    // Dibujo de linea entre las falanges

    function dibujarVector(ctx, inicio, fin, color) {

      ctx.beginPath();

      ctx.moveTo(inicio.x \* canvasResultado.width, inicio.y \* canvasResultado.height);

      ctx.lineTo(fin.x \* canvasResultado.width, fin.y \* canvasResultado.height);

      ctx.strokeStyle = color;

      ctx.lineWidth = 2;

      ctx.stroke();

    }

    async function predecirVideo() {

      canvasResultado.width = videoElemento.videoWidth;

      canvasResultado.height = videoElemento.videoHeight;

      if (marcadorMano && videoElemento.currentTime !== tiempoUltimoVideo) {

        tiempoUltimoVideo = videoElemento.currentTime;

        const resultados = await marcadorMano.detectForVideo(videoElemento, performance.now());

        contextoCanvas.clearRect(0, 0, canvasResultado.width, canvasResultado.height);

        if (resultados.landmarks) {

          for (const puntosMano of resultados.landmarks) {

            // Coordenas para el rango de las falanges que se tendra en cuenta

            const baseIndice = puntosMano[6];  // índice

            const puntaIndice = puntosMano[8];

            const baseMedio = puntosMano[10];  //medio

            const puntaMedio = puntosMano[12];

            const baseAnular = puntosMano[14];  // anular

            const puntaAnular = puntosMano[16];

            const baseMeñique = puntosMano[18];  //meñique

            const puntaMeñique = puntosMano[20];

            const vectorIndice = { x: puntaIndice.x - baseIndice.x, y: puntaIndice.y - baseIndice.y };

            const vectorMedio = { x: puntaMedio.x - baseMedio.x, y: puntaMedio.y - baseMedio.y };

            const vectorAnular = { x: puntaAnular.x - baseAnular.x, y: puntaAnular.y - baseAnular.y };

            const vectorMeñique = { x: puntaMeñique.x - baseMeñique.x, y: puntaMeñique.y - baseMeñique.y };

            dibujarVector(contextoCanvas, baseIndice, puntaIndice, "purple");

            dibujarVector(contextoCanvas, baseMedio, puntaMedio, "purple");

            dibujarVector(contextoCanvas, baseAnular, puntaAnular, "purple");

            dibujarVector(contextoCanvas, baseMeñique, puntaMeñique, "purple");

            // Claculo del coseno

            const cosenoIndiceMedio = calcularCoseno(vectorIndice, vectorMedio);

            const cosenoIndiceAnular = calcularCoseno(vectorIndice, vectorAnular);

            const cosenoIndiceMeñique = calcularCoseno(vectorIndice, vectorMeñique);

            //// se quiere que solo si los dedos que no son el indice estan en direcciones opuestas de acuerdo al calculo del coseno,

            /// grafique el circulo amarillo

            // Calculo de la Condición: si el coseno entre el índice y cualquiera de los otros dedos es cercano a -1

            if (Math.abs(cosenoIndiceMedio + 1) < 0.1 || Math.abs(cosenoIndiceAnular + 1) < 0.1 || Math.abs(cosenoIndiceMeñique + 1) < 0.1) {

              const x = puntaIndice.x \* canvasResultado.width;

              const y = puntaIndice.y \* canvasResultado.height;

              contextoCanvas.beginPath();

              contextoCanvas.arc(x, y, 5, 0, 2 \* Math.PI);

              contextoCanvas.lineWidth = 2;

              contextoCanvas.strokeStyle = "yellow";

              contextoCanvas.fillStyle = "transparent";

              contextoCanvas.fill();

              contextoCanvas.stroke();

            }

          }

        }

      }

      requestAnimationFrame(predecirVideo);

    }

    videoElemento.addEventListener("loadeddata", predecirVideo);

  </script>

</body>

</html>