ANEXO · TAREAS DEL ALUMNADO

GESTOS, CARAS Y AR + PRIVACIDAD + ACCESIBILIDAD

Nombre: _____ Fecha: _____

ANTES DE EMPEZAR · MANTENER PARA CONFIRMAR

Tablero **A** de Figma (15012027 · Gestures, face and AR). MediaPipe devuelve, por cada mano, **21 landmarks** y uno de estos gestos, con una puntuación de 0 a 1: `Closed_Fist`, `Open_Palm`, `Pointing_Up`, `Thumb_Down`, `Thumb_Up`, `Victory`, `ILoveYou` o `None`.

Tres protecciones contra el problema del **toque de Midas**: **puntuación mínima** (`MIN_SCORE`), **tiempo de mantenimiento** (`HOLD_MS`) y **una acción por mantenimiento** (`fired`).

1 ¿Puedo usar el gesto «decir adiós con la mano»? ¿Por qué?

2 En el código proyectado, ¿qué variables son `$state` y por qué esas?

3 Mantengo el puño cerrado cinco segundos. ¿Cuántas notas se borran? ¿Cómo la recupero?

TAREA 1 · ESCRIBE EL VOCABULARIO (7 MIN)

Duplicate to your drafts, tablero **C**: de 2 a 4 gestos de la lista A para acciones de tu app, con el motivo de cada gesto y el feedback. Elige tu `HOLD_MS` y tu `MIN_SCORE`.

OK	HAZ	ESTÁ HECHO CUANDO...
☐	Si puedes, el gesto **se parece** a su significado.	Pulgar arriba = «me gusta».
☐	Ninguna acción destructiva en un gesto que se hace sin querer.	`Open_Palm` (toda mano en reposo) no borra nada.
☐	Acción destructiva → mantenimiento más largo y forma de deshacer.	Hay un Undo.
☐	Cada acción es también un botón.	Todo se puede hacer sin cámara.

TAREA 2 · TUS GESTOS (28 MIN)

Nueva página `src/routes/gestures/+page.svelte` en tu proyecto.

OK	HAZ	ESTÁ HECHO CUANDO...
☐	`CameraStage` + `createGestureRecognizer` + tu vocabulario.	El modelo reconoce tus gestos.
☐	La lógica de mantenimiento: puntuación mínima, tiempo y `fired`.	Con el gesto mantenido, la acción se ejecuta **una vez**.
☐	Una etiqueta con un `<progress>` **dentro** de `<CameraStage>...</CameraStage>` (tablero D, izquierda).	La etiqueta y la barra aparecen sobre el vídeo mientras mantienes.
☐	Un `<p role="status">` bajo el vídeo.	Dice «Hold to ...» o «Show a gesture to the camera.»

Si algo falla:

- **La acción se ejecuta muchas veces.** Nunca pones `fired` a true, o lo reinicias en cada frame en vez de solo cuando cambia el gesto.
- **Nunca se ejecuta.** Comparas `progress > 1`; se para justo en 1 por el `Math.min`.
- **El mantenimiento nunca termina.** El gesto parpadea a `None` un frame y reinicia el reloj. Baja un poco `MIN_SCORE` o ilumina mejor la mano. No quites el mantenimiento.
- **La etiqueta sale en espejo.** La dibujaste en el canvas. El texto va en HTML.
- **La barra sigue ahí al apagar la cámara.** Usa `onstop` para reiniciar `label` y `progress`.
- **Sin webcam.** En parejas, cada uno escribe la página en su proyecto. Para probar solo la lógica, muévela a `handleGesture(name, now)` y llámala desde un botón temporal cada 100 ms.

Si acabas pronto: dibuja los 21 landmarks con `draw.drawConnectors(hand, GestureRecognizer.HAND_CONNECTIONS, ...)`, con `hand = result.landmarks[0]`.

EJERCICIO · EL PEGAMENTO (AR)

El detector da una caja en **píxeles del vídeo** (`originX`, `originY`, `width`, `height`). La pegatina es HTML con `position: absolute`. Calcula con un vídeo de 640 de ancho y una cara con `originX = 400` y `width = 120`:

CSS	FÓRMULA	CON NUESTROS NÚMEROS
`width`	`width / videoWidth × 100`	
`left`	`100 − (originX + width) / videoWidth × 100`	
`top`	`originY / videoHeight × 100`	

1 ¿Por qué `100 −`? ¿Y por qué porcentajes?

TAREA 3 · UNA PEGATINA EN TU CARA (16 MIN)

OK	HAZ	ESTÁ HECHO CUANDO...
☐	Añade `createFaceDetector` a `vision.ts` (mismo patrón, modelo `blaze_face_short_range`, sin `numPoses`).	El detector carga sin errores.
☐	Nueva página con el `frame` y el marcado proyectados; `stickers` es `$state`.	Aparece una pegatina por cada cara.
☐	La pegatina muestra algo de **tu** app (un título, un contador, un estado).	Te sigue al moverte, en la dirección correcta.
☐	CSS: `.sticker { position: absolute; transform: translateY(-110%); }` y `.blurred { filter: blur(8px); }`	Con dos caras, el contenido se difumina.

Si algo falla:

- **La pegatina se mueve al revés.** Falta el `100 −`. Muévete a la derecha y mira.
- **La pegatina está en la frente.** `top` es el borde superior de la caja: `translateY(-110%)` la sube su propia altura.
- **Píxeles en vez de porcentajes.** Funciona a un tamaño de ventana y se rompe a otro.
- **El `{#each}` no se repinta.** `stickers` no es `$state`.

TAREA 4 · DOS PREGUNTAS (7 MIN)

En parejas, auditad la app **del otro** con el tablero **E** y marcad:

OK	¿QUIÉN MIRA? (PRIVACIDAD)	OK	¿Y SI NO PUEDO? (ACCESIBILIDAD)
☐	Cámara y micrófono apagados hasta pulsar un botón.	☐	Cada acción de voz, pose y gesto existe también como botón.
☐	Mientras están encendidos, la página lo dice y ofrece «off».	☐	Cada botón funciona con el teclado.
☐	Off detiene de verdad las pistas: la luz de la cámara se apaga.	☐	Los cambios de estado se anuncian (`role="status"`).
☐	La página dice adónde van los datos.	☐	Ningún estado se muestra solo con color.
☐	No se guarda ni se envía nada. La app detecta, nunca identifica.	☐	Un permiso bloqueado o un dispositivo que falta dice qué ha pasado y qué hacer.

ENTREGA

En Classroom: (1) un **enlace de solo lectura** a tu copia de Figma con la tabla C y la checklist E hechas; (2) una **grabación de pantalla de 30 segundos**: una acción por gesto con su barra de progreso, y la pegatina siguiendo tu cara.

PARA CASA · PREPARA EL LUNES (EXAMEN)

- Termina gestos, pegatina y checklist en tu propia app.
- Ensaya el examen en 20 minutos: abre tu proyecto, bloquea la cámara y el micrófono desde la barra de direcciones y comprueba que cada estado de la semana aparece con su mensaje.
- Repasa: la tabla de navegadores del lunes, el eje y del martes y la fórmula de `left` de hoy.