ChatGPT

Dwell-Based Head and Gaze Tracking Interfaces Using MediaPipe

Head-Tracking Pointer Solutions (Hands-Free Mouse Control)

Google Project GameFace (Open-Source): An open-source, camera-based mouse that lets users control the cursor with head movement and facial gestures [1]. It leverages MediaPipe's Face Landmarks (478 3D facial points + 52 blendshape scores) for real-time face tracking [2]. Head motion (tracked via a stable forehead landmark) moves the cursor, and custom facial expressions trigger clicks (e.g. raising eyebrows to click/drag) [1]. To combat jitter, GameFace includes an adjustable smoothing filter on the cursor movement [3], allowing users to fine-tune stability for their setup. This results in very smooth tracking and a highly customizable hands-free interface. *Reusable components:* The **MediaPipe Face Landmarker API** (with blendshape output) is central to GameFace, and its approach to adjustable smoothing and per-direction cursor speed calibration can be adapted for other head-controlled UIs [4] [3].

Head-Tracking Pointer Chrome Extension (UCSC): A GitHub project from UC Santa Cruz's CV Lab provides a personalized head-tracking pointer as a Chrome extension for accessibility [5]. It uses MediaPipe (likely Face Mesh or Holistic) to track head pose in the browser, enabling the user's head movements to move the cursor. Dwell-based clicking is supported – the system triggers a left-click when the head-controlled cursor remains fixed over a target for a set "dwell time." (Right-click or other actions can be bound to alternative cues like eye blinks.) This extension demonstrates seamless UI integration: running in the browser, it overlays a head-controlled pointer on webpages. The repository notes it was developed for accessibility research and includes features like *dwell-clicking* for selections and possibly smoothing to filter small head jitters (e.g. via a short history average or threshold). *Reusable components:* It likely builds on **MediaPipe's JavaScript Face Mesh** and the browser's pointer APIs, and could be repurposed for other browser-based head gaze applications.

Face-2-Cursor (Python Desktop App): *FaceCursor* is a Python open-source project that uses MediaPipe Face Mesh to control the mouse via facial movements [6]. It tracks the nose tip position and moves the cursor based on the averaged nose coordinates (smoothing the movement) [7]. Blinks are detected (via eye aspect ratio) to simulate click events instead of using dwell [8]. The GUI (built with Tkinter) provides sliders to adjust cursor *sensitivity* and acceleration for X/Y axes [8] – effectively tuning how much movement translates to cursor motion, which also helps stabilize control. This project illustrates how MediaPipe's 2D landmarks can map to screen coordinates (nose movement mapped to Δx, Δy of the cursor) and how simple filtering (averaging frames or scaling down movements) can reduce jitter. *Reusable components:* It uses **MediaPipe's Python API** for face landmarks and **PyAutoGUI** for cursor control, which can be adapted in other Python-based assistive apps.

"MLI.Mouse" Dwell-Click Implementation: A research prototype (CEUR-WS 2022) called *MLI.Mouse* demonstrates an adaptive dwell-click technique using MediaPipe. The user controls the cursor with an arbitrary body part (e.g. head or hand) tracked by MediaPipe, and left-clicks are activated by dwelling: if the cursor stays within a small radius *R* for more than *S* seconds, a click is registered [9]. Both the radius and dwell duration are adjustable, allowing tuning of the tolerance for natural tremor vs. intentional hold. This method effectively filters out noise – small movements within the radius do not cancel the dwell – and prevents accidental clicks. *Insight:* Defining a “stay zone” around the cursor and requiring a sustained presence is a common strategy to stabilize dwell-based selection [9]. Developers can reuse this approach by applying an **exponential moving average** or **radius threshold** on the pointer coordinates before timing the dwell.

Eye Gaze Tracking and Dwell Interaction Examples

HueVision Gaze Tracker (Web Demo): *HueVision* is a browser-based real-time eye tracking demo built with MediaPipe FaceMesh and TensorFlow.js [10]. It detects 468 facial landmarks + iris positions in each frame, then extracts eye-region features to infer gaze direction [10]. The user calibrates by looking at on-screen targets, allowing the system to learn a mapping from eye features to screen coordinates via a lightweight TF.js model [11] [12]. The output is a continuously updated gaze point, visualized as a reticle or heatmap overlay on the page. While HueVision doesn’t natively implement dwell clicking, it provides the essential components for it – stable gaze point estimation and a UI overlay. The pipeline is fully client-side (no server) and emphasizes *stability*: MediaPipe’s dense landmarks and iris tracking greatly improved jitter and accuracy over earlier methods, and with careful throttling/drawing, the gaze point is smooth and usable in real time [13]. *Reusable components:* The **MediaPipe Face Mesh (with Iris)** in JavaScript gives robust face and eye landmarks, and the calibration approach (mapping normalized gaze features to screen coords) can be reused for other gaze-controlled interfaces. One could layer a dwell timer onto HueVision’s gaze reticle to trigger clicks when the user’s gaze stays on a button for, say, 800 ms.

MediaPipe Iris + Kalman Filter (Robust Gaze Estimation): Recent research has shown the benefit of filtering on top of MediaPipe’s eye tracking. For example, Ramesh et al. (2025) combine **MediaPipe Iris** with a **Kalman filter** to stabilize and predict gaze points [14]. The Kalman filter smooths the raw iris landmark movements and accounts for latency, yielding more consistent gaze estimation under real-world head motions, lighting changes, and occlusions [14]. Notably, their approach reduced the need for extensive calibration by using MediaPipe’s native 3D eye pose and depth data together with filtering [15] [14]. This technique can be applied in both web and native implementations – e.g. applying a Kalman filter to the sequence of gaze coordinates to dampen jitter and momentary noise. *Insight:* Developers building dwell-based gaze UIs can incorporate such smoothing filters (Kalman or exponential moving average) on top of MediaPipe outputs to improve stability before evaluating dwell time. A filtered gaze signal helps ensure the dwell timer isn’t reset by tiny involuntary eye movements.

Eye Gaze Dwell Selection in Practice: In an eye-controlled virtual keyboard scenario (Bhumika, 2023), gaze-based dwell selection was used to click keys: the system checks how long the user’s gaze stays on a particular button and triggers a “click” once a threshold is exceeded [16]. This is a common pattern for gaze UIs – for instance, a dwell timer circle can be shown on the focused element, completing when selection is made. Many projects integrate blink or gesture clicks as an alternative (as blinking can be an intentional “click” signal), but dwell-time clicking remains popular for its simplicity [17]. When implementing dwell selection with MediaPipe gaze tracking, one should consider a short delay and perhaps a confirmation highlight to avoid false activation. Ensuring the gaze-to-screen mapping is accurate (through calibration like HueVision’s or using known screen geometry) is critical so that dwell targets correspond to actual UI elements.

Additional Notable Implementations and Libraries

- **AirCanvas – Hand Gesture Dwell UI (Web):** This open-source web app uses MediaPipe Hands for cursor control and demonstrates dwell-based activation in a drawing interface. The user points a finger to move the on-screen cursor, and holding it still for ~0.5s engages "draw mode" (essentially a dwell-to-click for pen down) [18]. Tools are selected by hovering on them briefly [19]. Visual feedback (e.g. a color change or timer) indicates when a dwell selection registers. *Relevance:* Although it uses a hand pointer, AirCanvas shows how **dwell-based UI** can prevent accidental clicks and improve stability in a browser context – the same pattern can be applied to head or gaze pointers.

- **MediaPipe Holistic/BlazePose for Head Pose:** Some projects use MediaPipe **Pose (BlazePose)** or Holistic to get a rough head direction from the 33 body landmarks (e.g. nose, eyes, shoulders). This is less precise than Face Mesh but very fast. For instance, one can compute the angle of the line connecting the shoulders or use the nose's 3D position to infer yaw. However, for fine gaze control or dwell selection, the dedicated face/iris models are preferred for their higher fidelity. MediaPipe's **Face Geometry** utilities (in the solutions API) can convert face landmarks to a 3D head pose (using solvePnP under the hood), which some Python tutorials use to move a cursor based on head pitch/yaw. These lower-level tools or model outputs can be building blocks if a project specifically needs to integrate with body pose tracking (BlazePose) rather than a full face mesh.

- **Control Utilities and UI Integration:** MediaPipe's JavaScript library comes with helpful utilities (e.g., the `@mediapipe/control_utils` module) for drawing overlays and handling webcam input. For example, the official demos and CodePen templates provide a starting point for capturing video frames and rendering landmarks or cursor pointers on a canvas. Reusing these can accelerate development of a dwell-based interface. Likewise, if developing in Python, libraries like **PyAutoGUI** (for cursor clicks/movements) and GUI toolkits (Tkinter, PyQT) have been successfully combined with MediaPipe as shown in Face-2-Cursor [8]. Many of the above projects are open-source, allowing developers to study or repurpose code: for instance, GameFace's repository (on GitHub under `google/project-gameface`) contains modules for smoothing and custom gesture recognition, and HueVision's code illustrates a clean browser-based gaze pipeline.

Bottom Line: A variety of demos and projects have implemented dwell-based selection using MediaPipe's pose and face-tracking capabilities. Common best practices include using **facial landmarks or iris tracking for precise gaze/head pose**, applying **smoothing filters** (Kalman, moving average, or configurable gain) to reduce jitter, and employing a **dwell timer with positional tolerance** to trigger clicks reliably. By examining these implementations – from Google's GameFace and academic prototypes to open-source browser demos – one can find reusable components and techniques to build a stable, hands-free interface that selects UI elements by looking or pointing with one's head and dwelling. Each of these examples showcases a piece of the puzzle, whether it's high-fidelity tracking, calibration and mapping, jitter reduction, or intuitive dwell-click UX feedback. With MediaPipe's cross-platform tooling (JS WebAssembly and Python support), developers can mix and match these techniques to create accessible gaze and head-controlled applications [3] [9].

---

[1] [2] [3] [4] Project GameFace makes gaming accessible to everyone - Google Developers Blog
https://developers.googleblog.com/en/project-gameface-makes-gaming-accessible-to-everyone/

[5] head-tracking · GitHub Topics · GitHub
https://github.com/topics/head-tracking?o=desc&s=updated

[6] [7] [8] GitHub - kubilaiswf/face-2-cursor: FaceCursor is a Python project that enables mouse control by tracking facial movements over a webcam. Face tracking is done using OpenCV and facial movements are converted into mouse movements with the pyautogui library.
https://github.com/kubilaiswf/face-2-cursor

[9] ceur-ws.org
https://ceur-ws.org/Vol-3091/paper09.pdf

[10] [11] [12] [13] How to Build Real-Time Eye Tracking in the Browser
https://blog.roboflow.com/build-eye-tracking-in-browser/

[14] [15] MediaPipe Iris and Kalman Filter for Robust Eye Gaze Tracking | Atlantis Press
https://www.atlantis-press.com/proceedings/icsice-24/126011300

[16] [PDF] EYE - MOUSE AS A COMPUTER MOUSE USING MEDIAPIPE ...
https://epublications.vu.lt/object/elaba:192827889/192827889.pdf

[17] Redesigning Multimodal Interaction: Adaptive Signal Processing and Cross-Modal Interaction for Hands-Free Computer Interaction
https://www.mdpi.com/1424-8220/25/17/5411

[18] [19] GitHub - robert-mcdermott/aircanvas: A web-based drawing application controlled entirely by hand gestures and finger tracking. For video presentations, online meetings, and interactive demonstrations where you want to draw and annotate while being visible on camera.
https://github.com/robert-mcdermott/aircanvas