Demonstrating Headphone-Sensed, Accessibility-Informed Head Pointing with Snapping for Inclusive Interaction

MYKHAILO TARASENKO, National University of Kyiv-Mohyla Academy, Ukraine and MacPaw, Ukraine

IRYNA PASTUKHOVA, MacPaw, Ukraine

OLEKSANDR FRANKIV, National University of Kyiv-Mohyla Academy, Ukraine and MacPaw, Ukraine

ANASTASIIA SATARENKO, MacPaw, Ukraine

NATALIIA STULOVA, MacPaw, Ukraine

SERGII KRYVOBLOTSKYI, MacPaw, Ukraine

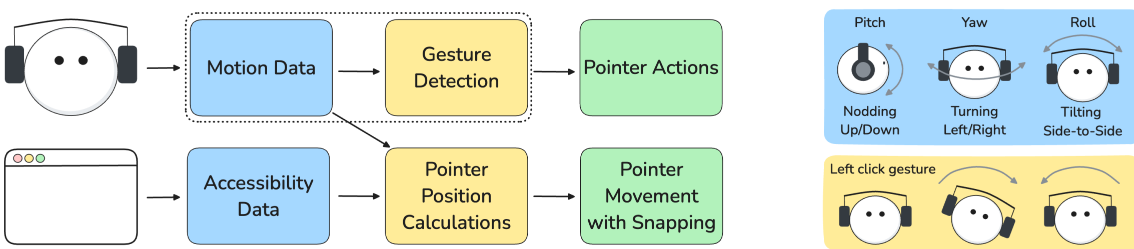


Fig. 1. General head pointing approach architecture with detalisation of captured data and detected gesture visualization

The mouse pointer is central to direct-manipulation graphical user interfaces, and modern desktop operating systems provide accessibility features that enable pointer control via eye or head movement. Existing solutions, such as Eye Control on Windows and Head Pointer on macOS, rely on continuous video capture, which makes them sensitive to lighting conditions and user position and raises privacy concerns. In this work, we demonstrate a head-pointing approach based on head movements captured by the gyroscope and accelerometer sensors of commercial headphones. To improve pointing precision, we implement pointer snapping that leverages accessibility information from application user interfaces. This approach is independent of camera placement and lighting conditions, offers privacy advantages, and requires no specialized hardware beyond commonly used headphones, supporting more inclusive and accessible interaction. Our demo highlights how reusing existing accessibility infrastructure can support more inclusive pointing interactions and contribute to creating more accessible interactive systems.

CCS Concepts: • **Human-centered computing** → **Pointing**; **Gestural input**; *Pointing devices*; **Accessibility systems and tools**; **Systems and tools for interaction design**.

Additional Key Words and Phrases: Head-based pointing, Head-based gestures, Accessibility, Gyroscope, Accelerometer, macOS

Authors' addresses: Mykhailo Tarasenko, National University of Kyiv-Mohyla Academy, Kyiv, Ukraine and MacPaw, Kyiv, Ukraine, ms.tarasenko@ukma.edu.ua; Iryna Pastukhova, MacPaw, Kyiv, Ukraine, iryna.p@macpaw.com; Oleksandr Frankiv, National University of Kyiv-Mohyla Academy, Kyiv, Ukraine and MacPaw, Kyiv, Ukraine, o.frankiv@ukma.edu.ua; Anastasiia Satarenko, MacPaw, Kyiv, Ukraine, an.sat@macpaw.com; Nataliia Stulova, MacPaw, Kyiv, Ukraine, nata.stulova@macpaw.com; Sergii Kryvoblotskyi, MacPaw, Kyiv, Ukraine, krivoblotsky@macpaw.com.

ACM Reference Format:
Mykhailo Tarasenko, Iryna Pastukhova, Oleksandr Frankiv, Anastasiia Satarenko, Nataliia Stulova, and Sergii Kryvoblotskyi. 2026. Demonstrating Headphone-Sensed, Accessibility-Informed Head Pointing with Snapping for Inclusive Interaction. 1, 1 (January 2026), 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND MOTIVATION

Mouse pointer and point-and-click workflows still remain the default expected way to interact with direct manipulation graphical user interfaces on personal computers and laptops. For computer users with impaired mobility, or for hands-free interaction scenarios in general, desktop operating systems vary in accessibility features and peripheral devices supported to accept alternative pointer controls. In Windows, the desktop OS with the largest user base worldwide, an eye mouse is available via Eye control technology. It relies solely on tracking the eye movement with a camera, and for pointer actions, the user is presented with a movable launchpad bar with predefined actions selected by dwelling the eye gaze on them. Moreover, its use is limited [9] "in locations with a lot of sunlight. Additionally, eye tracking works differently depending on eye color, eye size, or eye shape." On macOS, the second most used desktop OS, the Head Pointer technology embodies the face mouse approach. It also uses a camera, processing user face expressions and head movement for cursor actions and position change, respectively. Limitations [7] of this technology include "the ambient lighting being too bright or dark, the user being too close to or far from the camera, or them not being centered in front of the screen". Moreover, reliance of both solutions on camera feed requires either a built-in camera, an additional camera, or a camera-equipped device, and introduces user privacy concerns that need to be mitigated additionally.

Among the head-based pointing methods, head mouse approach is another possible implementation of a relative pointing device. Research to date shows that when head mobility is a possible scenario, relative head-based pointer controls are preferred over absolute eye gaze-based ones [1, 4, 5]. Current head pointing solutions differ in the hardware used to capture head movement, including head-based cameras [13], optical sensors on the neck [6], head and mouth pieces [3], or glass-mounted sensors [12]. Recently, modern headphones were considered as a general-purpose input device [10]. While the proposed solutions differ in their implementations of pointer actions, the use of gyroscope and accelerometer sensors has become a shared pattern, nowadays also present in commercial head- and earphones.

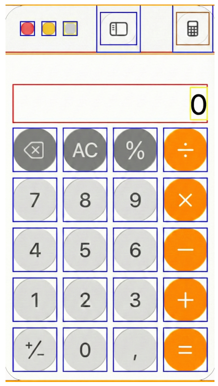
In this work, we demonstrate an approach to extend the capabilities of an existing wearable device, like headphones, to the task of head pointing. While only using two sensors for head position tracking, we take advantage of application user interface (UI) annotations, which are available in macOS as a part of its accessibility features, as an information source to improve pointing precision, naming it pointer snapping and making it informed, aligning with the user interaction intent. Using accessibility information instead of gaze allows for avoiding limitations of the environment and focusing on actual interaction. We implement left and right mouse click actions with head gestures and showcase user learning and app interaction.

2 BACKGROUND

We briefly outline the relevant for our approach software and hardware components of the Apple ecosystem.

Accessibility-related functionality in computer system interfaces helps people with different types of needs, such as vision, speech, mobility, cognitive, and hearing, to interact with their devices. Accessibility[1] framework in macOS supports developers in building systems that address a wide range of access needs through a comprehensive set of APIs. One of the framework features is the accessibility tree, like one shown in Figure 2 (right), that represents an app's user

[1] https://developer.apple.com/documentation/accessibility




```
"children":[
        {"id": "ad6e8c1d",
         "name": null,
         "role": "AXButton",
         "description": "Equals",
         "role_description": "button",
         "value": null,
         "absolute_position": "827.00;699.00",
         "position": "172.00;349.00",
         "size": "48;48",
         "enabled": true,
         "bbox": [172,349,220,397],
         "visible_bbox": [172,349,220,397],
         "children": []
        }
        ...
]
```


Fig. 2. A match between UI elements in a Calculator app and the respective JSON of the accessibility tree (the "=" button fragment)

interface and is used by assistive technologies like VoiceOver[2] to parse the UI hierarchy of the currently focused window opened on the user's desktop.

The other core framework of interest to us is CoreMotion[3] that processes motion- and environment-related data from the onboard hardware of wearable devices. Our approach supports all headphones that are compatible with the CoreMotion's CMHeadphoneMotionManager class according to the isDeviceMotionAvailable property. This class provides accelerometer, gyroscope, and other position data via a specific API. All headphones that support Apple's Spatial Audio technology with dynamic head tracking on the device are compatible with this class[4].

3 IMPLEMENTATION

Receiving data from headphones. To implement pointer movement using headphones that support Spatial Audio, we develop an application that uses the CoreMotion framework. Using its class CMHeadphoneMotionManager, we start tracking motion updates of the headphones by calling a method .startDeviceMotionUpdates(). As a result, our application starts receiving information about changes in the headphones' position in space. This information is encapsulated in a CMDeviceMotion class, which gives access to different data, such as attitude, rotationRate, userAcceleration etc. Here, attitude is a struct which represents the pitch, yaw, and roll aspects of motion, as shown in Figure 1.

Calibration and coordinate mapping. For our approach, headphone motion data must be mapped to the screen coordinates. In the considered setup, the yaw value corresponds to the horizontal coordinate (x), and the pitch to the vertical coordinate (y). Thus, for any pointer position (x, y), we have $x \in [0, width]$ and $y \in [0, height]$, where $width$ and $height$ are the screen dimensions. We assume that, on average, people can rotate their heads within a range of $\pm 30°$ (approximately ± 0.5 rad). Applying the linear interpolation, for the given yaw value, the x coordinate can be computed as $x = x_{min} + t \cdot (x_{max} - x_{min})$ with $t = (yaw - yaw_{min})/(yaw_{max} - yaw_{min}) = (yaw - (-0.5))/(0.5 - (-0.5)) = yaw + 0.5$, and consequently, $x = 0 + (yaw + 0.5) \cdot width = (yaw + 0.5) \cdot width$. By analogy, for any pitch value, the y coordinate $y = y_{min} + (pitch + 0.5) \cdot height = (pitch + 0.5) \cdot height$.

To generalize our approach and reduce dependence on the user distance from the screen, the yaw and pitch values are calibrated before applying interpolation. For this, the user is asked to approach four near-corner points on the screen

[2] https://developer.apple.com/documentation/accessibility/voiceover
[3] https://developer.apple.com/documentation/coremotion
[4] https://developer.apple.com/videos/play/wwdc2023/10179/

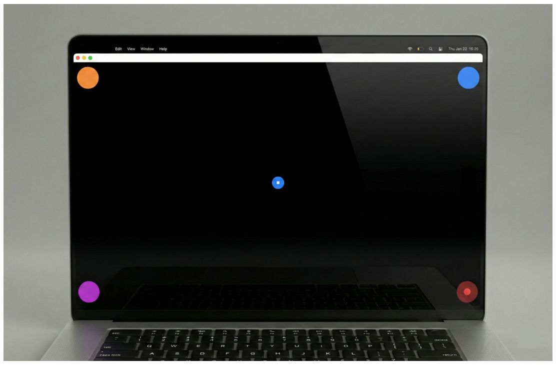


Fig. 3. Calibration process starts in a window

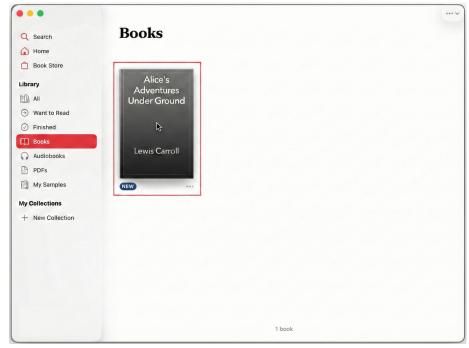


Fig. 4. Books app with pointer snapping

(see Figure 3) with the head pointer, providing user-specific extreme yaw and pitch values. Using the corresponding spans allows us to adjust the observed yaw and pitch values to the agreed-upon "universal" range above, ±0.5 rad.

Pointer snapping approach. Precise hovering, maintaining focus, and interacting with UI elements, especially small or closely spaced ones, can be challenging with head pointing. To address this issue, we introduce the snapping pointer approach. We equip each UI element with a targeting neighborhood that slows the pointer, snaps it, and targets it towards the center. Directing a pointer to the intended UI element requires determining the precise position and size of the closest available element. We continuously monitor the change of the active application to capture the user's context at each point in time. We traverse the accessibility tree of the active application to obtain UI elements, along with their positions on the screen, sizes, roles, actions, and other attributes. The Accessibility API allows us to respond to UI changes immediately. We further filter interface elements based on their accessibility roles such as AXButton, AXPopUpButton, and supported actions such as kAXPressAction, kAXShowMenuAction, relevant to pointer interaction.

Another consideration is the pointer snap itself. To make focusing on the UI element easier, we increase the motion threshold in the targeting neighborhood and highlight the element as it is approached. If the pointer is directly snapped to and released from the element's neighborhood, from the user's perspective, it feels like jumping and resistance. Also, even though there is almost always only one nearest element (target) at any given moment, in the case of close elements, because the pointer is very sensitive, even a small, often unintentional move can change the target. These rapid changes of the target generate oscillation between the close or intersecting neighborhoods, something like a "ping-pong" effect. All the above effects are mitigated by introducing the hysteresis, such that the threshold for entering the element is lower than for exiting it. At the same time, if the pointer enters the element itself rather than only its corresponding targeting neighborhood, it is selected immediately, enabling smooth navigation through list-like interfaces.

Gestures implementation. The ability to move the pointer alone does not provide complete interaction with the UI. For the purpose of the approach demonstration, we implement left- and right-click actions as a minimal set of interactions. We select head tilts for this purpose, as they are intuitive and rely on the motion roll values that are not yet utilized by our approach, unlike pitch and yaw. Our calibration provides the maximum and minimum roll values, which represent the range of neutral head positions. We will refer to them as the stability interval. We define a click gesture as a time-bounded (0.8s) sharp tilt toward the corresponding side initiated inside the stability interval, followed

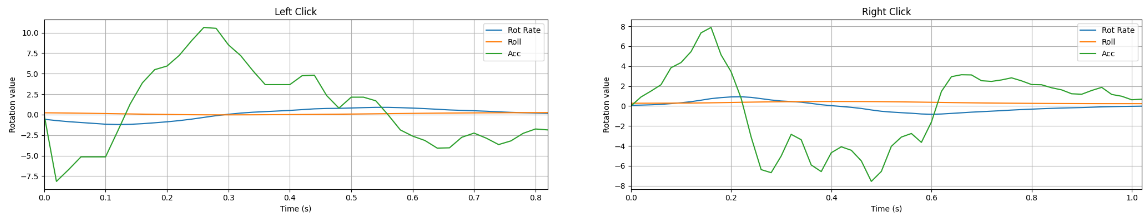


Fig. 5. Roll, rotation rate, and acceleration signals for left and right click gestures

by a rapid return to the starting position. Since the end point of the gesture does not always coincide precisely with the start point, half the stability interval tolerance is allowed.

Figure 5 presents the roll, rotation rate, and acceleration signals for the left and right clicks, respectively, highlighting that acceleration provides the most reliable indicator of the user's gesture intent. We observe that during a left-click gesture, acceleration changes sign from negative to positive. In contrast, during a right-click gesture, the opposite pattern occurs, enabling a reliable distinction between them. Thus, a left click is modelled as a sequence of three states in terms of acceleration: (1) neutral (head roll is within the stability interval), (2) tilt toward the left side, initiated by passing the acceleration negative threshold, (3) return to neutral, detected by acceleration change to positive and neutral head position adjusted according to the tolerance, as shown in Figure 1 (right bottom). This sequence must occur within a predefined time window. By analogy, the right click is identified using the inverse acceleration pattern. The acceleration threshold was initially set to the mean of the minimum absolute values observed across 200 recorded left- and right-click events from one of the authors, and then tuned through empirical evaluation to improve generalization with the final value of 7.0 rad/s^2.

Performing the gestures obviously induces significant changes in pitch and yaw, moving the pointer away from the intended element and resulting in clicks at unpredictable points. To prevent this, we further refine the snapping pointer with a freezing pointer effect. After holding an element in focus for some time, the pointer temporarily freezes, allowing the user to perform a gesture with greater precision. The focusing and freezing phases involve highlighting the corresponding element with different colors to provide visual feedback to the user (e.g., see Figure 4). In our experiments, a duration of 700 ms was sufficient to consider the element approach intentional without hindering natural movement, while 1.5 s of freezing allowed for deciding on and performing a gesture.

4 DEMO

To let users experience the proposed head pointing technique, we developed two training mini-games (Figure 6).

We've developed a mini-game called **Tiles** to introduce the user to basic pointer control and clicking gestures. This application allows users to capture an image with a built-in laptop camera and use it as a puzzle. The image is divided into a grid of 3 × 3, and the tiles are randomly shuffled. Users select and place tiles using the pointer input. We adapted the interaction to a click-based mechanism: a tile attaches to the pointer on a left click and can be released with the same click while hovering over the target location.

We developed the **Banners** mini-game to familiarize users with the effective use of the snapping pointer. It simulates the appearance of multiple intrusive advertisement-like windows on the desktop. Users are instructed to close all banners by clicking on a system-like close button on the header of each window. The close buttons are intentionally

Fig. 6. **Tiles** (left) and **Banners** (right) mini-games' interfaces

small, comparable to the default size in macOS. When the pointer approaches a button, a snapping mechanism attracts the pointer toward the target, facilitating reliable selection.

5 DISCUSSION AND CONCLUSION

We demonstrated an approach to desktop head pointing for mouse pointer control that leverages accessibility annotations in direct-manipulation graphical user interfaces. While our implementation is presented in the context of the Apple ecosystem, the proprietary technologies are not foundational to it. Instead, we showcase how leveraging the developer-facing parts of OS frameworks and building on existing system support for sensor data and accessibility information allows us to complement and extend accessibility functionalities. Our work highlights how accessibility technologies can benefit a broad range of users beyond those requiring accommodations, and we hope it will encourage operating system and application developers to further adopt and maintain accessibility annotations and functionalities.

In the hardware part, our approach relies on headphones and earphones equipped with a combination of accelerometer and gyroscope sensors. These two sensors are an integral part of various headset-based solutions to head pointing [2, 8, 11], and while full headsets allow for richer interactions with the desktop computers and other electronic devices, we believe we have shown that even with minimal hardware use full point-and-click workflows become achievable. Another consideration we had in mind was the reuse of existing technologies and electronics. By extending the use of widely available consumer hardware, our approach also points toward more sustainable interaction design practices, which could minimize the e-waste problem.

ACKNOWLEDGMENTS

We thank the Armed Forces of Ukraine for providing security to complete this work. We thank Mariia Kak and Vladyslav Makariuk for their help with accompanying video production, and Kseniia Dobrieva and Anastasiia Kyiashko for consultations on legal matters. This is an independent research publication, and it has not been authorized, sponsored, or otherwise approved by Apple Inc. Apple, Apple Books, AirPods Pro, and AirPods Max are trademarks of Apple Inc.

REFERENCES

[1] R. Bates and H. O. Istance. 2003. Why are eye mice unpopular? A detailed comparison of head and eye controlled assistive technology pointing devices. Universal Access in the Information Society 2, 3 (2003), 280–290. https://doi.org/10.1007/s10209-003-0053-y

[2] Aaron Castillo, Graciela Cortez, David Diaz, Rayton Espíritu, Krystle Ilisastigui, Bryce O'Bard, and Kiran George. 2016. Hands free mouse. In 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN). 109–114. https://doi.org/10.1109/BSN.2016.7516242

[3] Rodrigo Duarte, Nuno Vieira Lopes, and Paulo Jorge Coelho. 2025. A Low-Cost Head-Controlled and Sip-and-Puff Mouse: System Design and Preliminary Findings. Electronics 14, 24 (2025). https://doi.org/10.3390/electronics14244953

[4] David G. Evans, R. Drew, and Paul Blenkhorn. 2000. Controlling mouse pointer position using an infrared head-operated joystick. IEEE Transactions on Rehabilitation Engineering 8, 1 (2000), 107–117. https://doi.org/10.1109/86.830955

[5] Gareth Evans and Paul Blenkhorn. 1999. A Head Operated Joystick–Experience with Use. (1999). Technical report. https://eric.ed.gov/?id=ED430330.

[6] Ali Heydarigorji, S. M. Safavi, Ch. T. Lee, and P. H. Chou. 2017. Head-mouse: A simple cursor controller based on optical measurement of head tilt. In 2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB). 1–5. https://doi.org/10.1109/SPMB.2017.8257058

[7] Apple Inc. 2025. Move the pointer using head pointer on Mac. Online. https://support.apple.com/guide/mac-help/use-head-pointer-mchlb2d4782b/26/mac/26.

[8] Mohammad Ridwan Kabir, Mohammad Ishrak Abedin, Rizvi Ahmed, Saad Bin Ashraf, Hasan Mahmud, and Md. Kamrul Hasan. 2024. Auxilio and Beyond: Comparative Evaluation, Usability, and Design Guidelines for Head Movement-based Assistive Mouse Controllers. arXiv:2210.04483 [cs.HC] https://arxiv.org/abs/2210.04483

[9] Microsoft. 2019. Eye control troubleshooting guide. Online. https://support.microsoft.com/en-us/windows/eye-control-troubleshooting-guide-83352d28-688a-e29c-4b5f-e91aee324259.

[10] Payod Panda, Molly Jane Nicholas, David Nguyen, Eyal Ofek, Michel Pahud, Sean Rintel, Mar Gonzalez-Franco, Ken Hinckley, and Jaron Lanier. 2023. Beyond Audio: Towards a Design Space of Headphones as a Site for Interaction and Sensing. In Proceedings of the 2023 ACM Designing Interactive Systems Conference (Pittsburgh, PA, USA) (DIS '23). Association for Computing Machinery, New York, NY, USA, 904–916. https://doi.org/10.1145/3563657.3596022

[11] Rafael Raya, Javier O. Roa, Eduardo Rocon, Ramón Ceres, and José L. Pons. 2010. Wearable inertial mouse for children with physical and cognitive impairments. Sensors and Actuators A: Physical 162, 2 (2010), 248–259. https://doi.org/10.1016/j.sna.2010.04.019 Eurosensors XXIII, 2009.

[12] Andreia Sias Rodrigues, Vinicius Kruger da Costa, Rafael Cunha Cardoso, Marcio Bender Machado, Marcelo Bender Machado, and Tatiana Aires Tavares. 2017. Evaluation of a Head-Tracking Pointing Device for Users with Motor Disabilities (PETRA '17). Association for Computing Machinery, New York, NY, USA, 156–162. https://doi.org/10.1145/3056540.3056552

[13] Dariusz Sawicki and Piotr Kowalczyk. 2018. Head Movement Based Interaction in Mobility. International Journal of Human–Computer Interaction 34, 7 (2018), 653–665. https://doi.org/10.1080/10447318.2017.1392078 arXiv:https://doi.org/10.1080/10447318.2017.1392078