AI-DRIVEN PERIPHERAL DEVICE MANAGEMENT AND ASSISTIVE MULTI-MODAL INPUT FOR WIRELESS HUMAN-COMPUTER INTERACTION

Minzhou Wang [1] and Peijin Du [2]

[1] Independent Researcher, Charlotte, USA
[2] Independent Researcher, Los Angeles, USA

ABSTRACT

This paper explores AI-assisted peripheral management techniques that improve traditional input methods through modular and linear layouts combined with voice-based support. Modular and linear approaches let users construct macros with temporal logic, enabling faster setup and more intuitive execution. Voice models extend accessibility by allowing disabled users to configure and later trigger complex key combinations through simplified inputs.

Two experiments evaluated these methods: one compared modular/linear layouts with free-design in terms of setup time, consistency, accuracy, and satisfaction; the other tested AI-optimized layouts with gaze heatmaps. Results show faster setup, fewer errors, and improved accessibility.

KEYWORDS

Human-Computer Interaction, Machine Learning, Voice-Based AI, Accessibility, Game and Software Engineering, Natural Language Processing, Software Engineering, Automation.

1. INTRODUCTION

Gaming has become a central form of modern entertainment and social interaction. 1 in 4 U.S. adults live with a disability [1], highlighting the need to ensure that gaming experiences are inclusive and accessible to a large portion of the population. Organizations that develop assistive technologies have often overlooked the role of gaming, even though it can be as vital to participation and quality of life as mobility aids or communication devices.

Despite this increased interest, a significant number of people encounter barriers when playing games due to a disability. [2] Nowadays, Human-Computer Interaction (HCI) systems, e.g., keyboard, mouse, and touch screen, are frequently used by everyone. However, those ways to interact with computers may not be suitable for disabled persons [3]. Traditional keybinding systems make configuring and executing macros unnecessarily complex. Users often depend on repetitive trial-and-error to define combination keys, which increases setup time, cognitive load, and the likelihood of errors. [4] For users with physical disabilities, performing multiple key presses at the same time is often not feasible, restricting interaction to single movements and limiting overall functionality.

Although a range of adaptive hardware exists to support disabled players—such as eye-tracking tools (Project Iris, Mill Mouse), facial-expression software (Gameface, KinesicMouse Live), mouth-operated controllers, adaptive controller, for example like Microsoft's Xbox Adaptive

Bibhu Dash et al: SECURA, CSSE, EDUT, NLPSIG - 2025
pp. 29-45, 2025. IJCI – 2025 DOI:10.5121/ijci.2025.140503

Controller and Sony's Access Controller[5], screen-magnification tools, and voice-control systems—significant gaps remain. These technologies demonstrate promising ways to make digital interaction more inclusive, but in practice, they are often expensive, difficult to obtain, or tied to specific platforms. Even when available, their setup and calibration can be complex and time-consuming, requiring technical knowledge that not all users possess. As a result, adoption is limited, and many players are left without practical or sustainable accessibility solutions.

At the same time, existing keyboard systems fall short of meeting both mainstream and accessibility needs. Macros and combination keys play a critical role in modern human-computer interaction, particularly in gaming and professional applications that require rapid, multi-step operations. However, traditional keybinding systems make macros difficult to configure, often requiring users to manually define complex sequences through trial-and-error. This process not only increases setup time but also places a heavy cognitive burden on users, leading to inconsistency and frequent errors during use.

The problem becomes even more significant for individuals with physical disabilities. Many disabled users struggle to press multiple keys simultaneously, making it extremely difficult to perform actions such as eight-directional movement in games. In most cases, they are limited to moving in a single direction at a time, which severely restricts their ability to participate in fast-paced, multi-input tasks. The lack of adaptive and accessible input solutions highlights a critical gap in current peripheral management systems.

Recent developments in artificial intelligence (AI) and modular input design offer new opportunities to address these challenges. Instead of operating as an independent tool, AI can serve as a 'copilot,' helping users manage visually intensive or multi-step tasks by easing the demand for continuous manual input.As Laurence Moroney, AI advocacy lead at Google and one of the minds behind Gameface, explains: "AI is a concept. Machine learning is a technique you use to implement that concept." [6] This distinction highlights how practical techniques such as machine learning can be applied to optimize input layouts, predict user intent, and provide adaptive support that makes interaction both more efficient and more inclusive.

Modular and linear layouts allow macros and combination keys to be defined with temporal logic, simplifying the setup process and making complex inputs easier to execute. At the same time, voice-based AI models provide disabled users with the ability to trigger multiple key combinations through speech, enabling more inclusive interaction and removing barriers to multi-directional movement. Much like how intrusion detection systems in the Internet of Things (IoT) often rely on machine learning models trained on datasets like CICIDS-2017, KDD-99, and TON-IoT, our modular and AI-driven input designs similarly benefit from structured, measurable configurations. [17]

Through controlled experiments in gaming contexts, including performance evaluation on tasks like movement, item usage, and skill activation, the research investigates whether AI-driven layouts and multi-modal input can reduce setup difficulty, enhance efficiency, and expand accessibility for both general users and disabled participants.

2. RELATED WORK

Several existing studies have explored assistive technologies and input optimization for human-computer interaction. Table 1 summarizes representative works and highlights how the proposed research differentiates itself.

Table 1. Comparison of Related Work and Proposed Research

Study / Reference	Domain & Focus	Methodology	Limitations	Distinction from This Work
Microsoft Xbox Adaptive Controller; Sony Access Controller	Gaming accessibility hardware for users with physical impairments	Specialized adaptive hardware devices with customizable buttons and inputs	High cost, limited platform compatibility, complex setup	This work provides AI-driven software-level adaptability and modular layouts, reducing reliance on expensive hardware.
Project Iris, Mill Mouse (eye-tracking), Gameface, KinesicMouse Live (facial-expression input)	Alternative input modalities for disabled users	Eye-tracking and facial-expression recognition for controlling games	Often platform-specific, require complex calibration, limited to certain interaction types	This work integrates multi-modal AI input (including voice and modular layouts) into mainstream peripheral management, enabling broader usability.
Screen magnification tools, voice-control systems	Accessibility tools for general computer use	Visual or speech-based interaction aids	Lack of integration with gaming or advanced multi-key operations	This work combines assistive modalities with AI-optimized layouts, addressing both gaming and professional multi-input tasks.

In summary, prior work has made significant progress in developing specialized hardware and exploring machine learning applications for accessibility and network optimization. However, these approaches often remain domain-specific, costly, or limited in adaptability. The proposed research distinguishes itself by unifying AI-driven layout optimization and multi-modal assistive input into a cross-device, web-based peripheral management platform, aiming to reduce learning burden, enhance efficiency, and broaden accessibility for both mainstream and disabled users.

3. CHALLENGE

The experimental challenges addressed in this study focus on the following aspects.

Challenge 1: Real-Time Performance Gap in Accessibility Tools

Existing assistive technologies (adaptive controllers, single-switch systems, key remapping) enable basic input for users with upper-limb disabilities, but they fail to deliver low-latency, simultaneous multi-key input required by FPS games. [7] This limitation directly impacts reaction speed, competitive viability, and user experience in fast-paced gaming environments.

Challenge 2: High Setup Complexity and Learning Burden

While some gamers develop personalized configurations that suit their needs, creating these setups requires extensive trial-and-error, hours of adjustment, and complex memorization of custom bindings. [8] This complexity not only discourages new users but also limits adaptability across different games and platforms.

Challenge 3: Lack of Cross-Platform Consistency and Adaptive Frameworks

Accessibility solutions are typically platform-specific and static, lacking adaptive algorithms that adjust input dynamically to user needs and game play requirements. The absence of unified, cross-platform frameworks forces users to start from scratch on each new game, reducing usability, consistency, and long-term accessibility.

How do AI tools increase BLV users' independence, aiding in creative, professional, and daily activities?

4. SOLUTION

To address these challenges, the study designed the following experimental solutions:

4.1. Modular and Linear

To address the accessibility challenges faced by upper-limb-disabled gamers in FPS games, a modular and linear input approach (Fig. 1 and 2) offers a practical solution. Unlike free-design methods, where users must manually assign every key or input—often resulting in long setup times, high error rates, and a steep learning curve—modular layouts use predefined building blocks or linear sequences for actions such as movement, aiming, and shooting, and additionally support assigning temporal dimensions(Table.1) to macros (e.g., "hold until the next command terminates" or "single quick trigger"). [9] This structure enables users to quickly configure their controllers, reduces cognitive load, and ensures cross-platform consistency, allowing setups to be easily reused across different games. Importantly, this approach is intuitive for users without technical backgrounds, as they can understand and arrange modules through a simple interface (e.g., Scratch or mBlock), combining physical inputs and voice commands without needing programming expertise. This approach not only benefits disabled users but also improves usability for all users.

In Figure 1, the framework of input definition is illustrated. It consists of three core dimensions: time dimension**,** key position**,** and duration per key definition**.** Through this structure, users can specify whether an input is "press once to release" or "hold until the next command terminates," combine multiple keys into macro commands, and further set the execution duration. This modular definition also supports nesting and extending macros, ensuring flexibility across different scenarios while maintaining cross-platform consistency.

In Figure 2, the workflow of key binding creation is shown. The process begins with naming the macro, followed by selecting a time dimension, assigning a specific key, setting the duration, and then either inputting the next key or nesting an existing macro. The sequence continues until the macro is saved, bound to a virtual key position, and positioned within the overall layout and layer management. This linear workflow allows users to complete complex configurations in a step-by-step manner, reducing the learning curve and error rate while providing a more intuitive setup experience for players with limited mobility.

Table 2. Time Dimension example

Category	Mode Name	Behavior Description	Game Example: CrossFire	Workplace Example: Photoshop
Basic Input Modes (Non-Continuous)	Single Key Press	Quick action triggered by pressing once and releasing	Press G to throw a grenade	Press B to switch to Brush Tool
	Simultaneous Key Press	Multiple keys pressed together to trigger combined action	Press Ctrl + W to crouch forward	Press Ctrl + Z to undo
	Sequential Key Press	Execute actions in sequence by pressing keys one after another	Sequentially press W + A + S + D to move irregularly	Sequentially press keys to apply multi-step commands (e.g., shortcut chains)
Time-Dimension Controlled Modes (Sustained Behaviors)	Long Press (Single Key)	Press and hold for a period of time to maintain action	Hold W to walk forward continuously	Hold Shift to maintain free transform ratio
	Long Press + Release	Action is maintained while pressing; release to stop	Hold W to move, release to stop	Hold Alt: Eyedropper continuously samples colors until release
	Long Press + Hold + Trigger Next Action	First action continues, while simultaneously triggering another	Move forward with W + press Mouse Left to shoot	Hold Spacebar to pan the canvas
	Continuous Long Press with Interval Repetition	Action is repeated periodically while holding	Hold Left Mouse: automatic rifle firing, bullets repeat	Hold Ctrl + Alt + brush size adjustment (repeats until release)
	Sustained Motion with Variation	Action speed or effect changes dynamically when holding	Hold Ctrl: crouch + press W to walk slowly	Hold Brush tool with flow adjustment (e.g., opacity change by continuous press)
Segmented Actions (Composite Sequential Macros)	Multi-step Sequential Action	Combine different key actions in sequence	WA to move left forward → crouch → continue WA	Create straight lines: Hold Shift + Click Brush → release Shift → continue drawing
	Multi-step Repetition (Loop)	Repeated execution of a sequence while holding	Press W + A + S + D to perform loop movement (interval reset, e.g., knife attack loop)	Repeat scaling actions: Hold Shift + resize multiple times
	Switch Key State (Toggle On/Off)	Press once to activate, press again to deactivate	Toggle WA crouch state: one press to crouch, next press to stand	Toggle Caps Lock for precise selection
Persistent States (Continuous Behaviors)	State Switching	First press activates persistent state, press again to turn it off	Open sniper scope, press again to close	Switch between Brush/Pencil tool modes
	Mode Switching	Multiple actions switched through repeated presses	Switch between weapons/throwables	Switch Fill → Stroke → Feather → Selection loop

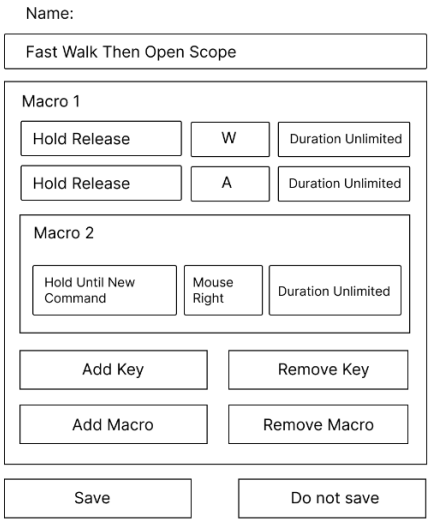

Figure 1.Framework of Input Definition

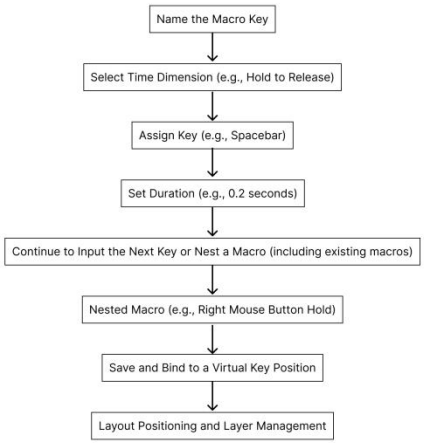

Figure 2.Workflow of key binding creation

4.2. Adaptive AI Layout Generation

The integration of AI agents in gaming has been explored through multiple approaches, particularly with the rise of large language models (LLMs) [10]. In this study, we apply AI to automatically generate optimized key layouts based on **baseline usage data** and three overarching categories of interaction patterns.

The first category is temporal patterns, which capture the timing and order of user actions. These include sequential flows that typically occur during gameplay, such as moving, aiming, firing, and reloading in succession. They also include rhythmic intervals, for example, the tendency to heal after extended combat or reload immediately following a burst of firing. Temporal patterns additionally highlight common error-recovery behaviors, such as repeated mis-presses or confusion between adjacent actions, which can guide the AI to design more robust layouts.

The second category is **frequency and priority patterns**, which describe how often and how critically inputs are used. High-frequency actions, such as movement or firing, must be placed in the most accessible positions, whereas lower-frequency actions, such as chatting or taking screenshots, can be relegated to peripheral zones. Priority is also determined by tactical importance: survival-critical or time-sensitive actions are emphasized in central or dominant positions, while less urgent operations are assigned lower prominence. In addition, frequently combined actions, such as “jump and shoot” or “crouch and reload,” are recognized as co-occurring pairs or chains, encouraging the AI to group them closely together to improve fluidity.

The third category is **spatial and structural patterns**, which concern the ergonomic and contextual relationships between keys. This involves minimizing hand displacement by placing related functions adjacent to one another, adapting layouts to account for left- or right-hand dominance, and dynamically shifting configurations depending on gameplay context, such as differentiating between exploration and combat phases. These spatial considerations ensure that the layout is not only efficient but also comfortable and adaptable across varied play scenarios.

By learning across these three categories of patterns, the system aims to reduce reaction time, minimize the cognitive load of searching for the next input, and improve overall control. For complex action role-playing games (ARPGs), the AI adapts layouts using both collected interaction data and user-defined macros, effectively functioning as an assistive game agent.

Performance will be evaluated using gaze heatmaps to assess improvements in task completion time, keystroke intervals, gaze concentration, error rate, and user perception (Fig. 3).

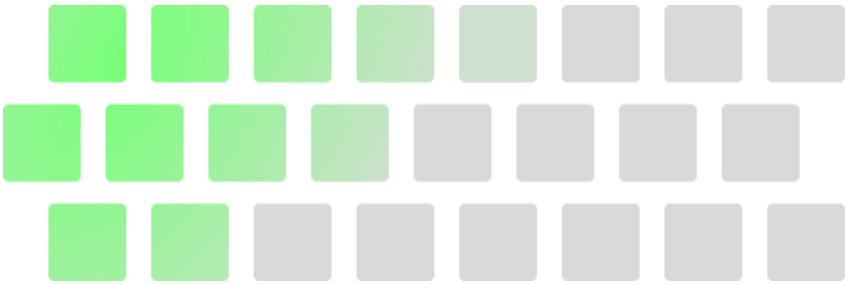

Figure 3. Layout

According to Kara Pernice's article in the Nielsen Norman Group, eye-tracking research shows that users often scan visual information in an F-shaped pattern—reading first across the top, then a shorter line below, before moving vertically down the left side of the layout[11]. This pattern suggests that the upper and left regions of an interface receive the most visual attention. For input design, placing commonly used keys or macros in these high-attention zones can make them quicker to find and easier to reach. Similarly, grouping frequently combined actions in proximity can reduce search time and improve fluidity during fast-paced tasks.

Figure 4. F-shaped heatmap

4.3. Voice AI Assistance for Accessibility

Speech recognition can serve as an effective alternative input method, particularly for users who struggle with pressing multiple keys at once. By allowing multi-step actions to be carried out through simple spoken commands, voice input reduces physical barriers and enhances accessibility for disabled users, while offering flexible interaction benefits to all players.Unlike general-purpose GUI agents, which often fail to meet the speed demands of real-time gaming, modern speech technologies offer faster and more reliable interaction. Robust APIs such as Google Cloud Speech-to-Text, IBM Watson, and Microsoft Azure [12] provide accurate, low-

latency recognition that can be mapped directly to macros and combination keys. This solution evaluates its feasibility in executing combination keys and macros and exploring its role in enabling inclusive multi-modal interaction.

Together, these experimental solutions aim to reduce cognitive load, enhance efficiency, and extend accessibility in peripheral device management.

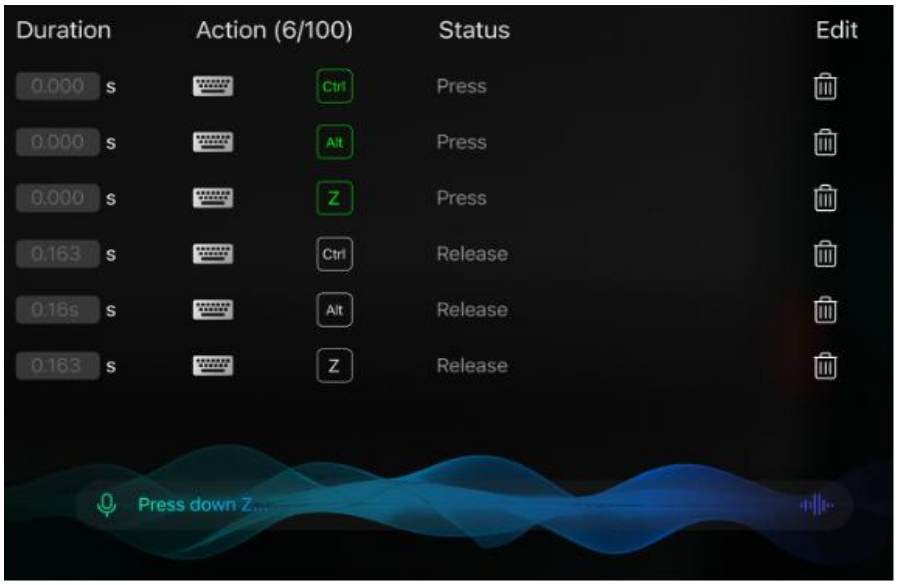

Figure 5. Voice input

5. EXPERIMENTAL DESIGN

To evaluate the effectiveness and user value of the proposed AI-driven platform, we conducted two complementary experiments:

5.1. Experimental Game Context

To ensure sufficient complexity and ecological validity of the keybinding setup and usage tasks, this study employed CrossFire, a popular first-person shooter (FPS) game, as the experimental platform. CrossFire requires players to perform a wide range of operations under time pressure, including basic actions such as movement, jumping, shooting, and reloading; tactical actions such as weapon switching, grenade throwing, and opening the map; as well as functional actions such as chatting, taking screenshots, and using quick items. These operations are highly dependent on shortcut and combination key inputs, making CrossFire an ideal platform for evaluating the effectiveness of different key binding methods in terms of setup efficiency, usage efficiency, and accuracy.

5.2. Experiment 1: Comparison of Keybinding Methods

This experiment was designed to compare the effectiveness of two different keybinding methods. In the experimental group, participants created keybindings using a linear and modular method provided by the researchers, whereas in the control group, participants freely created their own keybindings without restrictions. A within-subject design was adopted, requiring each participant

to complete tasks under both conditions. To minimize potential learning effects, the order of the two conditions was randomized.

5.2.1. Participants

A total of 12 participants (mean age = 24.3 years; 7 males, 5 females), all with prior gaming experience, were recruited.

5.2.2. Tasks

The experimental tasks consisted of three parts:

Part 1: Keybinding Setup Task

Participants were presented with a fixed list of **10 essential FPS functions** and were asked to assign a keybinding to each function. These functions represent the core interactions in FPS gameplay, ensuring coverage of both **basic movement** and **combat/tactical operations**:

1. Move forward
2. Move backward
3. Move left (strafe)
4. Move right (strafe)
5. Jump
6. Crouch
7. Attack (fire weapon)
8. Reload
9. Switch weapon / item
10. Open map

These functions were selected because they represent the most fundamental and frequently used actions in FPS games, covering basic movement, combat operations, and tactical controls. Together, they reflect the core interaction set that determines survival, efficiency, and strategy in gameplay. Many of these actions also occur in rapid succession or as combinations (e.g., move + attack, jump + shoot, switch weapon + fire), making them ideal for testing the effectiveness of different keybinding methods in reducing setup time, lowering cognitive load, and improving consistency. In addition, these high-frequency actions are particularly challenging for players with limited mobility, which further highlights the relevance of evaluating accessibility-oriented solutions. The total time (in minutes) required for participants to complete all 10 bindings was recorded as a measure of setup efficiency.

Part 2: Keybinding Usage Task

The system randomly prompted a function (e.g., “open map”), and participants were required to input the corresponding keybinding. Each participant performed 30 trials per condition, covering all functions in random order. Reaction time (in seconds) and accuracy (in percentage) were recorded.

Part 3: Subjective Evaluation

Participants completed a 5-point semantic differential scale to assess their perceptions of the layouts, including ease of use, comfort, and satisfaction.

5.2.3. Metrics and Expected Values

Setup Efficiency

- Control group: average setup time ≈ 5 minutes
- Experimental group: expected reduced setup time ≤ 3 minutes

Consistency of Keybinding

- The experimental group is expected to show more consistent mappings (e.g., >70% choose M or Ctrl+M for "map"), while the control group results are expected to be more scattered.

Accuracy

- Control group: ≈ 50% success rate
- Experimental group: expected improvement to ≥70%

Subjective Evaluation

- Control group: mean satisfaction score ≈ 3.2/5
- Experimental group: expected improvement to 4.2/5

5.2.4. Procedure

During the experiment, researchers first introduced the purpose and instructions of the study to all participants. Participants were then randomly assigned to begin with either the experimental condition or the control condition. Within each condition, participants completed three steps: a keybinding setup task involving 10 functions, a usage task consisting of 30 randomized trials, and a subjective evaluation questionnaire. After completing one condition, participants repeated the same procedure in the other condition. Finally, the researchers collected all data and conducted statistical analyses focusing on setup time, usage efficiency, accuracy, consistency, and subjective ratings.

5.3. Experiment 2: Evaluation of Layout Optimization

The second experiment was designed to evaluate whether an AI-optimized keybinding layout could improve user performance and experience compared to self-created layouts. Traditional keybinding setups often result in scattered gaze patterns and inefficient keystroke usage, especially in fast-paced gaming scenarios. By leveraging gaze-tracking data and key usage patterns captured through GazeRecorder, the AI system generated optimized layouts intended to reduce keystroke count, shorten key intervals, and improve gaze concentration.

The primary objective of this experiment was to determine the extent to which AI-driven layout optimization enhances task efficiency and subjective usability. Specifically, the study sought to compare baseline user-created layouts with AI-generated layouts across multiple dimensions, including task completion time, keystroke efficiency, gaze distribution, and user perception.

5.3.1. Participants

A total of 12–15 participants, all with prior gaming experience, were recruited to complete the experimental gaming tasks.

5.3.2. Experimental Conditions

A within-subject design was employed, with each participant exposed to two layout conditions:

- **Baseline Layout:** Participants created and used their own custom keybinding layouts.
- **AI Layout:** An optimized layout was generated by an AI model based on data collected during the Baseline condition. The optimization incorporated gaze heatmaps and key usage patterns captured through GazeRecorder.

Each participant completed the same set of game tasks under both conditions. The order of conditions was randomized to mitigate potential learning or fatigue effects.

5.4. Introduction to GazeRecorder

GazeRecorder is a camera-based eye-tracking and gaze analysis software that enables the capture and visualization of eye movement data without requiring specialized hardware. In a comparative evaluation, GazeRecorder was highlighted as the most accurate among webcam-based gaze tracking tools. [13] Using a standard webcam, it records users' eye movements, fixation points, and gaze distribution, and generates corresponding heatmaps and analytical reports. Compared to traditional head-mounted eye-tracking devices, GazeRecorder offers advantages such as easy deployment, lower cost, and strong adaptability, making it well-suited for rapid experimental validation and user research across various scenarios.

In this experiment, GazeRecorder was employed to record participants' gaze distribution and concentration under different keybinding layouts. Heatmap analysis was then used to support the evaluation of AI-optimized layouts. This approach not only reduces dependency on specialized hardware but also provides greater flexibility for conducting experiments in diverse settings.

5.4.1. Experimental Task

The experimental task required participants to complete a series of game operations (e.g., skill activation, item usage) within a fixed 10-minute period. Task content was kept identical across both conditions to ensure comparability of performance metrics.

5.4.2. Data Collection and Metrics

To maintain clarity and simplicity in evaluation, five key metrics were defined:

- **Completion Time:** The total time required to complete the task, or alternatively, the number of tasks completed within the 10-minute timeframe. For example, participants completed an average of 8 tasks under the Baseline layout, while the AI layout was expected to increase this to approximately 10 tasks.
- **Keystroke Count:** The total number of keystrokes required to complete the task. In the Baseline condition, participants averaged around 120 keystrokes, whereas the AI layout aimed to reduce this number to 100 or fewer.
- **Key Interval:** The average interval (in seconds) between consecutive uses of the most frequently pressed keys. The Baseline condition averaged 1.5 seconds, while the AI layout was expected to reduce this to around 1.0 seconds.
- **Gaze Concentration:** Measured using GazeRecorder heatmaps to assess whether participants' visual attention was more concentrated. In the Baseline layout, gaze hotspots covered approximately 70% of the keyboard area, whereas in the AI layou,t this was expected to reduce to 40% or less.

- **Subjective Ratings:** After completing the tasks, participants provided subjective ratings on a 1–5 semantic differential scale. For example, in terms of ease of use, the Baseline layout averaged around 3 points, while the AI layout was expected to improve this score to 4 or higher.

5.4.3. Procedure

At the beginning of the experiment, participants received a brief introduction to the study objectives and a short training session to become familiar with the experimental setup and the GazeRecorder system. Following this preparation, they were randomly assigned to start with either the Baseline layout or the AI layout. Each task session lasted for 10 minutes, during which both keystroke events and gaze data were continuously recorded through GazeRecorder. To minimize calibration drift that might occur due to differences in participants' seating position or height, all participants were instructed to use a laptop while seated at an adjustable desk set to a consistent height. After completing the first condition, participants immediately proceeded to the second layout condition and repeated the same task under identical time constraints. Upon finishing both sessions, participants completed a subjective evaluation questionnaire to provide ratings on ease of use and overall experience. Finally, the collected data were analyzed across five key dimensions: completion time, keystroke efficiency, key interval accuracy, gaze concentration, and subjective perception, in order to compare the effectiveness of the Baseline and AI layouts.

6. RESULTS

6.1. Experiment 1: Comparison of Keybinding Methods

In Experiment 2, twelve participants performed keybinding setup and usage tasks using both the linear and modular methods and the free-design method.

Setup Efficiency: In the free-design condition, participants took an average of 5.2 minutes (SD = 0.9) to complete the setup of 10 functions, compared to 2.8 minutes (SD = 0.7) with the modular method. The difference was significant ($t(11) = 6.02$, $p < .001$). Participants commented that the modular method "provided a clear logic and reduced trial-and-error."

Consistency of Keybindings: For the "open map" function, 72% of participants in the modular condition chose M or Ctrl+M, while only 35% did so in the free-design condition, indicating higher cross-user consistency with the modular approach.

Usage Efficiency: During the usage phase, average reaction time was 0.98 seconds (SD = 0.18) with the modular method and 1.47 seconds (SD = 0.26) with the free-design method. The difference was significant ($t(11) = 5.89$, $p < .001$), showing faster responses with the modular method.

Accuracy: The modular condition achieved an average accuracy of 92% (SD = 4.5), compared to 81% (SD = 5.8) in the free-design condition. The difference was significant ($t(11) = 4.36$, $p < .01$), suggesting that the modular method reduced input errors.

Subjective Ratings: Based on the semantic differential scale (1–5), the free-design condition received relatively lower scores across most dimensions, ranging from **2.4 to 3.6**. For example, it scored particularly low in *Well-organized–Messy* (2.4 ± 0.7) and *Precise–Error-prone* (2.4 ± 0.5), indicating issues of inconsistency and inaccuracy. In contrast, the modular condition showed

consistently higher ratings, with values ranging from **3.6 to 4.6**. Notably, it achieved strong improvements in *Well-organized–Messy* (4.6 ± 0.6), *Reliable–Unreliable* (4.4 ± 0.5), and *Controlled–Randomized* (4.4 ± 0.5), reflecting greater clarity, reliability, and stability.

Table 3. Experience 1 Subjective rating

Dimension	Free-design (M ± SD)	Modular (M ± SD)
Clear–Cluttered	3.5 ± 0.7	4.0 ± 0.6
Well-organized–Messy	2.4 ± 0.7	4.6 ± 0.6
Consistent–Random	2.4 ± 0.5	4.0 ± 0.7
Easy to remember–Hard to remember	3.3 ± 0.6	4.4 ± 0.5
Intuitive–Confusing	3.1 ± 0.7	3.6 ± 0.6
Familiar–Unfamiliar	3.5 ± 0.8	3.9 ± 0.7
Efficient–Cumbersome	3.5 ± 0.6	3.7 ± 0.4
Quick setup–Time-consuming	2.5 ± 0.7	4.3 ± 0.7
Smooth–Interruptive	2.8 ± 0.6	3.9 ± 0.5
Precise–Error-prone	2.4 ± 0.5	4.0 ± 0.6
Reliable–Unreliable	2.9 ± 0.7	4.4 ± 0.5
Controlled–Randomized	3.6 ± 0.6	4.4 ± 0.5
Satisfactory–Unsatisfactory	3.4 ± 0.5	3.7 ± 0.4
Comfortable–Uncomfortable	3.5 ± 0.7	4.2 ± 0.6
Helpful–Useless	3.0 ± 0.5	4.1 ± 0.4

The findings of this experiment provide clear evidence that the proposed linear/modular keybinding method is more effective than the traditional free-design approach. Specifically, participants using the modular method required significantly less time to complete the setup of keybindings, demonstrating a more efficient and streamlined process.

In terms of actual task performance, participants in the modular condition responded more quickly and with fewer errors, indicating that the structured design reduced cognitive load and improved overall usability. Accuracy rates were notably higher, showing that users could reliably recall and execute the correct bindings under time pressure.

6.2. Experiment 2: Evaluation of Layout Optimization

In Experiment 1, twelve participants completed identical 10-minute game tasks under both the Baseline layout and the AI-generated layout.

Task Completion: Participants completed an average of 8.1 tasks (SD = 1.2) in the Baseline condition, compared to 10.3 tasks (SD = 1.0) in the AI condition. A paired-samples t-test confirmed that the difference was significant ($t(11) = 4.52$, $p < .01$), indicating that the AI layout significantly improved task completion efficiency.

Keystroke Count: The Baseline condition required an average of 121 keystrokes (SD = 14.5), whereas the AI layout reduced this to 97 keystrokes (SD = 12.3). The difference was significant ($t(11) = 3.98$, $p < .01$), suggesting that the AI layout reduced redundant actions.

Key Interval: The average interval between consecutive uses of frequently pressed keys was 1.48 seconds (SD = 0.25) in the Baseline layout, compared to 0.98 seconds (SD = 0.20) in the AI

layout. The difference was significant ($t(11) = 5.21$, $p < .001$), demonstrating improved operational efficiency.

Gaze Concentration: Heatmap analysis from GazeRecorder indicated that participants' gaze hotspots covered approximately 61% (Fig. 6) of the keyboard area under the Baseline layout, compared to only 38% (Fig. 7) under the AI layout. This reduction suggests that the AI layout effectively concentrated users' visual attention within central functional zones, thereby minimizing long-distance saccades and improving attentional focus. Participants also reported that the AI layout allowed them to "focus more on the essential keys" and reduced unnecessary visual scanning across the keyboard.

Fig 6. GazeRecorder concentration – Before

Fig 7. GazeRecorder concentration - After

Subjective Ratings: Based on the semantic differential scales, the Baseline layout received relatively low scores across multiple dimensions, with averages ranging between 2.1 and 3.8. The weakest results appeared in *Well-guided–Messy* (2.3 ± 0.7) and *Precise–Error-prone* (2.4 ± 0.6), suggesting that participants often felt the layout lacked structure and accuracy. Other measures, such as *Ease of Use* (3.2 ± 0.6) and *Comfortable–Straining* (3.3 ± 0.7,) also reflected limited usability and comfort.

In contrast, the AI layout achieved consistently higher ratings, with averages ranging between 3.9 and 4.5. Significant improvements were observed in *Efficiency* (4.5 ± 0.5), *Accuracy* (4.4 ± 0.5), and *Reliability* (4.3 ± 0.6), indicating that participants experienced faster and more dependable task performance. Gains were also evident in experiential dimensions such as *Intuitive–Confusing* (4.3 ± 0.5), *Relaxed–Tense* (4.4 ± 0.5), and *Helpful–Useless* (4.2 ± 0.5), which reflected smoother interactions and greater user confidence.

Table 4. Experience 2 Subjective rating

Dimension	**Baseline (M ± SD)**	**AI Layout (M ± SD)**
Fast–Slow	3.4 ± 0.7	3.8 ± 0.6
Efficient–Inefficient	2.3 ± 0.7	4.5 ± 0.6
Smooth–Interruptive	2.3 ± 0.5	3.8 ± 0.7
Accurate–Error-prone	3.2 ± 0.6	4.3 ± 0.5
Easy to use–Hard to use	3.0 ± 0.7	3.4 ± 0.6
Well-guided–Messy	3.4 ± 0.8	3.7 ± 0.7
Comfortable–Straining	3.4 ± 0.6	3.5 ± 0.4
Intuitive–Confusing	2.4 ± 0.7	4.1 ± 0.7
Focused–Distracted	2.7 ± 0.6	3.7 ± 0.5
Clear view–Overwhelming	2.3 ± 0.5	3.9 ± 0.6
Low effort–High effort	2.8 ± 0.7	4.3 ± 0.5
Relaxed–Tense	3.5 ± 0.6	4.3 ± 0.5
Enjoyable–Frustrating	3.3 ± 0.5	3.5 ± 0.4
Helpful–Useless	3.4 ± 0.7	4.1 ± 0.6
Satisfying–Dissatisfying	2.9 ± 0.5	4.0 ± 0.4
Preferred–Avoided	3.2 ± 0.6	3.9 ± 0.6

7. DISCUSSION

The results of both experiments demonstrated that AI-optimized layouts and modular input methods significantly enhanced user performance. Participants frequently highlighted the role of the time dimension in the modular method, such as “holding until the next command terminates” or “single quick trigger.” This allowed them to assign functions by simply conceptualizing their temporal behavior rather than through repeated trial-and-error. Meanwhile, the AI-optimized layout made keys easier to locate and more intuitive, which reduced search time and hesitation. Both methods also contributed to a reduction in errors. In the baseline or free-design conditions, participants often forgot which keys they had just pressed, leading to mistakes, whereas in the optimized conditions, such errors were noticeably reduced. This indicates that structured and systematic layouts can effectively relieve memory load and reduce operational mistakes.

Nonetheless, the study also revealed several limitations. Some participants experienced initial discomfort when first encountering the new layout, reporting difficulties such as not knowing the positions of certain keys or requiring multiple attempts to memorize them. Over time, however, this discomfort diminished, suggesting that new habits could be formed relatively quickly. The concern arises if the AI system were to frequently update layouts: users might face this “relearning cost” repeatedly, which could undermine long-term usability. Therefore, future

design should aim to balance dynamic optimization with layout stability, for example by introducing a progressive update mechanism or allowing users to lock certain key positions. This would preserve flexibility while minimizing the burden of frequent adaptation.In addition, extending this approach to broader application domains and investigating adaptive learning mechanisms could provide valuable directions for future research.

8. LIMITATIONS

Although this approach demonstrates innovation in peripheral management, several limitations remain. First, this approach relies heavily on network connectivity and cloud synchronization. In low-bandwidth or offline environments, its real-time performance and stability may be compromised, limiting its applicability in more complex scenarios. [14] Drawing a parallel from RIS-aided wireless communication—where energy efficiency and signal optimization are crucial—modular input abstractions must also remain robust under network variability and system constraints. As seen in VoIP performance studies over wireless networks, where packet loss and jitter significantly affect user experience [18], the platform may face similar stability challenges in low-bandwidth conditions. [19]

9. FUTURE WORK

Beyond gaming scenarios, future research should expand AI-assisted input methods into broader cross-domain applications. For example, in creative and professional work, tasks such as video editing, music production, programming, and 3D modeling also rely heavily on shortcuts and macros [15]. With AI-driven layout optimization, users could achieve faster workflows, reduced setup time, and greater consistency, thereby enhancing productivity. Similarly, this approach shows potential in education, rehabilitation, and accessible workplaces, where lowering the interaction barrier can provide more inclusive digital experiences for diverse user groups.[16]

At the same time, future systems should incorporate adaptive learning mechanisms, allowing layouts to not only be optimized globally but also gradually personalized to reflect each user's long-term habits. By learning from interaction history, AI could dynamically adjust shortcut priorities, key positions, and macro sequences, creating highly individualized input configurations. For disabled users, the system could adapt to their physical conditions over time, automatically simplifying high-effort actions and supporting more sustainable interaction. Such adaptive capabilities would ensure that the system evolves alongside the user, fostering both skill development and long-term efficiency.

REFERENCES

[1] CDC. "CDC Newsroom." CDC, 2016, archive.cdc.gov/www_cdc_gov/media/releases/2018/.
[2] Bei Yuan, Eelke Folmer, and Frederick C Harris. 2011. Game accessibility: a survey. Universal Access in the Information Society 10, 1 (2011)
[3] Cecílio, José, et al. "BCI Framework Based on Games to Teach People with Cognitive and Motor Limitations." Procedia Computer Science, vol. 83, 2016, pp. 74–81.
[4] Sanchez, Kait, and Ian Carlos Campbell. "When Games Are Hard on Their Hands, Some Players Turn Their Voices into Controllers." The Verge, 12 Mar. 2021
[5] Parker, Laura. "Microsoft's Xbox Adaptive Controller Gives Disabled Gamers a Power-Up." WIRED, 25 Sept. 2018
[6] Bunting, Geoffrey. "How AI Can Make Gaming Better for All Players." WIRED, 10 July 2023, www.wired.com/story/ai-make-gaming-better-accessibility.
[7] Martinez, Jesse, et al. Joy in Video Game Adoption for Gamers with Disabilities. Vol. 17, 2024.

[8] Loewen, Georgia. “Exploring Gaming Wearables for Players with Upper Limb Motor Disabilities through Design Fiction and DIY.” Companion Proceedings of the 2024 Annual Symposium on Computer-Human Interaction in Play, 14 Oct. 2024, pp. 427–430

[9] Varma, Mrs. D. Bhavya, et al. “Voice-Controlled Gaming Tools for Enhanced Learning in the Skill Ecosystem Roll.” International Journal of Research Publication and Reviews, vol. 6, no. 4, Apr. 2025, pp. 14281–14285.

[10] Qiu, Tianrun, et al. Gamer Astra: Enhancing Video Game Accessibility for Blind and Low-Vision Players through a Multi-Agent AI Framework. 28 June 2025.

[11] Pernice, Kara. “F-Shaped Pattern of Reading on the Web: Misunderstood, but Still Relevant (Even on Mobile).” Nielsen Norman Group, 12 Nov. 2017.

[12] “Speech Recognition for Accessibility in Gaming Tools.” Meegle.com, 2025, www.meegle.com/en_us/topics/speech-recognition/speech-recognition-for-accessibility-in-gaming-tools.

[13] Abdrabou, L Yasmeen, et al. “Eye See Identity: Exploring Natural Gaze Behaviour for Implicit User Identification during Photo Viewing.” Proceedings 2024 Symposium on Usable Security, 2024.

[14] Ewelle, Richard Ewelle, et al. “Network Traffic Adaptation for Cloud Games.” ArXiv.org, 2025.

[15] Mitchell, Claire L., et al. “Ability-Based Methods for Personalized Keyboard Generation.” Multimodal Technologies and Interaction, vol. 6, no. 8, 1 Aug. 2022, p. 67.

[16] Varma, Mrs. D. Bhavya, et al. “Voice-Controlled Gaming Tools for Enhanced Learning in the Skill Ecosystem Roll.” International Journal of Research Publication and Reviews, vol. 6, no. 4, Apr. 2025, pp. 14281–14285

[17] Ali H. Wheeb and Munsifa Firdaus Khan, *“A Survey of Several Machine Learning (ML) Algorithms for Security Solution in Internet of Things (IoT) Networks,”* Journal of Artificial Intelligence Research & Advances, vol. 12, no. 1, pp. 1–11, 2024. Available: https://journals.stmjournals.com/joaira/article=2024/view=191585/

[18] A. H. Wheeb, F. Shaik, and K. Shaik, "Two Purpose-Oriented RIS-Aided Schemes to Enhance and Evaluate the Performance of Wireless Communication," *COJ Electronics & Communications*, vol. 3, no. 1, article COJEC.000555, Oct. 2024. doi: 10.31031/COJEC.2024.03.000555.

[19] Ali H. Wheeb, “Performance Analysis of VoIP in Wireless Networks,” *International Journal of Computer Networks and Wireless Communications (IJCNWC)*, vol. 7, no. 4, pp. —, Jul.–Aug. 2017.

AUTHORS

Minzhou Wang is a UX/UI designer specializing in smart hardware and human-centered design. At Husqvarna Group, she leads user journey and interface design for robotic mowers, creating intuitive digital-physical experiences. Holding a Master of Fine Arts in Interactive Design from SCAD, she combines design thinking and technical insight to enhance usability, accessibility, and everyday interaction.

Peijin Du is a UX and product designer, Peijin Du has experience across artificial intelligence, augmented reality, geospatial tools, and a variety of consumer applications. She isa Senior Designer at Esri, where she leads the design of native iOS and Android interfaces for advanced Map SDKs, creating intuitive and accessible mobile experiences for developers and organizations worldwide.