Proposal of an Alternative HMI Mechanism for Blind Android Users Based on Media Headsets as Input/Output Peripherals

Miguel Páramo Castrillo(✉), Silvia de los Ríos(✉),
Juan Bautista Montalvá Colomer,
María Fernanda Cabrera-Umpierrez, and María Teresa Arredondo

Life Supporting Technologies,
Universidad Politécnica de Madrid, Madrid, Spain
{mparamo, srios, jmontalva, chiqui, mta}@lst.tfo.upm.es

Abstract. This paper introduces an alternative method of interaction over Graphical User Interfaces (GUIs), using a button enabled headset as a main Input/Output (IO) peripheral. This paper is focused on the underlying basis which is based on the results and implementation of an Android prototype once the viability for this system has been remarkably proven. Being blind Android users the initial target beneficiaries does not restrict the scope of this solution since its conceptual approach is scalable to other various systems and may be applied to mainstream users. Hence, the main objective is to create an alternative, effective and low cost mechanism for blind users or eyes free scenarios by means of the buttons built in modern media headsets; or in other words, the main motivation of this article is to present this solution as an alternative Human Machine Interaction (HMI) mechanism with the objective of proposing an elegant and comfortable way of interaction over specifically designed software; resulting in a plausible better solution for many use cases where the eyes free approach may be beneficial.

Keywords: Interaction · Eyes-free · HMI · Android · Blind · Headset · Accessibility · UX · UI

1 Introduction

There are close to 40 million blind persons and 285 million of visual impaired worldwide [1]. A fraction of these people uses a smartphone on their daily basis and in any case, no one should be excluded from using it. Blind people face the challenge of using a mobile device that is designed for the non visual impaired and integrates a touch screen as the main input hardware. Blind users tend to interact with their devices with the assistance of a voice synthesizer or TTS (Text To Speech) combined with an "Explore by Touch" mode in detriment of the usage of traditional output to Braille accessories. This assistive solution is integrated by default on the main mobile operating systems (Android [2] and iOS [3]) and relies on a software that recognizes a set of gestures the user can "draw" with their fingers over the touch screen; enabling the navigation across the GUI, the identification of the different elements within it and the interaction with

M. Antona and C. Stephanidis (Eds.): UAHCI 2016, Part II, LNCS 9738, pp. 175–184, 2016.
DOI: 10.1007/978-3-319-40244-4_17

them while providing spoken feedback through the TTS at the same time. According to the general opinion of blind users, this current accessibility implementation has still slight room for improvement but is solving the obvious challenges they face and is satisfactory, especially the iOS implementation.

Despite the viability of the current solutions, they attempt to adapt the GUIs instead of doing the inverse process: not forcing the people to operate with an input hardware that was not originally conceived for them.

Some hypotheses are proposed: interaction based on gestures is not so intuitive and may lack precision; while the feedback given by the headset buttons themselves feels instead more accurate and precise. Apart of that, some Explore by Touch gestures are not natural or intuitive and require some training and learning [4]. Finally, the screen is also one of the major battery drainers of the smartphones (according to independent tests, on average it takes at least 30 % of the total or a regular usage) so this alternative system would certainly save power (using the screen to display any GUI is optional).

Modern mobile headsets have evolved and may integrate a built-in microphone for hands-free calls, up to three buttons for media player controlling [5] or even radio antennae. They keep the same size of a traditional headset and are retro compatible with standard jack plugs; meaning they can be plugged directly into a compatible jack port without requiring alternative hardware or further setup (Fig. 1).

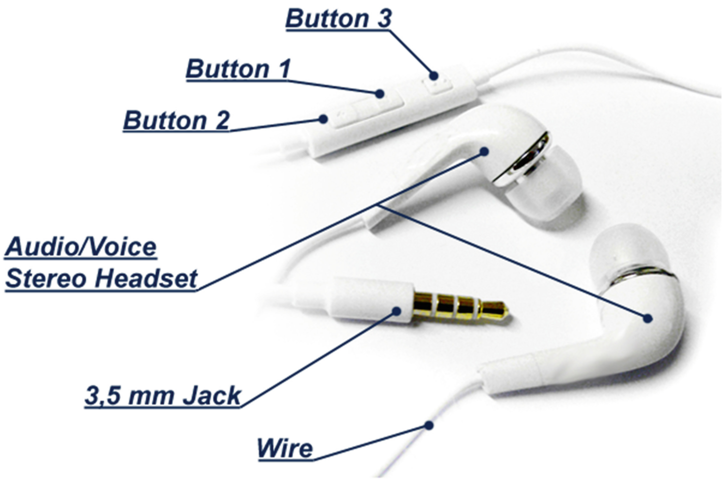


Fig. 1. A three-buttoned media headset by Samsung

2 Methodology

This proposal has been conceived following steps: Analysis, high level design, proof of concept implementation, prototype implementation and tests. As a prototype or Minimum Viable Product (MVP), the initial phase consisted on planning the solution and the HMI mechanism; this step also implied the analysis of use cases and potential benefits for the users. After that, the initial proof of concept (alpha) version was coded over a simple and function-less app in order to ensure that the implementation was feasible for the target system. Once the initial prototype was created and polished, then a full demo newsreader app with several functions was created. This second MVP (beta) was the solution subjected to evaluation with several users: 14 users: 12 non blind users and 2 blind users. All the feedback given was analyzed and conclusions were taken afterwards.

The track to achieve the proposed solution had two fundamental pillars which are the Software Development Methodology (SDM) and the User Centered Design (UCD) methodology [6]. Due to the cyclical SDM process and the also cyclical nature of the UCD methodology, both were combined into a single development process were users would participate and guide the successive iterations of the product. At the current state of the proposed solution, technical viability was granted and the proof of concept was made, hence not requiring a second iteration.

3 Implementation

This HMI implementation requires either a software layer built on top of some OS (Operating System) interaction mechanisms (acting as a hook for certain input events) or a custom made application that handles the behavior of the input events. Both solutions are feasible (at the moment this paper was written) but the first one relies much more on the Android OS and may not have granted stability and maintenance in the future, plus, it is less flexible and powerful since it limits the hooked events for security reasons and the hooking mechanism is more complex and slower, presumably lagging the response time and jittering the interaction. So for those reasons, the MVP was created following the second paradigm as described as follows. In any case, the specific implementation for Android can essentially be adapted to other systems.

When pressed, those headset buttons feed the device with a signal that is directly mapped by the OS into a keystroke as if generated directly from a keyboard. The Android API [7, 8] legitimately allows capturing and managing these input events directly from a foreground application (that could be the main launcher but for security reasons, not a third party app). Developers can use this eases in order to build apps based on the here exposed solution; where tactile exploration is not required and their users can control the whole application even keeping the device into their pocket.

When the prototype is launched, the Text To Speech engine is activated so it is ready to be called from the interface anytime. It operates in “Flush” mode, meaning that any message sent will not be queued but played immediately instead; even “cutting” the previous pending utterance if any. This way any user that has gain expertise with the system can interact far more quickly and also there are no problems of synchronization or delay among the focused element with the spoken feedback. Each time an element gains focus or is selected (both events are treated as the same) the TTS provides a description of it.

The given implementation considered different input events that are mapped to trigger certain actions depending on the operating mode of the user interface.

3.1 Input Events

Given a single commutable signal, there is a chance of extending the input it feeds into the system by controlling the intervals and duration of the "clicks". An extra degree of freedom (the time variable) allows a single input signal to "expand" into the following three:

- **Single click (S):** The user clicks on the button for a short time. The interval where the switch is commuted is shorter than the one established for the "long press" event.
- **Double click (D):** The user clicks twice times the commuter in a short period of time. The interval between the two clicks is shorter that the window time established for two separated "single click" events.
- **Long press (L):** The user presses the button for a time longer than the defined for a "single click".

3.2 Actions

On the other hand, the plausible actions that can be carried out are the following:

- **Activation (A):** It will activate the focused element. The response of the element will likely provide a description of it, navigate to another menu, modify its state or trigger a certain predefined action.
- **Next element (N)**: Action that navigates to the next element in the UI. It will un-focus the current element and focus/select the next one. The TTS will speak the description of the element being now focused automatically. As stated before, the interface is linear and cyclical so the last element will be concatenated with the first and vice versa.
- **Previous element (P):** The exact opposite of "Next", it gives focus to the previous element in the UI.
- **Back (B):** Action that returns back to the previous menu. If the user switched to another menu, since they are stacked, this event will "close" the current menu and enter back into the one where the current was coming from (Fig. 2).

3.3 UI Setups

There are certain particularities of the interface's behavior that must be considered. These HMI mechanisms are based on a sequential (linear) and cyclical UI. Cyclical means that it "chains" in both directions (next and previous) the first and last element within the interface. Also, being linear makes each menu to position the UI components in just one dimension (vertical or horizontal). Apart of those two important characteristics, the UI can also function under those following setups based on the way they handle the inputs, the logic within the UI or the UI design principles (Fig. 3).

- **Default (D):** Each action has a direct input which is triggered on the release of the input event. Interface that only responds to input events without any other internal behavior.

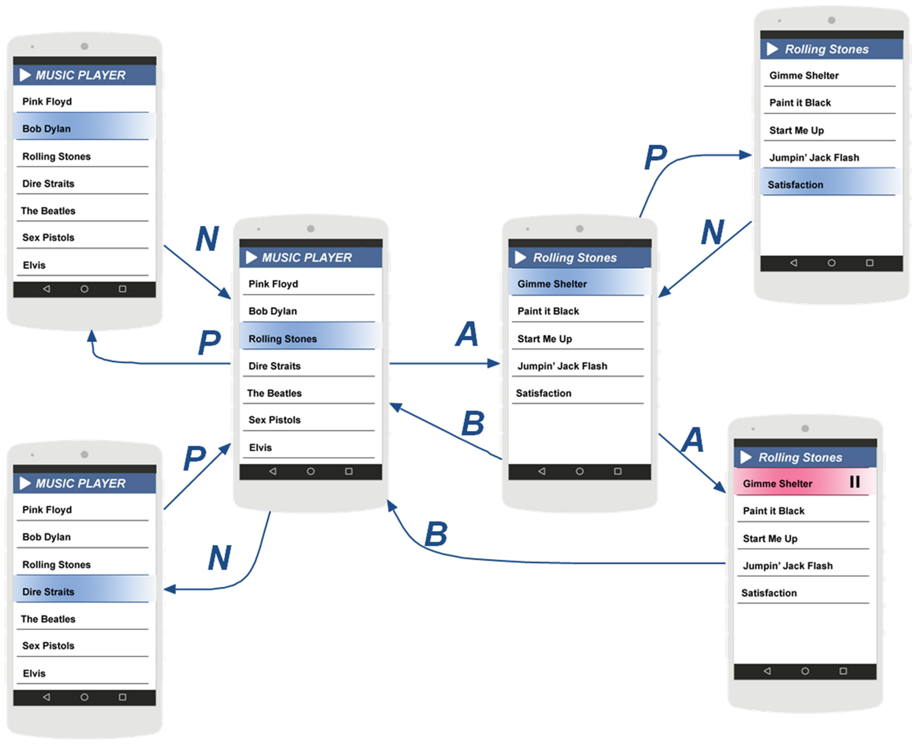


Fig. 2. Example of a diagram of inputs and its transitions under default (D) or continuous (C) GUI setup.

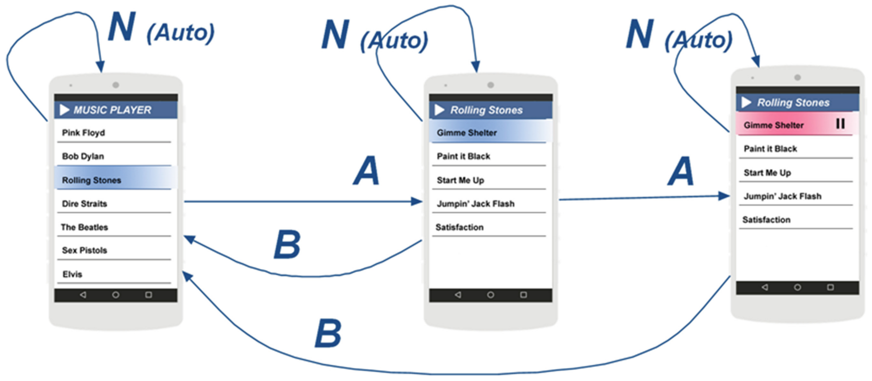


Fig. 3. Auto-scan (S) mode. There are no “Previous” actions and the scanner inputs “Next” automatically.

– **Continuous input (C):** Particular case of the default mode. An input signal that is kept in time (by definition, only applicable to long presses) can be kept acting even after the initial treatment of the input. Unlike the default mode, it delivers events instead of waiting for a release of the commuter. This continuous "Long press" input is interesting and supposed to be limited to generate a continuous flow of "Next" events (it would be similar to a manual auto-scan that stops when the user releases the button) or "Back" events (similar to going quickly from the hierarchy of menus to the root).
– **Auto-Scan (S):** The "Next" (or "Previous") action is automatically triggered at regular pulses by a timer; so the UI is continuously and automatically advancing from one element to the next. This mode is opened to other exclusive actions (these actions for scan control are referred as **R** from now on) such reverting the scanning direction, halting/un-halting it or even slowing it down briefly (Fig. 4).

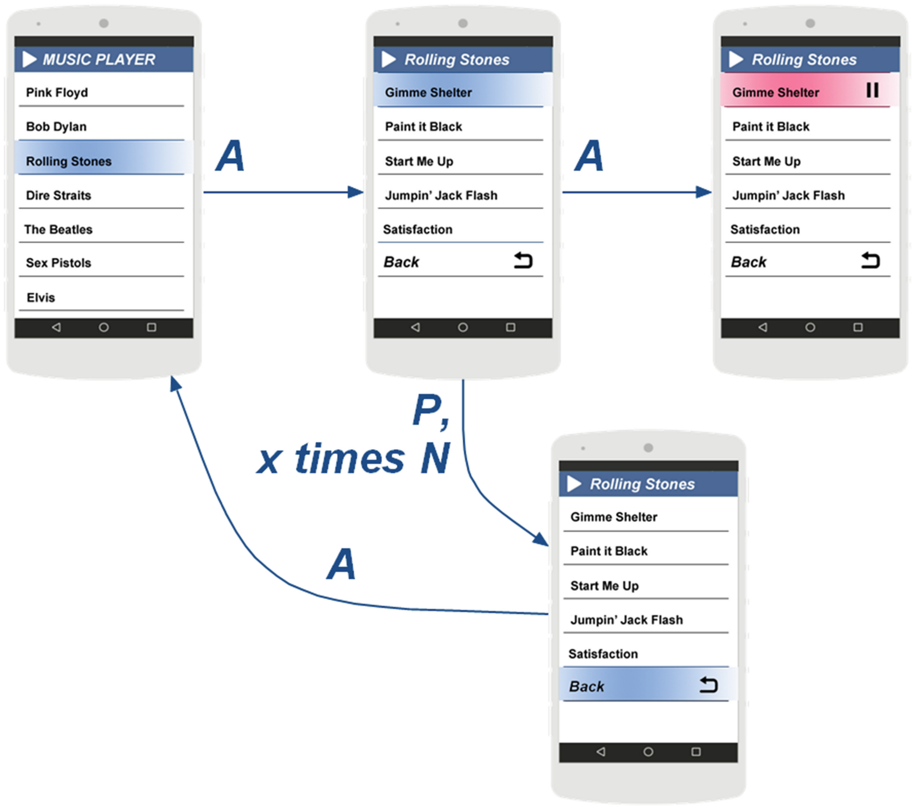
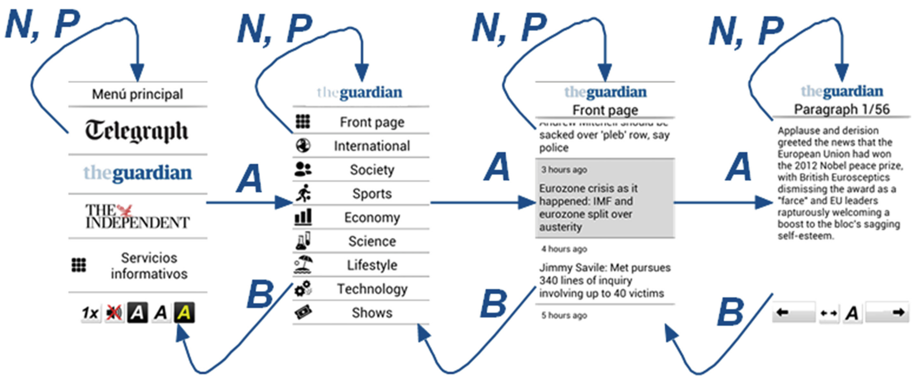


Fig. 4. Back as an element (E) mode. There are no Back actions since the UI includes an element that triggers this action.

– **Back as Element (E):** Design principle that includes the “Back” action as a focusable element within the UI. This setup is complementary to the rest (Fig. 5).



sacked over 'pleb' row, say police

Eurozone crisis as it happened: IMF and eurozone split over austerity

Jimmy Savile: Met pursues 340 lines of inquiry involving up to 40 victims

Applause and derision greeted the news that the European Union had won the 2012 Nobel peace prize, with British Eurosceptics dismissing the award as a "farce" and EU leaders rapturously welcoming a boost to the bloc's sagging self-esteem.

Fig. 5. Newsreader prototype on continuous mode

It is possible to define the behavior of **1 to 3** buttoned headsets by mapping the inputs (**S**ingle, **D**ouble, **L**ong) with the actions (**A**ctivate, **N**ext, **P**revious, **B**ack and scan control **R**) working under certain setups (operation modes) of the UI (**D**efault, Auto-**S**can, **C**ontinuous input and with Back as **E**lement).

3.4 Single Buttoned Headset Viable Configuration Mappings

It is available just a single button. Single click advances to the following element, double activates and long press returns. No previous action can be performed but the cyclical interface solves the issue. A variant integrates the “Back” as a focusable element at the end of each menu. In this second case, either double click, long press or both can be used to activate (these optional configuration is represented as -/symbol in all tables, where at least one of the inputs must handle the event):

Mode D	Input		
Button	S	D	L
1	N	A	B

Mode DE	Input		
Button	S	D	L
1	N	-/A	

Other alternatives; click activates while long press moves continuously to the next element. Double click returns back:

Mode C	Input		
Button	S	D	L
1	A	B	N

Its variation including the “Back” element can avoid using the double click:

Mode CE	Input		
Button	S	D	L
1	A	-/A	N

Mode CE	Input		
Button	S	D	L
1	N	A	N

Finally, under auto-scan, the button just activates the focused element since the logic behind the interface is responsible of the automatic and sequential navigation. Flux control of the scanner is optional as well:

Mode S	Input		
Button	S	D	L
1	A	-/R	B

Mode SE	Input		
Button	S	D	L
1	A	-/R	

3.5 Double Buttoned Headset Viable Configurations

Under those kind of headsets mappings for button 1 or 2 can be reverted. One button goes previous and its long press returns back. The second button goes next and its long press activates:

Mode D	Input		
Button	S	D	L
1	P	-	B
2	N	-/A	

Mode DE	Input		
Button	S	D	L
1	P	-/A	
2	N		

Also other configurations are viable without mapping the “Previous” action. One button activates and when long pressed it returns back. The second button navigates sequentially and optionally in continuous mode:

Mode D	Input		
Button	S	D	L
1	A	-	B
2	N	-	-

Mode C	Input		
Button	S	D	L
1	A	-	B
2	N	-	N

Under auto-scan, the first button controls activation and the second just the scanner flux. Long press returns. In case of “Back” element implemented, activation on other events apart from single click is optional:

Mode S	Input		
Button	S	D	L
1	A	-/A	-/B
2	-/R	-/R	

Mode SE	Input		
Button	S	D	L
1	A	-/A	
2	-/R		

3.6 Triple Buttoned Headset Viable Configurations

Headsets with more than three buttons can avoid “complex” interactions such the double click and can be set in a natural an easy way of interaction. Also, having such variety of inputs, there is no need to implement the auto-scan or the “Back” despite they can be set that way. The two main configurations are the same under default or continuous mode. It can be used one button for each next/previous event while the third button activates and its long press returns:

Mode D	Input		
Button	S	D	L
1	A	-	B
2	P	-	-
3	N	-	-

Mode C	Input		
Button	S	D	L
1	A	-	B
2	P	-	P
3	N	-	N

Other mappings for those three buttons may not feel much natural. Also any UI in auto-scan or with the back element over-complex or redundant since there is enough control capability with those 3 buttoned headsets.

4 Results

A first prototype was made for the Android platform and tested with 12 non blind users and two blind users, as mentioned above. In the test sessions they attempted to complete a variety of uses cases just using the buttons on the media headset and not being able to either touch or watch the screen. The observation of the trials and the focus groups taken afterwards for a preliminary evaluation were very positive and increased the interest of deeper research in order to improve the yet promising results. The prototype consisted on a GUI integrating a complete news reader among other functionalities. Users could consult a given newspaper, navigate among its categories and the latest news published in each and also control the article presentation paragraph by paragraph. The prototype was based on the continuous mode setup under a three buttoned headset and the previously detailed mapping.

The results were very promising since all of the users were able to interact with the prototype application without being able to see the screen and also without any training apart of a brief explanation. After few seconds they could interact naturally, quickly and with an extreme accuracy. There were no errors or misunderstandings as long as the TTS information was also enough descriptive. Also, two of the users were technically limited and never had used a smartphone before. One of them was a blind woman used to work with a Nokia device with a built in voice synthesizer. Even though they could complete all the trial use cases at its first attempt and even started using the system by themselves freely. This blind woman was totally impressed and delighted with the solution and demanded for a complete system based on this method of interaction.

5 Conclusions and Future Lines

The technique detailed in this paper is a viable alternative for eyes free control of UIs and has the potential of being extremely useful for blind users. It is yet pending to evaluate the solution with a wider sample of blind users and with a more sophisticated and improved prototype and also to test it in terms of satisfaction, accuracy, speed and power consumption against the commercial solutions (Touch to explore and Voice over). However the initial tests were promising and the feedback gathered during the trials will be considered in future iterations of the product despite those considerations are more related to the UI itself than the HMI mechanism.

Finally, the conclusions of this experiment put into manifest that other techniques involving accessible design, an ontology of interactive components and layout definitions of the UI are not mandatory but extremely relevant in order to achieve an excellent user experience when a GUI-less is meant to be designed under following this HMI mechanism. Those fields, along with a deep analysis and specification of the setup modes are beyond the scope of this paper but will be considered on future lines for an eventual development of a commercial solution under the precepts of Universal design.

References

1. WHO: Visual impairment and blindness fact sheet N°282. http://www.who.int/mediacentre/factsheets/fs282/en/
2. Google Inc.: Explore by touch in talkback. https://support.google.com/accessibility/android/answer/6006598?hl=en
3. Apple Inc.: Voice over getting started. https://www.apple.com/voiceover/info/guide/
4. Kane, S., Wobbrock, J., Ladner, R.: Usable gestures for blind people: understanding preference and performance. In: Proceedings of 2011 Annual Conference Human factors Computer System - CHI 2011, pp. 413–422 (2011)
5. US Patent: US20110263303, Multi-button control headset for a mobile communication device (2011)
6. Abras, C., Maloney-Krichmar, D., Preece, J.: User-centered design. In: Bainbridge, W. (ed.) Encyclopedia of Human-Computer Interaction, vol. 37 pp. 445–456. Sage Publications (2004)
7. Google Inc.: Android Activity (onKeyDown) (2016) http://developer.android.com/reference/android/app/Activity.html#onKeyDown%28int,%20android.view.KeyEvent%29
8. Google Inc.: Android KeyEvent http://developer.android.com/reference/android/view/KeyEvent.html