INTUITIVE

INnovative Network for **T**raining in To**U**ch **I**nterac**TIVE** Interfaces

Grant agreement: #861166 H2020-MSCA-ITN-2019

Start date: 2019-10-01 End date: 2023-09-30

Deliverable reporting document

Deliverable no: 5.6		WP: 5
Deliverable name: Design principles and basic version of the intuitive tactile user interface	Type: Report	Dissemination level: Public
Due Delivery date: 30 April 2022		Date delivered: 2 May 2022

Description:

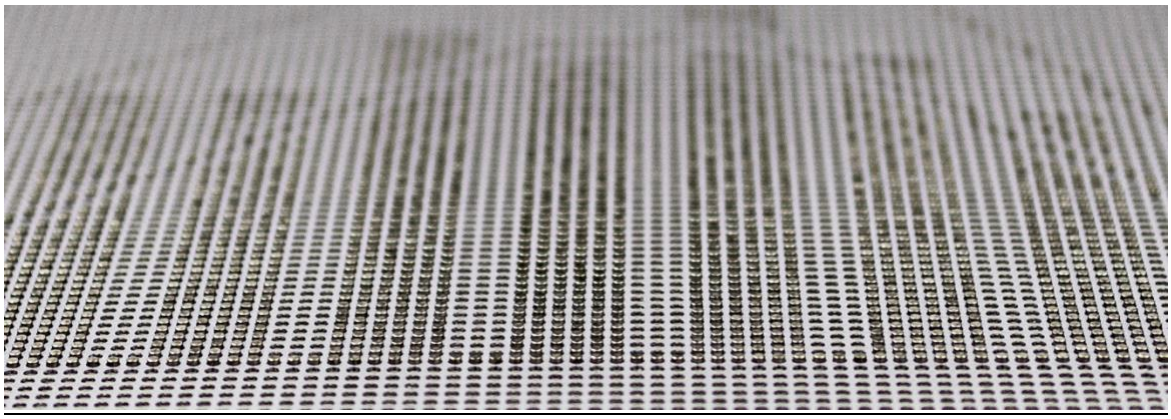
Software prototype for the new user interface allowing interaction between the Tactonom and computer

Design Principles and basic version of the intuitive tactile user interface

Software prototype for the new user interface allowing interaction between the Tactonom and computer.

Researcher:
João Gaspar Ramôa Gomes

Supervisors:
Prof. Dr. Rainer Stiefelhagen (KIT – SZS)
Dr. Thorsten Schwarz (KIT – SZS)
Dr. Alexander Hars (Inventivio GmbH)



Abstract

Enabling blind and visually impaired persons to perceive and interact with two-dimensional data is a complex challenge. Blind users cannot perceive graphical information in visual user interfaces and current assistive technologies only allows linearly text information access. Two-dimensional refreshable audio-tactile devices are an emerging technology capable of presenting two-dimensional data to blind persons through audio feedback and a braille matrix display constituted by tactile pins that can be raised up and down While hardware development for this technology has advanced, user interface development has not progressed likewise. There is a need to improve these devices' user interfaces to enable blind and visually impaired persons to access and use to-dimensional information. The user interface includes several challenging obstacles, such as audio-tactile synthesis representation, dialogue architecture, proper 2d tactile layout representations, and widgets for supporting the understanding of information by blind persons.

Introduction

To address the UI design principles and highlight the most interesting aspects for the development of a tactile user interface version, it is necessary to conceptualize the term of user interface.

Many different interpretations of user interfaces have been proposed. Some interpretations look primarily from the perspective of the design process of a user interface. Others addressed two different types of conceptualizations that delineate the user interface, software-based and interaction-based. Another conceptualization in the context of human computer interaction was proposed by the HCI ACM SIGCHI Curricula architecture. Our interpretation is based on ISO 9241-220:2019 where user interface is the set of all components of an interactive system (software and hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system. It was developed an architecture interpretation of user interface for 2d refreshable tactile user interfaces in order to identify the most interesting areas for research and define the design principles for this technology. The UI key aspects of this technology are conceptualized in Figure 1.

The user interface conceptualization was split into 6 categories:

- Input and output devices

  Include hardware aspects of the user interface, such as, the tactile display, the refreshment rate of the tactile pins, buttons and its arrangement, camera characteristics, weight and others.

- Input control

  Transformation of “raw” input into more detailed input signals, including pointing operations, gesture recognition and interactions, keyboard shortcuts techniques, speech-to-text and fingertip recognition.

- Information representation

  Includes the information represented on the tactile pin-matrix display, such as icons, braille text and audio-tactile widgets.

- Layout

  Arranges the previous elements, including windows designing, references lines, zooming and panning operations.

- Dynamic control

  Audio and tactile UI contents change according to user input over a period of time, in a dynamic interaction. An example is the use of dynamic navigation that uses dynamic audio-clues to navigate the user hand to a specific element.

- Dialogue architecture

  Comprises how the menu is structured, how files are presented, and how notifications are processed. Pagination, nomenclatures and status messages are included in this category.

This document will address each user interface category at time, including its tradeoffs and interesting aspects. Specific application examples will be presented for each user interface category as well, such as: *File Explorer*, *Plain text Editor*, *Image Viewer, Excel table, Widget library* and others. The user interface elements are implemented in the Tactonom device, a 2d refreshable audio-tactile reader developed by Inventivio GmbH in Germany.

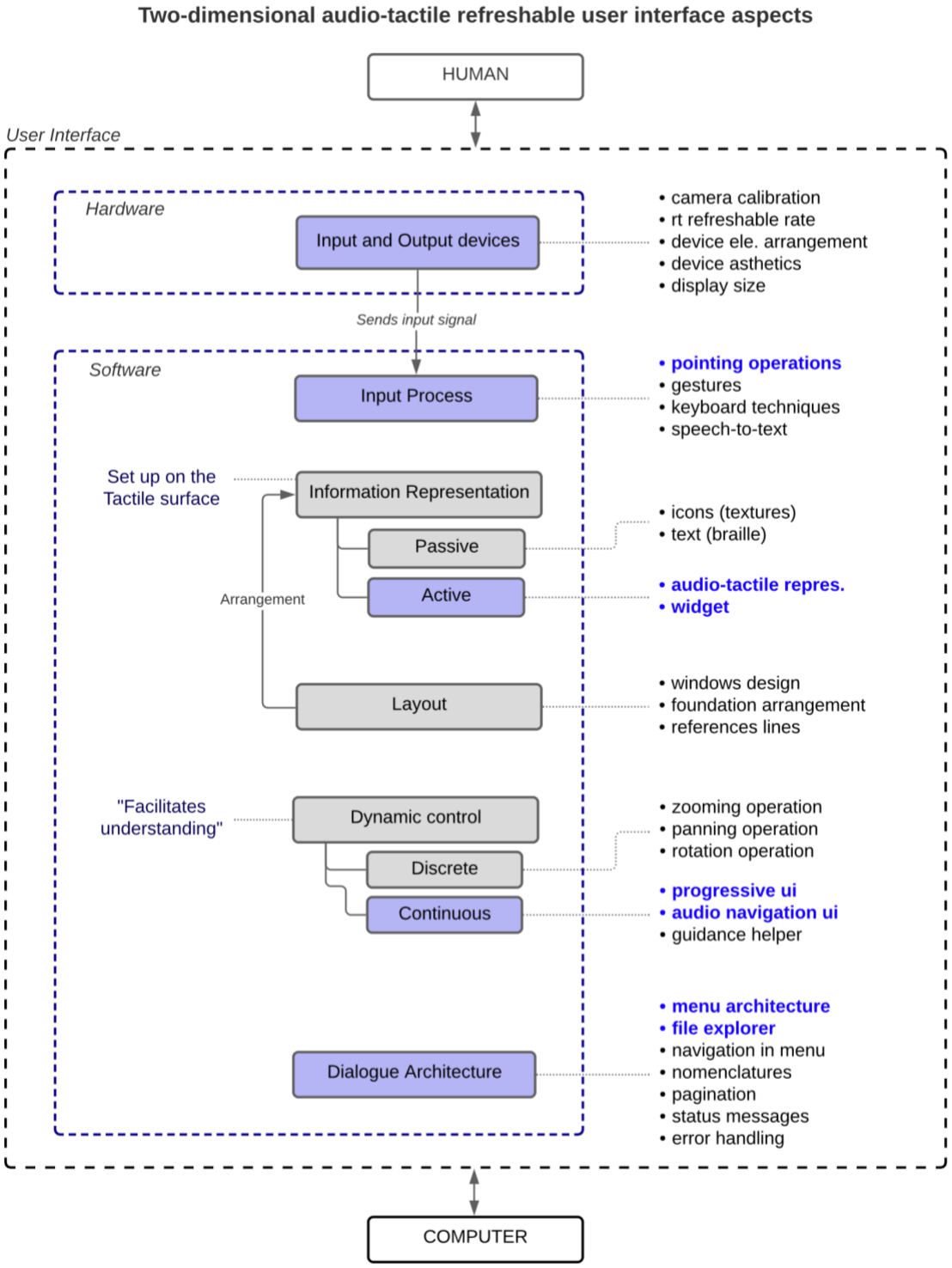


Figure 1 – User Interface aspects representation of two-dimensional audio-tactile refreshable devices. The highlighted elements in bold represent the most critical areas.

Input and output devices

The refreshable tactile surface of two-dimensional refreshable tactile displays is the most distinguished characteristic of these devices. The tactile surface comprises small tactile pins (taxels) distributed in a two-dimensional matrix. Related work has developed 2DRT displays that use different tactile surface resolutions and sizes. To evaluate the different tactile surface sizes, we have analyzed up to 18 2DRT devices from related research and measured the number of taxels per device (figure 2).

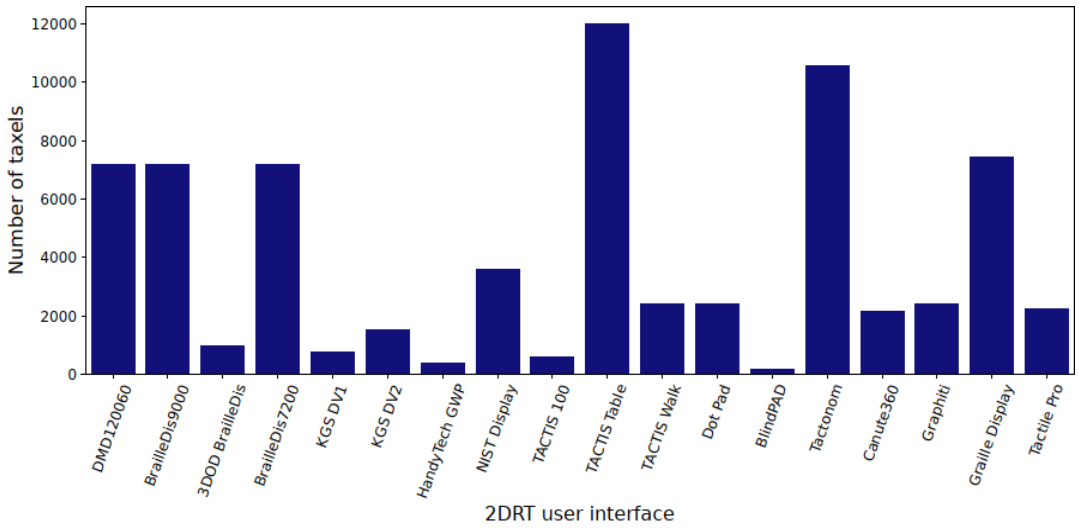


Figure 2 - Number of taxels (surface resolution) of 2D refreshable tactile devices.

Based on the collected data, 2DRT user interfaces can be split into large surface resolution (more than 7000 taxels) and small surface resolution (equal to or less than 7000 taxels). Larger surface resolutions enable the representation of more elements and more detailed tactile graphics, such as the Tactonom, TACTIS Table, the BrailleDis devices and the Graille display. However, higher resolutions imply a better organization of the information since the tactile surface can present a higher number of tactile elements, leading to information overload on the tactile surface. The Tactonom device has a total of 10.591 taxels.

Beyond looking at the surface resolution, it is interesting to look at the relation between the number of taxels and the screen size in square centimeters, since 2DRT tactile surfaces use different spacing distances between pins. A linear regression line of the previous correlation was drawn in figure 3. Results show that most 2DRT devices have a very similar relation between the number of taxels and screen size. Devices that use a higher pin spacing distance (Graphiti and BlindPAD) or do not have an equidistant spacing between taxels (Canute360) stood out from the average correlation. It is necessary to have a proper pin-spacing distance (between 2.4 to 3.0mm to support braille text presentation, which is used by the other 2DRT devices, including the Tactonom device (2.5mm). Equidistant distance between taxels is necessary to present a more extensive set of graphics, as conventional GUI screens use. Irregular pin distances have empty gaps that cannot display the visual information, leaving the two-dimensional output information with openings and holes.

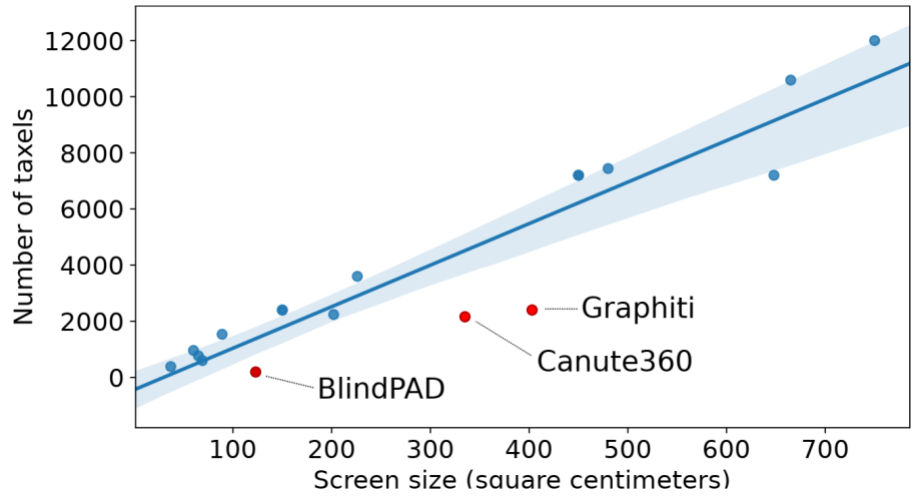


Figure 3 - Number of taxels as a function of the screen size in (cm2) of 2DRT devices.

We also correlated the number of taxels as a function of the refreshment rate of each device (represented in figure 4.) Results show that the BrailleDis 9000 and 7200 devices take an average of 0.2s and 0.05s, respectively, to refresh their 7200 pin-matrix surfaces, standing out from the other devices. Some device's pin-mechanisms take up to 10s to refresh the entire screen, limiting the representation of more tactile dynamic information. Still, devices such as the Tactonom and the TACTIS Table stand out positively from the average correlation due to their large surface size (bigger than 10.000 taxels.) The refreshment rate depends on the mechanism's speed that raises up and down the pins. Unlike the fast and expensive piezo-driven dots used in the BrailleDis devices, the Tactonom uses magnets that move beads so tactile pins can rest on top of them. If the bead is not positioned below the pin, the tactile pin will have nothing to rest on and will be raised down.

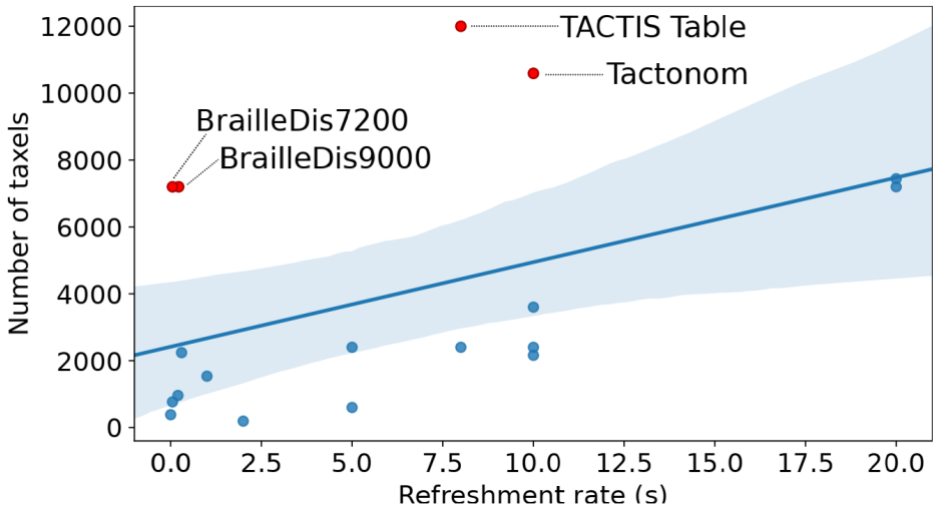


Figure 4 - Number of taxels as a function of the refreshment rate (s) of 2DRT devices.

Arrangement of the physical buttons on the 2DRT device is an important aspect of the UI. Beyond the type of buttons and their size/shape, their positioning on the device can avoid the Midas Touch effect. If buttons are positioned in the bottom part (south to the tactile surface), the user can accidentally press a button with their arm while touching the pin-matrix display. The Tactonom device has positioned its buttons on the sides (left and right). This way, the previous problem does not occur. Buttons are not the only important element whose position needs to be considered, audio speakers' positioning does also influence user interactions with the system. If both left and right speakers are positioned side by side, spatial audio is not perceived since audio is coming from the same direction. In order to create spatial audio which can be used to indicate positions to the user, left and right speakers should be positioned away from each other.

The most distinctive characteristic of the Tactonom in this technology is the use of an RGB camera. This camera is facing the device's tactile surface in order to detect the user's fingertip and recognize gestures. Other devices opted to use a touchable pin-matrix surface to keep track of the user's hand position, which replaces in this technology the mouse cursor in GUI.

Figure 5- Tactonom device input/output characteristics overview.

Input control

Differently from input and output devices' UI aspects, input controls are the interfaces responsible for interpreting raw signals from input-output devices into more detailed user inputs. These include camera-based finger detection and keyboard combinations and shortcuts.

Fingertip detection is a distinctive UI characteristic of the Tactonom from the rest of the state-of-the-art. The Tactonom uses a computer vision-based algorithm to detect the user's fingertip position on the tactile surface. Initially, the frame lighting is adjusted if lighting conditions are too dark or too light, this is known by calculating the L parameter from the LAB color channel of the frame. The user's hand is semantically segmented from the camera frame using a color filter threshold that correlates the RED value against the GREEN value in the RGB color channel space. This process returns several candidate clusters.

Further analysis is necessary to select the correct cluster (user's hand cluster) by using width and height, and minimal size values. After selecting the final cluster candidate, the fingertip position is located by using the highest pixel of that cluster. Figure 6 represents the aforementioned algorithm for fingertip detection.

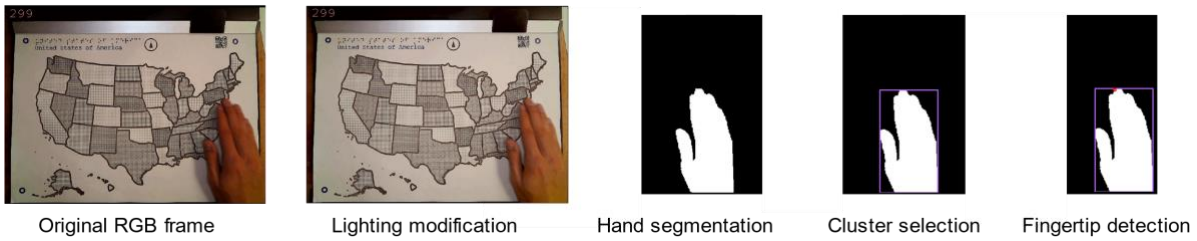


Figure 6- Tactonom' fingertip detection algorithm

Fingertip allows the user to execute operations since it can be used as a mouse cursor. Different alternatives can be used to select an element or simulate the equivalent process of “double-tap” in GUI. For example, let’s take the situation where the blind user explores the two-dimensional tactile surface with both hands, which is the most common way for blind persons to explore graphics and two-dimensional information. The 2d tactile surface has tactile elements that can be executed or “double-tap”. The user can indicate which element or object he/she wants to complete by moving their fingertip to the element located on the tactile surface. One approach to execute an operation on that element would be by using the other hand to press a physical button. After pressing the button, the user would move his/her hand back to the tactile surface. Another approach is to use a gesture with the hand that is on top of the element by using a camera-based gesture recognition algorithm. The other hand does not need to move away and remains on the tactile surface. Both approaches are interesting since they have their pros and cons. The physical button approach presents a big disadvantage since the user needs to move one of his/her hands away from the tactile surface to press the button. When the hand returns to the tactile surface, it is hard for the user to return it to the exact same

point, especially on large 2d tactile surfaces. Alternatively, the gesture approach does not require the user to move his/her hands away from the tactile surface. However, a gesture-based system does have two significant flaws. Firstly, it is more difficult to detect a specific gesture than to detect if a physical button was pressed or not. Even with the advance in deep learning gesture recognition methods, pressing physical buttons is more reliable. Gestures recognition also has the Midas touch effect problem, where the user performs the gesture unintentionally while exploring the tactile surface. In short, to enable gesture recognition, the gesture needs to be distinguished enough not to trigger actions and simple unintentionally to be recognized with high accuracy. Figure 7 exemplifies the two aforementioned approaches for executing an element/object on the 2d tactile surface.

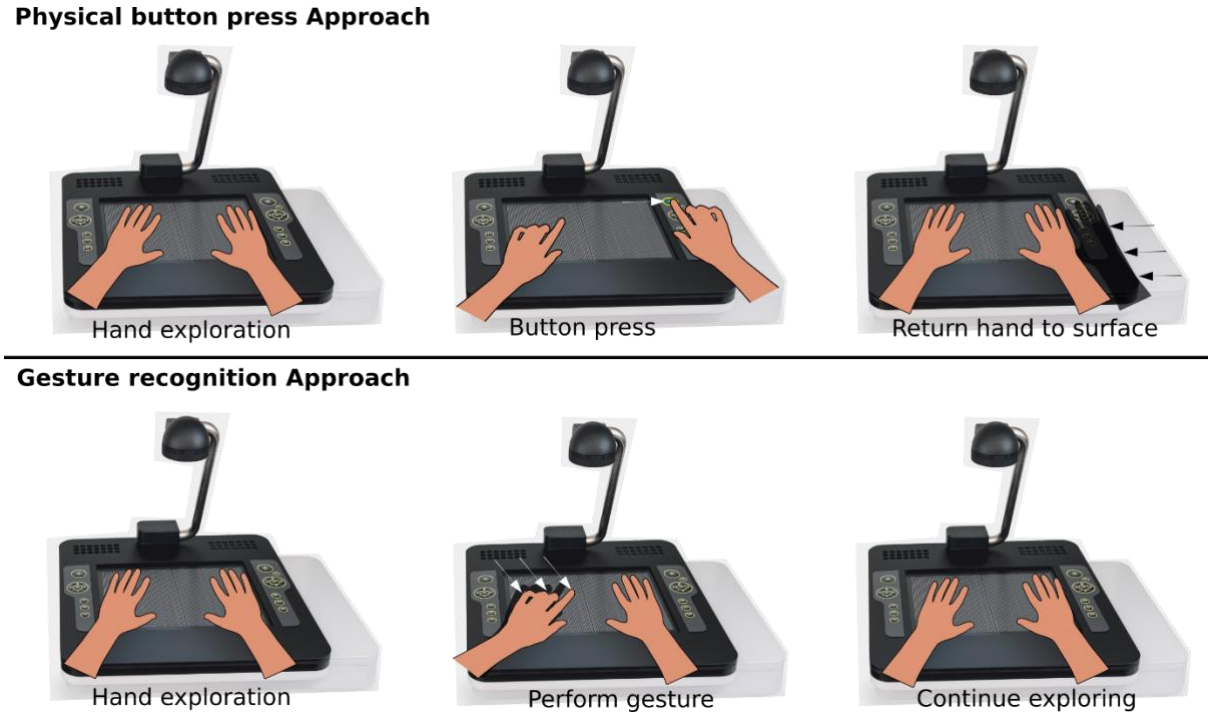


Figure 7 - Execution operation UI alternatives in 2DRT devices, (Tactonom process).

Commands can be used to execute more specific actions such as filtering or searching for a particular file. Another aspect of input control is the use of keyboard combination commands, also used in GUI. The user has a set of commands at his/her disposal that he/she can use in the current state or application of the Tactonom. Figure 8 presents the available commands of the current focus application (file explorer in this case).

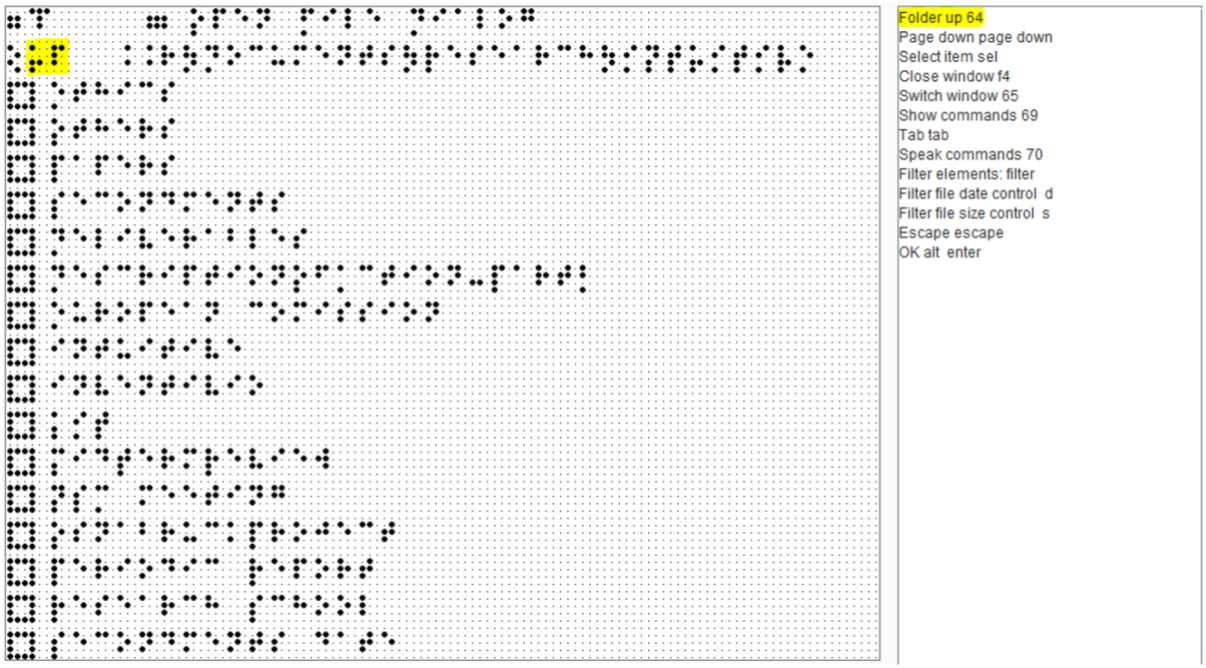


Figure 8 - List of commands for a file-explorer application of the Tactonom. Some use key combinations and other use single-key combinations.

These commands are intended to be executed by using the user's laptop keyboard or an external keyboard. Three different alternatives to execute these commands can be used on the Tactonom. Single-key commands, multi-key commands and gesture-based commands. The last one we have talked about the advantages and disadvantages of using gesture-based commands. On this case, we would have to add more and more gestures to be recognizable but at what point would these still be reliable and not unintentionally triggered by the user? The more gesture and different commands are implemented, easier it will be for the user to accidentally trigger these while exploring 2d information. The Tactonom supports single-key commands, where the user simply has to press a specific key to perform an action/operation in the current application. Multi-key commands require the user to press two or more buttons in a specific sequence, which is challenge for some blind persons and usually not the preferable way in other assistive technologies. Single-key commands seem the more indicate approach since they are easier to perform for blind persons. However, there are interesting situations where these commands cannot be used, such is in a text editor application. In the text editor these commands are not possible since the user needs to be able to write characters in the text document (figure 9). The alternative here is to only use ctrl-based commands (multi-key).

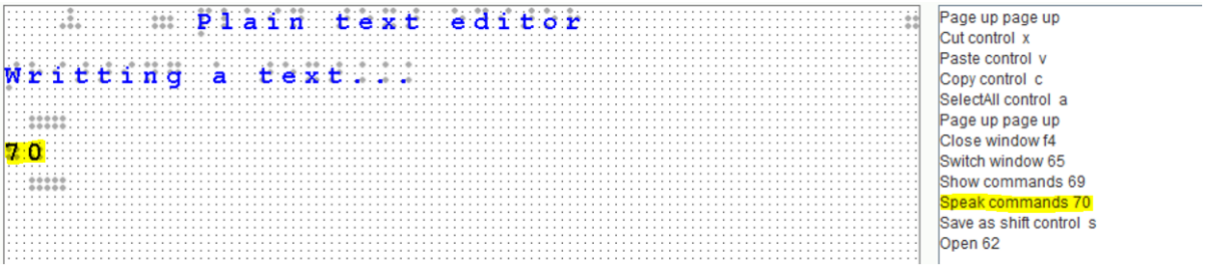


Figure 9 - Tactonom Text editor - commands presented on the right side of the image. The user fails to execute the command:" Speak commands" since the current focus is on a text box, which receives all text information from input keyboard.

Beyond the three presented alternatives, physical buttons can execute specific commands related to the application level or the system level, such as volume adjusting and text-to-speech engine selection. The disadvantage of this approach is the limitation on the number of available physical buttons. The functionality and use of physical buttons should be concise throughout the system and the set of available applications. One interesting strategy would be to use the left set of physical buttons for application or context related, meaning that the function of these changes concerning the currently active application, and use the right set for system-oriented functions.

Information representation

Information representation is the user interface aspect that includes the strategies to represent 2d and graphical information on the tactile surface and the audio modality, from text (static) representation to widgets (dynamic).

One aspect that has received attention by other researchers is tactile symbols (textures) and text representation (braille), which are not exclusive to refreshable audio-tactile displays. Single-line refreshable tactile displays represent braille cells and tactile graphics mainly focus on tactile textures. But most of these issues have not been examined in the context of two-dimensional refreshable tactile devices. For text information, **braille** is usually used by blind persons to represent text in a tactile format.

The Tactonom user interface does also use braille to represent text information. Using the **file explorer application** for demonstration purposes, braille is used in the title of the current application, and to represent the current folder, files and subdirectories. Several braille alphabets can be used depending on the internationality or preferable language of the user. Figure 10 represents the Tactonom' file explorer representation in braille text.

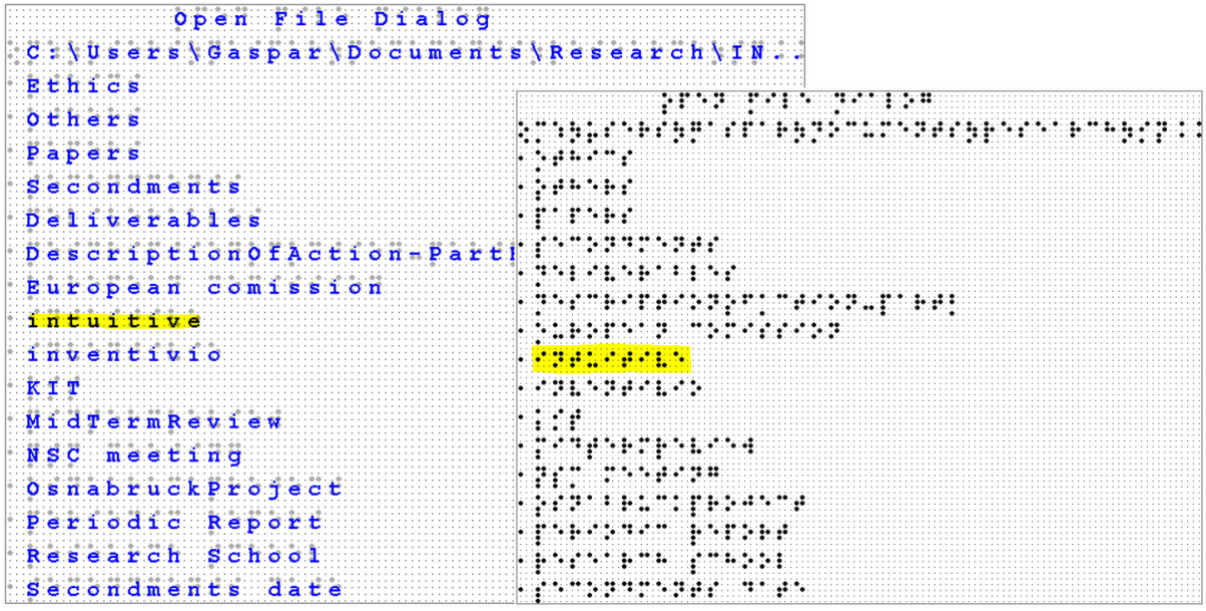


Figure 10 - Braille representation of text elements in the Tactonom' File Explorer application.

Beyond text elements, **tactile textures** and **symbols** can be used to indicate certain characteristics of the current folder state. For example, a down-arrow and up-arrow can give information regarding the current "page" of the files list when the number of files exceeds the display size (figure 11). These textures can also have an interactive element associated with them, such as a hyperlink button, making then a widget element (these will be approached later on this chapter).

Figure 11 - Implementation of tactile textures in the Tactonom' File Explorer

Although braille is widely used in assistive technology for blind people, it has its disadvantages. Representing a large amount of text in braille on a limited display size is challenging since braille does not scale down and uses a lot of space. With the current display size, it is possible to present 17 text lines (including the application title line) by using braille. Instead of presenting the information in text, textures and tactile symbols can be used to represent the files themselves, where two specific textures represent folders and files (figure 12).

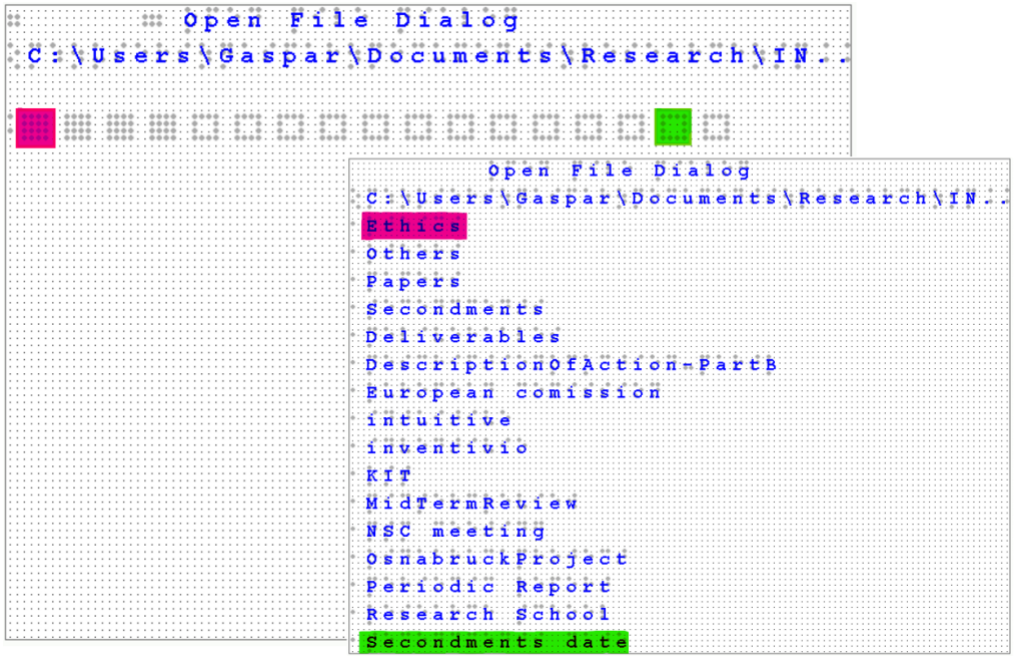


Figure 12 - Tactonom File Explorer - Representation of files with textures (left) and with braille lines (right)

The alternative of pressing files as textures (figure 12) has the advantage of being able to present more files at once in the tactile display. As the Tactonom currently has a refreshment rate of 10 seconds, it is beneficial for the Tactonom to use approaches that do not require as many page updates as other approaches. With the current display size, the texture-file approach is able to present 320 files (20 columns * 16 rows) or subdirectories at once, while the braille listing approach is only able to present up to 16 files (one per row) on the same page. In order for the user to have access to the text information, which in this case is the file name, audio modality can be used. The user can access the element information by using his/her fingertip and a text-to-speech engine can be used to play the file or directory name.

Another version of this file explorer user interface is to represent the files list with more detailed information, such as supporting files ordering, file size, file modification date and type of file or directory (represented in figure 13).

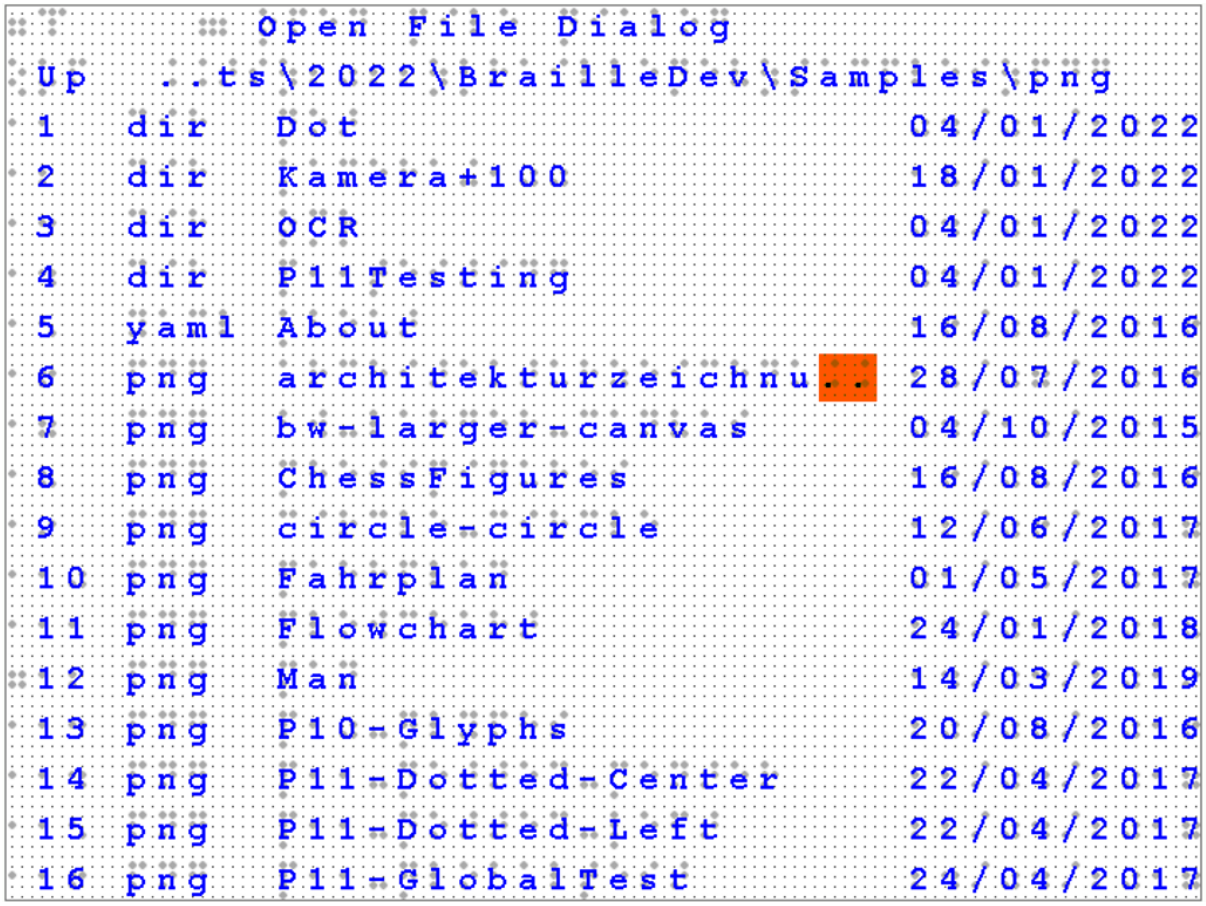


Figure 13 - Tactonom file explorer with braille text file list representation with extra information detail.

This version is able to present more in-line information to each file-folder and a numeration of the files which can be used as a guideline for blind persons to select of modify a specific file. The big drawback of this technique is the **overload of information** on the screen. For filenames and folder with relatively small size (< 10 letters) there is no problem, but for larger filenames, it is necessary to reduce one of the file properties on the tactile surface to fit all the information. Line 6 of figure 13 represents exactly this problem, where the file name representation had to be reduced to a limit number of characters.

If the file explorer view also presents the file size (which is normal in GUI), then the problem of overload of information is even more perceptible. Figure 14 represents this approach of representing the file explorer with file order, type, name, size, and date of modification.

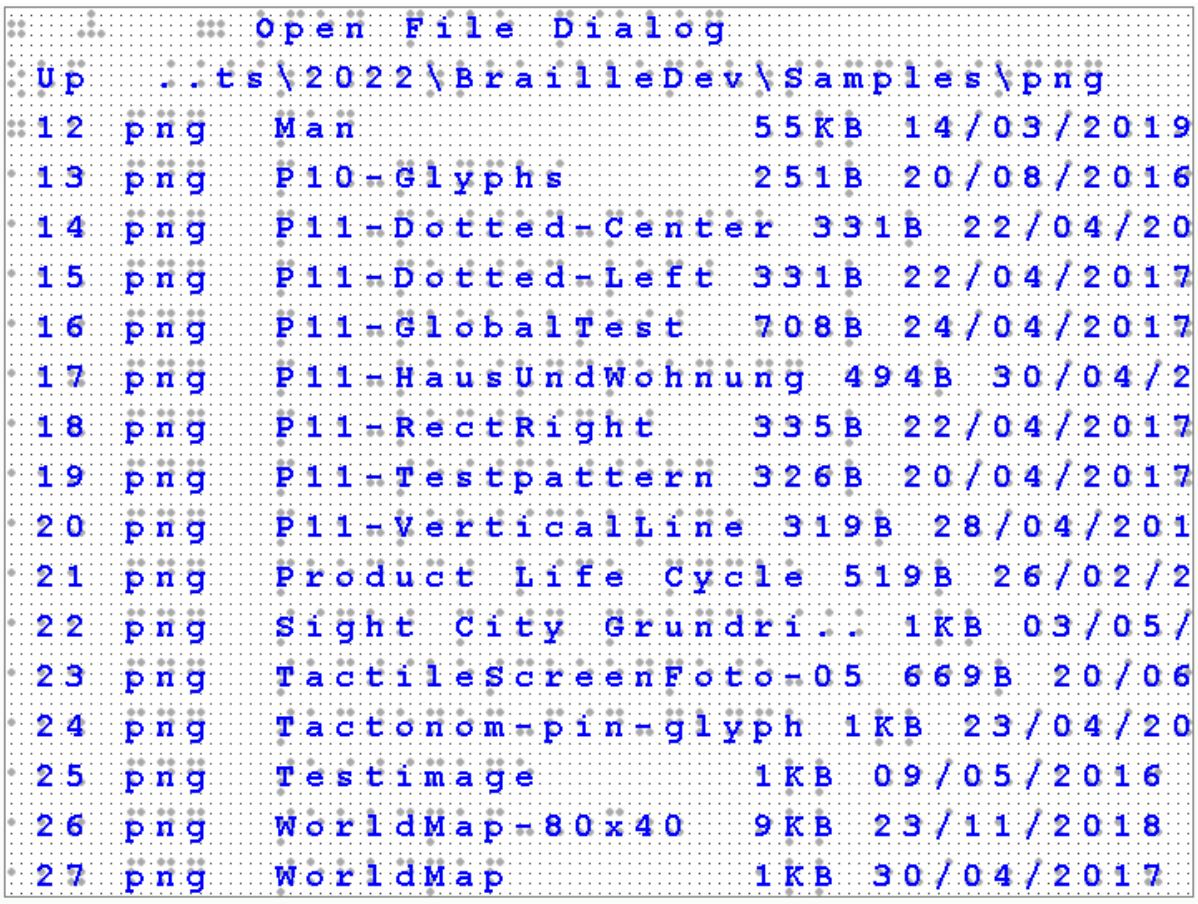


Figure 14 - Overload of information in braille representation of file name and details on the Tactonom' file explorer

If it is intention of the user to search for a specific file or type of files, a **filter operation** can be used to filter for example the filename of a specific file or directory (figure 15). This can reduce the number of information presented on the tactile screen.

Figure 15 - Filter operation with words "P1" on the Tactonom's file explorer

Widgets have a great importance for building any kind of user interface. They encapsulate and standardize user interface approaches and patterns for a particular class of devices. Buttons can be set up on the tactile surface to navigate inside file explorer, by using the braille text of filenames and directories as tactile representation of these buttons (figure 16).

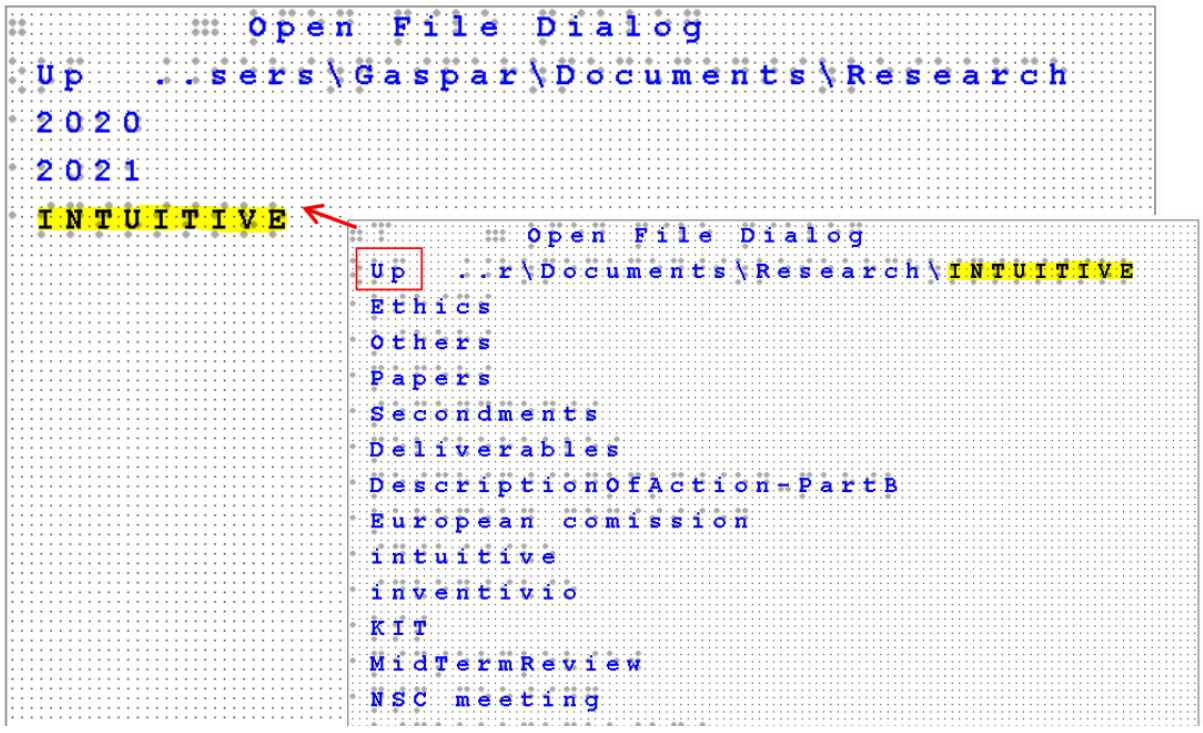


Figure 16 - Using braille elements as widgets buttons to navigate through directories in the file explorer

The major drawback of this approach is that these braille buttons do not have a representation difference from braille text, leaving to the user the question if the current braille element is interactive (a widget) or just text for reading. Audio information with fingertip detection can be used to indicate if the current braille

element is a braille text or a dynamic widget, by playing a background sound when the fingertip is on top of a braille button (widget).

A list of **checkboxes** can also be implemented in the file explorer, which would indicate if a file were selected or not to perform a standard operations, (copy, delete, rename). In the example where textures are used as files, these could be also selectable for these operations, making these textures dynamic widgets.

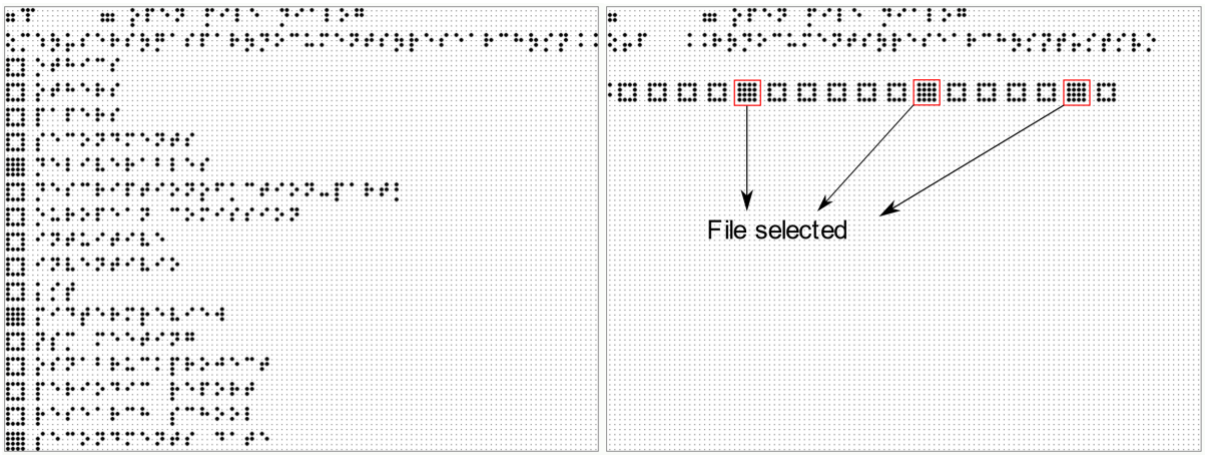


Figure 17 - Usage of checkboxes in braille-list and texture file explorer representations

Besides the best ways to address the size and resolution limitations of tactile devices, a key open question is how to use audio feedback to present more detailed information easing the overload on the tactile surface. Buttons can have audio feedback as well as checkboxes once selected and unselected.

Figure 18 - Implementation of audio feedback when user interacts with tactile widgets (file explorer)

Another interesting widget element is the scrollbar (also used in GUI). Scroll panning was already implemented in other technologies such as the BrailleDis devices from the HyperBraille project, however their implementation in the Tactonom might not be beneficial to increase user interaction (implementation

in figure 19). As it was already concluded before, braille text lists are not ideal for the Tactonom, since this as a bigger refreshment rate of 10 seconds. The scrollbars in GUI are useful for the user to scroll through a big list of elements, while maintaining feedback on what is on that list. They are also used to tell of depth the user is in the list. These are very dynamic widgets, meaning that are only usable with a faster refreshment rate. An easier alternative to these would be to implement a button that would just scroll down page to page, since the user will have to wait 10 seconds for the page to update.

Figure 19 - Implementation of a scrollbar in the Tactonom's file explorer

The **plain text editor** is an application plugin that enables users to write and edit text-based documents. This was implemented in the Tactonom device, where the user can open empty documents and write using a connected keyboard.

Figure 20 – Plain text editor application with an empty text file. Title text symbol identifier in red. User document position identifier in green.

A small rectangle constituted with 6 markers (2 height by 3 width) represents the title of the current application or menu stage. In order to perceive the current user position (user focus) in the text file, a tactile symbol is used. The Tactonom implementation uses a distinctive rectangle symbol with 10 markers (2 height by 5 width) that is placed on the same x user position and next bottom text line (y position). The rectangle two central points out of 10 (vertical center line of the rectangle) represents the exact user position in the

document (figure 21). If the document is empty, this identifier is only represented in a 2x2 square symbol as it can be seen in figure 20.

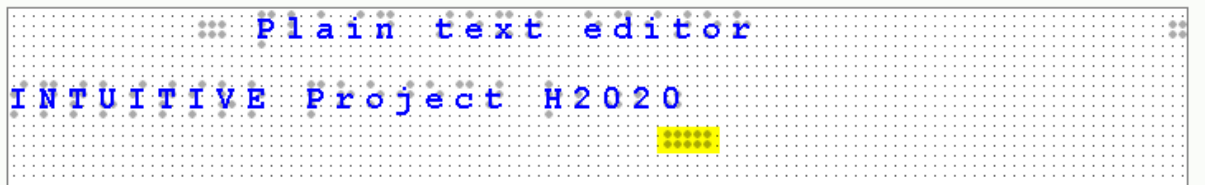


Figure 21 – Plain text editor application with user position in the text document identified in yellow.

In order to increase the perceptibility of the current user position in the document, an additional rectangle symbol is used. This rectangle is placed on the same x user position on the top text line (y position). The document position is now between the two rectangle symbols (figure 22).

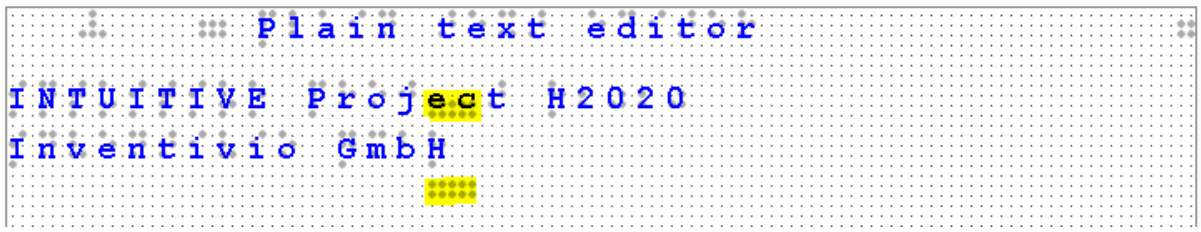


Figure 22 - Plain text editor application with two user document position markers (in yellow).

The major drawback is that in order to represent the current user position, the two top and bottom braille characters become concealed as it can be seen in figure 22. This is necessary in order for the user to perceive and locate the current focus position.

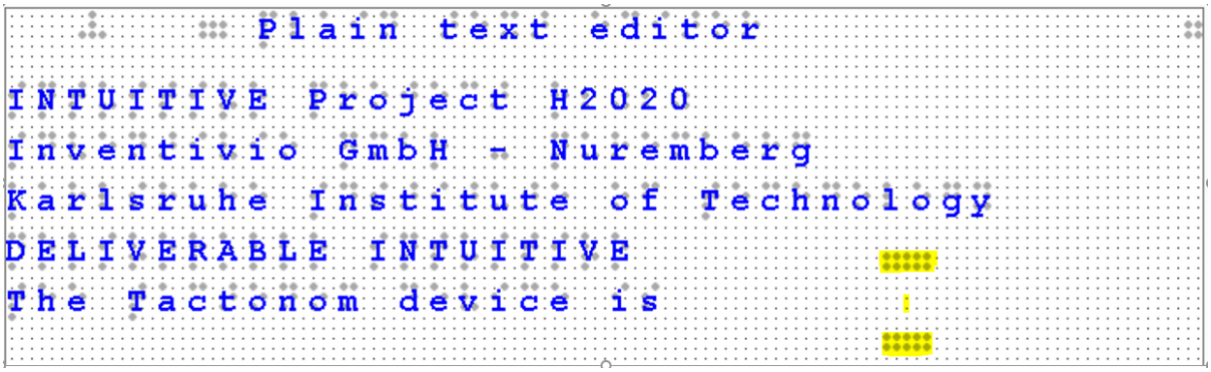


Figure 23 - Text editor application, user document position is independent on text, and can be represented in blank spaces (user document position in yellow).

The user document position is highlighted in figure 23. The user can move the document position to a non text-element (blank spaces) using the keyboard key arrows.

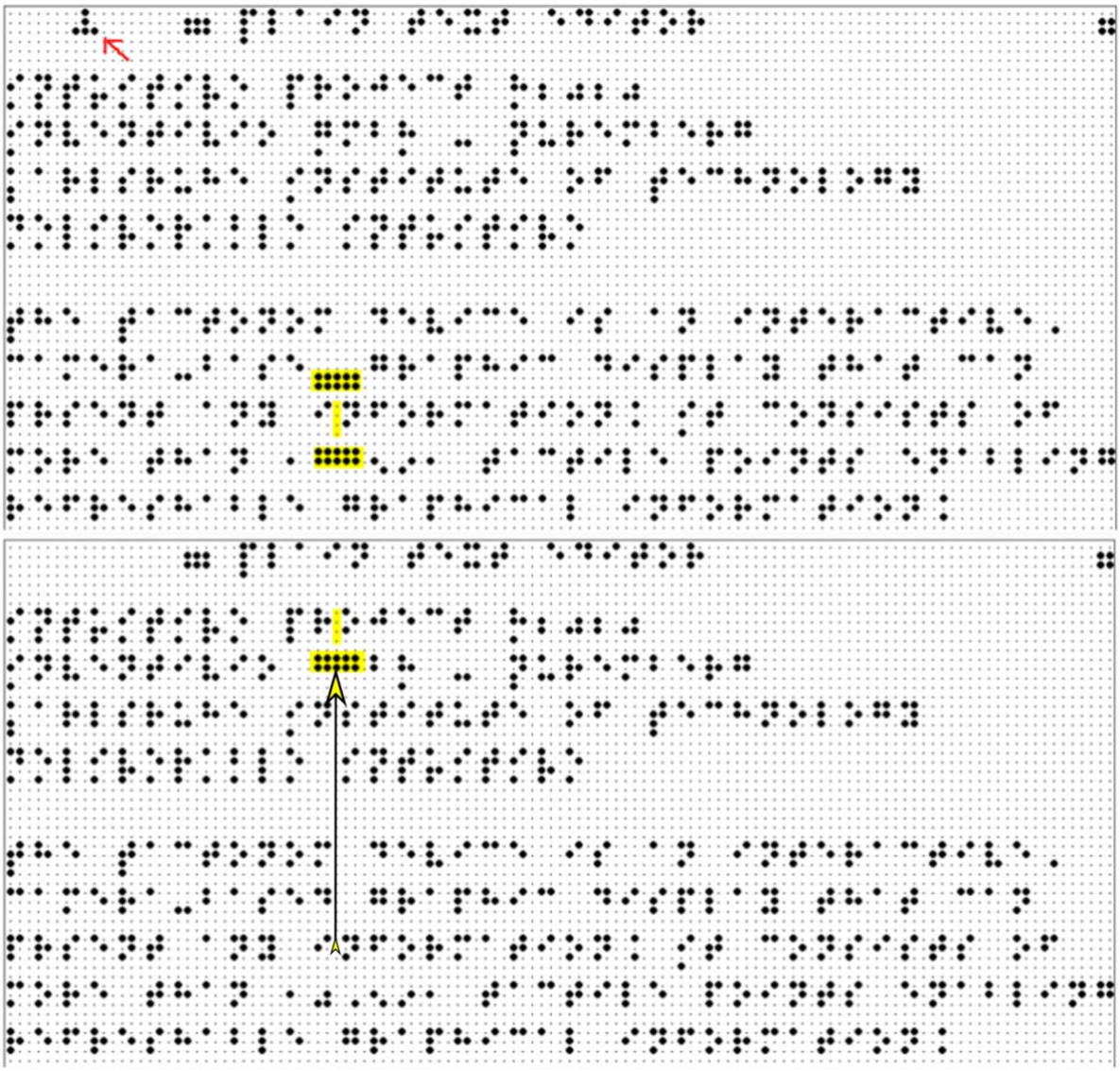


Figure 24 - Function page up in the plain text editor application. Once the user activates the page up button, the focus user point move to the top line and the page up button is removed from th

The plain text application page-up functionality replaces the current user document position to the first text document row, enabling the user to move faster to the first line of the text (figure 24).

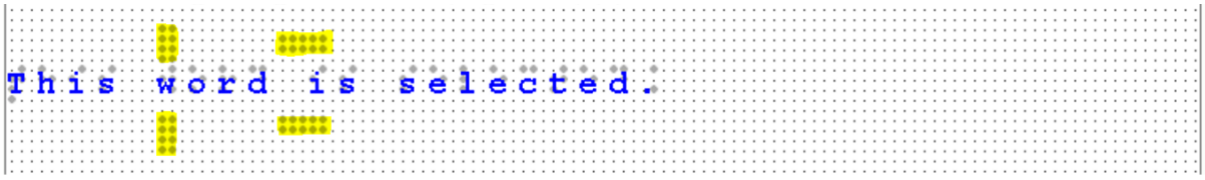


Figure 25 - Text selection of "word" in text editor application (markers in yellow).

Allowing text selection, enables other document text operations, such as copy, past and cut. In order to indicate which part of the text is selected we can use once again tactile rectangle symbols that are placed over the bottom and top text lines of the selected line(s). The tactile symbols for the selection are vertical rectangles with 8 points (4 height by 2 width). These vertical rectangles delineate the start of the selection, and the horizontal rectangles delineate the end of the selection. For relatively long text, it might be difficult to perceive the current user position with only these small symbols. Bigger symbols can be used but would hidden more nearby text which is still context for the user to locate specific sentences and words.

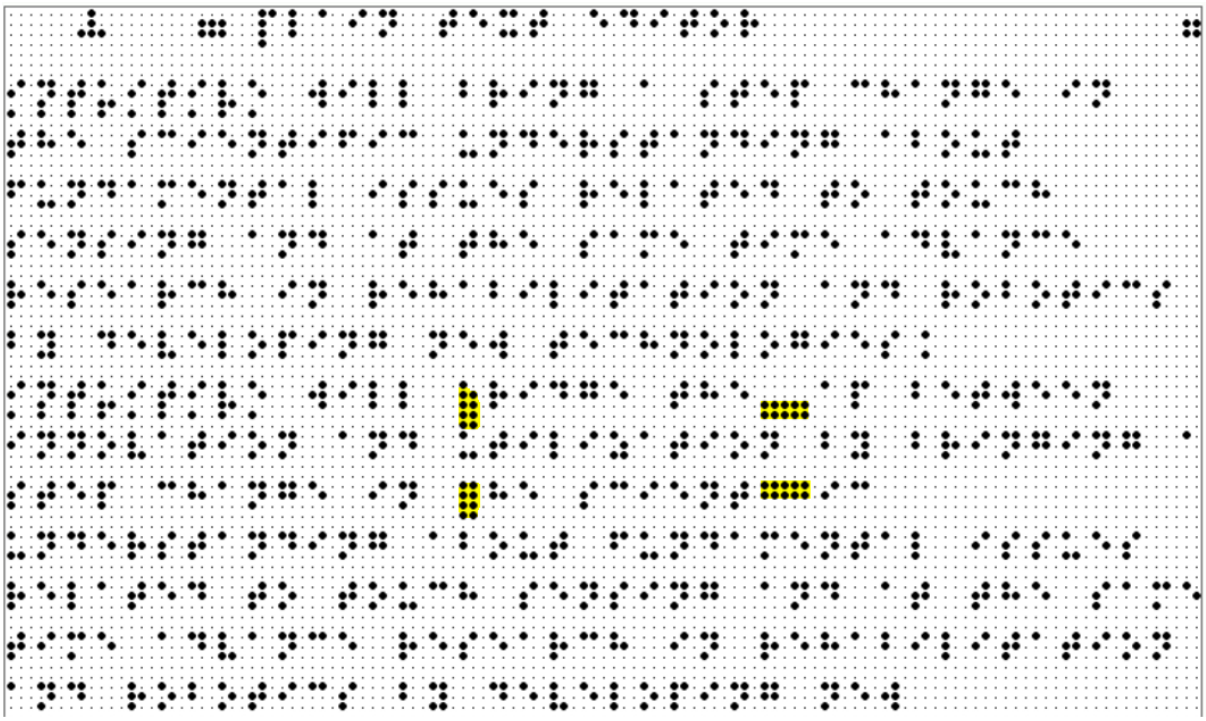


Figure 26 - Word selection in text editor application with long text.

Several **widgets** were implemented in the Tactonom software, including labels, checkboxes, buttons, textboxes, text areas, table representations and listboxes. The Tactonom includes a widget library application containing an example of all the previous listed widgets (figure 27).

Although labels are a very basic widget in GUI, in 2DRT user interfaces, these represent text in the form of braille, as it was mentioned previously. As different countries use different braille alphabets, the label widget in the Tactonom needs does support different braille languages. **Labels** use a conversion policy, which maps standard alphabet letters to a specific braille alphabet letter, by implementing a dictionary. As in GUI, labels are not dynamic and for reading purposes. These are used for presenting information and in titles on the Tactonom device (figure 28).

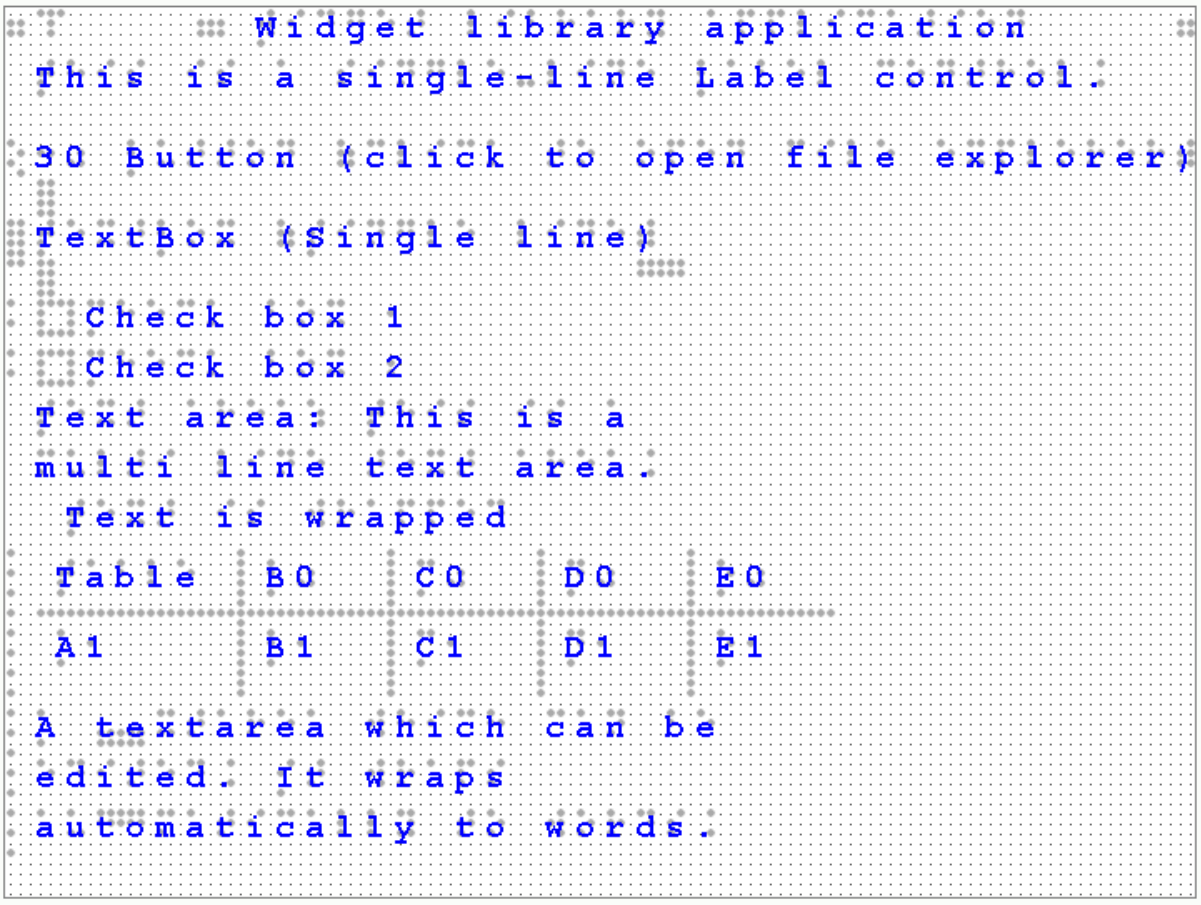


Figure 27 - Widgets library application of the Tactonom

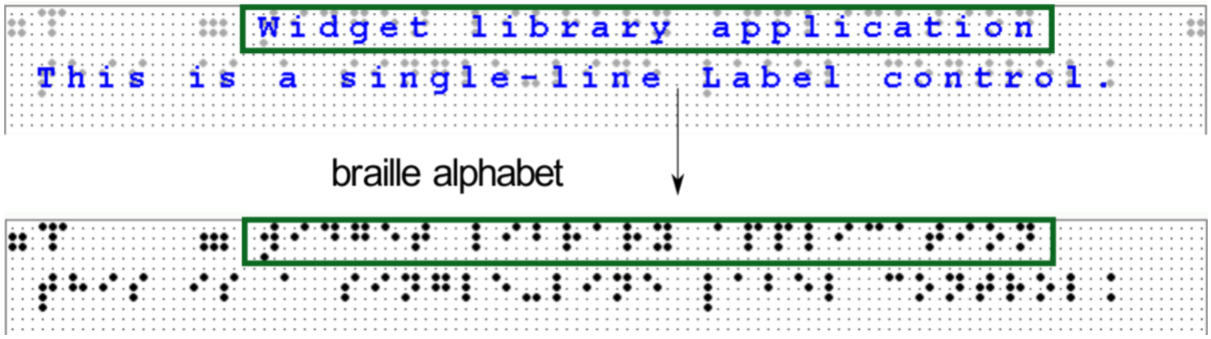


Figure 28 - Label widget (Tactonom)

A more dynamic widget is **textboxes**, consisting of a single line frame where the user can dial characters to be displayed on it. As labels, text boxes do not trigger any event if the user slides his hand on top of these. However, these are still dynamic since their tactile representation can be modified by user input. A focus is also draw on the tactile surface if the textbox is focused, as well the user current text position (figure 29).

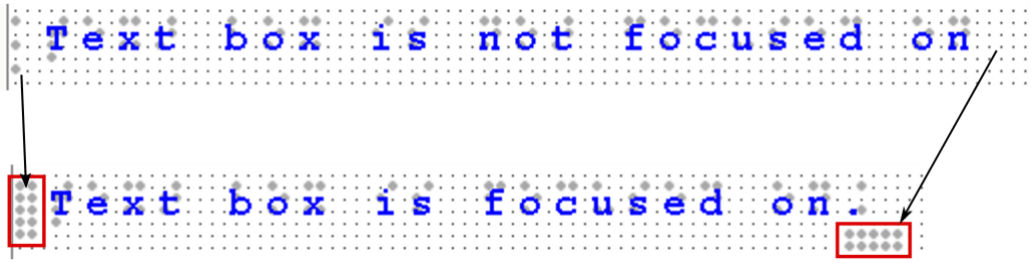


Figure 29 - Textbox widget (Tactonom)

For multi-line textboxes, the user can use **text areas** which can be edited as textboxes but are not limited to one line. Beyond this difference, the text wraps automatically to words and the user can scroll down on the text area to uncover text that was not represented on the tactile surface. It is possible to create a text are from 1 line up to 16 lines. In figure 30, we have a text area with 3 lines, but this is not limited to 3 lines of text information only, if the user continues to write more text, the text area will adjust the view by pushing up all of the other text lines. It is interesting since it can store more information than it is presenting in the tactile surface.

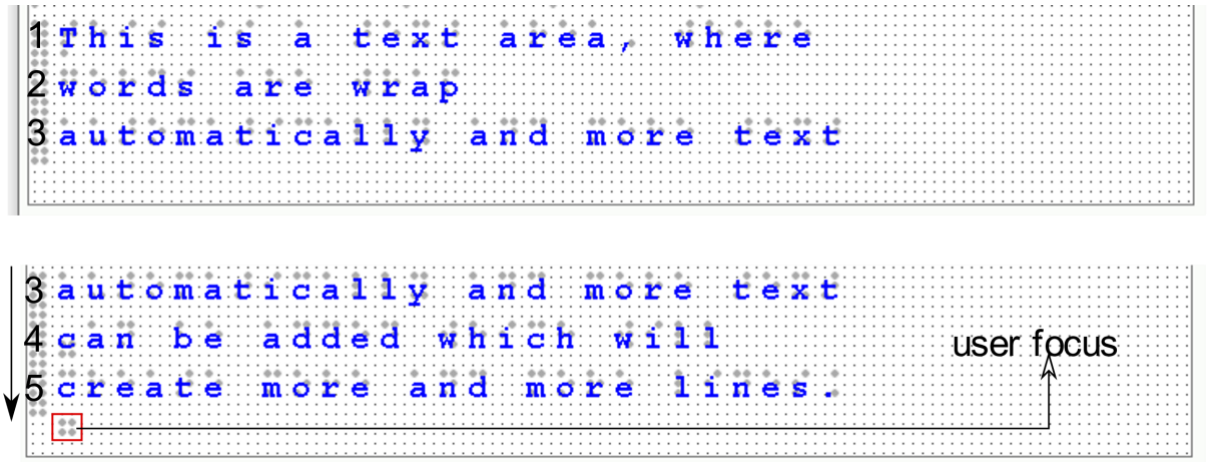


Figure 30 – Text Area implementation in Tactonom

Checkboxes are dynamic widgets which were already discussed in this document. These can come together with text label elements and include two different states, selected and unselected. A 4 x 4 texture square is used to represent the checkboxes. If the four interior taxels of the square are not raised up, that means that the checkbox is unselect and the opposite case means that it is selected (figure 31). Alternatively, audio clues can be used to indicate if the checkbox is select or not, which would work in real-time independently on the 2DRT device refreshment rate. The approach of moving the interior pins of the square is not ideal for Tactonom since it would take 10 seconds for those pins to be raised up.

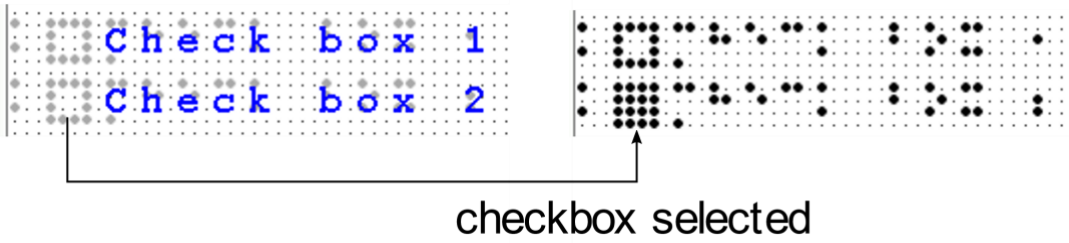


Figure 31 - Checkbox widget (Tactonom)

Another interesting widget implementation in 2DRT user interfaces are listboxes. These present a set of elements, one by line, represented by a tactile symbol and braille text. Each element is selectable and once one is selected, the tactile symbol changes from 1 taxel to a 2x2 square symbol, to indicate that the element was selected (figure 32). Beyond that, it is also interesting to present the last modified element, which is useful for situations when the user does take his/her hands out of the tactile surface for a moment and than needs to find the last focus point (which is the last item selected).

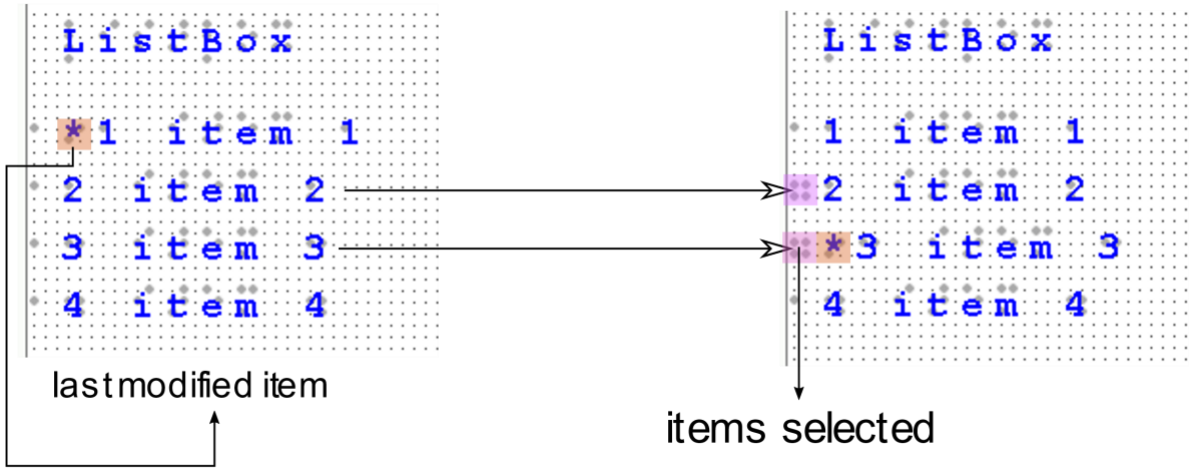


Figure 32 - Listboxes widget implementation on the Tactonom

Layout

Another important aspect of user interfaces, especially in two-dimensional refreshable audio-tactile devices is the arrangement of the information elements (discussed in the previous chapter). This arrangement has a crucial importance in the interaction and overview of the elements for blind people. For instance, in the file explorer, files can be represented as interactive checkboxes, which allows for more elements to be on screen at the same time comparing with the braille text listening. However, it is difficult to distinguish which elements are files and which are subdirectories. A solution is to organize the arrangement in a way that subfolder and files are separated from each other. This is a concrete example on how layout influences user interaction.

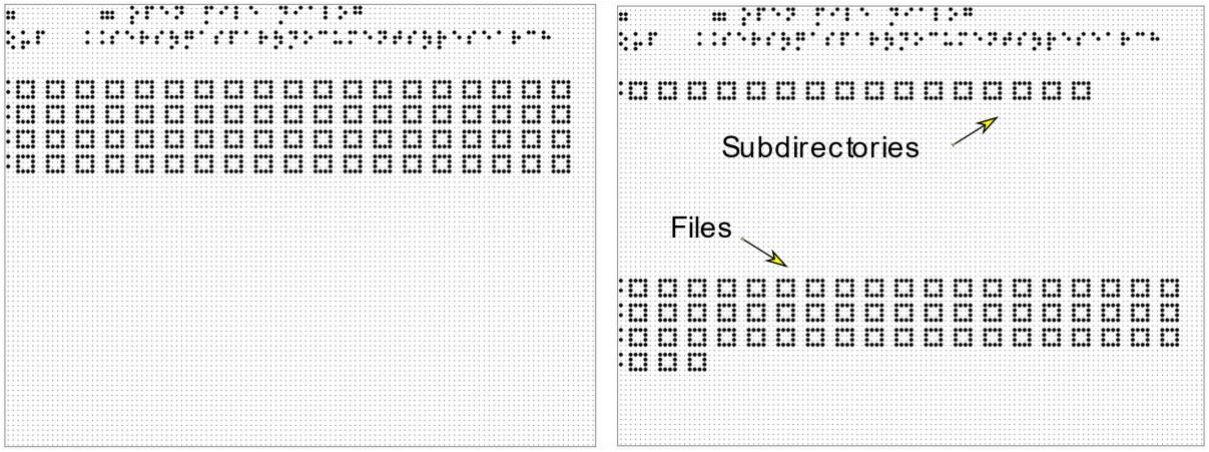


Figure 33 – Layout approach to distinguish files from subdirectories in texture representation on file explorer.

This last layout is very interesting because beyond representing more files, it gives a fast overview of the folders content be just panning the hand over all the checkboxes. With a list, the overview is different since the user would have to go through all of the pages to know how many elements there are while reading braille to know which ones are folders and files.

On this same layout, since there is a lot of space left, the structure of the file system can be represented in two-lines in top area of the tactile display. Or it can be represented in a braille list instead in the form of a **tree list**, (figure 34). The representation in two-lines leaves more space available to present more files at once. However, the representation in a braille tree list can make use of audio buttons that would allow the user to navigate within the Tactonom file system by pressing the respective braille text leaf element of the braille tree.

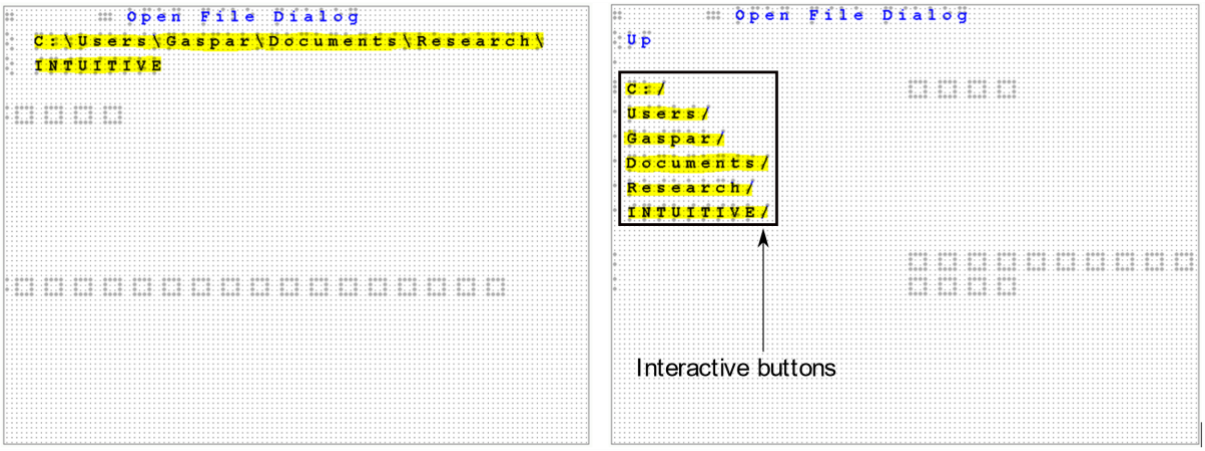


Figure 34 - Two different layouts for representing the absolute user-focus path in the file explorer application.

Going back to the original layout with a list of files with name, more information per line could be represented, such as the file type, file size and file date. However, for relative long file names, representing all of this information at once might not be the best approach since it would limit the max number of visible characters of a filename.

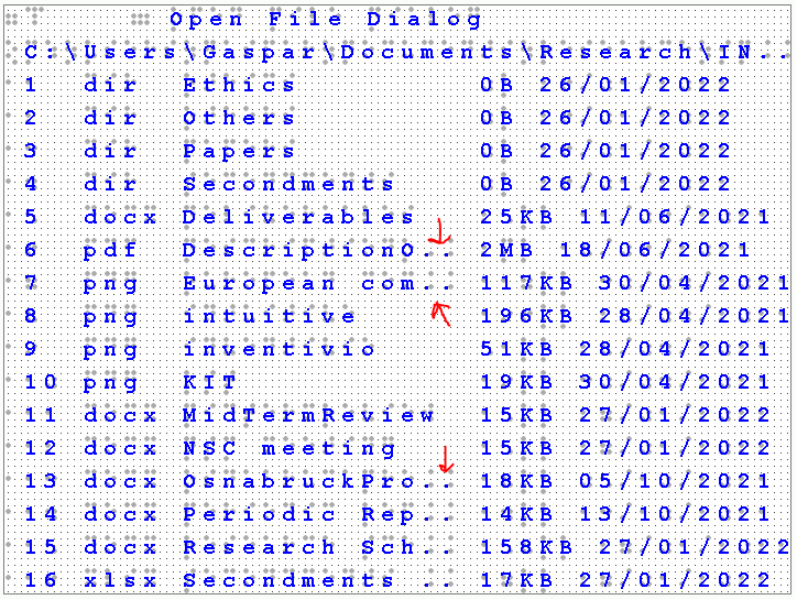
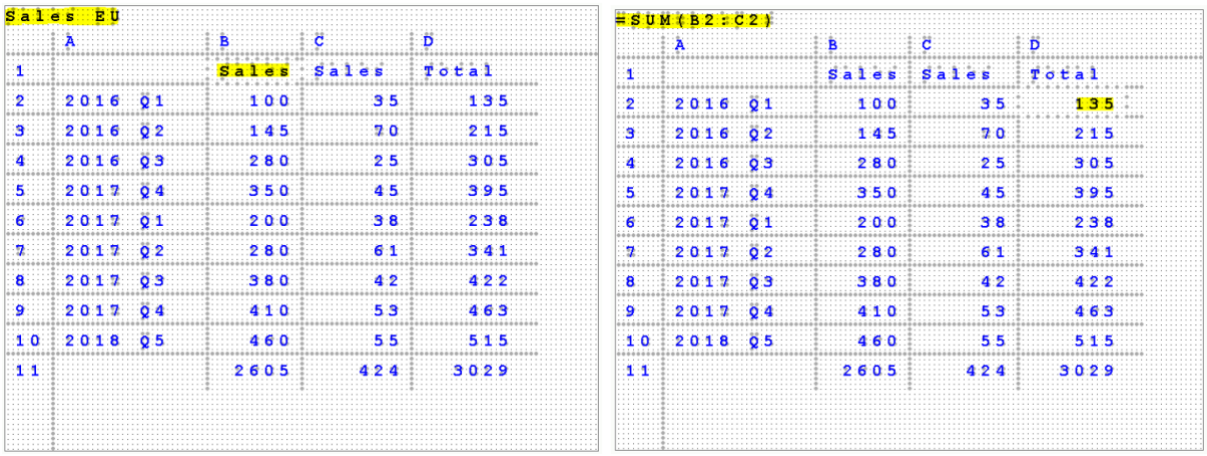


Figure 35 - Tactonom file explorer with file details layout information.

Another interesting applicability of layout aspects of the user interface in this technology is in excel sheet's view type. The default excel Tactonom application layout uses lines of width = 1 to delineate the excel sheet cells. Each cell compromises a single line of braille text that is restricted to a few numbers of braille characters. Since braille is not down scalable, we are limit to these small number of texts per cell if we use braille. One user interface design strategy is to just present the first characters of a word in a cell preview and use one entire line (focus line) to present the braille text information inside the specific cell. The focus line can also be used to represent built-in formulas information of the cell, such as sums and other arithmetic

operations. The downside of presenting both the formulas implementations and the cell content on a "focus" line is that if the number of the cell content gets too larger, the user will not be able to access it via braille text. This occurs because for cells that implement arithmetic operations, the "focus" line prioritizes the formula instead of presenting the cell content (figure 36).



	A	B	C	D
1		Sales	Sales	Total
2	2016 Q1	100	35	135
3	2016 Q2	145	70	215
4	2016 Q3	280	25	305
5	2017 Q4	350	45	395
6	2017 Q1	200	38	238
7	2017 Q2	280	61	341
8	2017 Q3	380	42	422
9	2017 Q4	410	53	463
10	2018 Q5	460	55	515
11		2605	424	3029

Figure 36 - On left the cell content is present in the focus line (top-left in yellow). On the right, the cell content is not presented in the focus line. As this cell as an operation associated with it, this is the information that is represented in the focus line.

A completely different layout can be used for the excel sheet program. Instead of presenting the cell sheet in the previously explained format (dividing lines with braille text), this can be presented in a more minimalistic layout. This new representation does not use dividing lines, instead it uses two different tactile symbols to represent the cells content. The bigger symbols representing cells with content and single points to symbolize empty cells. This minimalistic layout view allows the user to have a better perception of the excel sheet document. Additionally, audio can be used to present the content of a specific cell, which is not limited to a limited number of characters as the previous layout view. This view can also present more element cells at once in one single page, reducing the necessity of refreshing the tactile surface when moving between cells(panning) in large excel tables (figure 37).

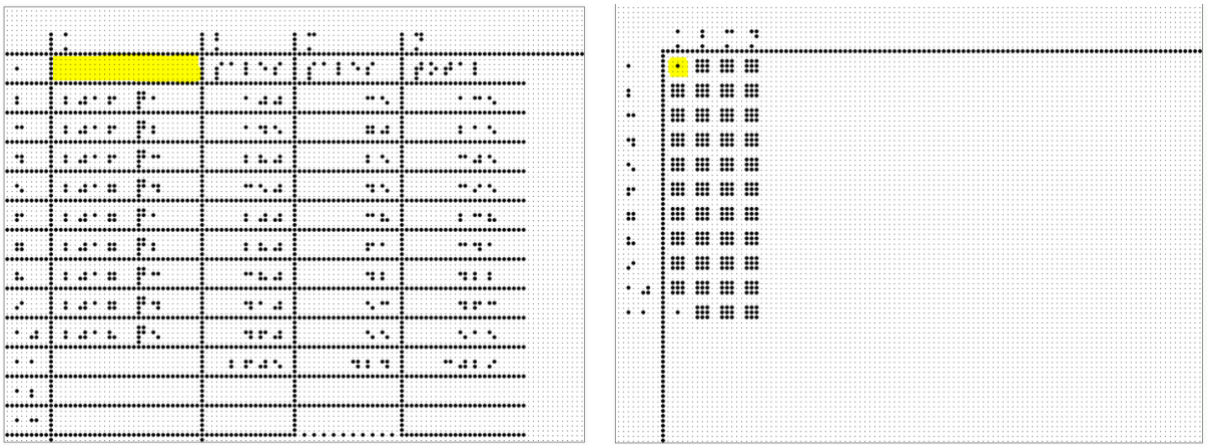


Figure 37 – Minimalistic excel table view, cells are represented as interactive textures containing audio information of content.

The highlight of a cell is actually an interesting aspect. On the more detailed view, the highlight changes the border cell lines from continuos to discontinuous lines. On this view, the tactile display is overloaded with braille text, and highting an element like this might not be as perceivable with just tactile feedback. On the other hand, the simplified view, sets disconinuous set of border points to highlight a specific cell, but as the screen is just presenting two very distinctive forms of information for each cell, it is easier for the user to notice and perceive this highlighted element (figure 38).

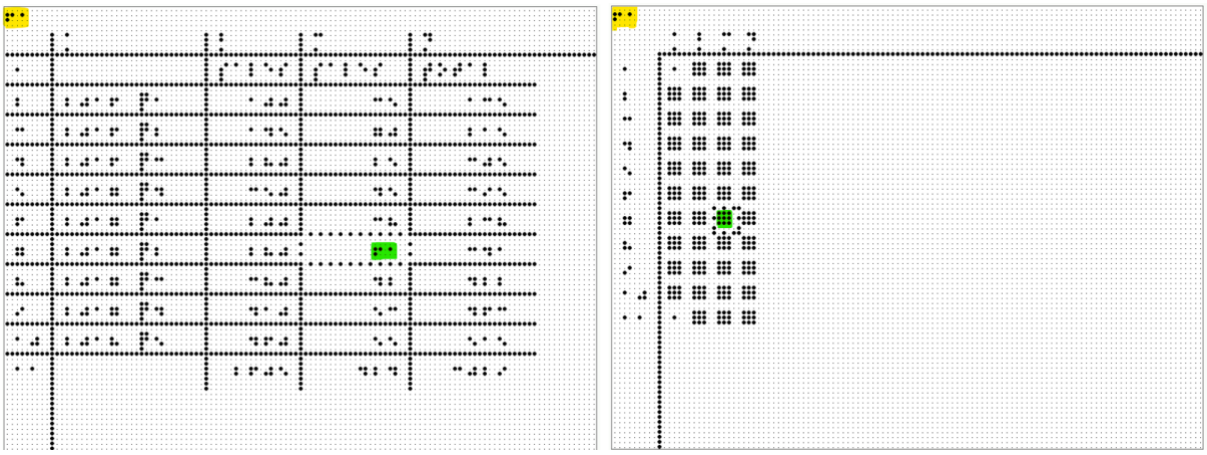


Figure 38 – Different layouts on how to highlight a cell in a table sheet application.

Dynamic control

Beyond static and widgets representations, the system can use dynamic user interfaces, where audio and tactile contents of the user interface change according to user input over a period over a period of time. An applicability of dynamic interfaces is to develop audio navigation interfaces that help the user locate a specific element by using audio clues that are dynamically interacting with the user's hand fingertip position. These are interesting since there is a challenge for blind persons to find elements in graphical documents such as floor plans, maps, or excel sheets.

Three different dynamic navigation algorithms are implemented in the Tactonom device, sonar-based, axis-based, and voice-based.

The sonar-navigation algorithm is based on submarine sound navigation. If a fingertip is detected, a brief (0.8 seconds long) background sound is played in a loop with constant frequency. The volume and pitch of this sound are negatively proportional to the distance between the user's fingertip and the element target in the tactile graphic. The closer the user's fingertip gets to the target, the higher the volume and pitch of the background sound will be, indicating to the user that he is getting closer to the target element.

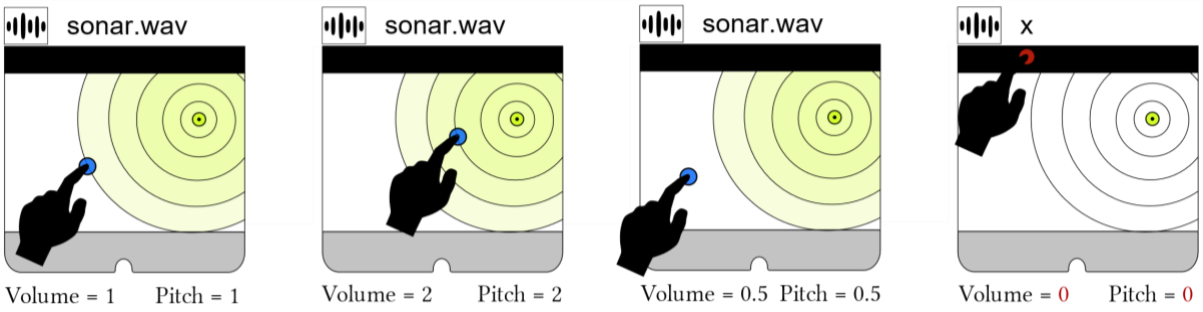


Figure 39 - sonar-navigation algorithm

The axis-navigation algorithm is based on the x and y-axis of the coordinate plane. In this method, there is no constant background sound being played. The user will only start to hear audio feedback once it reaches this target x or y-axis. If the person's fingertip is positioned either on the same x or same y coordinates as the target, a "trigger" sound is played. This trigger audio is only played once, followed by a short background sound (0.4 s) that will play in a loop. In this situation, the user has the information that he or she is in the same x or y coordinate as the target. As the user moves on the axis and gets closer to the target, the background sound gets louder and higher in pitch. The volume and pitch of the background sound are negatively proportional to the target distance. If the user's fingertip leaves the axis coordinate at some point, the background sound is silenced. It was also implemented a short "error" sound (0.4 s) when the user moves within the axis but in the opposite direction to the target to have faster responsive feedback that the user is moving his or her hand in the wrong direction.

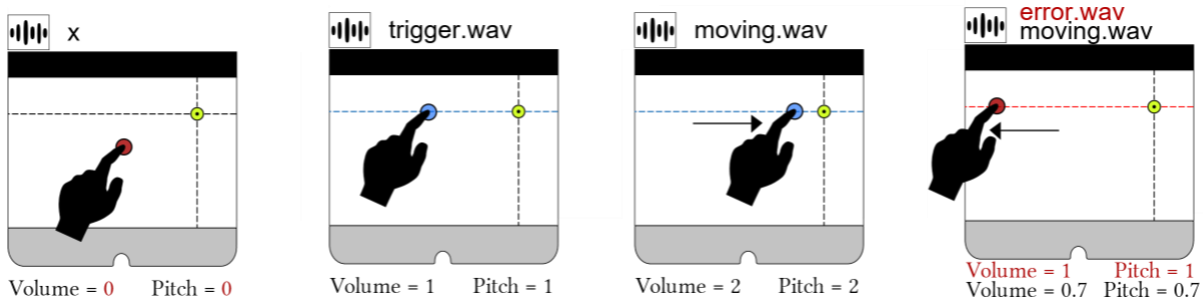


Figure 40 - axis-navigation algorithm

The voice-navigation algorithm is based on voice instructions commands that indicate in which direction the target is located proportionally to the user's fingertip position. Four different audio instructions are used to indicate four different directions, "go top," "go bottom," "go left," and "go right." These directional sounds are text-to-speech-based audios. To avoid situations where audio instructions are constantly changing, a priority direction move was defined. The volume of the voice commands is negatively proportional to the distance from the fngertip to the target.

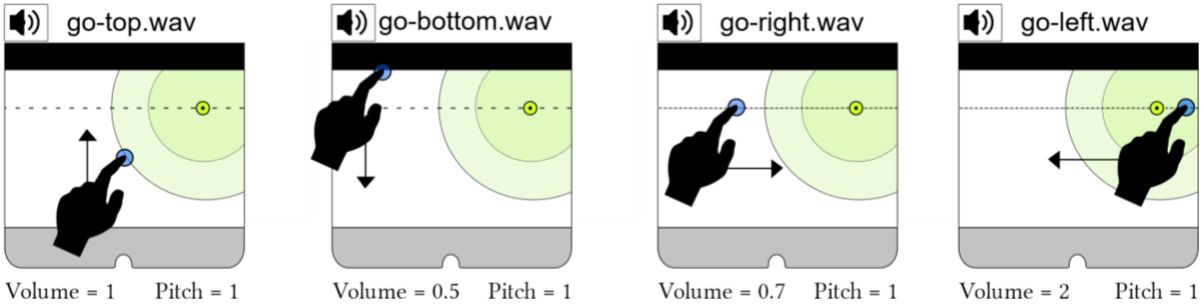


Figure 41 - voice-navigation algorithm

All of the algorithms distribute the played sounds in the stereo space, meaning that if the user plugs his/her earphones' panning effect is perceived.

The proposed navigation algorithms have different advantages, depending on the type of assistive technologies the user mostly use. Some use technologies that make use of sonar feedback, with varying pitch and volume, while others use technologies with speech-based interfaces. Depending on what blind persons usually use, a different algorithm is more appropriate.

An interesting aspect of dynamic control user interfaces is the maintenance and responsiveness of the interaction. An non-dynamic alternative to inform the user about an element location would be a speech description of the location, "element A is located in top-right corner on top of element B and on the left side of element C". However, this non-dynamic interaction does not react to user input, it only reacts once the user finds the target element. The dynamic interface reacts continuously to user input, enable rectifications of the user movement and actions.

Dialogue Architecture

Another aspect is the user interface dialogue architecture which comprises how the menu is structured, how files are presented, and how notification messages are provided. Pagination, nomenclatures, and status messages are included in this category.

The Menu interface of the Tactonom is presented in figure 42. These are the initial menu options that are presented once the Tactonom is turned on, entitled “Desktop”. The menu options are numbered to be easier to perceive and locate the desired option by sliding the fingertip in a vertical direction from top to bottom.

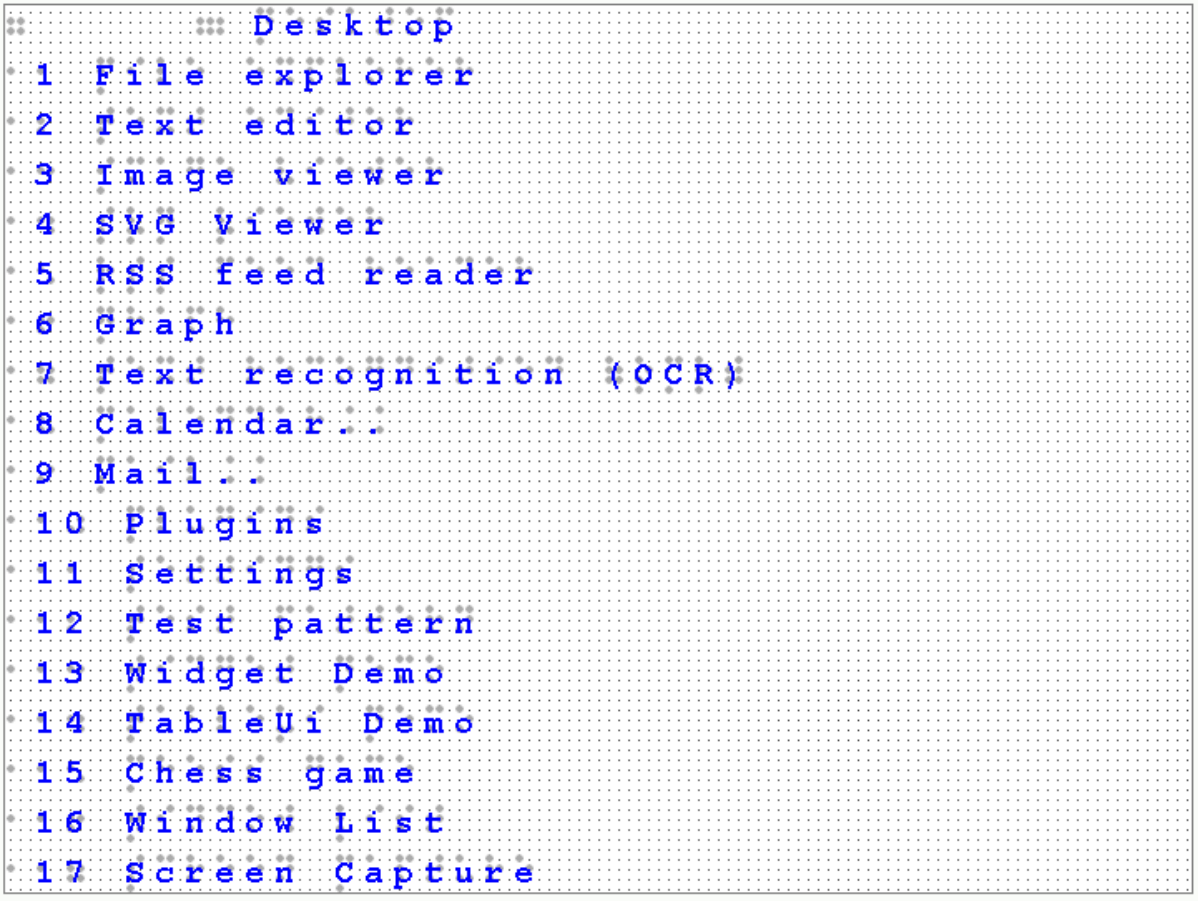


Figure 42 - Tactonom initial menu interface

The user only needs to dial the specific number or select with finger detection the menu option. After using the specific selected application (option), the user can return back to the main menu. The menu interface is able to inform the user regarding the last option used, by displaying a “*” symbol before the used option.

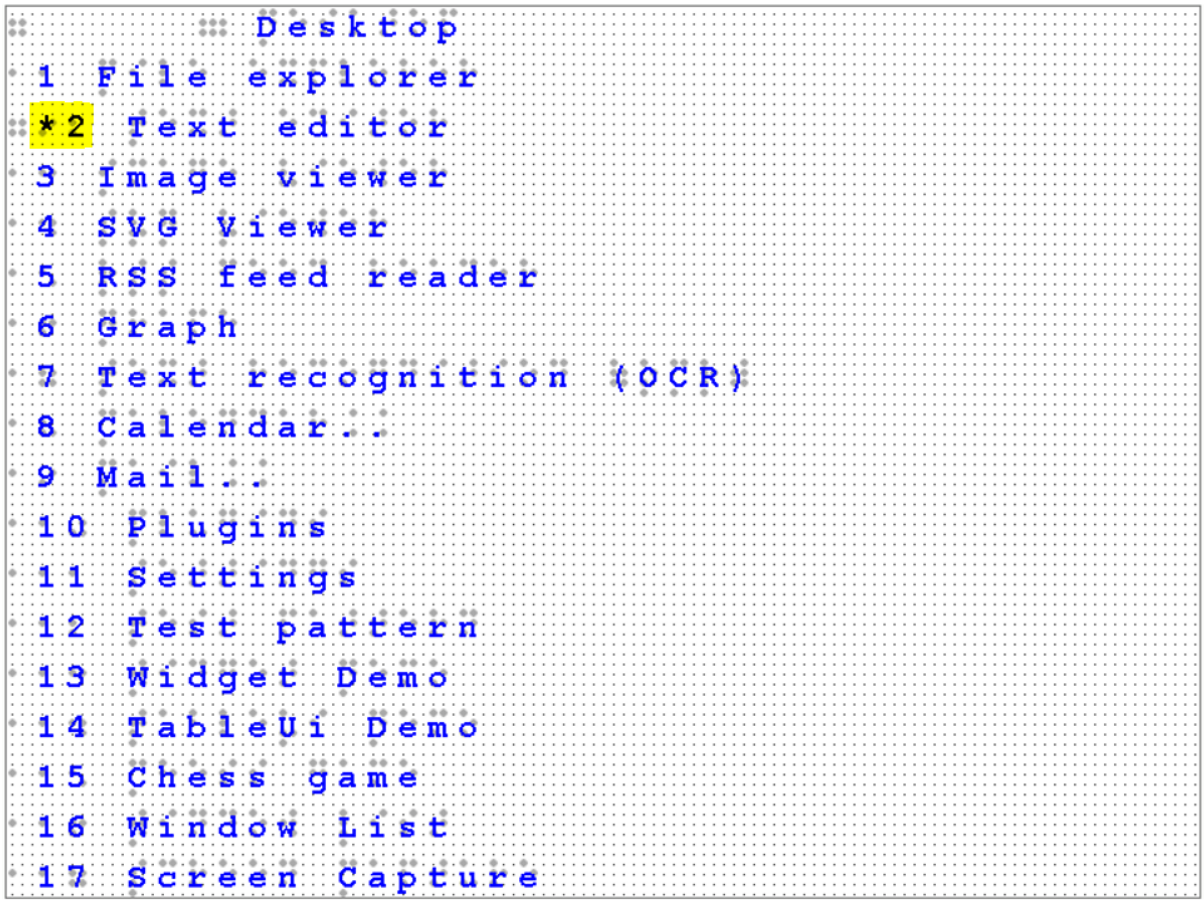


Figure 43 - Menu user interface with highlighted last used option.

It is also possible to change the current menu window to another desktop as it is available in conventional operating system. Having additional windows allows users to customize a menu desktop with specific applications options. Multiple windows support is also convenient for domain division of the available Tactonom applications.

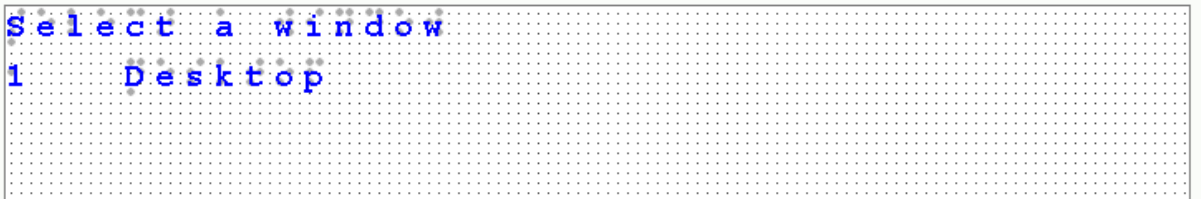


Figure 44 - Switch between menu windows.

Beyond having the option to customize and change windows. The user is able to adjust the speech playback rate and braille language, which takes an important role when translating characters exclusive to specific braille alphabets.

Conclusion:

To develop and define new design principles and user interface solutions, some aspects of the Tactonom tactile user interface were discussed, including the application contextualization of a **plain text editor,** a **file explorer**, a **widgets library**, and an **excel table sheet plugin**. All of the user interface elements presented in this document are part of the current state of the Tactonom user interface.

Not all GUI elements implementations to have a beneficial effect and usage for blind and visually impaired people when used in 2DRT user interfaces. Some widgets, as the scrollbar do not have a big impact on user interaction since dynamic interaction is limited to a refreshment rate of 10 seconds. However, audio feedback can be used more dynamically to present dynamic elements that the tactile feedback can not.

As the Tactonom is limited to this refreshment rate, further analysis and discussion are necessary in order to design more usable user interface elements for this technology. This user interface design discussion should evolve the participation of blind and visually impaired participants, since what is usable for sighted persons is not necessarily what is beneficial for blind persons.

As an emerging technology, investigation will focus on the scientific gaps in user interface design, mostly on audio-tactile widgets design, dynamic control elements and fingertip and gesture recognition.