Executive Summary
SpeedyType: Improving communication using only the eyes

AILIE FRASER, RYAN KRAL, KRISTINA DO, JONATHAN HO, CHRISTINE PHAM, DARREN SYU, DUWEI WANG, University of California, San Diego

A. KEY PROBLEMS ADDRESSED

Our project aims to help patients with Locked-In Syndrome (LIS), a neurological disorder in which patients are unable to control any muscles in their bodies except for their eyes, yet they remain fully conscious and able to think [Loc 2007]. There are existing systems to help such patients communicate with only their eyes, but these tend to be slow and inefficient. In addition, many commonly used typing systems for LIS patients rely on another person's assistance. We sought to increase the speed and ease at which LIS patients can communicate, decrease the amount of dependence on other people, and improve their quality of life with an enjoyable interface that can be customized based on the specific user's needs and abilities.

Our vision for the project was to build a typing system that did not rely on blinking or dwell time, as both of these input methods can be slow and error-prone. We also envisioned a keyboard that was more efficient for eye control than a traditional QWERTY keyboard, namely a circle. Finally, we wanted to provide shortcuts for communication to reduce the need for repetitive actions, such as predicting words and phrases based on who the patient is communicating with.

B. DESIGN IDEAS

The design ideas for SpeedyType originated in various stages of our team's development cycle. The initial idea was that a keyboard should be implemented in a circular layout with keys placed around a center confirmation button. Typing with the eyes would occur by looking at a character on the outer ring, then looking to the confirmation button in the center in order to type the selected character (a method we refer to as "speedy"). A rough version of this typing approach was developed using Java and then tested for feasibility. It was determined that though the circular arrangement of characters was a promising layout, the large quantity of buttons (30) was reducing the precision of letter selection.

To improve upon this idea, we grouped letters together and reduced the 30 selectable characters to just ten selectable buttons. T9 typing was then implemented so that each button could be selected once, and all possible words which could be made from the chosen letter combination were displayed. This change to the design increased the button size, increased the precision and usability with eye control, and also increased the speed and ease of typing because T9 is already a well-developed technology that has been proven to work in the past. This design idea was ultimately settled on and named the AlphaDial. A suite of features were developed around the AlphaDial to further show its usefulness and effectiveness: contact storage, the ability to send emails and quick messages, piano and emoticon keyboards, and variable settings. The AlphaDial combined with these features ultimately resulted in the full SpeedyType application.

DOI: http://dx.doi.org/10.1145/0000000.0000000

C. IMPLEMENTATION

The SpeedyType application described above was developed using Java with the NetBeans Integrated Development Environment. Java was chosen as the programming language because all team members were familiar with it and it could be run on a wide array of operating systems. BitBucket in combination with GitHub was used for version control of the software. This allowed for all team members to easily access the latest version of the code in order to add features, test the application, and document the progress of the project. For eye tracking capabilities, we used the Eye Tribe technology. It provided an inexpensive, portable eye tracker which could easily replace mouse control on a variety of devices. Finally, the Microsoft Surface was chosen as the target platform for the application as it runs on Windows 8 which supports Java, has a USB 3.0 port so it can make use of the Eye Tribe, and is portable enough to be easily moved around with a wheelchair.

The above design decisions were made and ultimately allowed for parallel development to occur on the various development machines that our team members had. In the end, the Eye Tribe was used to replace mouse control with eye movements; this gave all team members the ability to design and test on their local machines, and only final testing had to be performed on the Surface tablet.

D. FEATURES

The following are some of the key features developed for our project:
Contact Storage (”Speedy Contacts”) - Contact information can be saved, edited, and deleted. The user of the application can then select contacts for use with contact-specific features (such as Send Email).
Send Email - The AlphaDial keyboard is displayed and allows users to type a message using a circular or vertical layout. Once finished, the typed phrase is sent to the contact’s email address.
Piano Keyboard - We repurposed the AlphaDial keyboard by replacing the letter buttons with the notes of the chromatic scale, to allow users to play melodies using only their eyes. Input is made in the same way as with the AlphaDial keyboard.

E. EVALUATION

The AlphaDial keyboard was evaluated in comparison with one of the primary typing methods used by LIS patients: the alphabet board. We also implemented the ability to select letters using dwell time for another point of comparison. Dwell-time selection works by gazing at a button for half a second to select it. We tested all five methods (alphabet board, circular and vertical speedy typing, circular and vertical dwell-time typing) and found that both speedy typing methods were significantly faster than the alphabet board and were equally as accurate. Circular and vertical dwell-time typing methods were faster than the alphabet board but slower than the speedy method and had more errors. After this testing we were able to conclude that our keyboard is more efficient than the alphabet board, and through our own subjective experiences we also found it much more pleasant and intuitive to use.

F. FUTURE DIRECTIONS

Based on our original vision to make a faster and more efficient keyboard, future improvements to the system should begin by focusing on the AlphaDial keyboard. The keyboard can benefit from improving the priorities of suggested words, enhancing the dictionary used to include more ”slang” terms, and by adding grammatical elements like symbols and capitalization. Following these changes, improvements can be made to the piano keyboard to allow for features such as playback, rhythm, multiple octaves and chords. With these future extensions in mind, it is clear to see that our project has laid the foundation for many further benefits.

SpeedyType: Improving communication using only the eyes

AILIE FRASER, RYAN KRAL, KRISTINA DO, JONATHAN HO, CHRISTINE PHAM, DARREN SYU, DUWEI WANG, University of California, San Diego

1. INTRODUCTION

Locked-In Syndrome (LIS) is a neurological disorder that causes patients to become ”locked” inside their own bodies. The severity of symptoms varies, but generally patients are unable to control any muscles in their bodies except for their eyes, yet they remain fully conscious and able to think [Loc 2007]. The biggest problem LIS patients encounter is the inability (or limited ability) to communicate with others. Our project aims to help increase this ability. The project was motivated by a specific case: a patient named Bob Veillete, who developed LIS as a result of a stroke [Bob 2014]. Bob is completely paralyzed except for some eye movement and blinking. Before his stroke, Bob was a jazz pianist and a journalist, but LIS has made it impossible for him to play piano or make music, and very difficult and time consuming to write articles and communicate with others [Bob 2014]. Our goal for this project was to develop a system using eye tracking that makes it easier for Bob to communicate and improves his quality of life.

We first looked into existing methods of communication using the eyes, and found that many of them are inefficient and are far from optimal. We sought to create a keyboard for typing messages that can be used quickly and accurately, and does not require as much movement as a traditional keyboard, and for this we decided on a circular layout. To allow for people with varying amounts of eye movement ability, we also envisioned a vertical version of the same keyboard. In addition, we wanted to make it easier for Bob to communicate with specific people, so we aimed to personalize the message-sending interface based on who the message is being sent to. We called our system SpeedyType, and it involves the storage of contact info which we called Speedy Contacts, and typing messages with a circular (or vertical) keyboard called the AlphaDial. Once we had the main components of our program in place, we thought about Bob’s case in particular and how we could extend it to improve his quality of life based on his interests. Since Bob used to be a journalist, we added the option to write an article and save it to a file. We also demonstrated that the AlphaDial keyboard can be repurposed for any variety of uses and implemented a Piano Keyboard version of it to allow Bob to create piano music in real-time.

The rest of the report is structured as follows: In Section 2, we introduce the motivation behind our program, provide background information on existing solutions and discuss the problems with existing solutions that we aim to solve. In Section 3, we walk through our design idea, and give a timeline on how it evolved over time and arrived at the end product. In Section 4, we give an overview of the architecture of our program and the technologies we used, and then provide detailed descriptions of the many features in our program. In Section 5, we describe the results of our testing, and confirm our hypothesis that our typing program is faster than current methods and is just as accurate. In Section 6, we discuss the team dynamics and organization and outline each member’s contribution to the project. Finally, in Section 7 we discuss some future extensions that could further improve our program and conclude with some notes on how the project went and how we achieved our initial vision.

2. MOTIVATION AND BACKGROUND

Our main objective for the project was to aid those with Locked-In Syndrome (LIS) or Motor Disabilities. Those with Locked-In Syndrome are fully conscious but are almost completely paralyzed, being limited to varying degrees of eye movement. There are three categories of LIS. The first is Total LIS, which means there is quadriplegia, anarthria and no eye movement [Keen 2014; Duchowski 2002]. The next is Classic LIS, which has "preserved vertical eye movement and blinking" [Keen 2014]. The last is Incomplete LIS, which has eye movement and some other forms of voluntary movement [Keen 2014; Duchowski 2002]. The amount and rate of recovery of motor skills varies quite a bit from case to case, ranging from no recovery at all to full recovery [Duchowski 2002]. This imposes several factors to account for since each person will be in their own stage of the syndrome. With the limitations each person possesses, we wanted to ensure that as many people as possible would be capable of using our product.

Individuals with LIS have to be equipped with a means to help them communicate. On top of that, many lose the ability to eat and must be supplied with a feeding tube [Keen 2014]. It can be seen that no ordinary task will be simple for a person diagnosed with LIS. The range of motor capabilities an LIS patient has varies widely and there is still no readily available prognosis. Most LIS patients are bound to a wheelchair, therefore the need for assistance is great especially if their motor ability is severely limited [Keen 2014]. Because of this, we focused on creating a product that can help ease the difficulty of communicating. In order to improve quality of life for LIS patients and those who are assisting them, we wanted to create a better keyboard that could effectively alleviate the caretakers' parts in trying to understand what the patient is in need of.

This idea to help with communication is important since care is the main way for patients to do anything. Patients usually use some sort of computer device to communicate, but that only works if there is a way to track eye movement [Beaudoin and De Serres 2010]. We looked into several products for eye typing on the market to see what strengths they have and what areas still need improvement. We will now discuss these other products and their functionality.

2.1 Alphabet Chart / Board

A popular method used by LIS patients to communicate is an alphabet chart (Figure 1) [Khanna et al. 2011]. To use this, someone must assist the patient by going through each line, and the patient blinks when the line they want is reached. Then, the helper goes through each letter on that line and the patient selects a letter by blinking again [Khanna et al. 2011]. This would be used for every sentence or phrase the person would want to say. Another way of using this chart is for the helper to go through each letter of the alphabet one by one rather than row by row, and have the user blink when the letter they want is reached. Bob uses a method like this, but instead of going through the list alphabetically, his helper reads letters in order of how commonly they are used [Bob 2014]. These methods are generally accurate but are quite slow and inefficient as they require cycling through letters repeatedly.

2.2 Eye Typing with a Keyboard

There are several applications out there that allow users to provide input to a keyboard with their eyes. The Jet Propulsion Laboratory and LC Technologies have both built systems involving QWERTY keyboards that users control by gazing at each letter in turn [Kaplan 1999; Eye 2013]. The eye tracking company Tobii has also built a similar keyboard program, as part of a larger system that allows users to control their entire computer with their eyes (Figure 2) [Tob 2013] . These keyboards all use dwell time (gaze time) for the selection of each letter, which is the time spent gazing at one point on the screen. For typing, it is usually set to about half a second [Kaplan 1999; Eye 2013]. Using gaze as input can be

A	B	C	D	End of word	
E	F	G	H	End of sentence	
I	J	K	L	M	N
O	P	Q	R	S	T
U	V	W	X	Y	Z

Fig. 1. An alphabet chart where each row begins with a vowel. Image from [Khanna et al. 2011].

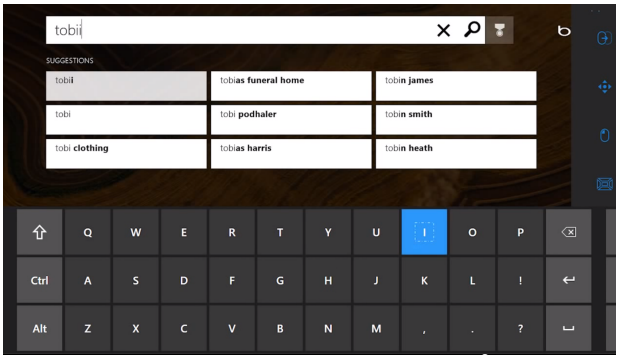

Fig. 2. The Tobii eye-typing interface. The system suggests words based on what has been typed so far. Image from [Tob 2013].

an issue since the user must be constantly aware of what they are looking at. If they happen to just be staring somewhere then opening an unwanted application could be a side effect of this. Also, as stated before, not all those with LIS have full range of eye movement and the ability to only do vertical eye movement would greatly limit their capabilities to communicate with such a keyboard. The use of a regular keyboard generally has the most common letters in hard to reach spots and are on opposite ends [Key 2000] which works well for typing with both hands, but is far from optimal when there is only one point of input, such as eye gaze. The circular keyboard we created helps to eliminate this distance issue by placing all keys in the same range of each other.

2.3 8pen Keyboard

Another keyboard is the 8pen keyboard, shown in Figure 3 [Payne 2014]. The 8pen is a gesture-based keyboard for touch screens. Letters are arranged around a central ring, with more commonly used letters closer to the center. Users form gestures with their finger by moving it through the different sections based on the letter the user wants to select [Fitzpatrick 2010]. Like the Dvorak keyboard [Key 2000], the 8pen wants to help eliminate the distance traveled to each commonly used letter. This is efficient, but society is already accustomed to using keyboards with buttons, therefore those with LIS may not want to learn a new typing method since they are already limited in other ways. The learning curve for the 8pen can be very steep due to its unconventional arrangement of the letters.

2.4 Our Vision

We envisioned a keyboard that would help conquer the issues raised above, and decided on a circular keyboard. Since the letters are formed around one area, movement would be much more efficient than with a standard QWERTY keyboard. It would also provide an effective way to communicate without much assistance from others. We wanted this keyboard to rely less on dwell time and blinks since these can slow things down considerably. Dwell time for selecting buttons can be error-prone if it is set to be quick, like half a second, and can be tedious if it is set to be slower, like several seconds. Blinks are another way of selecting, such as used with the alphabet chart, but they can be ineffective since blinking also happens naturally and it may be difficult to differentiate between intentional and unintentional blinks. We wanted to bypass these problems to develop a keyboard that would not rely solely on either of those aspects.

Fig. 3. The 8pen gestural keyboard interface. Image from [Payne 2014].

3. DESIGN

We will now discuss the timeline of our project, and show how our ideas and prototypes evolved over the course of the development. Figure 4 shows a timeline of our project's evolution, with key developments and their dates marked.

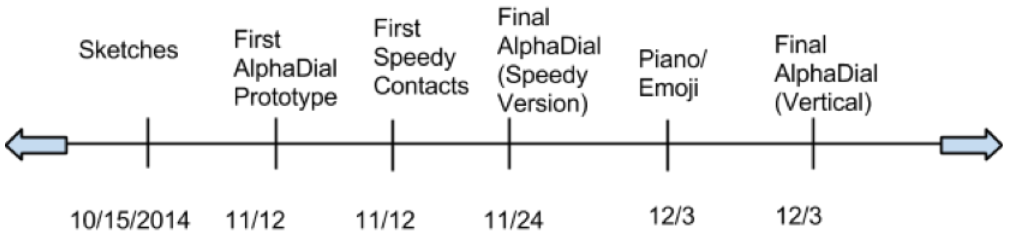

Fig. 4. A timeline of our project's evolution and key developments.

3.1 Initial Sketches

Our initial wireframe sketches both featured the circular design (Figure 5). The design on the left was the earliest design (envisioned during the first group meeting by Darren) and the one we decided to implement. It would feature many different sections (one for each letter). The design on the right was a backup design that would be implemented should the EyeTribe fail to be accurate enough. At the time, it was thought that users would select from a group of letters, and then select a specific letter out of that group. The design was dubbed "AlphaDial" at the end of the first meeting. The first team meeting also produced ideas of a contacts/messaging system, drawing system, home screen, and instruments (piano) system. Some of the ideas were further developed by the 10/22 meeting, with the primary focus being the development of a home screen, AlphaDial-based messaging system, and contact storage. The next week, we split into three subgroups: one for AlphaDial, one for the Eye Tribe, and one for Contacts (dubbed "Speedy Contacts").

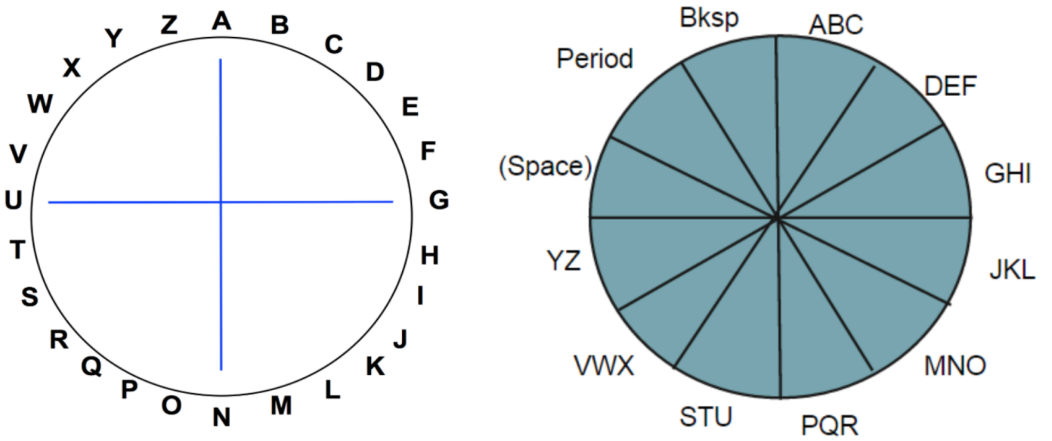

Fig. 5. Wireframe sketches for the AlphaDial.

3.2 Development of AlphaDial

The first working implementation of the AlphaDial was delivered on 11/12 and was built in Java (Figure 6). It had 30 buttons; 26 for letters, a few for punctuation, and one for space. The Alpha Dial subgroup notably implemented the ”plus sign” method of inputting letters, which was intended to be faster than dwell time or blinking. This system of input involves mousing over the desired letter button, then mousing over the plus sign to input the selected letter. This plus sign is the confirmation that allows the user to make mistakes before the final selection takes place; only the last letter they mouse over before mousing over the plus button will be selected. Because of this, a delete button was not initially implemented. We integrated this version of the AlphaDial with the Eye Tribe by enabling mouse control with the eyes, but it failed to work sufficiently well. The Eye Tribe was not accurate enough and the squares were too small. Thus, the second wireframe had to be implemented instead.

By 11/17, a T9 version of AlphaDial was implemented. At this time, AlphaDial was no longer its own separate application, and had been integrated in with Speedy Contacts. We could now send emails by selecting a contact in Speedy Contacts and typing a message with AlphaDial. This version of AlphaDial uses the same ”plus button” method for entering text as the previous version. However, it uses the T9 method for entering and word prediction in lieu of the user selecting each letter. T9 was chosen for its speed - it was faster than double selection. It has a long and proven history, as it was the main method of text input used by cell phones before smartphones with full keyboards became popular. Lastly, it was straightforward to implement by adapting existing open-source code for word prediction. All these reasons made it clear why T9 would be superior to the double selecting method that was envisioned at the first meeting.

The screenshot in Figure 7 is the final version; however, the older versions look practically identical with a few exceptions. Initially, users were unable to select a suggestion from the row at the top (which is needed when a key combination has multiple possible words) but that was quickly fixed in the coming week. By 11/24, a delete button was added, sounds were added for feedback, and green highlighting was added (not visible in screenshot). The sounds were added (one for the initial mouse over, one after final input) after we spoke to Bob’s manager at the Midterm Madness demo; he suggested that Bob would find auditory feedback helpful.

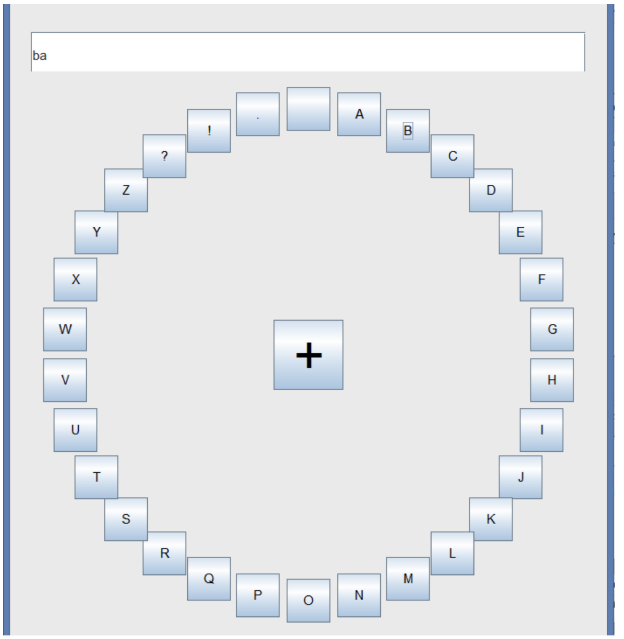

Fig. 6. First working prototype of AlphaDial.

Fig. 7. Final version of the AlphaDial.

On 11/24 we had a replanning session to prioritize the remaining tasks. We deemed eye control, vertical and dwell versions of AlphaDial, sounds, and a ”Write Articles” function to be the most important. We also discussed ways of repurposing the AlphaDial for other uses, namely displaying emotions and playing music. The latter was motivated by the fact that Bob used to play piano. Thus, several versions of AlphaDial were made by 12/3: emoji, piano, and vertical AlphaDial. We also implemented dwell time for both circular and vertical typing, mainly so that we could compare the speed and accuracy of our ”plus button” method with the more commonly used dwell time method. We called our plus button method ”speedy” to differentiate between it and dwell time. In the piano version, the T9 buttons are replaced with the 12 notes of the scale. One note is played at a time, with all notes being of equal loudness and duration. Likewise, the emoji keyboard replaces the T9 keys with emoticons. Once an emoticon is confirmed, a large emoji is displayed on screen until the exit of the application. A vertical version of AlphaDial was implemented after hearing confirmation at Midterm Madness that Bob had limited horizontal eye range. Since Bob has some slight horizontal range, our vertical design included two large buttons, one on either side of the vertical keyboard.

As we developed the various versions of the AlphaDial, we tweaked the design and button placement based on testing and experimentation. The final results were four versions of the AlphaDial for typing (circular and vertical layouts, both with speedy and dwell-time selection methods), and three different uses for the keyboard (typing text, emoticons, and piano). All have sound, green highlighting to indicate selection, and fully integrated eye control via EyeTribe.

3.3 Development of Speedy Contacts

Speedy Contacts came into being a little later than AlphaDial, with its first implementation on 11/12. This first version was mouse controlled, and was designed to store contacts and use AlphaDial to send messages. It had small text, and consisted of two larger scroll buttons at the top and bottom, a center area for displaying contacts, and two 'add contact' buttons that were next to the scroll buttons.

By 11/17, Speedy Contacts had a screen that displayed relevant information once a contact was selected; this included "Send Email", "Send Text Message", "Edit Contact", "Delete Contact", and "Back". It featured the same green highlighting as AlphaDial. There was tentative email functionality at this point; there was a hard coded email sender and receiver, and the writing function utilized AlphaDial. "Edit Contact" and "Delete Contact" were not functional, though they would be in later versions. We decided at the 11/17 meeting that the texting function should be removed since it would be too time consuming and potentially costly to implement.

By 11/24, an "Edit Contact" popup interface was added, giving it full functionality; a popup confirmation was also added for the "Delete Contact" functionality. Lastly, texting was fully abandoned in favor of a "Quick Message" function that allows the user to quickly select a message to send from a list of messages previously sent to the selected contact. A sound upon the initial mouse-over of a button was implemented across all areas of Speedy Contacts. Lastly, the "Write Article" function was made, which resembled the "Send Email" functionality, but would save the text to a file rather than send it to a contact.

12/3 brought the end of the development cycle. The final application was named SpeedyType, and Speedy Contacts thus became part of SpeedyType. A home screen was added, which includes "View Contacts" (Speedy Contacts), "Write Article", "Piano Keyboard", "Emoticons", "Settings", and "Quit". Settings allowed the user to change various aspects of the interface, including the keyboard layout (circular or vertical) and input method (speedy or dwell). We also implemented dwell-time selection for all menu screens in the program. A bit of final debugging and polishing occurred after 12/3, but 12/3 was essentially the last major update of SpeedyType.

4. SYSTEM DEVELOPMENT

We will now discuss the implementation of our program, including the architecture and technology used, followed by descriptions of the many features in our program.

4.1 Architecture

Our entire program was written in Java. The program is modular, with a different class for each functionality. Every screen is implemented as a subclass of DwellMenu, which is an extension of JDialog that we created. The DwellMenu class provides the ability for buttons to be selected using dwell time. Since this was needed for all screens, we implemented it as an extension of the JDialog class. Thus, any class that extends DwellMenu will have dwell time functionality. When the user selects an option in a menu, the current Dialog opens a new Dialog, passing it the necessary information. It then waits for the new Dialog to be closed, and retrieves the information passed back by the new Dialog and acts on it if necessary. Since each Dialog is implemented as its own independent class, it is easy to add new components to the program without worrying about merge conflicts within files.

As an example, we will discuss the control flow that happens for sending an email. Other functionalities work in a similar way. It is important to note that only the HomePage and ContactsDialog classes create and open new dialogs.

To send an email, user activity starts at the HomePage class. When "View Contacts" is selected, the HomePage creates a ContactsDialog and opens it, thus displaying the Contacts screen. When a

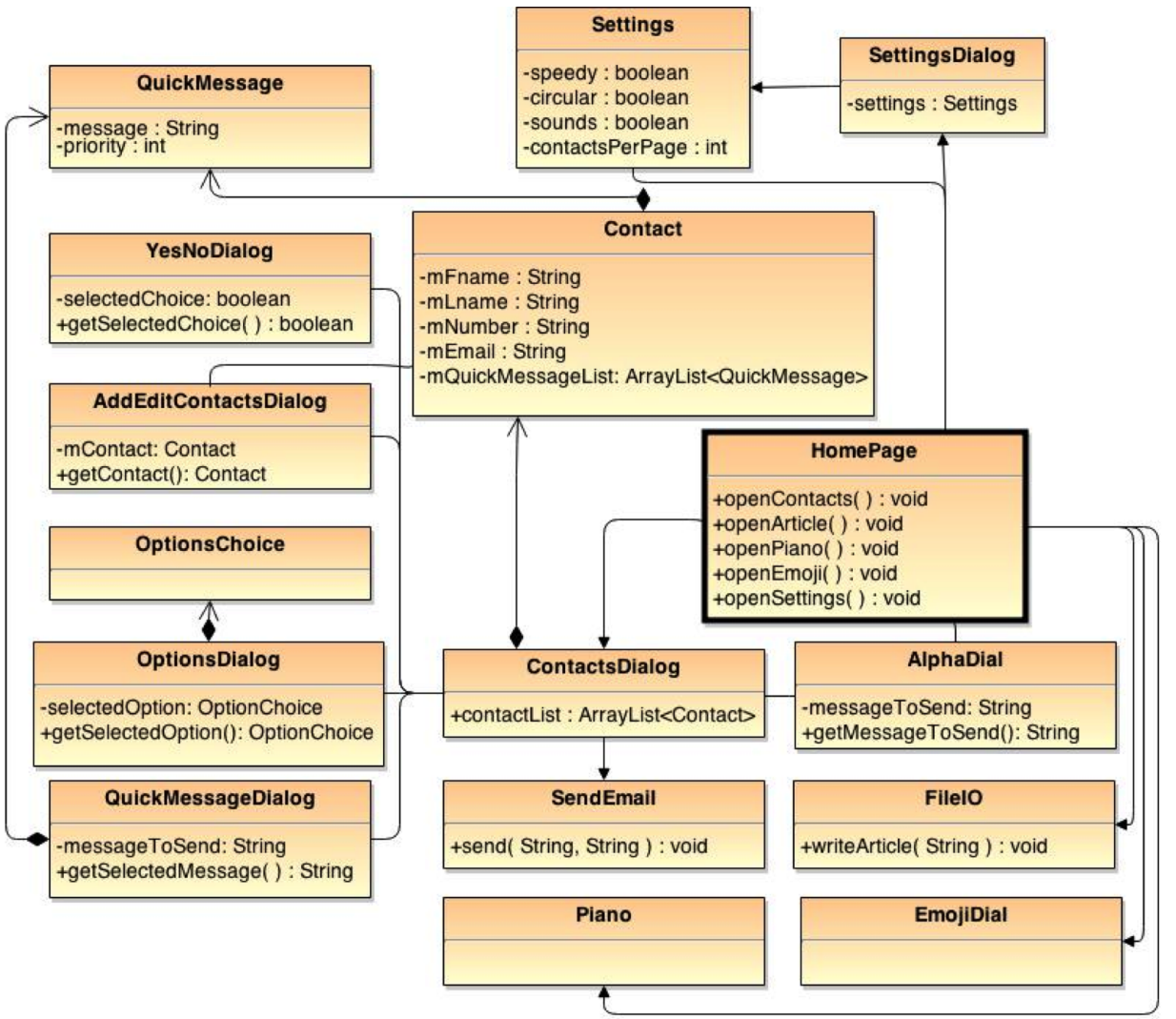

Fig. 8. A UML diagram showing the flow of information, class relationships, and important variables and methods in our program.

contact's name is selected, the ContactsDialog creates an OptionsDialog and opens it. When "Send Email" is selected, the OptionsDialog saves that as the selected option and sends it back to the ContactsDialog. From here, the ContactsDialog creates an AlphaDial dialog and opens it, thus displaying the AlphaDial keyboard. The AlphaDial stores the message the user is typing and passes it back to the ContactsDialog when the user presses Send. From here, ContactsDialog calls on SendEmail and passes it the message and the email address of the contact to send it to. SendEmail then sends the message using a Gmail account we created through SMTP.

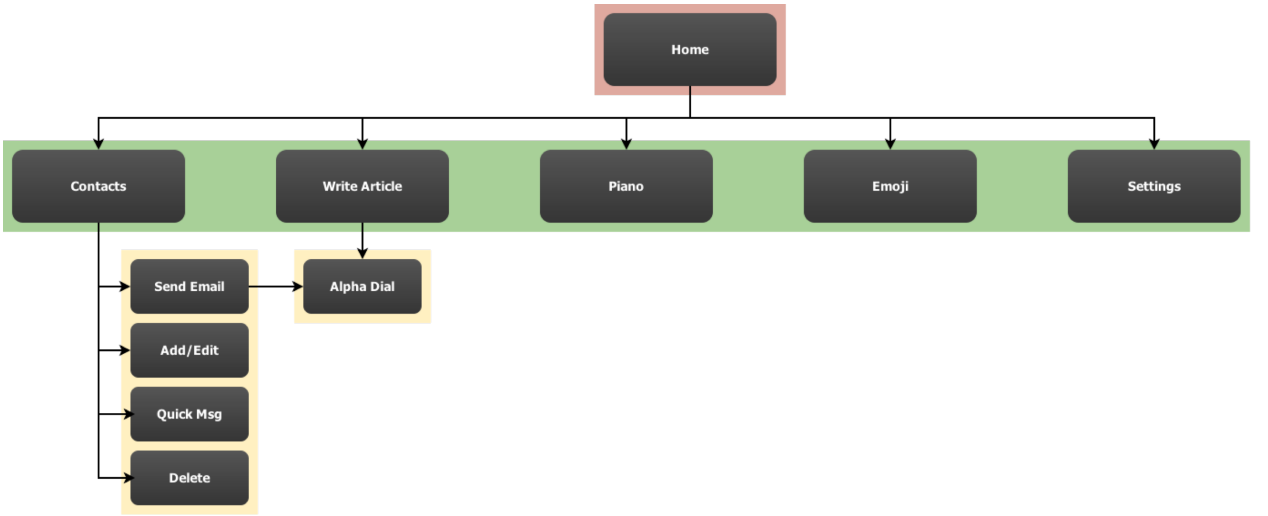

Fig. 9. A high-level overview of the dialogs involved in our program.

4.2 Technology Used

4.2.1 *Eye Tribe.* The Eye Tribe was an important device to use in our project because it would allow the user affected with LIS to communicate. The Eye Tribe software enables people to use eye control on their computers, which allows for hands free navigation of websites and applications. The Eye Tribe uses a standard USB 3.0 connection and runs on both Windows and MAC operating systems. This enables anyone running on Windows or MAC with a USB 3.0 port and $99 to access this technology [Eye 2014]. On top of these features, the Eye Tribe is small enough to be portable and will not be a hassle to carry from one place to another.

The Eye Tribe was used in our program in place of a mouse. This was done by starting the Eye Tribe server, calibrating the user with the device, then turning on mouse cursor control through the EyeTribe server before starting our application. These initial steps must be performed with the help of an individual without LIS, but allowed for our application to be developed for use with the mouse or Eye Tribe. Because of the features the Eye Tribe provides, its inexpensiveness, and the ease with which it can replace mouse control, it was a good choice of device to use for this project.

4.2.2 *Microsoft Surface.* We chose to use the Microsoft Surface for our project for a variety of reasons. It ran the Windows 8 operating system and had a USB 3.0 port which allowed for connectivity with the Eye Tribe. Also, because it runs Windows 8, it allows support for many different programming languages. By making the target platform the Surface, we were able to work on the application on separate computers without the need for the Surface at all stages of development and testing. The final application, though developed for the Surface, can ultimately be run on any device which supports Java and can be connected to the Eye Tribe.

The smaller, tablet-like, size of the Surface allowed for portability and made sense for the target user who will likely need a small, portable device which can be moved around with a wheelchair. The Surface also allows for a multitude of other applications to be run, and was ultimately chosen for its compatibility with the Eye Tribe, ease of use, and versatility.

Fig. 10. The Eye Tribe. Image from http://eyetribe.com.

Fig. 11. Microsoft Surface. Image from http://media01.versus.io/microsoft-surface-pro-2/front/front-1380271251350.flat.jpg

4.3 Features

4.3.1 *Implementation Details.* The SpeedyType application supports many features which have all been implemented in a similar manner. Java was used as the development language across the entire application due to its ease of use, and cross-platform capabilities. For each screenshot shown below, the layouts were designed as Dialogs using the NetBeans GUI designer in combination with hand-coded alterations. Each Dialog, not including the HomePage, can be opened by another Dialog and only closes when its intended functionality has been completed or cancelled. Dwell time was placed on all menu options by extending a DwellMenu class which controlled timers and button selections. In addition to these implementation details, it is important to note that all Dialogs except for the HomePage and ContactsDialog, implement one unique feature of the application. Group members were assigned to different features and could create these in separate classes without worrying about merge conflicts. Then, once new features were completed in a standalone manner, one team member was designated to merge features into the full-fledged application. Because all features were implemented using a single Dialog and supporting data structure classes, merging the features became a simple task which was performed in the same manner for each new feature.

For further information on implementation details (including details on the flow of data), please see the Architecture section of this paper.

4.3.2 *Home Screen & Navigation.* The SpeedyType home screen has a linear, vertical layout and is clean and easy to navigate. It displays a welcome message and a menu with the following options: ”View Contacts”, ”Write Article”, ”Piano Keyboard”, ”Emoticons”, ”Settings” and ”Quit” (Figure 12). Users can navigate the home screen using their eyes, and whichever button they are currently looking at is highlighted in green. To select a button, the user must dwell on a menu item for 3 seconds. Highlighted buttons become darker each second to signify to the user that they are in the process of making a selection. To exit the program, users can choose ”Quit” at the bottom of the Home Screen menu.

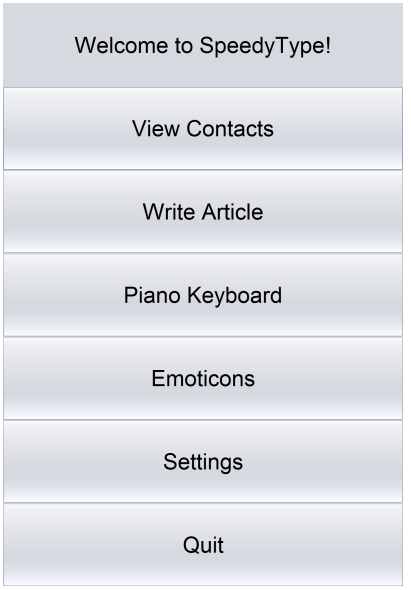

Fig. 12. The SpeedyType Home Screen with the user looking at "View Contacts".

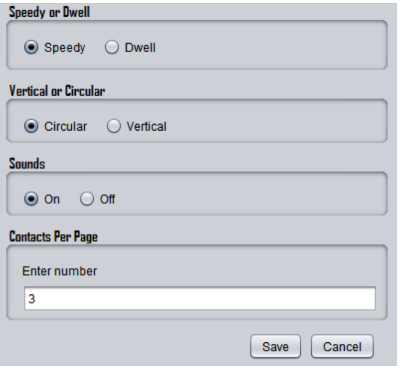

Fig. 13. The Settings Dialog box.

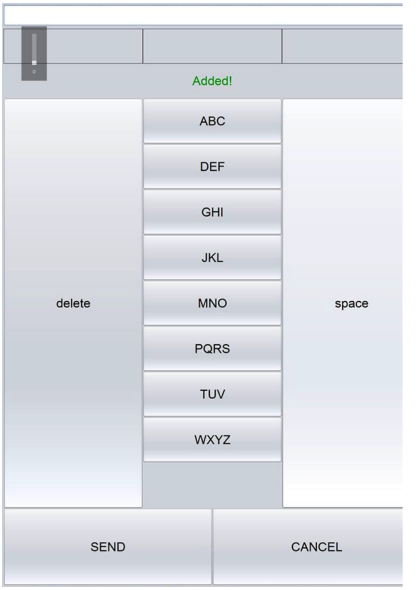

Fig. 14. Vertical Keyboard in dwell mode.

4.3.3 *Settings.* When the user selects "Settings", a dialog box will appear with the options to customize 4 categories: keyboard input method, keyboard layout, sounds, and contacts per page. Users who can only utilize their eyes will require assistance to select options, as this dialog requires input from a mouse and physical keyboard.

Input Method - Speedy vs. Dwell: Users are able to choose from two input methods: speedy or dwell. Speedy mode allows users to choose a key (example: letter, piano key, emoticon) by looking at it, then confirm the key selection by looking at the plus button or confirm key. Dwell mode allows users to choose a key (example: letter, piano key, emoticon) and confirm selection by dwelling on that key for half a second. In dwell mode, if users would like to select the same key, they are required to move the cursor outside the key and dwell on it again. These input options are applied to the AlphaDial keyboard, the Piano keyboard, and the Emoticon keyboard.

Keyboard Layout - Circular vs. Vertical: Users are able to choose from two options for the keyboard layout: Circular or Vertical. The circular keyboard interface has the appearance of a rotary dial. If it is in speedy mode, the crosshair or confirm key will appear in the center of the circle. If it is in dwell mode, an "Added" message will appear in the center after each character is selected. The vertical keyboard interface is split into 3 columns, with the keys in the center and the delete button on the left. In speedy mode, the crosshair or confirm key will appear in the rightmost column, and the space key will appear at the bottom of the center column. In dwell mode, the space key will appear in the rightmost column. An "Added" message will appear above the alphabet keys after dwelling on the desired character for 0.5 seconds.

Sounds: The sounds option allows users to choose whether they want to hear sound when hovering over a menu option or key (this setting does not impact the Piano keyboard). For those with extremely limited eye movement, the sound feature serves as auditory feedback on the accuracy of their movement and whether or not they have switched to a different key.

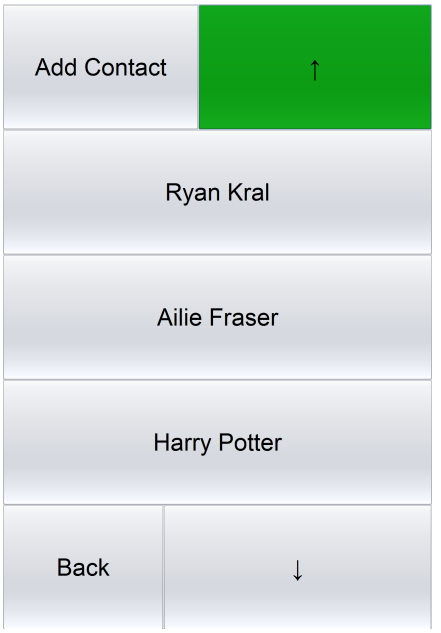

Fig. 15. Main contact navigation menu.

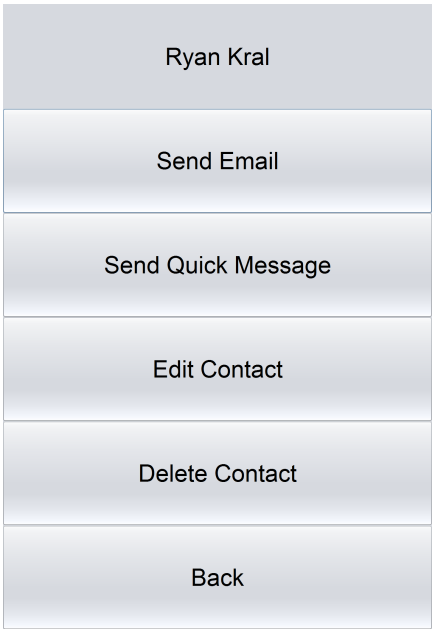

Fig. 16. Contact options page with specific contact name at the top and the actions users can select to perform.

Contacts Per Page: Contacts per page lets users set the number of contacts they wish to see on the ”View Contacts” page. This gives users the freedom to control how many contact buttons are shown at a time. The more the buttons, the smaller they are and so the user can customize this based on their preferences and abilities for precision.

4.3.4 *View Contacts.* When users choose ”View Contacts”, they can add a contact or select a specific contact, which brings up a submenu with the options: ”Send Email”, ”Send Quick Message”, ”Edit Contact”, or ”Delete Contact”. The ”View Contacts” page has a linear, vertical layout with contact names stacked. The navigational arrow keys allow users to scroll between pages of contacts.

4.3.5 *Add Contact.* If users choose to add a contact, a dialog box with fields for the contact’s first name, last name, phone number, and email address will appear. This requires input using a physical mouse and keyboard, as the intention is for the person being added to enter their own information for the patient.

4.3.6 *Send Email.* The ”Send Email” feature allows users to compose a message using the Alpha-Dial. The message appears in the white text area at the top of the page. As the users type, suggested words appear in the grey boxes at the top. To select a suggested word, users hover over the word and it replaces the current word in the white text area. Like with most smartphone keyboards, if the user enters two spaces in a row, a period will appear. When users are done composing, they can dwell on the send button, and it will email the message to the specified contact. If users decide they no longer want to send an email, they can select the cancel option.

4.3.7 *Send Quick Message.* ”Quick Message” is a quick and simple feature that displays a list of previously composed email messages to that specific contact. Users select a message to send by dwelling

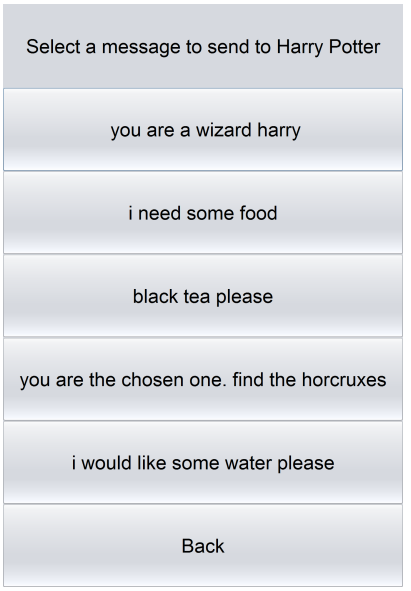

Fig. 17. Quick Message feature showing messages previously sent to a contact.

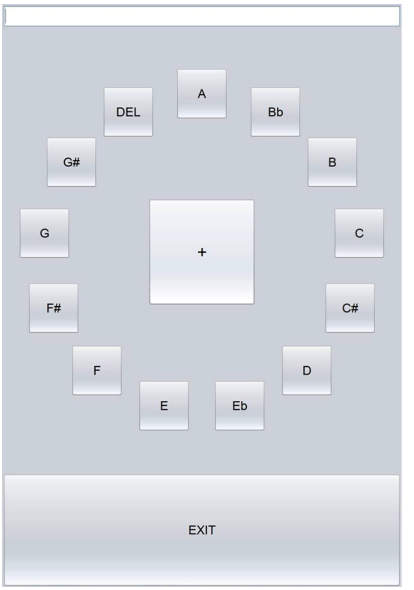

Fig. 18. Piano Keyboard containing the twelve notes of the chromatic scale.

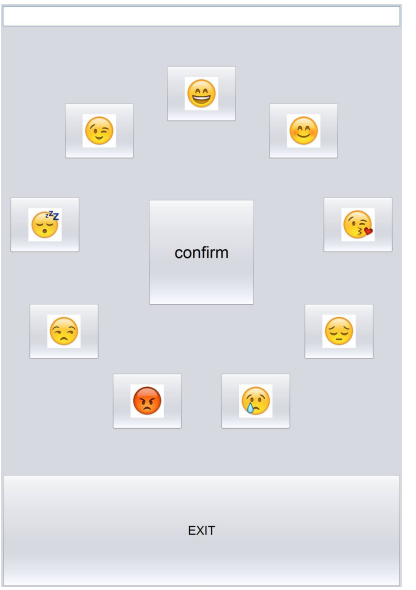

Fig. 19. Emoticon keyboard.

on it for 3 seconds, as in the other menus. The messages displayed to the user are ordered based on the number of times that specific message has been sent in the past.

4.3.8 *Edit Contact and Delete Contact.* The ”Edit Contact” feature is almost identical to the ”Add Contact feature”, the only difference being that the dialog box already contains the contact’s information and users can edit the existing information. As with ”Add Contact”, input requires a physical mouse and keyboard. To delete a contact, the user dwells on the delete button. A confirmation page will appear and the user must dwell on ”Yes” or ”No” to proceed with the process.

4.3.9 *Write Article.* The ”Write Article” feature allows users to embark on creative writing and self expression. Like the ”Send Email” feature, this uses the AlphaDial keyboard. When they are finished, dwelling on the ”Send” button will save their composition as a text file.

4.3.10 *Piano Keyboard.* ”Piano Keyboard” is a feature aimed at improving quality of life by allowing users to have fun and create music. The piano keyboard serves as a proof of concept because it shows how flexible the AlphaDial keyboard is, since only minor tweaks and additions were needed to convert the AlphaDial keyboard to a piano. Some of the changes include switching the letters on keys to the names of musical notes and adding the correct piano sound to each note. The piano keyboard currently features the twelve chromatic notes of the scale (in only one octave) and a delete key. The usage and input method for the piano keyboard is identical to the AlphaDial except that when a note is added, the corresponding sound is played. The notes selected can be seen in the top text area. Users can also delete the last note they played and it will be removed from the composition history in the text area.

4.3.11 *Emoticons.* Since most users cannot make facial expressions, the Emoticon feature was added to help users more easily express their current emotions. This feature once again serves as a proof of concept because it shows how flexible the AlphaDial keyboard can be. Here the AlphaDial

keys are replaced with a selection of emojis. Once an emoji is confirmed, it is displayed larger in the center until the user chooses to return to the home screen.

5. TESTING AND EVALUATION

The testing process was designed to compare the speed and accuracy of our developed keyboards with the most common method of communication for patients with LIS: the alphabet chart. Our preliminary research showed that there were multiple techniques which could be used to communicate using the alphabet chart. The two most prominent methods were going through the alphabet one letter at a time, and going through the chart to first select a row (which begins with a vowel) and then traversing the selected row to obtain the letter. Because we wanted to test our methods against the fastest current implementation, we set our control test to be the alphabet chart technique which cycled first through rows and then through the letters within that row. For testing, our tester would start a one minute timer to notify when the test has concluded. During that one minute, our tester would attempt to spell the sample sentence: ”hello my name is bob and I would like a glass of water and an apple.” The tester would record how many words of the sentence were completed and how many errors occurred during testing. This procedure was repeated ten times for each entry method and an average was calculated from the results. Figure 20 shows the results we achieved following this procedure.

Type of Communication	Average WPM (after 10 trials)	Average Accuracy (after 10 trials)
Alphabet Chart (Control)	4 Words Per Minute	1 Error Per Trial
Speedy Circular	12 Words Per Minute	1 Error Per Trial
Speedy Vertical	12 Words Per Minute	1 Error Per Trial
Dwell Circular	10.5 Words Per Minute	2 Errors Per Trial
Dwell Vertical	10 Words Per Minute	3 Errors Per Trial

Fig. 20. Speed and accuracy results for our program and the alphabet chart.

As shown through these results, we found that our speedy keyboards were approximately three times faster than the control results using the alphabet chart. In addition, our accuracy using the speedy circular keyboard or the speedy vertical keyboard is comparable to the alphabet chart. When using the dwell input method, we found that we were able to achieve faster results than with the control, but slower results than with the speedy method. In addition, the dwell method produced more typing errors such as adding an unneeded letter or choosing the wrong letter when typing. By backtracking to delete undesired letters, the user would undoubtedly achieve slower results using the dwell typing format compared to the speedy typing format.

From a subjective point of view, we found that the circular layout seemed more intuitive than the vertical layout when typing using the eyes. While the vertical typing format is obviously important for those with more severe cases of LIS, circular typing seemed more natural. This is because the center plus button acted a default resting position for users and required minimum effort for letter selection. It also allowed users to rest their eyes after a selection while pondering the next word or letter to type without causing typing errors.

When comparing the speedy typing format with the dwell typing format, we found that speedy did indeed live up to its name and was faster than the dwell format. While both had fairly higher words per minutes compared to the alphabet chart, the usage of the speedy format gave the sense of the rapidity that most people get from texting on a phone. By moving between the letters and the plus symbol, the user has the comforting feeling of being able to type as quickly as their eyes allow.

One issue we ran into during testing was the fact that even slight movements of the upper body would cause the Eye Tribe calibration to be slightly off. This meant that the tester would face difficulty selecting the desired buttons accurately, or, if the calibration issue was severe, full recalibration was necessary. We attempted to minimize calibration issues by sitting in comfortable and stable positions when using the Eye Tribe. We also required a secondary helper to assist in actions such as starting the application in order to limit movements of the tester as much as possible.

6. COLLABORATION

6.1 Team Structure

Our team was comprised of two graduate CSE students, one Masters (Ryan) and one PhD (Ailie); and five undergraduate students, three in CSE (Christine, Darren and Jon) and two in Cognitive Science (Duwei and Kristina). We first ran some initial brainstorming sessions and decided on a rough plan for our program. We then assigned a different technology or task to each person, to get a better idea of which technologies would work best with our program. Darren and Kristina took the Eye Tribe, Duwei took the Kinect, Ailie took the Google Glass and Ryan took the Surface. Christine and Jon began prototyping some basic interfaces, with Christine focusing on a home screen and Jon on a form to add contact information. We decided that the Kinect and Google Glass would not be needed for our program.

6.2 Team member contributions to the project

We then split into three subteams to work on the three major components: AlphaDial, Contacts and Eye Tribe. The subteams were decided based on group members' interests and availabilities. The AlphaDial subteam consisted of Ailie and Duwei, the Contacts subteam consisted of Jon and Christine, and the EyeTribe subteam consisted of Darren and Kristina. Ryan floated between the AlphaDial and Contacts subteams. As the project went on and peoples' strengths and weaknesses emerged, each person ended up taking charge of different areas. We avoided merge conflicts by ensuring that whenever someone added a functionality, they did so in a separate class file. Once all the different pieces of the program were completed, Ryan and Ailie merged them together and finalized the flow of the program.

Darren focused mainly on the EyeTribe, integrating it with the Surface and testing our application with it. He practiced typing with the AlphaDial and conducted tests of the different methods' speed and accuracy. Darren also built the Piano Keyboard, basing it off of the AlphaDial code, and implemented all sounds in the program. Christine implemented the email sending, and worked on the home screen layout. She also took charge of taking meeting minutes at every meeting. Jon emerged as our Bitbucket "expert" and helped out other group members who were less familiar with it. He also implemented the Contacts menu and designed the interface for adding and editing contacts as well as Settings.

Duwei and Kristina, being Cognitive Science students, were less experienced with programming and thus their main roles were in storyboarding and documenting. Duwei drew up initial mockups and storyboards of our ideas, and also kept detailed development notes throughout the project as a record of when features were implemented, and the various iterations that our program went through. In addition to this, she wrote up a "User Guide" with instructions for using the program. Kristina wrote the wiki page for our Bitbucket repository, and wrote up an initial outline for our final report.

When we decided not to implement some of our ideas due to lack of time or for other reasons, Kristina wrote up descriptions of these decisions to be used in the final report.

Ailie took charge of the AlphaDial, implementing both circular and vertical layouts and both speedy and dwell typing. Ailie also implemented the DwellMenu dialog, which allowed all menus to use 3-second dwell-time selection, and built the EmojiDial keyboard based off of the AlphaDial code. Ryan took charge of Write Article, Quick Message, and overall flow of information throughout the program. He implemented the backend storage of contact information and quick messages as well as the ability to save an article to a text file. Ryan also connected all the various components in the project together and implemented the passing of information between dialogs, as well as the persistence of the selected settings throughout the program.

6.3 Team member contributions to the report

Our overall plan for the writing of the report was to assign each member one or two sections to write, and then for Ryan and Ailie to bring the sections together, and make revisions to ensure flow and cohesion throughout the paper. We split up the sections as follows:

Introduction: Ailie
Motivation and Background: Christine
Design: Duwei
System Development:
Architecture: Jon and Ryan
Technology used: Christine
Features: Kristina
Testing and Evaluation: Darren
Collaboration: Ailie
Conclusion and Future Work: Ryan

Once the report was completed, Ryan and Ailie put together the Executive Summary by extracting the key points from each section in the paper.

6.4 Organizational Tools and Procedures

Our group met once a week for about an hour and a half on Wednesday evenings. We kept detailed minutes of every meeting that we projected onto a large screen so that everyone could see them and contribute. By the end of each meeting, we would have made a to-do list for the week with the tasks that each member would complete. After a few weeks, we realized that we needed to keep in closer contact throughout the week and make it easier for group members to ask for help along the way, so we made a Facebook group. This helped allow everyone in the group to be aware of what was going on, and meant that we had less catching up to do at each group meeting since everyone was already on the same page.

6.5 Issues and Solutions

Our group was a bit slow to get started in the first few weeks, but once we made the Facebook group we were able to keep in closer contact and get more done each week. Had we done this right at the start we likely would have accomplished more in the first few weeks. Another issue we encountered was with the varying levels of programming and git experience within our group. Many group members had not worked with git before, and/or were not very experienced with Java programming. To deal with this, we assigned tasks based on each member's strengths, for example by assigning writing and storyboarding

tasks to Duwei and Kristina. We also established certain members as the ”point of contact” for issues with particular things. Jon was our point of contact for git and Bitbucket, and Ryan was our point of contact for issues with Java and Netbeans. Another challenge with a large group size would be to allow multiple people to work on code without having conflicts. We managed this quite well by having a modular design and having each member work on specific classes in different files.

7. CONCLUSION AND FUTURE WORK

The vision for this project was set at the beginning of the quarter, and our progress has not led us astray from our initial goal. We aimed to develop a keyboard which would provide a fast and convenient way for users suffering from LIS to communicate using only eye movements. Through our development, we have done exactly this by creating the AlphaDial keyboard. We have shown that the AlphaDial keyboard allows users to type more quickly and with equal accuracy compared to the commonly used alphabet chart typing approach. The speedy version does not rely on dwell time or blinks which other eye-typing methods utilize heavily. Thus, we have not only created a faster approach, but one which is significantly different. This different approach takes advantage of the natural resting place of the eyes (the center of a screen) by allowing selections to be made with minimal effort required. Because of this, we can claim that our typing approach is different and more intuitive than other existing approaches.

To demonstrate and enhance the effectiveness of our AlphaDial keyboard, a full suite of features was compiled into the SpeedyType application. This application went above and beyond our original vision by providing a way to store contacts, compose emails and quick messages, write articles, play piano, and display emoticons. These extensions, though not important to typing quickly and efficiently, further supported the idea of the AlphaDial keyboard and the many ways which it can be utilized in daily life. Through the piano keyboard, we demonstrated a way that our typing approach could be altered to improve the quality of life for patients who suffer from LIS. Specifically, we developed a functional piano which could be played with eye movements only. Though this idea was not completely flushed out, it shows a way in which patients who suffer from LIS can be reminded of their passions and follow them to a deeper extent than they previously could.

7.1 Future Work

When viewing the SpeedyType application as a whole, a wide array of changes could be made as improvements: a cleaner user interface, text-to-speech capabilities, the ability to send text messages, etc. However, based on our original vision of a faster and more efficient keyboard, future work should first be suggested to improve upon the AlphaDial keyboard. T9 word prediction is currently in use, but the suggested words are displayed in alphabetical order. In future iterations of the keyboard, these word suggestions should be made in a prioritized way based on items such as frequency of use, what the preceding words were, and which person a message is being sent to. Another improvement which should be made is to select a more concise dictionary which includes fewer strange, unused words and more ”slang” terms which are common in today’s vernacular. One final improvement which could be made to the AlphaDial keyboard is the implementation of grammatical elements such as capitalization and symbols.

Secondary to improving the AlphaDial keyboard, improvements should be made to the piano keyboard. Based on feedback from the class and other users, this feature is something which could greatly benefit from further work. Currently, the user can play notes and display the history of what they have played in a text area, but there is no way to save this or play it back later. Implementing this ability could allow users to compose melodies and replay them. To make this work, rhythm would also have to be incorporated. The system could either remember the timing the user played the notes at and play it back in the same way, or it could allow the user to select the duration of each note (quarter note, half

note, etc.). Other useful extensions would be to support multiple octaves, perhaps by having two more buttons to move up or down an octave, and to allow multiple notes to be played at once (chords). We originally allowed chords to be played; all the buttons that the user looked at before looking at the plus button would be played at once when the plus button was looked it. We removed this feature, however, because it was very easy to accidentally select unwanted notes and given that Bob has an involuntary twitch in his eyes, we thought this could be a major a problem for him. Alternative methods for playing chords could be integrated into this to further richen the capabilities of our piano keyboard.

With the current project state explained, and future work identified, this project should ultimately be deemed a success. Our initial project goal was met, and extensions were made to push this project above and beyond initial expectations.

REFERENCES

2000. Why are the keys arranged the way they are on a keyboard? (Sept. 2000). http://computer.howstuffworks.com/question458.htm

2007. Locked-In Syndrome Information Page. (Feb. 2007). http://www.ninds.nih.gov/disorders/lockedinsyndrome/lockedinsyndrome.htm

2013. Communicate with the world using the power of your eyes. (2013). http://www.eyegaze.com/eye-tracking-assistive-technology-device/

2013. Computer Access through eye gaze. (2013). http://www.tobii.com/PCEye2011

2014. About Bob Veillette. (2014). http://www.bobveillette.com/about_bob.html

2014. The Eye Tribe. (2014). http://theeyetribe.com

Nicole Beaudoin and Louise De Serres. 2010. Locked-in Syndrome. (2010). http://cirrie.buffalo.edu/encyclopedia/en/article/303/

Andrew T. Duchowski. 2002. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers* 34, 4 (Nov. 2002), 455–470. DOI:http://dx.doi.org/10.3758/BF03195475

Jason Fitzpatrick. 2010. 8pen Is a Speedy Gesture-Based Keyboard for Android Phones. (Nov. 2010). http://lifehacker.com/5678488/8pen-is-a-speedy-gesture-based-keyboard-for-android-phones

Karen Kaplan. 1999. High-Tech Dawn for People with Disabilities: Typing by Eye Movement, Non-Visual. (1999). http://www.independentliving.org/docs5/karenkaplan.html

Shannan Keen. 2014. Locked-in Syndrome. (2014). http://brainfoundation.org.au/medical-info/205-locked-in-syndrome-lis

Kunal Khanna, Ajit Verma, and Bella Richard. 2011. The locked-in syndrome: Can it be unlocked? *Journal of Clinical Gerontology and Geriatrics* 2, 4 (Dec. 2011), 96–99. DOI:http://dx.doi.org/10.1016/j.jcgg.2011.08.001

Olivia Payne. 2014. 8pen - alternative touch screen keyboard. (Oct. 2014). https://blogs.commons.georgetown.edu/cctp-711-spring2014/8pen-alternative-touch-keyboard/