Making Bridge Accessible to the Visually Impaired

Allison DeJordy

Contents

List of Figures

Introduction and Motivation

Problem and Importance

Visual impairment (defined in the U.S. as visual acuity of less than 20/40 in the better eye with corrective lenses [10]) is one of the most common problems to arise as people age. According to a 2004 National Eye Institute study, 2.7% of U.S. adults over age 40 have either low vision or are legally blind. Further, the risk of visual impairment rises dramatically with age: while only 0.3% of adults aged 40-49 are visually impaired, this percentage increases to 23.7% for adults 80 and over. The study estimates that nearly 3.3 million Americans are visually impaired, a number that is likely to rise as the "baby boom" generation ages [1].

As shocking as these numbers are, an even more common problem in aging is the development of dementia. Characterized by memory loss and a decline in mental faculties, dementia can rob affected persons of the ability to perform such simple tasks as eating or getting dressed. A study conducted in 2007 estimates that 13.93% of U.S. adults over age 70 have some form of dementia, a figure that rises to 37.36% for those over 90 [13]. The cost of supporting such an individual can be staggering; the Alzheimer's Association's 2011 Facts and Figures report states that the average per-person cost for Medicare beneficiaries with dementia was $42,072 in 2004, compared to $13,515 for beneficiaries without dementia [2]. In addition, dementia severely decreases the affected person's quality of life, as well as putting a tremendous emotional strain on their caregivers.

While dementia cannot currently be treated or reversed, there are methods of preventing its onset or slowing its progress. The Alzheimer's Association suggests a number of ways to preserve "brain health" on its website, among which are maintaining an active social life and engaging in mentally stimulating tasks [3]. One activity that seems ideal for combating dementia, based on these recommendations, is playing bridge. Bridge is played in groups of four and therefore contributes to a person's social life. It also relies heavily on complex strategy and memorization, so it is mentally stimulating as well. As a bonus, bridge is especially popular among senior citizens, who are at the highest risk of dementia.

A standard game of bridge, however, is almost completely inaccessible to the visually impaired. A player may have up to thirteen cards in his or her hand at any given time, making large-print or Braille-marked cards difficult to read; additionally, the 2007 Annual Report from the American Printing House for the Blind states that only 10% of blind children are taught to read Braille, and those who go blind later in life may not learn it at all [4]. The presence of a dummy hand, which is face-up and may be anywhere on the table, also makes Braille impractical. A fully computer-based game is an option, but this all but removes the social aspect of the game; while online socialization is a possibility, no research has been conducted on its efficiacy in treating dementia. Our solution was to develop a game using RFID technology to let players hear the face values of cards.

The game uses an RFID system designed for video poker to track the cards as they move in and out of the players' hands. This information is sent to a computer running a Java program, which keeps track of the cards in the visually impaired player's hand, the face-up dummy hand, and the current "trick" or hand of play. Through an earbud or other audio output device, the visually impaired player hears the cards as they are played, and can also press buttons on a keypad to hear other information about the game. To identify their own cards without the use of Braille or other tactile identifying markers, the visually impaired player can scan them over a special identification antenna. This system gives the visually impaired player all the information they need to play a game of bridge with sighted players, without significantly changing how the sighted players play. I hope that this will eventually result in improved quality of life and lower incidence of dementia for visually-impaired people.

Product Requirements

A local blind woman who had played bridge before losing her vision offered to help us develop and test the product. In speaking with her, we identified a set of five requirements that the game absolutely must fulfill. The first of these is that the game must support play with others; there is little point in designing a game whose benefit rests on social interaction otherwise. The game must also be as unobtrusive and easy to learn to operate as possible, to reduce the extent to which it affects the sighted players. Since bridge is already a fairly slow-paced game, the game should provide play in real time, rather than slowing gameplay down any further than necessary. Finally, the tester was adamant that the game provide no advantage to the blind player; the only information available from the computer should be information that would otherwise be available visually.

Background

Introduction to Bridge

This is a brief overview of the gameplay and strategy of bridge, summarized from McLeod [7].

Bridge is a four-person trick-taking card game similar to Hearts. The players are referred to by cardinal directions (North, East, South and West), and each player is partnered with the player facing them; i.e. North and South are partners, as are East and West. The object of the game for each partnership is to win more tricks, or hands of play to which each player contributes one card, than the opposing partnership. However, bridge's complex rule structure makes this much more challenging than it might at first appear.

Bridge has two distinct stages of gameplay: the bidding stage and the playing stage. The bidding phase begins as soon as all the cards have been dealt. Beginning with the dealer, each player proposes a trump suit and a number. Cards of the trump suit automatically beat cards of any other suit in a trick, giving players with many cards of that suit an advantage in play. A player may also choose to bid "notrump," in which case there is no trump suit. The number corresponds to the number of tricks, minus six, the bidder thinks their team can win. For example, a bid of two means the bidder's team must win eight tricks or suffer point penalties. Bids must be made in ascending order of suit (from lowest to highest, the suits are clubs, diamonds, hearts, spades, notrump) and number. For instance, if the first bid was "1 Diamond," the next player could bid "1 Heart" or "2 Clubs," but not "1 Club."

During the bidding phase, a player may also choose to pass, or not to make a bid. The bidding phase ends once three consecutive players have passed. The last bid made becomes the "contract" for the round, and the player who first bid the winning suit becomes the "declarer." The declarer's partner also becomes the "dummy." Once the first card is played in the playing phase, the dummy player will arrange their cards faceup so that they are visible to the other players. The dummy player does not actually play during the playing phase; rather, the declarer plays the cards out of the dummy hand.

Play begins with the player to the declarer's left. Each player plays a card into the trick, proceeding clockwise around the table. The player who starts the trick also determines the suit of the trick by the suit of the first card played. Each player must play a card of this suit, if they have one. If not, any card in the player's hand may be played, including a trump card. The winner of a trick is determined by which player played the highest-ranked trump card, or which player played the highest-ranked card of the trick's suit if there are no trump cards. This trick is counted as "taken" by the winner's team, and the winner then begins the next trick.

After thirteen tricks have been played, the number of tricks won by each team is added up. If the declarer's team failed to make the number of tricks specified in the contract, the opposing team wins an "undertrick" point bonus for the number of tricks the declarer's team failed to make. For instance, if the contract was 4 Hearts and the declarer's team only took eight tricks, the opposing team would receive undertrick bonuses for two tricks. If the declarer's team did make their contract, they receive points for every trick taken, as well as an additional bonus for any tricks taken above the number required by the contract. After a round is scored, another may be bid and played; games are typically played until one team attains a preset number of points, or until a certain number of rounds have been played.

Needless to say, the game of bridge is highly social. In addition to the conversation that takes place during any group card game, partners who play together frequently will often develop bidding "conventions"--sort of a secret code to let each other know generally what suits their hands are strong or weak in. For instance, a bid of 1 Notrump after a bid of 1 Spades might mean "I have a good hand in spades," while a pass might mean "my hand is weak in spades." This means that partners who socialize outside of the game have an advantage over those who do not, since they have the opportunity to set up these bidding conventions ahead of time. Partners who regularly play together will also get a sense for each other's playing style, allowing them to predict each other's actions without explicitly discussing strategy, so there is a clear advantage to staying with one partner.

Bridge is also a very intellectual game. Before play even starts, the bidding phase is extremely intellectually challenging. There is a complex system for bidding in which each card or group of cards is worth a certain number of points, and it is generally accepted that a player must have a certain total number of points to make higher bids. This is much more complicated than simply having one or two high cards in a particular suit. Suppose a player has only the ace, four, three and two of spades. While it might superficially seem like a good idea to bid spades because of the ace, after the ace is played the player actually has very little power in the trump suit. If one or both of the opposing players has a lot of spades, they can simply outrank the player's trump cards, rendering them useless. On the other hand, if the player's partner also has a strong hand in spades, it might pay to bid that suit--but in order to know this, the player must remember the bidding conventions previously agreed upon with their partner. Paying attention to the other team's bidding might prove useful as well, since this allows the player to guess to some degree what the opposing players' hands look like--if they bid progressively higher in spades, one can probably guess that their hands are strong in that suit and that, consequently, bidding spades might be a bad idea after all. The player must absorb and retain a great deal of information to even know what to bid. It is clear from the very start that this is not a simple game.

The playing phase offers even greater intellectual challenges. The first player's choice is especially crucial, since they play before the dummy hand is revealed. All this player has to work with is the information they gathered during the bidding phase. This is especially important because the winner of the previous trick begins the next trick, so teams will often win several consecutive tricks. With only thirteen tricks in a game, there is not much room for error when it comes to winning tricks. The first card can make or break the game for that player's team.

Once the dummy hand is revealed, it adds another layer of complex strategy to the game. The other players now know exactly what cards are in at least one other player's hand, and taking this into account is important for both teams. On the dummy's team, the declarer must effectively use the dummy's cards to their advantage. For example, suppose the bid is 2 Spades, the declarer has the king, queen and six of spades, and the dummy has the ace, jack and ten. The declarer could play the king or queen, but then they would either be wasting a high card in the dummy's hand by outranking the jack or ten, or wasting a high card in their own hand by letting the dummy's ace outrank their own card. It is far better to play the six from their own hand and choose the dummy card based on what the intervening player does.

The defending team must also take the dummy hand into account when deciding what to play. Obviously, if the dummy has a high card in a particular suit, playing a lower card in that suit is self-defeating, but the strategy goes deeper than that. If the dummy is out of cards in a particular suit, playing a card from that suit gives the dummy an opening to play a trump card, automatically winning the trick unless the starting player's partner happens to have a higher trump card. But this might also work to a player's advantage if they have some trump cards themselves and can afford to sacrifice a few lower-ranked cards in the suit the dummy hand has no cards in. By depleting the dummy's supply of trump cards, the player might be able to win a few subsequent tricks with their own trump cards. Whether this is advantageous, however, is yet another complex decision.

Of course, there are also strategical challenges that do not directly involve the dummy hand. Obviously, none of the other players has direct information about the cards in the defending team's hands, but a shrewd player might deduce some information by noticing what cards the defending team plays. The obvious example is noticing when a player plays a card that is not of the trick's starting suit. This means that that player has no cards remaining in that suit, which can be either good or bad news for the opposing team depending on what suit it is and whether that player has cards left in the trump suit. It is also possible to guess what cards a player has based on what rank they play when they do follow suit, since players will usually play either the lowest card that will win the trick, or their lowest card in that suit if winning the trick is not possible. A player playing a six of hearts when the trick already contains the queen and five of hearts might suggest that the six is their lowest heart and that, consequently, the low hearts are in another player's hand; similarly, a player playing an ace in the same situation suggests that they do not have the king of hearts. It is also possible for a player to mislead the opposing team if they have a long string of consecutively ranked cards in the same suit. In the first example above, if the player not only has the six but also the four, three and two of hearts, they might try to trick the other players into thinking they do not have the low hearts by playing the six. Experienced players will be prepared for this as well, however, and will react accordingly. It should be obvious by now that the opportunities for strategizing in this game are virtually endless. It is easy to see why bridge games are typically played very slowly--they require quite a lot of thinking from the players.

There are other card games, however, that combine both the social and intellectual aspects of bridge. One might wonder, considering bridge's complex strategy and the difficulty of translating its various rules to code, why bridge in particular was chosen for the project. The obvious answer is that the blind client was an intrepid bridge player before losing her sight and specifically wanted a bridge game, but there is more to it than that. Bridge was a popular game in the 50's and 60's, and is thus especially well-known among senior citizens. As previously mentioned in the introductory chapter, a large proportion of the visually impaired are over the age of 65. Given that the aim of this project was improving quality of life for visually impaired persons, it made sense to choose a game many of them would already be familiar with rather than attemping to teach them a new one; it is also fairly simple to learn the basics of bridge, so that even those who are not already proficient at the game can pick it up quickly. Bridge is also, due to the presence of the dummy hand, much more intellectually challenging than many other games of its type. This makes it an engaging activity for the visually impaired and increases its benefit in preventing dementia. The final reason for the choice is that bridge's bidding phase encourages social interaction beyond the limits of the game, through bidding conventions such as those mentioned above. Again, this maximizes the game's usefulness for dementia prevention.

There is far more to bridge than can be described in a few pages. It is superficially simple to learn, but can take years to fully master. However, this background information is enough to explain the reasons for choosing bridge in particular, its benefit to players, and its unique social and strategic dimensions. With this in mind, we can move on to explaining the technical details of the project.

Introduction to RFID

This section provides a brief overview of the basic concepts involved in RFID. It summarizes the information available in Bonsor [5].

RFID, or radio-frequency identification, technology is ubiquitous in most people's daily lives. Some uses are obvious and easily recognizable to the layperson. The EZPass transmitters that enable drivers to pass through tollbooths without stopping use RFID to transmit the driver's information for billing. When a pet is taken home from a shelter or pet shop, it is now customary to offer the new owner an RFID chip to be placed under the animal's skin; in the event that the animal gets lost and its collar or tag comes off, this allows animal control officers to recognize that the animal is a pet and to easily identify the pet's owner. RFID is also used to track packages as they are shipped, to manage retail inventory, and to identify the owner of a passport to prevent fraud. Other uses are not so obvious; for example, surgical sponges are now commonly equipped with RFID chips to allow the surgeon to quickly determine, by waving a wand over the patient's body, whether a sponge has accidentally been left inside the body cavity. RFID-equipped police badges are even available to prevent people from posing as police officers by stealing badges; the police department can identify which officer a badge should belong to by reading that information from the tag. Chances are that if something needs to be quickly and uniquely identified, an RFID device is available or will soon be available for that purpose.

RFID requires three basic components: an antenna, a reader (also called a transceiver or decoder), and a tag (also called a transponder). The tag is the chip embedded in the pet, EZPass device, or whatever else needs to be RFID-enabled; at the very least, it contains a serial number unique to that tag, though it may contain up to 2 KB of other data. As one might expect given that these tags are sometimes injected with a needle, they can be made extremely small. The antenna emits radio waves to search for tags within range and to read from or write to them. The range can be as small as one inch or as large as 100 feet, depending on the antenna's power, the power (if any) provided to the tag, and the specific frequency used. When a tag enters the antenna's range of detection, it detects the antenna's radio signal and transmits the information stored on it back to the antenna. The antenna then passes this information on to the reader, which in turn decodes the information and passes it on to a computer for processing. A simple diagram of this relationship is provided in Figure 1 below.

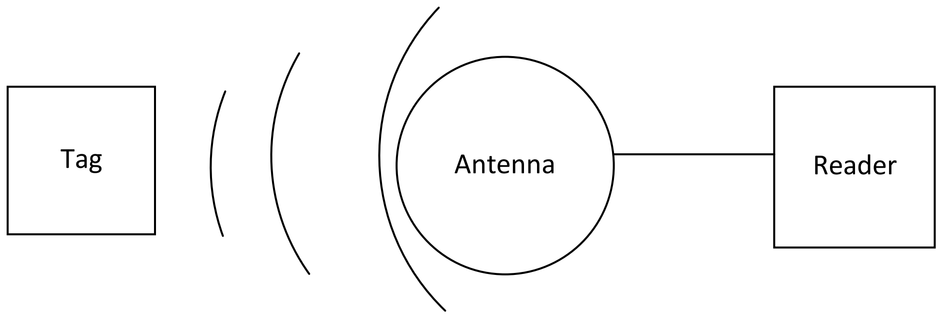

Fig. 1. A diagram of a basic RFID setup.

There are two types of RFID tags: active and passive. An active tag, as the name suggests, actively transmits information to the reader by emitting its own radio frequency. Obviously, active tags require power (usually provided in the form of a battery) to function. Active tags are more expensive, but can be read from a greater distance; for this reason, they are used in applications like EZPass where the tag needs to be read from far away. Passive tags do not have their own power source. Rather, they get power from the antenna's radio frequency signal. Due to the lack of a battery, they are usually both cheaper and smaller than active tags, but need to be closer to the antenna to be read. Tags such as those used on pets or surgical sponges are often passive, since the antenna is expected to be quite close to the tag.

While RFID is a useful technology and has in many ways revolutionized the modern world, it does have its failings. The largest of these is interference. Interference is an unwanted radio frequency signal occuring within the range of an RFID antenna. This is not necessarily another RFID device; cell phones, remote controls, and even light dimmers can emit unwanted radio frequency signals. When interference occurs, the antenna's signal may not be correctly recognized by the tag, or the antenna may receive incorrect or invalid information from the tag. Managing interference is one of the greatest challenges of designing an RFID system.

Project-Specific Hardware

The hardware used in this project was developed by Andrew Milner, using an RFID system manufactured by SkyeTek [8]. It consists of a reader, up to eight antennae, and a set of passive RFID-tagged playing cards. In our case, we used five antennae, one at each position to identify cards as they are played, and one to allow the visually impaired player to identify their cards before playing. The reader connects to a computer by means of a USB connection.

Figure 2 shows the hardware configuration we used. Here, there are four antennae corresponding to the players (labeled N, E, S, and W). There is also a fifth ID antenna, to be used by the blind player in identifying their own cards before playing them. The antennae are all connected to a central reader, which is also connected to a computer. A simple diagram of the hardware setup is shown below.

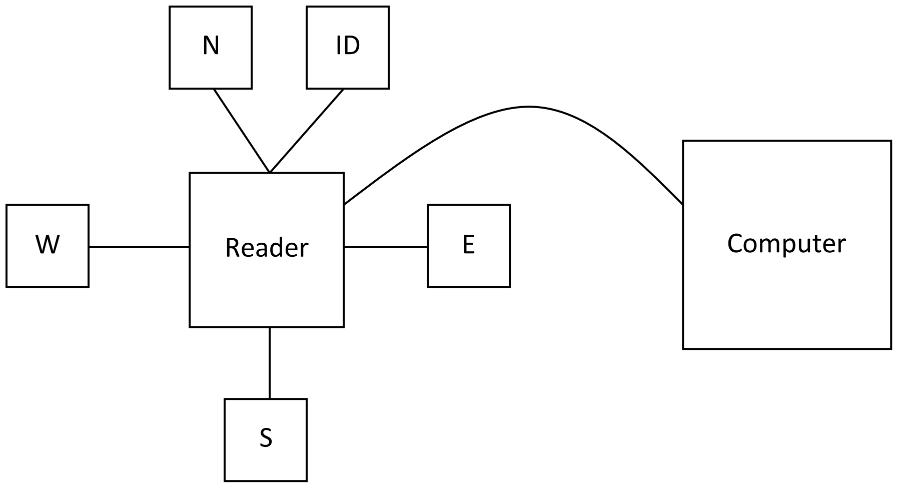

Fig. 2. A diagram of the hardware used in this project. Visible are the five antennae, the reader, and the USB connection to the laptop.

This particular hardware was selected for several reasons. First, it was relatively inexpensive, costing only $1400 total (not including the computer). This was important in order to make it practical for businesses such as home care facilities to actually obtain the device. It was also packaged as a kit, so we knew that the reader, antennae and cards were all compatible. The system also comes with a C# library for interfacing; this enabled us to easily write a server that retrieved information from the system and passed it on to a Java program running on the laptop. The antennae's short range, only about two inches, made interference a nonissue. Finally, the hardware draws power from the computer rather than from an external power source, minimizing the amount of wiring and therefore setup involved.

Related Works

The hardware used in this project was not originally intended as an assistive technology device for the visually impaired. Rather, it was developed to facilitate the playing of video poker. In video poker, RFID technology is used to track the cards and display them on a video feed, mainly for the benefit of spectators. One can deduce from the use of video that this game is not only not intended for visually impaired players, but is not even accessible to them. In my research, I was unable to find any other examples of RFID being used in games for the visually impaired.

Most of the games available to visually impaired players are either computer-based or make use of an existing video game system. Some use a tactile interface to interact with the player, such as VI Bowling, which uses a Nintendo Wii controller and its vibration capabilities to guide the player to the correct spot [9]. Others use an entirely audio-based interface, including the free games available from Spoonbill Software, which provide information on the state of the game using simulated speech [14]. Still others combine both types of interfaces; "Rock Vibe," an accessible version of the popular video game "Rock Band," uses both auditory and tactile cues to prompt the player to take different actions [1]. While Spoonbill Software does provide several audio card games such as cribbage and solitaire, I was unable to find any accessible versions of bridge, even as a completely computer-based game. In addition, none of these games support play with others, even online, limiting their potential benefit in preventing dementia to their level of intellectual challenge.

While the concept of using RFID in games for the visually impaired may be new, RFID is used in numerous assistive technology devices for the visually impaired. Students at Central Michigan University have developed a "Smart Cane" that uses RFID tags embedded in the sidewalk to help visually impaired people navigate, and hope to develop a mobile robot that will use RFID to act as a seeing-eye dog [12]. An RFID device to allow the visually impaired to easily identify buses is also in development [11]. Combined RFID and QR-code devices also exist to assist visually impaired people when shopping [6].

Procedure

Interaction with Hardware

As shown in Figure 3, there are five antennae: one for each player, and an identification antenna for the blind player to identify their own cards. At the beginning of the game the blind player's cards are scanned over their antenna before bidding takes place. After bidding, the first card is played and the dummy's cards are scanned in over the dummy antenna. Play then proceeds as outlined in the Introduction to Bridge section, with each player passing their card over their antenna when playing it. A photo of the hardware setup is provided below.

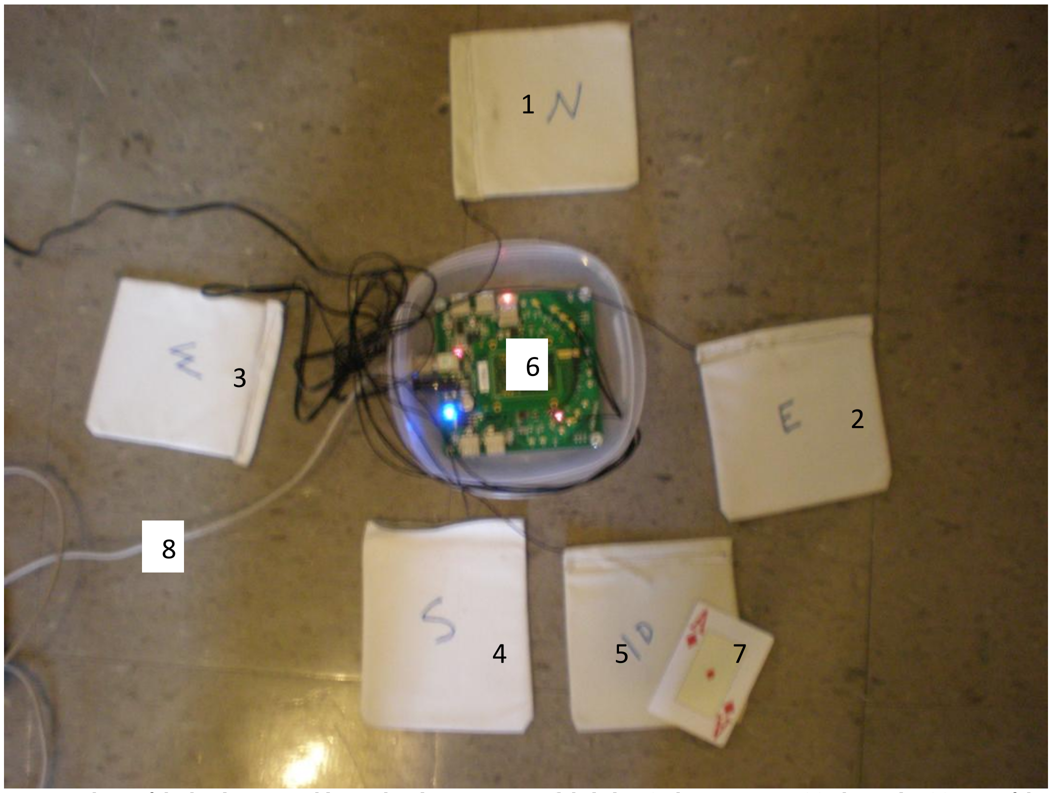

Fig. 3. A photo of the hardware. Visible are the player antennae (labeled 1-4), the ID antenna (5), the reader (6), one of the cards (7), and the USB connection to the computer (8).

The SkyeNet hardware comes with very little documentation on the included C# libraries, only detailing how to perform a few basic functions. Initially we considered having the RFID reader send the tags to a database, from which the Java program would retrieve and interpret them, but it quickly became clear that this would be unnecessarily complicated. Thus we settled on having the reader and the Java program communicate through a server coded in C#.

The server first opens a port on the computer and begins sending information coming from the reader to that port, where it waits to be retrieved by a client. Meanwhile, a Java program handling all details of actual gameplay connects to the same port and listens for tags being transmitted. When a tag is received, it interprets the data and acts accordingly. A diagram of the flow of data through the system is provided in Figure 4.

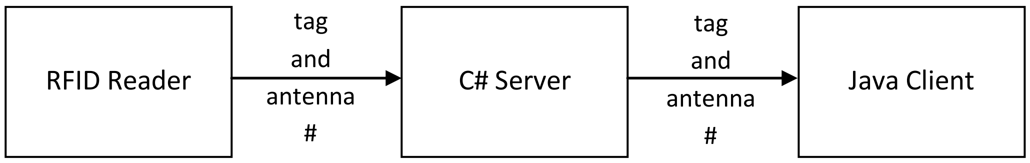

Fig. 4: The initial architecture of the hardware-to-software interface. Arrows represent the flow of data.

As with all RFID systems, each card has a unique identification tag, which is transmitted to the USB connection in the form of raw bytes. Having the server parse these bytes as a string and send them through the port to the Java client was simple enough. The challenging part was that because every card had a unique tag, the tags were not consistent across decks--for instance, the king of clubs in one deck will have a different tag from the king of clubs in another deck. Since we had two decks, this left us with 104 different tags to map to their respective card identities, a number that could become larger still if support for more decks needed to be added. Coding this as a simple array would make later referencing awkward, as well as making the code less readable for future developers. The problem was solved by use of a hash table. Each tag and its associated card can be accessed by a simple lookup function, resulting in more understandable, elegant code.

Programming Game Rules

The first and most important issue was to make sure that the game responded appropriately when one of the antennae detected a card. It needed to know which antenna the card was found on, as well as whether that antenna corresponded to a player or to the identification antenna. To accommodate this, a CardListener interface was created, intended to represent an antenna in the virtual representation of the game. The interface provided a cardFound method to be used when a card was detected on the corresponding antenna (initially there was also a cardIDed method, intended to be used when a card was found on the identification antenna, but as you will soon see this was redundant). There were two implementations of this interface: the Hand class, obviously corresponding to a player's hand, and the CardIdentifier class, corresponding to the identification antenna. We also built an AntennaHandler class, which managed the flow of information from the hardware to the Java client and contained all the CardListeners as fields. When the AntennaHandler received a tag, it would parse the tag to determine which card had been detected and which antenna number it had come from (appended to the tag by the C# server). It would then match the antenna number to one of the CardListeners and call that object's cardFound method. From there, the CardListener could appropriately inform the rest of the program about the new card. A UML diagram of the architecture described in this paragraph is shown below.

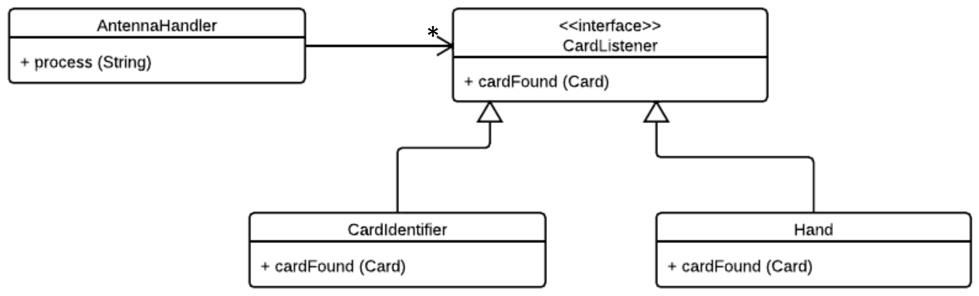

Fig. 5. Architecture of the CardListener interface.

Internally, the cards were represented as instances of a Card class. This was one of the simplest parts of the game to program, since Java provides the capability for enumerated types: essentially, classes containing little more than a list of named integer values. The compareTo method is automatically created for these classes and, as one might expect, returns the difference between the integer values of the two instances being compared. By representing a card's rank and suit as enumerated types, the cards could be ordered by both rank and suit within a hand with a minimum of actual codewriting. This cut down on extraneous code and made the rest of the game easier to write.

Due to the heavily audio-centric nature of the project, it was necessary to provide a way for the sound file associated with each card to be retrieved easily within the code. This was achieved by standardizing the names of the sound files. Each file was given a simple two-character name corresponding to the card's rank and suit: for instance, the file for the three of hearts was designated "3H.WAV." All the sound files were clustered inside a /sounds/cards folder inside the project directory. It was therefore a simple task for a Card, when created, to determine the path to its own sound file. This was then saved as a variable within that Card, which could easily be retrieved whenever the sound was needed.

The structure of the game as a whole was handled by the Game class. This class contained all four Hands and the CardIdentifier as fields. It also managed such information as the current trick, trump suit and game state. As mentioned earlier, bridge has three distinct phases: a dealing phase, a bidding phase, and a playing phase. The bidding phase involves only human interaction and can thus be ignored by the computer, but a distinction still needed to be made between dealing and playing.

Again, the problem was solved with the use of enumerated types. A GameState type was created, with the values "DEALING" and "PLAYING." The game held a field of this type and would check it each time a card was detected, adding the card to the player's hand or to the current trick as the game state dictated. Originally this was a simple boolean indicating whether or not it was the playing phase. An enumerated type was ultimately chosen because there are actually two sub-phases of the dealing phase; one before the first card is played in which the cards are dealt, and one after the first card is played in which the dummy hand is revealed. Using an enumerated type made it easy to add a third "FIRSTCARD" state.

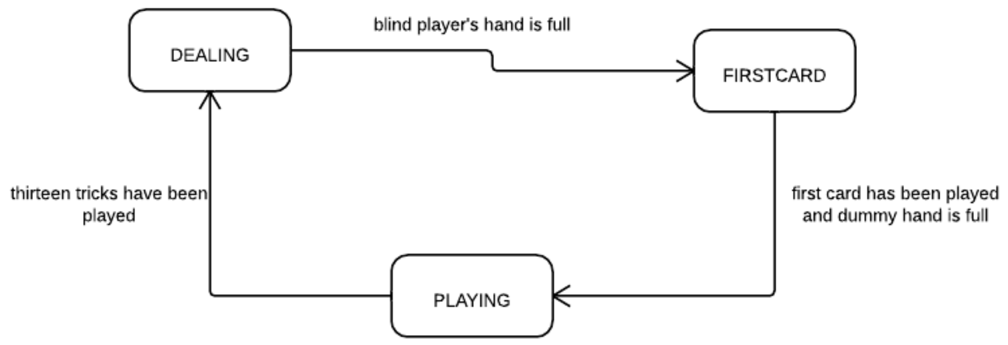
DEALING
blind player's hand is full
FIRSTCARD
first card has been played and dummy hand is full
PLAYING
thirteen tricks have been played

Fig. 6. State diagram of the game.

The task of informing the Game about cards detected by the antennae was implemented by making the Game class a listener to the Hand and CardIdentifier classes. When the Hands and CardIdentifier were created during the Game's initialization, the Game passed itself to their setListener methods. The Hands and CardIdentifier would then save the Game as a field within their own instances. When a card was detected, the instance would call the Game's cardFound or cardIDed methods as appropriate. The Game would then analyze its own internal state and which antenna the card had come from and respond correctly.

Several challenges appeared as we tried to implement the dealing phase. Initially the mechanics of this phase seemed obvious enough; simply add a card to the blind player's or dummy's hand when it was detected on either of those antennae, and play the appropriate sound file. The first issue with this was that the antenna polls constantly for tags within range. Therefore, a card held over the antenna for even a short length of time was detected more than once. With our original setup, this resulted in not only duplicates of the same card being added to the hand, but also the card's sound file being played repeatedly.

The solution was to ignore cards during the dealing phase if they already existed in the hand they were being scanned into. In the Hand class, the addCard method was modified to only add the passed card to the hand if the hand did not already contain it; if it did, the card was ignored. Originally a similar fix was implemented in the Game class to prevent the sound from being played; however, during testing it became clear that to have the game simply not respond to a card being scanned in any event was confusing to the player. Therefore this change was later reverted.

Another challenge was that due to the limitations of the hardware, only one antenna can be listening for tags at a time. A C# command must be invoked on the computer to change the antenna that is currently listening. To get around this, we originally had the C# server send commands to the hardware at a fixed rate, having it cycle from one antenna to the next constantly throughout the game. However, in practice this became unweildy. If the commands were sent more often than once every 200 milliseconds, the hardware could not process the commands fast enough to keep up with the computer, and the server would become flooded with commands and crash, forcing both the server and the Java program to be restarted. If we observed this limit on the rate of commands, the server would take at least a full second to cycle through all five antennae, potentially forcing the players to wait that long to play a card. This was obviously unacceptably slow in gameplay. Another failure of this model was that since in bridge it can only be one player's turn at a time, three of the antennae were essentially unnecessary at any given moment. Having the server cycle through these three unused antennae wasted not only time but CPU cycles, making the entire program less efficient.

The solution lay in the fact that the Java program, by which the actual gameplay portion of the product was controlled, also controlled when the turn shifted from one player to the next and kept track of whose turn it currently was. If we could find a way for the Java program to send commands to the C# program to be relayed to the hardware, we would only need to cycle between two antennae: the antenna corresponding to the current player, and the ID antenna. The cycling could therefore be done much slower and still be fast enough not to disrupt gameplay. This would solve both the problem of the server crashing and the slow speed of gameplay when the server cycled through all five antennae.

Doing so proved a bit of a technical challenge. The C# program and the Java program were originally written with information only flowing from the C# side to the Java side and not in the other direction. Having the Java program send information to the C# program was simple enough; we only needed to have the Java program write the commands to the output stream of the port. The difficult part was having the C# server listen for these commands and pass them along to the hardware. In the end, the C# server constantly read information from the port's output stream, watching for a single-letter code that identified which antenna to switch to. Whenever a single character was received, the server would interpret it and send the appropriate command to the hardware. A diagram of this revised setup is shown below.

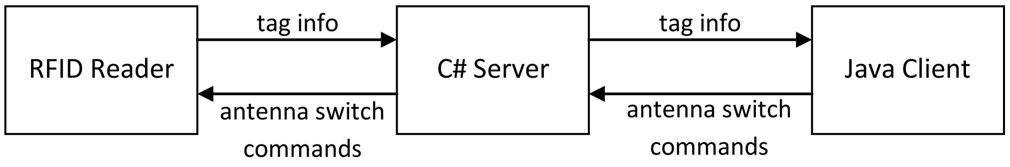

Fig. 7. The revised architecture of the hardware-to-software interface.

One other hardware-related challenge bears mentioning. When a card is held over an antenna that is currently listening for tags, the antenna detects it each time it polls for tags; several times per second. Under the vast majority of gameplay circumstances, this is not a problem, since cards will not be added to the blind player's or dummy's hand if they are already there during the dealing phase, and the antenna listening for tags changes as soon as a card is played during the play phase. But since the rules of bridge dictate that the first card must be played before the dummy hand is revealed, the dummy's turn comes as soon as the dummy player is finished scanning his or her cards. The system would erroneously interpret the last card scanned into the dummy hand as the card the dummy player intended to play.

The situation was remedied by inserting a pause() method into the Hand class. The method would set a boolean variable, "paused," to true, initialize a timer that would set the variable to false after a short time period, and then start the timer. Meanwhile, an if statement was added to the cardFound() method that would cause the Hand to ignore all incoming tags if the paused variable was true. Finally, the Hand was modified to call its own pause() method whenever a card was detected while the hand was full. This had the net effect of making the dummy hand ignore all cards detected for a short period after the last card in the dummy hand was scanned. With some experimentation, the interval was made short enough that it did not interfere with gameplay speed, while not being so short that it was ineffective

Interaction with the Blind Player

To facilitate the blind player receiving information about the game state, a simple USB number pad was used. When turned upside-down, the pad had four keys across each of the bottom two rows--one row for both the dummy and the player's hand, and one key for each suit. Since in bridge the player is often limited as to what suits he or she may play, separating this information by suit was a natural choice. The pad also had two long, easily-identifiable enter and zero keys. A spoken tutorial was mapped to the enter key, while the function to repeat the last thing spoken by the computer was mapped to the zero key.

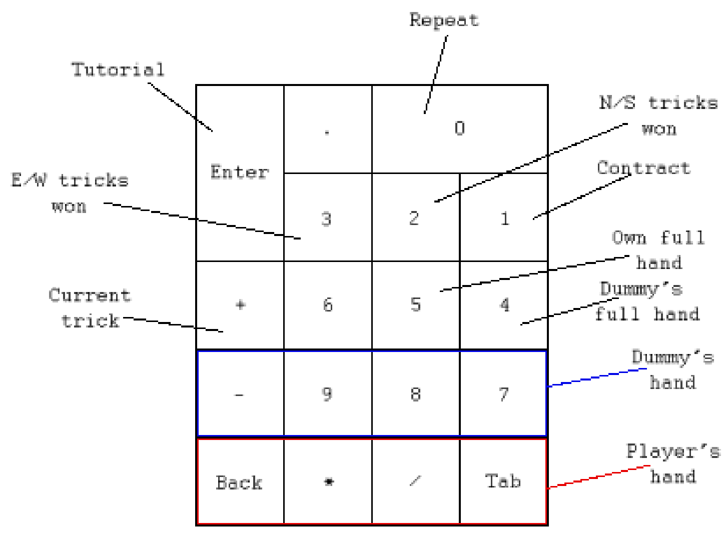

Fig. 8. The layout of the keypad.

Using a keypad presented a unique challenge, however. In a word processor, holding down a key results in an endless string of the same character. In our project this translated to an endless stream of signals to play certain information, causing the program to repeat the requested information over and over even after the key was released. This was problematic since in testing, the tester's first instinct was to hold the key down until the sound began playing, sending multiple signals even during that brief time frame. The obvious solution was to have repeated signals from the last key pressed ignored, but there was a possibility that the player might actually want to hear that information twice in a row.

The solution was to have the program "forget" the last key pressed after a certain length of time, similar to what was done with scanning the dummy hand. When a signal was received, the keycode was stored and all signals with that same keycode ignored until the next key press. At the same time, a timer was started that would reset the stored keycode to zero after a preset period of time. As with the dummy hand, this required some experimentation to get the timing right, but it solved the problem once it was perfected.

Evaluation

As with any product intended for direct end-user interfacing, user evaluation was integral to the development of the bridge game. While we did test the product amongst ourselves, due to our knowledge of the game's inner workings, we would sometimes unconsciously test in a way that minimized the potential for bugs to arise; additionally, we never actually tried to play a game of bridge, just tested specific functionality. User error, which needed to be corrected for if the product was going to be used by the public, was also not addressed by developer testing. Along these lines we scheduled semi-regular evaluations with a local blind bridge player.

The blind tester, along with three other sighted testers, would attempt to use the product to play a game of bridge. The testers commented throughout the game on any difficulties they were encountering, as well as any suggestions they had for improvement. Meanwhile, a note-taker would record these comments and any other bugs or problems she personally noted with the game. The evaluation would also be videotaped for later study; the developer would subsequently transcribe these tapes, noting the length of time spent playing compared to that spent trying to get the game to work.

Initially, as might be expected, the game was barely usable. Bugs and problems mentioned in the Procedure section were a constant distraction, and more time was spent correcting for the myriad sources of error than actually playing bridge. However, as development progressed and the problems found in evaluations were taken into account, the product gradually became less frustrating and more functional. By the time of the third evaluation in December, the product was nearly in a workable state, and the conversation around the testing table turned more toward the rules and strategy of bridge than toward the problems with the product.

First Evaluation

Date: August 20-21, 2011

Participants: Carol Lerner (blind), Arthur Lerner, Barbara Lerner, Richard Lerner

Evaluation length: 3 hours total

Observations

This was the first time the game had been tested in any real capacity. Until this evaluation we had been in contact with the blind tester for feedback during the product's development, but she had not actually used the product itself. Additionally, since the product had spent only a few months in development at this point, the developers themselves had not tried to play a full game of bridge with the product. Needless to say, a variety of problems and suggestions arose during the evaluation.

The first and most obvious problem was the slowness of the hardware. We had anticipated that the hardware would be slow in detecting cards, because at this point the server was cycling through all five antennae as described in the

Procedure section. What we had not anticipated was the drastic extent to which card detection would be delayed. The product sometimes took up to ten seconds to detect a card after it was placed on the antenna. With thirteen cards in a hand, this meant that even in the best case a hand could take up to a minute and thirty seconds to scan in at the beginning of the game. In practice scanning was of course much slower, at one point taking up to five minutes. By the time all the cards were scanned in, the blind player had frequently forgotten what they were, necessitating use of the keypad and further slowing down the game.

The server setup also proved to be unstable during testing. In just over three hours of testing, the server crashed a total of four times. Each crash required the entire program to be restarted, further slowing the game as the blind player's hand then had to be scanned in again. This turned out to be related to the previous problem; the constant cycling was overloading the server so that it either took an inordinately long time to send information to the Java program, or failed entirely.

There were several instances during the evaluation in which the game code did not work correctly. The game did not test whether a hand already contained a particular card before adding it to the hand. This resulted in the same card being added repeatedly if it was detected more than once by the hardware, while cards that were scanned later (after the game thought the hand was full) were not added at all. The game also required the dummy hand to be scanned even if the blind player was the dummy, causing the cards to be unnecessarily scanned twice in this instance. There were also oversights in coding simply due to the developers' incomplete understanding of the game of bridge. As mentioned in the Introduction to Bridge section, the first card must be played before the dummy hand is revealed, but the game expected the dummy hand to be scanned first, which would expose it to the first player. There were also instances in which all players passed during the bidding phase, requiring the cards to be re-dealt, but there was no way to reset the game state without restarting.

Problems also arose with the keypad that the blind player would use to get information about the game. First, sound playback was unacceptably slow, with long pauses between some sounds. This was because the sounds were being read on a timer based on the length of the longest sound, introducing empty space when the sounds were not that long. The keypad also read the entire name of the card (e.g. "Three of hearts") when this was not necessary, since each key corresponded to a single suit in the player's hand; it was suggested that the keypad read only the number of cards of that suit, and then the rank of each card. The repeat button did not work correctly, as it would re-add the sound it had just played to the list of previous sound files without clearing the list first, resulting in the same sound being played repeatedly if the button was pressed more than once in a row. A final, minor problem with the keypad was the lack of a button to read out the blind player's or dummy's full hand.

Changes Proposed

- change the server to respond to antenna change commands instead of continuously cycling between antennas

- change keypad to ignore multiples of the same key press within a short time frame
- fix repeat button
- eliminate need to scan dummy hand when blind player is dummy
- support all players passing
- improve speed of sound playback
- add command to read entire hand

Quantitative Data

Frustration time (time spent fiddling with the hardware or software)	51:57
Time spent discussing the game's hardware or software	20:46
# hardware-related restarts	5
# software-related restarts	2
Avg. time between restarts	19:13
Avg. time to scan a hand	3:36
Time spent playing bridge	0:00

Second Evaluation

Date: November 4, 2011

Participants: Carol Lerner (blind), Arthur Lerner, Barbara Lerner, Richard Lerner

Length: 3 hours total

Changes Made

- repeat button moved to 0 key
- hand modified to ignore duplicates during dealing phase
- game now ignores multiples of the same keycode
- full hand button added

Observations

This evaluation was distinct from the last in that some time was actually spent playing bridge with the device. In the previous evaluation, the product was so buggy and crash-prone that progress was rarely made beyond the dealing phase. Several of these bugs had been fixed based on observations from the last evaluation, and this time it was possible to play out several hands. There were some lingering problems, however, and new issues also arose.

The biggest fix made between the last evaluation and this one was the modification of the card-listening method to ignore the last card that had been scanned on a given antenna. This meant that a card would no longer be added to the hand twice if it was held over the antenna longer than a few seconds. The Hand class was also modified so that it would not add a card to the hand if it was already there, so the duplication issue of the last evaluation was not an issue in this release.

Changes had been made to the hardware that would supposedly allow it to cycle faster than the previous limit of 200 milliseconds per antenna. This resulted in a significant decrease in the time required to scan a hand, and made gameplay more efficient as well. However, the issue of commands overflowing the stack and causing the server to crash was not entirely resolved, as the server did fail once during testing. This problem would not be completely resolved until the new server setup was implemented.

Several minor suggestions had been implemented since the last evaluation. In particular, the repeat button was moved to the 0 key, and the key-listening code was modified to ignore repeated signals from the same key to avoid repetition. Full-hand buttons for both the visually impaired player's hand and the dummy hand were introduced.

Changes Proposed

- cards should be read in "3 clubs: x, y, z" format
- play next sound immediately after previous one is finished
- restart button for the entire game

Quantitative Data

Frustration time (time spent fiddling with the hardware or software)	33:49
Time spent discussing the game's hardware or software	21:26
# hardware-related restarts	2
# software-related restarts	5
Avg. time between restarts	18:21
Avg. time to scan a hand	1:40

Time spent playing bridge	19:33

Third Evaluation

Date: December 9-10, 2011

Participants: Carol Lerner (blind), Arthur Lerner, Barbara Lerner, Richard Lerner

Length: 3 hours total

Changes Made

- repeat button fixed
- cards now read in "3 clubs: x, y, z" format
- server now responds to antenna change commands

Observations

By the time of this evaluation, the new server setup had been implemented. The server now cycled between the ID antenna and the antenna of the player expected to play next, and only changed when directed by the Java program. As a result, there were no hardware-related restarts during this evaluation. However, the more functional the hardware became, the more opportunity there was to discover problems with the software. The changes to the software that the new hardware setup demanded also introduced their own problems.

The sounds had been rerecorded to play in the format suggested in previous evaluations. The suit buttons now only read the number of cards in that suit, followed by the rank of each card; the full hand buttons read the same information as all four suit buttons, in sequence. This did speed up playback considerably. However, the sounds were still played on a timer, as threading complications were making the resolution of this issue difficult. Therefore, playback was still unacceptably slow in a gameplay situation. The button to restart the entire game was also still not functioning correctly, although this did not explicitly come up during the evaluation.

Several new problems were also identified. In particular, the keypad would sometimes stop responding after the game had been running for some time, even though it had worked as expected earlier. This was determined to be due to a flaw in the way multiples of the same keycode were ignored; the timer would sometimes be stopped before it fired the event that caused the game to "forget" the last keycode. The ID antenna would also sometimes behave as though it were the blind player's playing antenna, and the other way around. This was because at this point, the game decided whether a card was meant to be identified or played based on the currently active antenna. The active antenna would often change before the game checked it, and it would misidentify a card scanned on the ID antenna as one scanned on the player antenna, or vice versa. In response, the CardIdentifier class was modified to call a different method in the Game class than the one the Hand class used. This ensured that the game would always know, accurately, which antenna had actually been used.

A problem not directly related to either the hardware or the software also arose. As the game became more functional and more time was spent playing bridge, the tester had to scan and sort her cards more often. It became clear that her method of sorting--holding the cards between her fingers so that they were separated by suit--was not optimal. She would sometimes drop some or all of her cards, forcing her to rescan them in order to organize them again, as well as exposing them to other players in the process. This did, however, draw attention to another user interface concern: the game would not respond during the dealing phase if a card that was already in the hand being scanned was scanned again. This was confusing when the tester had already scanned some of her cards and did not know whether silence indicated that the card had already been scanned, or a problem with the product. Therefore, the game was modified so that a card scanned twice during the dealing phase would be spoken again, but not added to the hand again. The root issue of the tester not being able to hold her cards effectively remains unresolved, but will probably be easily repaired with the use of one of the commercially-available card holders already manufactured for the visually impaired.

Some miscellaneous concerns were also raised during the evaluation. At one point, a card came too close to an active player antenna and was inadvertently scanned. Since there was no way to undo the last card played and the card that was accidentally played was an illegal card, the entire game had to be restarted. The need for an undo button that would take back the last card played was noted.

The bidding user interface was also mildly confusing, as bids in bridge are stated with the number first and the suit second, but the suit selection screen came first in the game. This was easily reversed by changing their positions in the GUI class's screen array.

Changes Proposed

- undo button
- swap bid number and bid suit GUIs
- use a card holder to prevent cards from being dropped

Frustration time (time spent fiddling with the hardware or software)	19:51
Time spent discussing the game's hardware or software	10:05
# hardware-related restarts	0
# software-related restarts	9
Avg. time between restarts	17:11
Avg. time to scan a hand	1:42
Time spent playing bridge	43:13

Fourth Evaluation

Date: April 12, 2012

Participants: Carol Lerner (blind), Arthur Lerner, Barbara Lerner, Richard Lerner

Length: 90 minutes total

Changes Made

- sounds now played one directly after the other
- bid number and suit GUIs swapped

Observations

For this evaluation, the sounds had been fixed so that they were now played one directly after the other, rather than on a timer. This shortened the time needed to read out information to the visually impaired player considerably. The bid number and bid suit GUIs had also been swapped so that the number came first, as is customary in bridge bids.

Several new issues were identified. The ID antenna now did not work properly. When a card was scanned over it, it detected the tag (as evidenced by debugging output) but did not speak the card. Most troubling was that the game was now taking an extremely long time to switch from listening to one antenna to another, and this delay seemed to increase the longer the game had been running. The game was stopped from switching between the ID and player antennae in an attempt to solve this, but it did not help.

Old issues also reappeared in this evaluation. The most salient of these was the fact that the tester had trouble with the physical cards. In this evaluation she dropped her entire hand before she was finished scanning, causing her to miss scanning two cards. With the ID antenna not working, it took a significant amount of time to find which two these were. The issue in which the last card scanned into the dummy hand was immediately played also recurred, though this is likely due to having the dummy antenna pause for too short a time after scanning and should be easily fixed.

Changes Proposed

- fix lagging problem on antennae
- fix ID antenna
- fix dummy hand's problem of playing last card scanned

Quantitative Data

Frustration time (time spent fiddling with the hardware or software)	18:03
Time spent discussing the game's hardware or software	7:26
# hardware-related restarts	1
# software-related restarts	2
Avg. time between restarts	25:05
Avg. time to scan a hand	1:55
Time spent playing bridge	22:14

Metrics for All Evaluations

	Eval. 1	Eval. 2	Eval. 3	Eval. 4
Frustration time (time spent fiddling with the hardware or software)	51:57	33:49	19:51	18:03
Time spent discussing the game's hardware or software	20:46	21:26	10:05	7:26

# hardware-related restarts	5	2	0	1
# software-related restarts	2	5	9	2
Avg. time between restarts	19:13	18:21	17:11	25:05
Avg. time to scan a hand	3:36	1:40	1:42	1:55
Time spent playing bridge	0:00	19:33	43:13	22:14

Issues Remaining

The largest issue that remains to be solved is the lag when the game needs to switch antennae. At present I believe this is due to the large amount of output generated by the C# server, but I will need to obtain a new version of Visual C# to test this.The problem in which the ID antenna does not function also persists, and testing is required to find the source and correct it. The problem in which the dummy hand plays the last card scanned is the least crucial; most likely, the time for which the dummy hand is paused simply needs to be adjusted. The most minor change suggested, the "undo" button, should be fairly easy to implement. The issue of the tester not being able to hold her cards properly also remains to be solved.

Conclusions

Visually Impaired UI Needs

Sighted people innately rely on vision to gain information about the world they live in, perhaps more than any other sense. It is one of our most natural and most immediate methods of interacting with each other and with our environment. Without sight, a person's perception of the world is radically different than it is for a sighted person. Information must come through channels that sighted people do not rely on as heavily, and this introduces unique difficulties when a sighted person must design a user interface for a blind person.

Initially, I had naively assumed that as long as all necessary information about the game was provided in an audible format, the user interface would be adequate. I was far more concerned with the technical details of the game than with the way the visually impaired person would interact with it. In retrospect, I can see that my priorities were extremely confused. Of course the functionality of an assistive technology product is important, but equally important is making the interface accessible to the user.

Vision is a very time-efficient sense. Sighted people can take in a great deal of information in seconds just by glancing at something. When information is provided audibly, the opposite is true. Audible information must be slow enough to be understood, but fast enough to be efficient. I did not fully realize this when I began this project, and it affected the user interface several times during testing. Initially the sounds were too slow; they were played on a timer rather than one directly after the other, there was a significant amount of "dead space" in the sound files themselves, and I included unnecessary information that slowed down playback further. When I repaired these issues, the sounds were too fast and ran together, making it difficult for the visually impaired player to understand. The middle ground was to add a separating sound between distinct pieces of information, dividing the sounds into discrete groups without inserting too long of a pause between them. It should be obvious from this process that simply providing information non-visually is not enough to design an interface for a visually impaired user. Information must also be provided in the right way; as quickly as possible while still being easily understood.

Issues also arise with the way the visually impaired player requests this information, in the cases in which it is not automatically provided. The fact that the information is there means nothing if the user cannot access it. In my project, this came up with the use of a keypad to allow the visually impaired player to access information on demand. I would often catch myself saying things like "Press the six" to explain the keypad to the visually impaired player. To a visually impaired person, there is no six; there is the key on the third row from the bottom, second from the right. In order for a visually impaired person to understand a user interface, it must be explained in their terms and not those of a sighted person.

The organization of the keypad also proved to be important. Initially the functions were mapped to keys rather haphazardly, not necessarily grouped with other related functions or organized in an intuitive way. For a sighted person this is not a significant problem, as they can keep a written list of keys and their functions close at hand. For our visually impaired tester, this made all but a few keys on the keypad entirely useless. If she could not guess where a function might be from the location of others, she could not access it without listening to a several-minute-long tutorial explaining the keypad's layout. By shifting a few keys so that their location was more intuitive--for instance, putting the numbers of tricks won by each team on adjacent keys--the interface became much more accessible. One area in which the keypad's layout did work was in organizing the cards by suit. Since suits have an order in bridge, the blind tester was able to instantly deduce which key on the bottom two rows corresponded to which suit simply by placing one finger on each key. Which row corresponds to which hand is also intuitive, as the player's hand is always closer than the dummy hand (unless the player is the dummy, in which case both rows give the same information). This was immensely helpful to the tester and is something I feel should be kept in mind when designing other audio card games for the visually impaired.

Visually impaired people also derive a great deal of information by touch. However, when touch fails, interacting with the physical world can prove difficult. One unique difficulty in this regard that I did not anticipate was in the blind tester's ability to interact with the physical cards. In one evaluation session, she complained that the cards were "too slippery" and that it was difficult for her to pick them up off the table after they were dealt. She also had trouble identifying which side of a card was which, sometimes holding the cards face-out. One adaptation that might mitigate both factors would be to imprint a slight, but noticeable texture on one side of the cards, while leaving the other side smooth. This would give the visually impaired person something to grip when picking up cards, while also allowing them to differentiate between the two sides. If the texture was shallow enough, it would not interfere with shuffling.

When playing cards, most people organize the cards in their hand by suit. This is especially important for the visually impaired, as they have no other way to approximate where in their hand a certain card might be. Our tester also faced unique challenges in this regard. Only able to identify one card at a time through the ID antenna, she tried to keep them separated in her fingers, and frequently dropped them. Using a card holder would prevent this difficulty. Another adaptation I feel would be helpful would be to emboss tactile symbols, simpler than Braille but still distinct, on each card to identify its suit. This would allow the visually impaired player to organize the cards by suit without having to spend time scanning each one.

Future Work

Once the game's existing issues are solved, there is a great deal of potential for future work. One of the most obvious is testing the game with a wider pool of visually impaired testers. At present, the game has only been tested by a single blind player. This leaves us with no information as to how the game will be received by those who have some functional vision, as well as other blind players from different backgrounds. Feedback from many more different testers is needed to make this game usable for everyone.

The question of whether the cognitive load on the visually impaired player is simply too great also needs to be answered. Our current tester was adamant that no extra help be provided to the visually impaired player, but this may not be the case for all people, particularly those who already have some level of dementia. For these users, the strain on their memory added by not being able to see, combined with the memory-intensive nature of bridge, may make the game too difficult to be enjoyable. To accommodate both opinions, options might be provided at the beginning of the game to turn certain helpful features on or off.

Another possibility is making the game accessible to more than one visually impaired player. At the moment, the hardware can only support a single visually impaired player; adding a second would require, at the very least, introducing another audio channel for the second visually impaired player's information. However, given the statistics on visual impairment mentioned in the Introduction section, the chances that two visually impaired people might want to play bridge together are fairly high. Adding support for more than one visually impaired player would certainly make the game attractive to a wider range of users.

Alternative user interface devices might also be considered. The keypad is useful for blind players because of its highly tactile nature, but not everyone using this product will be blind. For those who have some functional vision, a touchpad device similar to an iPad might prove useful. These players might also appreciate a separate visual display, in larger print than that provided for the sighted players.

References

[1] Allman, T.; Dhillon, R.K.; Landau; M.A.E.; Kurniawan, S.H. (2009). Rock Vibe: Rock Band® computer games for people with no or limited vision. *The 11th International ACM SIGACCESS Conference on Computers and Accessibility* (51-58). New York: ACM.

[2] Alzheimer's Association. (2011). 2011 Alzheimer's Disease Facts and Figures. *Alzheimer's and Dementia, 7(2):* 1-63.

[3] Alzheimer's Association. Brain Health. Retrieved from http://www.alz.org/we_can_help_brain_health_maintain_your_brain.asp.

[4] American Printing House for the Blind. (2007). 2007 Annual Report. Retrieved from http://www.aph.org/about/ ar2007.pdf.

[5] Bonsor, K.; Keener, C.; Fenlon, W. (2007, November 5). How RFID Works. Retrieved from http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid.htm.

[6] López-de-Ipiña, D.; Lorido, T.; López, U. (2011) In J. Bravo, et al. (Ed.) *Ambient Assisted Living - Third International Workshop* (33-40). Springer.

[7] McLeod, J. Bridge: rules and variations of the card game. Retrieved from http://www.pagat.com/boston/bridge.html.

[8] Milner, A. Home Page. Retrieved from http://videopokertable.net.

[9] Morelli, T.; Foley, J.; Folmer, E. (2010). Vi-bowling: a tactile spatial exergame for individuals with visual impairments. *The 12th International ACM SIGACCESS Conference on Computers and Accessibility* (179-186). New York: ACM.

[10] National Eye Institute. (2004). Prevalence of Blindness Data Tables [data file]. Retrieved from http://www.nei.nih.gov/eyedata/pbd_tables.asp.

[11] Noor, M.Z.H.; Ismail, I; Saaid, M.F. (2009). In M.N. Taib, et al. (Ed.) Bus detection device for the blind using RFID application. *5th International Colloquium on Signal Processing & Its Applications* (247-249).

[12] O'Connor, M.C. (2009, September 14). Michigan Students to Develop RFID-enabled Robotic Guide Dog. *RFID Journal.* Retrieved from http://www.rfidjournal.com/article/view/5214/1.

[13] Plassman, B.L.; Langa, K.M.; Fisher, G.G.; Heeringa S.G.; Weir D.R.; Ofstedal M.B... (2007). Prevalence of dementia in the United States: the aging, demographics, and memory study. *Neuroepidemiology, 29(1-2):* 125-32.

[14] Spoonbill Software. BG Games. Retrieved from http://www.spoonbillsoftware.com.au/bggames.htm.