Grand Valley State University
ScholarWorks@GVSU

Technical Library | School of Computing and Information Systems

2014

Say-it: Design of a Multimodal Game Interface for Children Based on CMU Sphinx 4 Framework

Mashael Alsulami
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/cistechlib

Recommended Citation
Alsulami, Mashael, "Say-it: Design of a Multimodal Game Interface for Children Based on CMU Sphinx 4 Framework" (2014). *Technical Library.* Paper 184.
http://scholarworks.gvsu.edu/cistechlib/184

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.

Say-it: Design of a Multimodal Game Interface for Children Based on CMU Sphinx 4 Framework

By
Mashael Alsulami
April 2014

Say-it: Design of a Multimodal Game Interface for Children Based on CMU Sphinx 4 Framework

By
Mashael Alsulami

A project submitted in partial fulfillment of the requirements for the degree of

Master of Science in

Computer Information Systems

At

Grand Valley State University

April 2014

Dr. Yonglei Tao	April 2014
Professor	**Date**

Table of Content:

Abstract:

Nowadays, computer games are involved in a child's education as tools for learning or practicing some academic skills. However, most educational games are designed without any considerations about children with motor system difficulties. Thus, the main benefit of multimodal interfaces is to allow for inclusive design, which will enable children with motor disabilities to use the same applications other children use. Since focus and concentration are major skills for children in the learning process, many physical games can be performed with assist from parents or teachers to practice these skills. This project aims to explore the effectiveness and the use of multimodal applications as computer-based exercises by implementing a multimodal system that offers an inclusive design for three interactive games to practice the skill to focus. In this project, Say-it is implemented as a functional multimodal prototype to demonstrate the value of multimodal interfaces in the education of young children. It is designed to provide the advantages of both current physical games and computer exercises and make these exercises available for children with wide rang of abilities. Also, Say-it can be considered as an experimental prototype to explore the performance of using CMU Sphinx 4 framework as an underlying speech recognition tool for the application. The results of using Say-it show that multimodal games can be designed using the existing speech recognition technologies, such as CMU Sphinx 4 framework. Also, using different input modalities in the proposed prototype makes the games more enjoyable and challenging for children.

INTRODUCTION

According to Daniel Goleman, focus is the hidden driver of excellence. Focus and concentration are defined as thinking skills that allow people to start working on a certain task without any delay or distraction and pay a complete attention until the task is completed. Children are the most group of people that suffer from many different forms of the inability to concentrate. Furthermore, Some children are visually oversensitive which make them easily distracted by any smile while others have auditory processing problem that make it hard for them to distinguish different noises from a noise. [5]

The goal of this project is to emphasize the value of multimodal systems in a child's education. In this project, a multimodal prototype that offers three interactive games is implemented to help practicing the concentration skill for children. Basically, Say-it provides two modes to interact with the application: speech-only mode that uses speech for inputs and graphical mode that uses mouse and keyboard for inputs. To implement the speech-only mode, CMU Sphinx 4 framework has been used as the underlying speech recognition technology. The three designed games are: memory game, counting game and noise game.

Unlike traditional matching memory games, Say-it implements a memory game that aims to help children to memorize certain items that are related to a

certain object. Essentially, the game starts by giving two different orders for two different customers and asks the child to remember the items for a given customer. A child can visualize the items for both orders, which are displayed as graphics with texts and then say or select the items of the given customer based on the selected modality.

Also, Say-it implements a counting game to help practicing the ability to focus on a sequence of numbers and catch the missing one. Basically, the system will provide a sequence of numbers with one number missing. A child needs to recognize which number is that after counting is finished. Finally, a traditional noise game is implemented in which a child needs to listen to a noise and recognize the noises in it.

The interaction of the three-implemented games can be done using speech-only mode or graphical mode. At any point of the game, a child can switch from the default mode, which is speech-only mode to the graphical mode, which enhances the usability of the proposed prototype. The interaction using speech modality is based on "Say what you see" principle where the commands need to be displayed to make them easy to remember by the users [10]. This command-based style requires a graphical interface to be effectively implemented [10].

Background and Related Works

The main goal of this project is to experiment with multimodal interfaces to demonstrate their value on a child's education. Say-it is implemented as a multimodal game for children to practice the skill to focus. These exercises can be played inside a classroom or at home without any help from somebody else. A child can take the advantages of using this prototype to explore working with multimodal applications.

In this section, a brief overview of multimodal interfaces is given including a discussion of what multimodal interfaces are and what the benefits of using and designing these type of interfaces.

An interface is defined as a multimodal interface when there is more than one modality to interact with the system [9]. Generally, the motivation of using multimodal interfaces is to simulate human-to-human communication and to make the interaction between the user and the system being used as natural and efficient as possible. In multimodal interfaces, a combination of speech, pen, touch and motion can be used. The most common combination that is used by many multimodal systems is speech and touch modalities [9]. Few systems combine more than two modalities due to coordination technical issues.

Several benefits of preferring multimodal systems rather than single modal systems have been discussed in the literature. Some of these advantages include accessibility, usability, flexibility and error recovery [1]. In terms of accessibility and usability, multimodal interfaces allow the application to be accessible and usable by wide rang of people with wide rang of capabilities. For instance, applications that use speech as one of their input modalities give free hands

users the same opportunity to use the application as normal users have. Also, switching modalities can be considered as an error handling strategy [2]. When the user has a choice to switch to another modality as a result of poor recognition or occurrence of long error sequence [2], the error recovery of the system will improve. Essentially, several studies show that multimodal interfaces are more flexible, effective and powerful than single modal interfaces for certain context, however; the challenging part is how to coordinate these different modalities in an efficient way and how to make switching between input modalities a smooth process for users.

This project explores the use of a multimodal application as an activity exercise for children to practice the skill to focus. This multimodal prototype allows children to have a control over the application by selecting the modality that they like and enjoy to interact with the system. This can indeed build their confidence in using computer programs in many contexts in their education. Also, many studies indicate that offering different modalities in computer games would improve the performance of the players [6]. In fact, When using different modalities in a computer, improving the performance of the game itself is not a goal but the goal is to make playing the game more enjoyable [6].

Program Requirements

Both functional and nonfunctional requirements were taken into consideration in the very early stages of designing Say-it. Also, general guidelines for designing multimodal interfaces were considered in the design phase.

In terms of nonfunctional requirements, effectiveness and error recovery were the main goals for the application as usability attributes. In order to achieve an overall effectiveness of the application, two aspects were considered. The first aspect was to make the voice-based interaction between the system and the user as natural and efficient as possible. The second aspect was to come up with a modality switching approach that is flexible and meets the user's expectations.

In order to make the voice-based interaction efficient, several design decisions were made, such as providing visual information on each screen of what the user said and what he/she heard. Visualizing this information reduced the pressure on the user to remember what have been said and increased the flexibility of the application. In terms of modality switching, there was a stop microphone button on each screen of the application to make it easy and obvious for the user to switch from speech input modality to a graphical one. It is worth mentioning that the default modality of Say-it was speech modality to demonstrate the benefit of using speech as an initial modality. According to myth#4 in [9], speech is usually the primary modality in many multimodal applications since it is the most usable input modality that can be performed by many users. Furthermore, all this information and all of these system prompts were designed in such a way that makes this information clearly understood by users of different ages. Short

commands, slow synthesized speech, clear font and different colors were used for that purpose.

In general, effective speech-based interaction and the flexible modality switching approach drove the application to be efficient and effective as a human-computer interaction system.

In terms of the error recovery, speech recognition errors were handled in two ways. First, every speech input needed to be confirmed by the user before considering it a correct input and that was done by a simple question after each input, such as *"Do you mean memory game?"* These questions were represented as visual texts where the user could read them and as synthesized speech where the user could hear them. Moreover, these confirmation questions enhanced the accuracy of the application by avoiding wrong detection that may have occurred as a result of surrounding noises, quality of the microphone or wrong pronunciation. Second, using modality switching as an error handling strategy [2]. As mentioned earlier, speech was the default input modality in the application, however; sometimes users repeated what they wanted to do several times and the system doesn't recognize what they said well. That frustrated them and discouraged them to continue playing. In this case, multimodal systems in general and Say-it in specifics allows users to switch to another modality when performing a task with speech modality becomes problematic for the users. In addition, rapid re-prompts mechanism to error recovery was used with friendly instructions to encourage the user to try again when recognition was failed, such as *"Sorry I can't understand you, Please repeat that again!"*

In terms of functional requirements, the basic functions of the main menu, memory game, counting game and noise game will be described in this part. Each screen in the application had a text area and a stop microphone button as static elements. The text area allowed to visualize the system prompts and what the user said while the stop microphone button allowed switching from speech input modality to graphical input modality at any point of time. The main menu allowed the user to choose one of three games using either speech or graphical (mouse/keyboard) modality. Using speech modality, user could say any command that was written in any button to perform its action. As mentioned earlier in this paper, command-based style was considered which is based on "Say what you see" principle. When the user said the name of any of the games, the label of the game's name will change its color to red and a text version of the synthesized speech will be displayed in the text area followed by what the user was saying. After the user confirmed his/her choice, a description about the selected game was given.

The main goal for the implemented games is to provide computer-based exercises for children to practice their skills to focus.

Memory Game:

The basic idea of the memory game is to practice memorizing a set of elements that are related to a certain object. At the beginning of the game, a screen that contains an order of a customer called Bob (See appendix figure 7) is displayed. After visualizing the photos and the item's names of his order, the user needs to say *"Next"* to move to the next order in case of speech-only mode or press the Next button in case of the graphical mode. The final screen in this game is a complete menu that contains all the items that have been ordered by both customers. The user is asked to try to remember what was Bob's order for example. After all four elements have been selected using speech input modality; the system asks the user if the selected items are what he/she said. If the user says "Yes", the result will be shown in the next screen, otherwise; the user will be asked to select the items again. The domain of acceptable words in this game is restricted to only 8 words. Those words are the names of the items that have been ordered by both customers.

Counting Game:

The main goal of the counting game is to practice focusing on a sequence of numbers and recognize the missing one. The game starts when the user either says, "start counting" in the case of speech input modality or presses on "start counting" button in the case of graphical input modality. The system counts from 1 to 15 and miss one number. The user needs to recognize which number is missing from that sequence. If the user says the correct missing number, the system prompts the user to confirm the answer. If the user confirms his/her answer, the result will be shown in the next screen, otherwise; the system prompts for another answer. The domain of acceptable words in this game is restricted to only digits.

Noise Game:

The motivation of playing this game is to help recognizing noises from a noise environment. The game starts when the user either says, "start listening" in the case of speech input modality or presses on "start listening" button in the case of graphical input modality. After the audio is done, the system prompts the user to recognize three noises from that audio file. After each input, the focus will move to the next input field. When all the input fields are filled, the user needs to say, "done" to move to the result screen.

Implementation

To implement this multimodal prototype, several tools were required. First, CMU Sphinx 4 framework was the chosen framework to implement the speech recognition part. Second, FreeTTS, which is an open source text to speech library, was used to implement the synthesized speech as the system prompts. Third, WindowsBuilder design tool was used to design the GUI for the prototype.

Finally, I have used Java as a programming language to code the functionality part of my prototype.

CMU Sphinx 4 is an automatic open source speech recognition framework that is developed in Carnegie Mellon University and entirely written in Java and imported to other programming languages such as python [3]. It provides two main features: speech recognition and audio transcription [7]. CMU Sphinx is a common speech technology that has been investigated and explored by many researchers in the human computer interaction community. Moreover, it is known to be a flexible and powerful framework due to its high performance [3]. CMU Sphinx 4 is one of the Sphinx family members and the last updated version of them [4]. My first experience with CMU Sphinx 4 was implementing a prototype of a small application in CIS623 and I get interested in the technology at that point. My goal of working with CMU Sphinx 4 is to adopt a multimodal prototype using Java as a programming language, however; there are very limited speech technologies that worked well with Java. That reason makes me willing to experiment with CMU Sphinx 4 to investigate the performance of using it in a multimodal interface with voice-based interaction.

The main alternative that I considered during the implementation phase was Microsoft System.speech namespaces. These namespaces mainly designed for developers who are interested in designing speech-based applications in Windows Vista and Windows 7 [8]. This constraint on working only on Windows versions was one of the main reasons that drives me to look for more generalize option that can serves any operation system using different programming languages.

The core component of CMU Sphinx 4 framework architecture is the recognizer, which is the speech engine for CMU Sphinx 4 [3]. It can be demonstrated as an xml configuration file that consists of three modules: front end module, decoder module and linguistic module. The modularity feature of CMU Sphinx 4 architecture makes it a powerful framework because each of those modules can be configured without affecting the source code or any other module. In term of the front-end module, the source of speech input is specified in this module.

The speech input source can be a microphone or any data source format, such as .wav or .au [3]. For the purpose of my prototype, the microphone was used as the speech input source. In other context, CMU Sphinx 4 is capable of transcribing an audio file into text. On the other hand, the decoder module is responsible to perform a search using the input that are coming from the front-end module through the search graph to find the best path and generate the recognition result [3].

Finally, the linguistic module consists of three sections: acoustic model, dictionary and language model. CMU Sphinx 4 supports different acoustic modules, such as US English acoustic models for microphone and telephone speech. Developers may use an existing dictionary for their application (which is recommended) or create their own one. The most common dictionaries that are used are: WSJ (Wall Street Journal) and TIDIGTS. Generally, CMU Sphinx 4

provides these two dictionaries by default in its downloadable package. For the purpose of my prototype, WSJ was used. The final piece of the linguistic module is the language model. Basically, any speech-based application needs either a language model or a grammar file. The language model is a file that contains all possible sequence of words while a grammar file contains a predefined structure of words that follow certain rules such as JSGF grammar. The JSGF grammars are specified to match the speech input with the words or phrases that are in the grammar file. Furthermore, the location of the .gram file needs to be determined in the xml configuration file. It can be in the same package as the application or in another package. [3][4]

Specifically, in Say-it prototype, each one of the games had its own JSGF grammar file. Also, a more general grammar file was created to serve more general purposes, such as the confirmation grammar file. It is worth mentioning that in the configuration file of confirmation grammar, I had to change the value of *allowingMissingWord* property from false to true to allow unformed words to be acceptable, such as *"yup"* and *"nop"* because when testing this prototype by some children, I observed the use of these words instead of "yes" and "no".

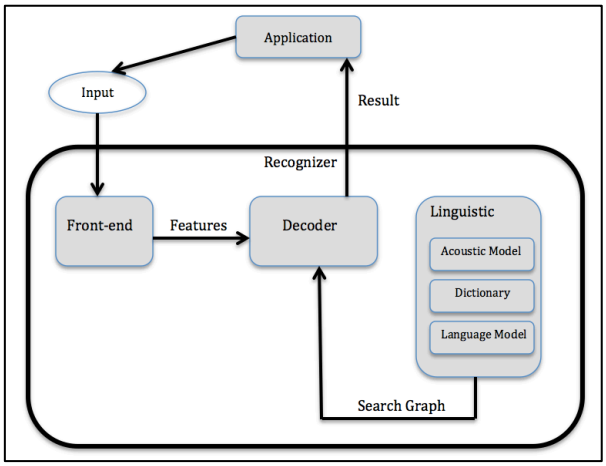

Figure 1: CMUSphinx Architecture

To get started with CMU Sphinx 4, three related objects need to be instantiated. First, the configuration manager object accesses the xml configuration file for a specific grammar. Second, a valid recognizer object needs to be looked up using the configuration manager object. Finally, a microphone has to be resolved as a resource for the application. Essentially, each recognizer object has its own copy of the microphone object. After all three objects are created, a method called "recognize" will be called to recognize the speech from the user. Testing the result is done programmatically to execute the proper event.

The challenging task in the implementation stage was to find an efficient and easy way to switch from speech input modality to graphical input modality and that because CMU Sphinx 4 does not have start/stop recognition methods. In order to stop the recognition to switch to the graphical mode, I had to do some workaround by calling stop recording method from the microphone object and as a result of that the recognizer will still recognize but it hears nothing. Then, I

faced another issue where the microphone still had some data even when I called stop recording method, which caused some confusion to the user. Thus, I needed to clear the microphone from all data that are possibly being there and then called the stop recording method. The use of these two methods seemed to solve the issue with switching modality. Figure 2 illustrates how the switching between speech and graphical modality was done.

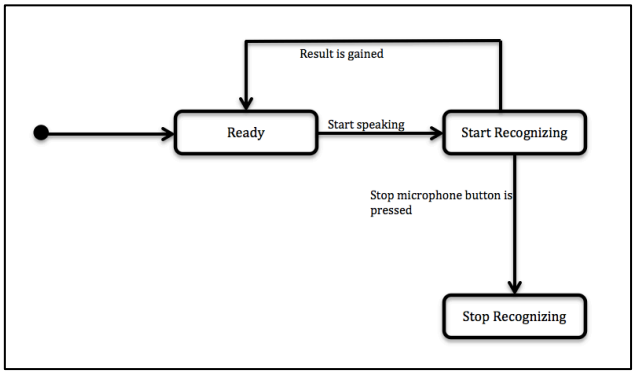

Figure 2: A state transition diagram for speech-only mode

A final implementation aspect that I needed to deal with was the synchronization between the content of the text area that included the visual text of the system prompts and user's responses and the recognizer. The issue was that when the recognizer started the recognition, I was not able to change the text in the text area to make it compatible with what the user said or heard. To solve this issue, I used SwingWorker thread to dynamically update the content of the text area to visualize the information during the recognition. SwingWorker thread allowed the text area to update its content while the recognizer was still running. The benefit of such behavior is to allow updating the text in the text area without waiting until the recognizer was done recognizing. Updating the content of the text area could be considered as a small task that did not have to wait for the recognizer which could be considered as a long task to finish to start running.

Results, Evaluations and Reflections

The main objective of this project was to create a functional multimodal game system that used speech and graphical (in form of mouse/keyboard) modalities to interact with the system. The result of this project included implementing Say-it as a functional multimodal prototype that had a main menu and three interactive concentration games for children. The prototype explored the multimodality by providing two main modes of interactions: speech-only mode and graphical mod using mouse/keyboard as input modalities. In addition, visual texts were provided and displayed in each screen in form of a text area that contained what the user said and heard during the speech-only interaction mode. Also, in the case of speech-only mode, visual clues were provided in form of changing colors or gaining focus on a certain element on the screen.

Different colors were used to emphasize certain things. For example, when the user switched from speech-only mode to graphical mode, the color of the text in the text area became red to get the user attention and to emphasize the current state of the system. Another example is when the user selected a game from the main menu; the color of the label that indicated the name of the game changed to red to get the user's attention.

The usability evaluation of this prototype was conducted with 7 potential users, 3 of which were children from 9 to 12 years old and 4 were adults' users. All the participants know how to use a computer and they are familiar with computer-based games. An oral introduction about the system and the purpose of it was given for each of the participants. Also, I asked them about their expectations of what they will see before using the system to compare it with their later feedback. Each participant was asked to choose only one game to play without any constraints on the modality being used. In addition, external microphone was provided to gain better accuracy.

The results showed that all participants from both groups (adults and children) tried to complete the game using speech-only mode at the first time. However, only 25% from the adults group was able to complete the selected games using speech input modality while the other 75% switched to the graphical mode at some point during the game. On the other hand, it was interesting to find out that the children group was more eager and patient to complete the selected games using speech input modality. Moreover, two of the three children were able to complete the game successfully using speech input modality.

For adult group, accuracy was the main factor to switch to the graphical mode. Another reason was that in the speech-only mode, the flow of the games was too slow comparing with the graphical mode. However, the children group found that the speech-only mode was more fun than the graphical mode and made the game more challenging. Also, the results showed that the counting game was the most favorite games for both groups while the noise game was the least favorite one. Another thing that was noticeable by both groups and I was not aware of was the voice of the synthesized speech. All the adult users found it noisy and not so friendly while all the children thought that it was a funny robot voice.

Also, positive comments about the overall design were given and most of the participants liked the use of the colors and the graphics in the user interface.

Generally, I was not satisfied by the overall accuracy of the prototype. Several factors may affect the accuracy such as surrounding noise, the quality of the microphone or the user's pronunciation [6]. However, the accuracy had been improved a lot when I used an external microphone. The confirmation questions after each speech input was a way to overcome the issue of the accuracy and wrong recognition issue, however; these questions slowed down the flow of all the games. At the end, it was a matter of a trade off between the speed and the accuracy. Finally, I really found working in this project was a great learning

experience in terms of the technologies that I used and the experience that I had in engaging children as potential users in the evaluation of my prototype.

Conclusion and Future Work

The evaluation of this multimodal prototype showed the beneficial use of this kind of applications as a computer-based exercise for children. The result of using Say-it indicated that speech recognition applications could be implemented and developed using existing speech recognition frameworks. In particular, this project demonstrated that CMU Sphinx 4 framework is an ideal choice to develop voice- based Java applications that can work with any operating system. In general, Say-it was a fun multimodal computer-based exercise for both adults and children to practice the skill to focus. Furthermore, the usability evaluation with potential users showed that the designed games could be used as tool to make practicing the skill to focus an enjoyable process. The two different modalities that were implemented in Say-it prototype were ways to make the application usable by wide rang of people with wide rang of capabilities.

Since there was only limited time to implement this prototype, a more advanced version of it with many ideas could be implemented in the future to enhance its performance and functionality. Here are some of these ideas:

- The proposed prototype is a desktop application that based on CMU Sphinx 4 framework. It is a good idea to implement it as an android application and bring it to portable devices to server more users anywhere and at anytime. This will require using pocketsphinx framework as underlying framework for speech recognition to allow activating audio object in the target device.
- All the implemented games are one level games, it will be more interactive to have several levels for each game started from easy to difficult.
- In case of implementing several levels for each game, it will be more intuitive to save sessions with each user to allow for later completion. This will require creating a database to store levels that are completed and ones that are left for each user in each game.
- In each screen in the application, there is a stop microphone button to switch from speech modality to graphical modality. It will be more reasonable to have a start microphone button to switch back to speech modality. However, this is not implemented in this version because each screen has only one task to accomplish which makes changing modalities back to speech is an expensive process especially with CMU Sphinx 4 framework.

Finally, designing a multimodal interface is a challenging, ongoing process that needs a lot of effort to simulate the interaction between the application and the user as natural and smoothly as possible.

Bibliography

[1] Baljko, M. (2005). The Contrastive Evaluation of Unimodal and Multimodal Interfaces for Voice Output communication aids. *ACM*. Retrieved March 18, 2014, from http://dl.acm.org.ezproxy.gvsu.edu/citation.cfm?id=1088463.1088515&coll=DL&dl=ACM&CFID=304928293&CFTOKEN=19962465

[2] Bell, L., Boye, J., Gustafson, J., & Wirén, M. Modality Convergence in a Multimodal Dialogue System. *Centre for Speech Technology, KTH*. Retrieved March 19, 2014, from http://www.diva-portal.org/smash/get/diva2:641612/FULLTEXT01.pdf

[3] Carnegie Mellon University, "Sphinx-4: A Speech Recognizer Written Entirely in the Java™ Programming Language", http://cmusphinx.sourceforge.net/sphinx4/, last accessed on February 4, 2014.

[4] Carnegie Mellon University, "CMUSphinx: Versions of Decoders", http://cmusphinx.sourceforge.net/wiki/versions, last accessed on February 4, 2014.

[5] Elias, M. (2013, October). Helping Students Develop the Skills to Focus. *EduTopia*. Retrieved March 19, 2014

[6] Gürkök, H., Hakvoort, G., & Poel, M. (2011, November). Modality Switching and Performance in a Thought and Speech Controlled Computer Game. *ACM*. Retrieved March 18, 2014

[7] Niyizamwiyitira, C., & Lundberg, L. (2013, October). Performance Evaluation and Prediction of Open Source Speech Engine on Multicore Processors. *ACM*. Retrieved March 18, 2014

[8] Microsoft, "System.Speech Programming Guide for .NET Framework", http://msdn.microsoft.com/en-us/library/hh361625(v=office.14).aspx, last accessed on March 18, 2014

[9] Oviatt, S. (1999, November). Ten Myths of Multimodal Interaction. *COMMUNICATIONS OF THE ACM*, *42*. Retrieved March 19, 2014, from http://www.kevinli.net/courses/mobilehci_w2012/papers/p74-oviatt.pdf

[10] Tchankue, P., Vogts, D., & Wesson, J. Design and Evaluation of a Multimodal Interface for In-Car Communication Systems. *ACM*, 314-321. last accessed on March18, 2014.

Appendices

Figure 3: The speech-only mode of the main menu screen

Figure 4: The GUI mode of the main menu screen

Figure 5: The speech-only mode of counting game screen

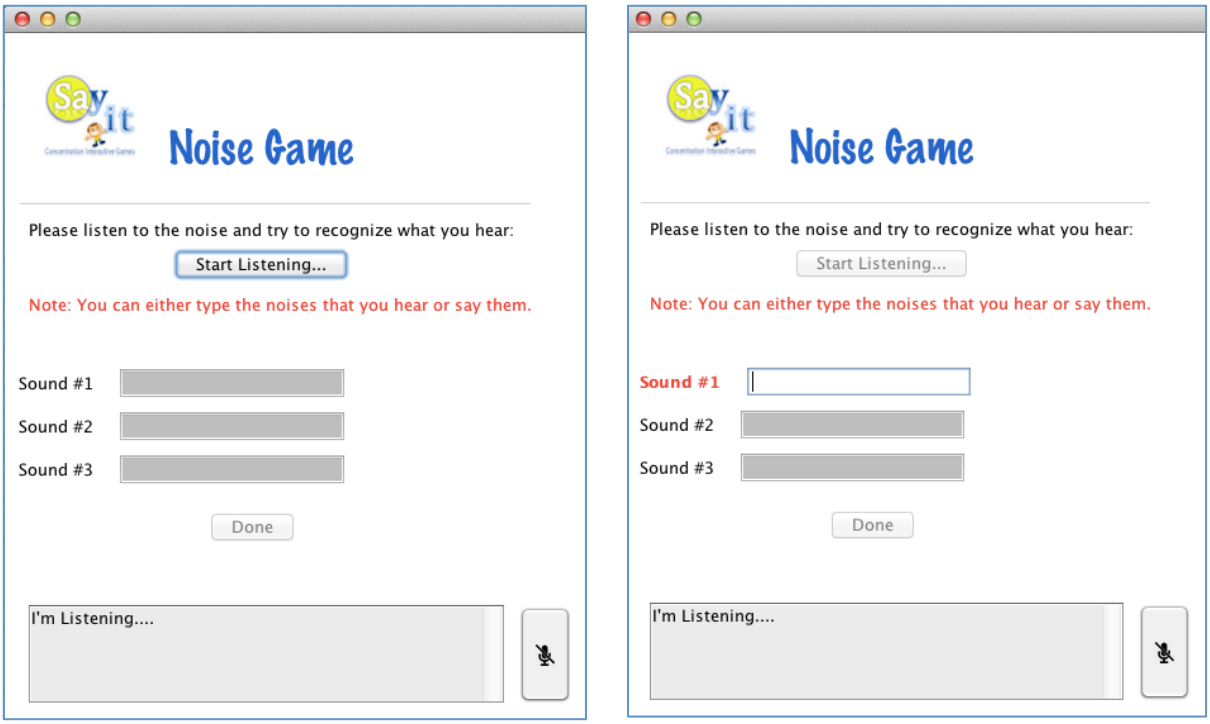

Figure 6: Screenshots from the noise game

Figure 7: Screenshots from the memory game