

A Multimodal User Interface for Public Information Kiosks

Roope Raisamo
Department of Computer Science
University of Tampere
P.O. Box 607 (Kehruukoulunkatu 1)
FIN-33101 Tampere, Finland
rr@cs.uta.fi

Abstract

Current information kiosks and automatic teller machines are mostly accessed by pressing large on-screen buttons that are displayed on a small touchscreen. This kind of interaction is suited for simple kiosk systems since their use does not require versatile interaction. When public information kiosks gain new markets and new applications are developed, the need for more expressive human-computer interaction grows stronger. As a solution we have developed a kiosk user interface framework called Touch'n'Speak that makes use of advanced touchscreen interaction techniques and speech recognition. This paper describes the system and several multimodal interaction techniques that are used in it.

1. Introduction

Automatic teller machines and other simple interactive computer systems are used increasingly in the modern society. We call them computerized public kiosks or simply kiosks. Some kiosks just respond to key presses while others present large on-screen buttons on a small touchscreen. An automatic teller machine is one of the simplest kiosk systems and has a specific predefined function.

Many companies provide information kiosks in their lobby area with which visitors can get to know their business and spend the waiting time usefully. These information kiosks can be based on conventional desktop computers equipped with a keyboard and a mouse, or on a computer that communicates with the user using a large touchscreen that is used for both input and output. Still, with these modern kiosks human-computer interaction is limited. Current kiosks may have good multimedia presentation capabilities, but the user can only use predefined buttons or a slow on-screen keyboard to give input to the system.

This paper presents a kiosk user interface framework

that provides many different techniques for touchscreen input and supports them with speech recognition. These input modalities can be used separately or in parallel. The choice is left to the user. The system presents the resulting information in a Web browser and allows browsing by touching or speaking.

The paper is organized as follows. First, we present some previous work on interactive kiosks. Next, the Touch'n'Speak user interface is described using a restaurant information system as an example. This is followed by a detailed description of our touchscreen selection techniques. After that the system structure and framework classes are briefly explained. We conclude by discussing our development plans for the future.

2. Related work

Even though most public information kiosks are very limited in their interaction with the user, some experimental research prototypes have been developed. Digital Smart Kiosk [1] is a highly interactive advanced kiosk prototype that senses the user with computer vision and allows touchscreen input. It presents the information using a standard Web browser. Cybcérone [3] relies solely on a touchscreen when interacting with the user and can present Java-based Web pages in a kiosk setting.

Speech recognition has been utilized in several kiosk prototypes, like in the MASK kiosk [5] that was used to provide train travel information and tickets. DINEX [7] is a speech-aware restaurant guide that is used for the same purpose as our restaurant system example in this paper. However, DINEX does not provide advanced touchscreen techniques to support its speech recognition. WebGALAXY [4] allows browsing the Web by speaking or typing natural language queries and navigation commands. Speech is used as an augmentation to the traditional keyboard and mouse input in their system. None of these previous kiosk systems has used speech and touch in the same way as our Touch'n'Speak system that is presented in this paper.

3. Touch'n'Speak user interface

In this section the Touch'n'Speak system is presented as it appears to the user. Our goal was to make the interface as seamless as possible, and this led us to use just speech recognition and touchscreen. Using a keyboard and a mouse would require some training for the novices and that is not adequate in public information kiosks. Unfortunately, our decision also caused some problems for us since we cannot expect the users to wear close-talk microphones and use their time to teach the speech recognition system to understand them better.

We developed a demonstration application that lets the user select restaurants in Cambridge, Massachusetts. The restaurant system is based on the Touch'n'Speak framework that will be described later in this paper. The restaurant application is used here to explain the user view of the system.

In the Touch'n'Speak framework and therefore also in the restaurant system the user has two available input modalities: touch and speech. The modalities can be freely mixed. For example, the user can start the interaction by pressing some touchscreen buttons, but can select items by voice. There are also synergistic interaction techniques that make use of both modalities. These techniques are used in picture area selections, for example in the map selection in Figure 1. When this picture was taken, the user was selecting a circular area on the map with his or her finger. Speech commands were given during the selection to control the selection process, and to carry out the search operation. These interaction techniques will be described in detail later in this paper.

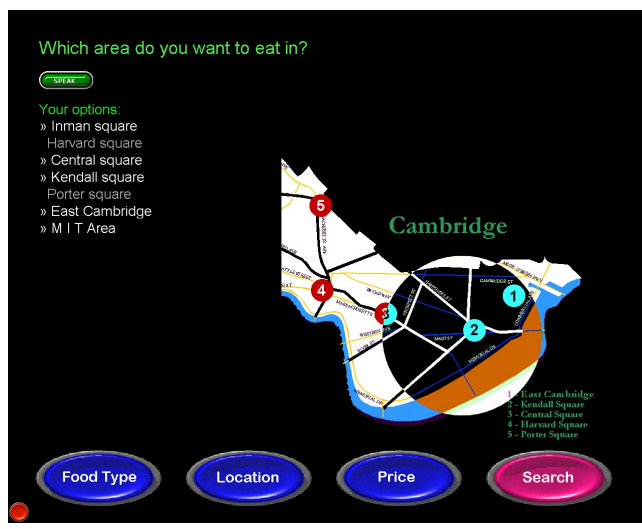


Figure 1. Area selection in the restaurant system. The user is selecting a circular area on the map of Cambridge, and the options are automatically selected based on this area.

3.1. Touch alone

The touchscreen is used in several ways. First, the conventional large touch buttons are presented on screen when needed. For example, in Figure 1 the buttons let the user select the criteria that he or she wants to refine. Pressing a button shows a special input screen, as the one for location choice in Figure 1. The search can be started any time by pressing the *Search* button. Second, the user can select options in the *Your options* list by touching them. Finally, the last use of touchscreen is the most interesting and involves the use of different selection techniques in picture selection. These techniques will be described in Section 4.

3.2. Speech alone

In addition to touching the options, the user has always an option to speak the same alternatives. For example, in the restaurant system, the user can select different criteria by saying "Food type", "Location" or "Price". The options in the *Your options* list can be selected by voice, and the user can also use some additional voice commands in the selection process, including "Help", "All", "None", "Undo" and "Not option". The undo command is vital since speech recognition cannot be guaranteed to be always accurate.

When the user has defined the criteria and carries out the search, the system generates results in HTML form and presents them in a Web browser. The resulting pages can be browsed using speech in addition to pressing buttons. This is possible even when the Java implementation in the browser does not yet support the Java Speech API [9], since currently the main application handles the speech recognition also when the Web browser is displayed to the user.

3.3. Touch and speech together

Touch'n'Speak makes use of multimodal input allowing both touch and speech to be used in the same tasks. However, true multimodal input techniques make use of both modalities in parallel. In our framework speech recognition is used to supplement touchscreen input in different picture selection techniques. These techniques are described next.

4. Picture selection techniques

We have developed different kinds of selection techniques that can be used to select a part of a picture. All of the techniques rely on touchscreen, but speech adds some value to them if used with touching. Speech interaction is not required to make selections, since it may not be usable in all conditions. For example, current speech recognition techniques are not adequate in trade shows in which there is too much continuous and unpredictable noise.

Two speech commands, “Lock” and “Unlock”, can be used in all picture selection techniques. They lock and unlock the current selection so that it is not changed accidentally. In addition, speech has an important role in the time-based selection technique.

The four picture selection techniques are described next.

4.1. Time-based selection

This selection technique is based on a timer. When the user touches the picture, the selection starts growing as long as the user keeps the finger inside the selection. The selection area can be simultaneously moved while it is growing by sliding the finger on the screen. This technique has two parameters: timer interval (default 100 ms) and increment or decrement (default 5 points). These values seemed to work well with our 20” touch monitor. Optimal values need to be determined with user testing.

In the simplest case, the selection grows as long as the finger is being pressed on the screen and stops growing when the finger is lifted off the screen. The selection can still be moved around the screen, but its size cannot be changed after it has been completed. During the selection process, speech commands “Smaller” and “Larger” give the user an option to change the direction in which the selection is changing. In this technique speech gives additional functionality that would not be possible with just the touchscreen when the user is making the selection. This is because commercial touchscreens recognize only one touch point at a time and it is needed for making the selection.

4.2. Incremental z-value based selection

Incremental z-value based selection technique makes use of special z-level detection capabilities of Elo IntelliTouch touchscreens [2]. The screen can detect finger pressure in 256 levels. This first z-value based selection technique functions simply so that when the pressure level is greater than 128, the selection area is increased by 10 pixels at a time and when the level is smaller than 128, the selection area is decreased by 10 pixels at a time. 10 pixels have proved to be a suitable amount in our experiments.

This kind of selection technique obviously has two changeable parameters: threshold value (128) and increment or decrement (10). Optimal values depend on touchscreen capabilities, i.e., how the pressure is defined, and on the speed in which the change is being added or subtracted from the current value. Elo IntelliTouch screens send a mouse event every time the pressure level changes or the mouse position changes. This has seemed to result in a natural speed in changing the selection area when using our default parameter values. Other possible implementation alternatives include using a timer as in the previous technique, but defining the size of the area by sampling the pressure level when a timer event happens. This

implementation would add timer interval as the third parameter for this technique.

4.3. Nonlinear z-value based selection

In our preliminary experiments we noticed that direct use of the z-value did not result in an appropriate selection area. The selection area changes too fast, and since the pressure level decreases rapidly when the user lifts the finger off the screen, the final selection area was not what the user thought it would be. This problem can somehow be cured with the “Lock” command that can be given using speech, but this is awkward and still a linear z-value based selection behaves unsoundly.

To cure the hastiness of direct z-value mapping, we decided to introduce a coefficient function to transform the z-value. Obviously, since direct z-value mapping did not work well, all linearly behaving coefficients would give the same results, just in different scale. That’s why we decided to experiment with nonlinear functions. One function that seems to behave well is multiplying the z-value by sine of z-value/3, in which z-value/3 represents an angle in degrees. Incidentally, the cosine function did not work well in a similar experiment. The function that we used here is shown in Figure 2 and presents the form of a function that works well, at least with Elo IntelliTouch screens.

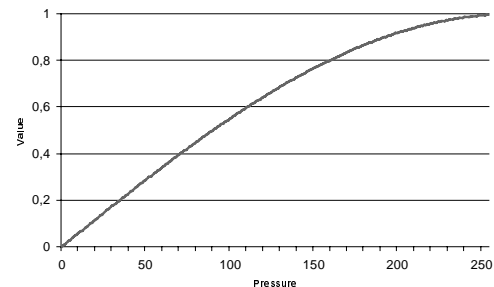


Figure 2. The nonlinear coefficient function $\sin(z\text{-value}/3)$ that we used in our z-level based selection technique.

Naturally in this kind of z-level selection technique an important decision is which kind of coefficient function is used. Since this depends on the touchscreen properties, especially on the way the pressure is defined, there may not be any general solution to this problem. With the Elo touchscreens, the selection was too hasty, and thus this sine function makes it steadier as all z-values are multiplied by a value that is smaller than 1. Also, this function speeds up growing the area at the highest pressure levels, which is good since pressing the screen hard is not as comfortable as pressing it normally. Especially moving the finger while pressing it hard on the screen is not nice.

4.4. Direct selection

The direct selection technique is the simplest of the techniques that are presented in this paper, and works as follows. The user draws the radius of a circle with his or her finger pressed on screen. The circle is drawn at the same time, and this technique is based on the principles of conventional direct manipulation [8]. This technique was implemented to allow comparing a basic direct manipulation technique with our new time- and pressure-based techniques.

5. General structure of the system

The structure of the Touch'n'Speak system is presented in Figure 3. The basic functionality of the system is inherited from the abstract framework classes *ActionFrame* and *DataFrame*. The actual implementations depend on the application. In our example the implementation is written in classes *RestaurantSystem* (action) and *Restaurant* (data).

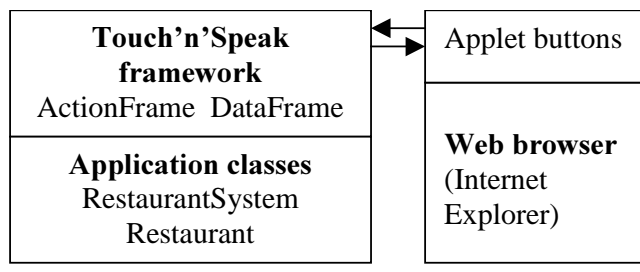


Figure 3. The Touch'n'Speak system structure.

A Web browser (Microsoft Internet Explorer) is used to present the results to the user. The resulting HTML pages are generated in the application class. The control in the Web browser is handled by using Java applets that communicate with the Touch'n'Speak system using local socket connections. Each applet tells the main system which speech command it wants to be associated with it, and what will be done when the user speaks this command. When the user presses a button or speaks its name, the button applet sends a message to the main system that executes the action that was associated with this button. Figure 4 shows an example of the resulting HTML pages in the demo application.

The system is written in C++ and uses a 20" Elo TouchSystems IntelliTouch screen [2] to detect touch position and pressure. Speech recognition is handled using Microsoft's Whisper technology [6]. Whisper is speaker-independent and requires no training. It also offers pretty good command and control capabilities that are used in the Touch'n'Speak system.

When Java Speech API has been released and supported in major browsers, the complete Touch'n'Speak system can

be ported to Java and used via the Internet. In the current prototype the reason for using C++ was that Java did not support speech recognition.



Figure 4. A resulting HTML page with three applets that respond to both touch and speech.

6. Touch'n'Speak framework

In the system level, Touch'n'Speak is not fixed at any application domain. The framework simply allows the user to construct an application by writing subclasses for two framework classes and implementing the application level functionality in them. The basic interaction services will be available to the new application by inheritance and will be accessed by implementing the abstract methods that are present in the framework classes. We will describe the framework here by using the restaurant system as an example of an existing implementation.

6.1. Actions

We have defined abstract action methods for certain events. The actions are defined in an abstract class *ActionFrame* that the actual application class inherits. In our example, the application class is *RestaurantSystem*. The general action methods in class *ActionFrame* are:

- InitAction(): called to initialize the system when it is started.
- ExitAction(): called to make the necessary finalizations when the program is closed.
- ItemSelected(): called when an item in the menu has been selected or deselected (by either touch or speech).
- PictureSelection(): called when an area in the Picture has been selected by touch.
- FinishAction(): called when the main task (i.e., *Search* in our example) is to be carried out.

ProgrammedPicture(): called when a user-drawn picture needs to be updated.

FeatureOptions(): returns a list of possible values for a given feature.

ActionFrame also has three utility methods that the application class can call. These are:

LoadHtmlPage(): loads an HTML page in the browser screen.

ModifySelection(): modifies the selection on screen.

Exit(): exits the program.

This set of callbacks and utility methods is certainly not complete, but is enough for implementing straightforward information seeking interfaces, as the restaurant system in our example.

6.2. Data

Data classes present the data that is being manipulated in the interface. The data is organized as a collection of objects that can tell the names of the features that are stored in them and their possible values. This way the framework is not fixed in any specific application domain, and new data can simply be defined by inheriting the framework class. The application data class is a subclass of an abstract class *DataFrame* that has the following abstract methods:

FeatureCount(): How many distinct features exist in this application?

FeatureName(int index): Gives the feature name for feature with index number *index*.

GetFeature(String feature): Returns the value of the feature named *feature* in this object.

Specifically, in the restaurant system the Restaurant class has five features: restaurant name, food type, location, online menu URL, and price category. It will tell about them to the RestaurantSystem class through the inherited methods listed above.

6.3. Dialogue states

In addition to the ActionFrame and DataFrame classes, the dialogue is defined in a configuration file that is presented in Windows INI file format. The file describes all states in which the system can be during its use. Below is the configuration file that is used in the restaurant system.

```
[Choices]
Choice1=Food type
Choice2=Location
Choice3=Price
```

```
[Food type]
SpeechEntry=What would you like to eat?
Attributes=SAVE MULTISELECT
```

```
[Location]
SpeechEntry=Which area do you want to eat
in?
Attributes=SAVE BGPICTURE(20inch_map.bmp)
MAPSELECT MULTISELECT
```

```
[Price]
SpeechEntry=Which price range do you prefer?
Attributes=RANGESELECT MULTISELECT SAVE
BGPICTURE(priceoptions.bmp)
GotoDefault=Start
```

```
[ConfirmQuit]
SpeechEntry=Are you sure you want to quit?
SpeechUnknown=Excuse me?
Option1=Yes
Speak1=Goodbye. $QUIT
Option2=No
Goto2=Start
```

```
[Start]
SpeechEntry=How can I help?
Attributes=BGPICTURE(cambridge.bmp)
```

The different criterium categories are defined in the Choices section, which is followed by a section for each criterium. The program begins by loading the Start state. ConfirmQuit is a default state to which the system changes when the user tries to exit the program.

There are a few attributes that can be defined for each state, the most important of which are described here:

SpeechEntry	What the system speaks when the user enters the state.
OptionXX	An available speech command, numbered successively starting from 1.
GotoXX	The state to which the control moves when command number XX is spoken.
Attributes	Special attributes that control the characteristics of the user interface. The values are separated by spaces and can consist of the following items:

MULTISELECT: this state allows multiple selections

SAVE: the selection is to be saved as an input parameter for the main task.

BGPICTURE(picture.bmp): the name of the picture that is used in the selection process.

MAPSELECT: circular picture selection mode.

RANGESELECT: rectangular picture selection mode.

PROGRAMMEDPICTURE(name): a user-drawn picture is defined for this state. The system will call the method ProgrammedPicture(name) in the application class to draw this.

All criterium buttons and the start button (named *Search* in the example) are always available. The data framework can be initialized in the inherited application classes. For example, in the restaurant system the possible restaurants and their properties are read from a URL that lists Cambridge area restaurants. That's why there are no Option

attributes in most dialogue states in the example configuration file; the options are parsed within the data class. If the MULTISELECT attribute is not defined, the state is a single selection state which changes to the next state when a button has been pushed or its speech command spoken, as in state ConfirmQuit. As an illustrative example of a multiselection state, the definition of the state Location can be compared to Figure 1, which presents it on screen.

7. Discussion

We decided to use speech input partly because we did not want to waste limited screen space in an on-screen keyboard. Our current techniques do not allow arbitrary input, but are still more versatile than simple touch button interfaces.

We were limited to a pretty small vocabulary due to our design decisions. Since we cannot expect the users to train the system, a speaker independent speech recognition engine needs to be used. We would have used a keyword spotting engine, but none were available for us at the time. Thus the speech recognition capabilities are pretty limited in our current system.

During the system development we observed that it is not very convenient to drag a finger over a touchscreen for extended periods of time. This is because some pressure is always needed, and pressing the display while moving the finger just does not feel right. This led us to develop our new interaction techniques that do not require the dragging operation as does the last presented direct selection technique. The dragging is still used to move the selection with all techniques, but the user can choose not to use it, but just make a new selection.

The techniques presented here seem to work well. However, user tests are needed to compare our selection techniques and check our presumptions of convenient parameter values for the techniques. We will be carrying out such an evaluation later this year.

8. Conclusion

Touch'n'Speak demonstrates the use of speech recognition and touch screen input in public information kiosks. These input modalities were selected to make using these systems as natural as possible, thus allowing them to be used by a wide range of users. Touch'n'Speak is an object-oriented framework that was used to build a restaurant information system in our example. Preliminary user tests show that our multimodal interaction techniques are adequate for this kind of input. We will be carrying out an empirical study in which we compare different picture selection techniques and evaluate the system with a larger subject group. Speech recognition can be made more natural when we have a good speaker-independent keyword spotting engine that does not limit available phrases to

predefined ones as does a common command and control based engine.

Our prototype system is one step closer to a natural information kiosk that is easy to use and powerful both in presenting the information and in finding out what the user wants. Much work is still to be done in developing new touchscreen techniques and augmenting them with speech recognition. We are just beginning to take full advantage of these input modalities.

Acknowledgments

This work started during my research internship at Digital Equipment Corporation's Cambridge Research Laboratory in 1997. I thank my host Andrew Christian for providing an interesting project and environment to work in. This work was partially supported by the Academy of Finland (Project 38118, Grant 52936).

References

- [1] Andrew C. Christian and Brian L. Avery, Digital Smart Kiosk project. *Human Factors in Computing Systems, CHI '98 Conference Proceedings*, ACM Press, 1998, 155-162.
- [2] Elo TouchSystems, *IntelliTouch surface-wave products* [<http://www.elotouch.com/itover.html>]
- [3] François Grize and Mehdi Aminian, Cybcérone – A kiosk information system based on the World Wide Web and Java. *Interactions*, 4 (6), 1997, 62-69.
- [4] Raymond Lau, Giovanni Flammia, Christine Pao, and Victor Zue, WebGALAXY: Beyond point and click – a conversational interface to a browser. *Proceedings of Sixth International World Wide Web Conference*, 1997, 119-128.
- [5] A. Life, I. Salter, J.N. Temen, F. Bernard, S. Rosset, S. Bennacef, and L. Lamel, Data collection for the Mask Kiosk: WOz vs prototype system. *Proceedings of International Conference on Speech and Language Processing (ICSLP'96)*, vol. 3, 1996, 1672-1675.
- [6] Microsoft Research speech technology web pages, [<http://www.research.microsoft.com/stg/>]
- [7] Stephanie Seneff and Joseph Polifroni, A new restaurant guide conversational system: issues in rapid prototyping for specialized domains. *Proceedings of International Conference on Speech and Language Processing (ICSLP'96)*, vol. 2, 1996, 665-668.
- [8] Ben Shneiderman, The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology* 1, 1982, 237-256.
- [9] Sun Microsystems, *Java Speech API home page*, [<http://java.sun.com/products/java-media/speech/>]