

TECHNICAL UNIVERSITY OF CRETE

Multimodal System for Preschool Children

by

Meliopoulos Spiros

A thesis submitted in partial fulfillment for the
degree of Electronic and Computer Engineer

in the
Speech Processing and Dialog Systems
Department of Electronic and Computer Engineering

October 8, 2008

Supervisor:

Prof. Alexandros POTAMIANOS

Committee:

Prof. Katerina MANIA

Prof. Vassilis DIGALAKIS

Prof. Alexandros POTAMIANOS

“KISS: Keep It Simple, Stupid.”

TECHNICAL UNIVERSITY OF CRETE

Abstract

Speech Processing and Dialog Systems
Department of Electronic and Computer Engineering

Electronic and Computer Engineer

by Meliopoulos Spiros

In this diploma thesis, we used a multimodal game platform under a WoZ environment to acquire information for pre-school children area. Informal evaluation by children was found positive especially for the animated agents (84.6% preference) and the graphics. Further, flexible choice of input modality made the application easy to use even for little children with disabilities. Finally, there seems to be a relation between multimodality and time-errors where time increases and errors decreases as children fuses modalities. We expect that data acquired from WoZ experiment will help further conversational child machine interaction technology.

Contents

Abstract	iii
List of Figures	vi
List of Tables	viii
Abbreviations	ix
1 Introduction	1
2 Multimodal Systems	3
2.1 Introduction	3
2.2 Advantages of Multimodal Systems	3
2.3 Basic Design Principles	5
2.4 Multimodality & Children	9
3 Language & Acoustic Modeling	11
3.1 Introduction	11
3.2 Language Modeling	12
3.3 Acoustic Modeling	14
4 Game Platform	17
4.1 Introduction	17
4.2 System Information & Architecture	17
4.3 Games Overview	18
.	20
.	20
.	20
.	20
.	20
.	20
.	20
.	20
.	20
4.4 The Farm	22
4.4.1 Description	22
4.4.2 Software Analysis	24

4.5	Speech Synthesis	28
4.6	Voice Activity Detection	29
4.7	The WoZ Part	30
5	Evaluation	32
5.1	Introduction	32
5.1.1	Evaluation of Speech Recognition	32
5.1.2	Evaluation of Multimodal Systems	33
5.2	Our Approach	34
6	Results & Conclusions	38
6.1	Results & Discussion	38
6.2	Conclusions	45
6.3	Future Work	45
	Acknowledgements	46
A	System Setup	47
A.1	Introduction	47
A.2	Repository Connection	47
A.3	Checkout Modules	48
A.3.1	Controller Server	48
A.3.2	Application Manager	50
A.3.3	Children Interface	50
A.4	Scripts Module	51
A.4.1	Phoneme Configuration File Format	51
A.4.2	Pronunciation Lexicon Format	52
A.4.3	SONIC Configuration File	52
A.4.4	Acoustic Model	53
A.4.5	Language Model	54
	54
A.4.6	Game Metrics	54
A.5	Connect as Client	55
A.5.1	Children Home Page	55
A.5.2	Registration Phase	56
A.5.3	Start Playing	57
A.5.4	Microphone Adjustment	57
B	Questionnaire	60
	Bibliography	62

List of Figures

3.1	Architecture of a hidden Markov model	11
3.2	Parameters of HMM	15
3.3	Recognizer	16
4.1	The Architecture	17
4.2	Town - Games	18
4.3	The Shapes	19
4.4	The Numbers	20
4.5	The Arithmetics	20
4.6	More or Less	21
4.7	Find Number Task	21
4.8	The Agent	22
4.9	Welcome Scene	22
4.10	The Farm	23
4.11	End of Farm	23
4.12	State Transition	25
4.13	Activity Diagram	26
4.14	Class Diagram	27
4.15	A Text To Speech System	28
4.16	VAD	29
4.17	The Wizard GUI	31
6.1	Preference per Character	39
6.2	Child with Disabilities	39
6.3	Errors per Modality	41
6.4	Time per Modality	42
6.5	Evaluation test 1 for recognition	43
6.6	Evaluation test 2 for recognition	44
6.7	Evaluation test 3 for recognition	44
A.1	Repository Connection	47
A.2	Repository Structure	48
A.3	Run Controller	49
A.4	Compile Manager	50
A.5	Sonic Recognizer	51
A.6	Phoneme Configuration File	52
A.7	SONIC Configuration File	53
A.8	Metrics	54

A.9 Children Home Page	55
A.10 Registration	56
A.11 Registration Form	56
A.12 Games Page	57
A.13 Trust Applet	57
A.14 Allow Microphone	57
A.15 Settings	58
A.16 Microphone	58
A.17 Calibrate Microphone	59

List of Tables

5.1	Game Sessions per Modality	35
5.2	Corpus Characteristics	36

Abbreviations

VAD	V oice A ctivity D etection
ASR	A utomatic S peech R ecognition
TTS	T ext T o S peech
RMS	R oot M ean S quare
OOV	O ut o f V ocabulary
WoZ	W izard o f O z
PC	P ersonal C omputer
VTLN	V ocal T ract L ength N ormalization
TCP	T ransmission C ontrol P rotocol
CCI	C hild C omputer I nteraction
HCI	H uman C omputer I nteraction

Dedicated to my parents

Chapter 1

Introduction

It is a fact that multimedia and speech technology has flourished in the last 5 years. As a result, multimodal applications [2](#) became more attractive than before. Nowadays, most natural spoken dialogue systems target toward adult user population. That's because the task of building such systems for adults poses less challenges than for children [\[1\]](#).

In this diploma thesis, we designed and implemented a multimodal system for pre-school children. The internet-based platform that was developed could be used in many research fields.

We utilized the WoZ technique in order to acquire as soon as possible user-data. This technique allows the observation of a user operating an apparently fully functioning system whose missing services are supplemented by a hidden wizard *Chapter 4*.

First of all, the conversational multimedia interaction data be obtained could help training of language and age-dependent acoustic models [\[1, 2\]](#). These trained models could be employed in interactive speech applications for education. Moreover, linguists or psychologists based upon acoustic analysis of data could further survey speech pathologies and language acquisition among pre-school children.

The second research field which is more adjacent to this diploma thesis is the investigation of multiple modalities in child-machine interaction. The understanding of how children use different input modalities when interacting with the PC will be the guide for future development of efficient interfaces and multimodal systems in general.

At this point, I feel obligated to clarify my own contribution to the Game Platform which mainly includes: a multimodal game named **The Farm** and the **WoZ part** which are analyzed in *Chapter 4*.

The rest of the thesis is organized as follows: *Chapter 2* introduces some basic material on multimodal systems. *Chapter 3* describes the fundamentals of Speech Recognition including theory from Language and Acoustic Modeling though the platform will integrate ASR in future. *Chapter 4* presents the Game Platform designed by Theofanis Kannetis and my own contribution. Evaluation methods for multimodal systems and our approach in assessing multimodality usage from children are discussed in *Chapter 5*. The results based upon child-game interaction are discussed in *Chapter 6*. This Chapter also summarizes the conclusions of this thesis and outlines interesting directions for future work.

Chapter 2

Multimodal Systems

2.1 Introduction

Nowadays, there is a tendency towards multimodal interactive applications. Such systems give the opportunity to utilize several modalities. So, what does modality mean? As a modality, we could define the means that convey information between user and system. For example, keyboard and mouse are traditional input modalities in comparison with voice and touch. Graphics, animation and speech are some typical output modalities as well. Interfaces may be grouped by the modalities they utilize.

A unimodal system offers one kind of modality. On the other hand, multimodal interfaces combine many simultaneous input modalities and may present the information using synergistic representation of many different output modalities. Someone may wonder why to utilize a multimodal system.

2.2 Advantages of Multimodal Systems

The use of multimodality holds promise to enhance human-computer interaction. Since human perception is based largely upon multimodality, it is an important factor in the realization of human-friendly interfaces. As applications become more complex, a single modality alone does not permit varied users to interact effectively across different tasks and usage environments. However, a flexible multimodal interface offers people the choice to use a combination of modalities, or to switch to a better-suited modality, depending on the specifics of their abilities, the task, and the usage conditions.

Multimodal interfaces permit our highly skilled and coordinated communicative behaviors to control system interactions in a more transparent interface experience than ever

before. Our voice, hands, and whole body together, once augmented by sensors such as microphones and cameras, now are the ultimate transparent and mobile multimodal input devices.

Accessibility for diverse users and usage contexts

Perhaps the most important reasons for developing multimodal interfaces are their potential to greatly expand the accessibility of computing to diverse and non-specialist users, and to promote new forms of computing not previously available. Since there can be large individual differences in people's ability and preference to use different modes of communication, multimodal interfaces will increase the accessibility of computing for users of different ages, skill levels, cognitive styles, sensory and motor impairments, native languages, or even temporary illnesses. This is because a multimodal interface permits users to exercise selection and control over how they interact with the computer. For example, a visually impaired user may prefer speech input, as may a manually impaired user with a repetitive stress injury or with her arm in a cast. In contrast, a user with a hearing impairment, strong accent, or a cold may prefer pen input. Well before the keyboard is a practiced input device, a young preschooler could use either speech or pen-based drawing to control an educational application. A flexible multimodal interface also permits alternation of input modes, which prevents overuse and physical damage to any individual modality during extended periods of use.

Performance stability and robustness

A second major reason for developing multimodal architectures is to improve the performance stability and robustness of recognition-based systems. From a usability standpoint, multimodal systems offer a flexible interface in which users can exercise intelligence about how to use input modes effectively so that errors are avoided. To reap these error-handling advantages fully, a multimodal system must be designed so that the two input modes (e.g., speech and pen) provide parallel or duplicate functionality, which means that users can accomplish their goals using either mode. A well-designed multimodal architecture also can support the mutual disambiguation of two input signals. For example, if a user says "ditches" but the speech recognizer confirms the singular "ditch" as its best guess, then parallel recognition of several graphic marks in pen input can result in recovery of the correct spoken plural interpretation.

One of the most exciting recent discoveries is that multimodal systems demonstrate a relatively greater performance advantage precisely for those users and usage contexts in which unimodal systems fail. For example, recognition rates for unimodal

spoken language systems are known to degrade rapidly for accented speakers or children, and in noisy field environments or while users are mobile. However, recent research has revealed that a multimodal architecture can be designed that closes this recognition gap for challenging users and usage contexts. As a result, next-generation multimodal systems may be capable of harnessing new media in a way that makes technology available to a broader range of everyday users and usage contexts than ever before.

Expressive power and efficiency

Systems that process multimodal input aim to give users a more powerful interface for accessing and manipulating information, such as increasingly sophisticated visualization and multimedia output capabilities. In contrast, interfaces that rely on keyboard and mouse input are limited or inappropriate for interacting with small mobile systems, virtual environments, and other new forms of computing. Since spoken and pen-based input are human language technologies, they can easily provide flexible descriptions of objects, events, spatial layouts, and their interrelation.

2.3 Basic Design Principles

To exploit the advantages of multimodal systems, the design should conform to some principles. In this way, such systems obtain the maximum functionality. Basic principles are:

1. **Learnability** depicts the user's adaptation to the system in relation with the time. This characterizes how fast the user become familiar with the services given.
2. **Flexibility** can be attributed with the term modality as well. Application becomes more interactive as flexibility increases.
3. A system may obtain **Robustness** if it is capable of coping well with unpredictable variations in its operating environment with minimal damage, alteration or loss of functionality.

Apart from these basic principles which should be the guide when deploying a software, a multimodal platform should conform to some rules that are relative to each one modality which is incorporated in the system.

First of all, multiple modalities need to be synchronized. Spoken interaction is highly temporal, whereas visual interaction is spatial. When combining these modes of interaction in a multimodal interface, synchronization is a key feature that determines overall usability of the interface. Synchronization is key to the following multimodal acts:

Point And Talk

User points at a location on the map while speaking a question.

Redundant Confirmation

The user interface supplements visual output with a spoken confirmation; for example, a travel reservation system might visually highlight the user's selection while speaking an utterance of the form: *"Leaving from San Francisco"*.

Unless synchronized, such supplementary use of modalities can significantly increase the cognitive load experienced by the end-user and prove a source of confusion.

Parallel Communication

The user interaction leverages the availability of multiple streams of output to increase the band-width of communication. For example, a travel reservation system might visually present a list of available flights and speak a prompt of the form: *"There are 7 flights that match your request, and the flight at 8:30am appears to be the most convenient"*.

To be effective, this form of complementary use of multiple modalities needs to be well-synchronized with respect to the underlying interaction state.

Moreover, multimodal interaction should degrade gracefully. Human interaction degrades gracefully; for example, a face-to-face conversation degrades gracefully in that it still remains effective when one of the participants in the conversation is functionally blind, e.g., when talking over a telephone. This form of graceful degradation is due to the high level of redundancy in human communication. As man-machine interfaces come to include multimodal interaction, we need to ensure that these interfaces degrade gracefully in a manner akin to human conversation. Such graceful degradation is important since the user's needs and abilities can change over time e.g., a user with a multimodal device moving between a noisy environment where spoken interaction fails and an eyes-free environment where visual interaction is unavailable. The use of multiple modalities to supplement one another leads to user interfaces that degrade gracefully. Portions of the interface that use multiple modalities to complement one another are natural points where the interface will fail to degrade gracefully. When complementary modalities are used, the underlying system needs to be aware of the modalities that

are currently available and ensure that all essential items of information are conveyed to the user. Ensuring the above is a key accessibility requirement in ensuring that the user interface is usable by individuals with different needs and abilities. Capabilities can change rapidly in the case of mobile users. Such changes include available band-width between the mobile device and the network, as well as changes in the band-width of communication between device and user. To be useful, multimodal interaction that is deployed to mobile devices need to adapt gracefully to such changes.

Furthermore, multiple modalities should share a common interaction state. Successful task completion during a conversation requires that the participants share a common mental model of the conversation, and this is true in the case of man-machine interaction as well. When using multiple modalities in a user interface, it is important that the various modes of interaction affect and share a common interaction state that is used to update the presentation in the various available output media. Such a common interaction state is also essential rapid completion of the conversation, since the various multimodal interactors can examine this shared interaction state in determining the next step in the dialog. A shared interaction state is important for the following multimodal acts:

Switching Modalities

User switches between interaction modalities owing to a number of external factors such as the available band-width between the user, the device and the network. For such transitions to be seamless, the data collected by each interaction modality, as well as the information conveyed via the available output media need to be driven by a shared interaction state.

History

The shared interaction state can track the history of user interaction, and this history can be useful in determining the most appropriate path through the dialog to achieve rapid task completion.

Multi-Device Interaction

A user with a personalized mobile device might wish to use a large visual display upon entering a conference room. To achieve a synchronized multimodal experience, the user's mobile device and the conference room display will need to share some interaction state.

Distributed Multimodality

It might be advantageous to offload complex speech processing tasks to network servers when using thin clients such as cell phones. As an example, a cell phone

might be capable of local speech processing sufficient to enable the user to dial a small number of hotlist entries by speaking a name. If the name is not found in the hotlist, it may need to be looked up in a larger phone book, e.g., a company directory, and the speech processing required might be best offloaded to a network server. Sharing a common interaction state between the visual and spoken components of the cell-phone is essential for synchronized multimodal interaction in such distributed deployments.

Further, multimodal interfaces should also be predictable. Multimodal interaction provides the user with a multiplicity of choices and often enables a given task to be performed in a number of different ways. But to be effective, the user interface needs to empower the user to intuitively arrive at these different means of completing a given task. Symmetric use of modalities where appropriate can significantly enhance the usability of applications along this dimension; for example, an interface that can accept input via speech or pen might visually highlight an input area while speaking an appropriately designed prompt. Where a specific modality is unavailable for a given task, e.g., “*signatures may only be entered via pen input*”, appropriate prompt design can help make the user implicitly aware of this restriction. Predictable multimodal user interfaces are important for:

Eliciting Correct Input

Appropriately designed prompts are important for eliciting the desired user input. This in turn can lead to rapid task completion and avoid user frustration when using noisy input channels such as speech.

What Can I Do?

Rich user interfaces can often leave the user impressed with the available features, but baffled as to what can be done next. Spoken interaction –combined with good visual user interface design– can be leveraged to overcome this lost in space problem. Rich multimodal interfaces can use the shared interaction state and dialog history to create user interface wizards that guide the user through complex tasks.

Finally, multimodal interfaces need to adapt to the user’s environment to ensure that the most optimal means of completing a given task are made available to the user at any given time. In this context, optimality is determined by:

- The user’s needs and abilities

- The abilities of the connecting device
- Available band-width between device and network
- Available band-width between device and user
- Constraints placed by the user's environment, e.g., need for hands-free, eyes-free operation.

2.4 Multimodality & Children

It is beyond saying that children would be benefited from a well designed multimodal interface. In recent years, there has been an increasing trend for children to use information and communication technology in its various forms. Children now grow up immersed in technology to a level that keeps surprising earlier generations, but which, to them, is simply an inherent element of their habitat. Children form a crucial segment of customer population for interactive multimedia systems and are eager and quick to embrace, and use, new technologies.

Multimodality is attractive in the creation of conversational interfaces for children in the sense of both overcoming inherent limitations in speech technology and exploiting the ubiquitous availability and/or familiarity with conventional modalities such as the computer mouse, keyboard, joy stick and pen. There are several open research issues that need to be addressed including multimodal input integration and interpretation, multimodal dialog design, multimedia output presentation and performance evaluation. Realistic case studies and prototype designs are crucial to further our understanding of multimodal interactions. Although the task of building multimodal dialogue applications for children looks appealing, it poses great challenges.

First of all, CCI ¹ is still finding its way. On the other hand, HCI (Human computer interaction) has been growing in importance over the last 25 or more years, and, as a discipline, has matured and settled. Furthermore, most speech recognition systems that have been deployed target adult users and experience severe ASR performance degradation when used by children.

Children speak and interact with computers differently from adults. Several aspects of these differences can be identified such as in the acoustic and linguistic characteristics of speech, dialog interaction strategies, problem solving skills and user preferences. Further, the sources of these differences reflect physiological and anatomical changes associated with development of articulators and the effects of socio-economic factors during a child's

¹Child computer interaction which is the sub-field of HCI studies how children use interactive products

growth. Acoustic variability in children's speech, which renders pattern classification difficult, is identified as a major hurdle in building high performance ASR applications for children.

Chapter 3

Language & Acoustic Modeling

3.1 Introduction

In this Chapter, we will introduce the reader to some basic concepts from speech recognition technology. Speech recognition converts spoken words to machine-readable input. The term “voice recognition” may also be used to refer to speech recognition, but can more precisely refer to speaker recognition, which attempts to identify the person speaking, as opposed to what is being said.

Modern general-purpose speech recognition systems are generally based on HMMs. These are statistical models which output a sequence of symbols or quantities. One possible reason why HMMs are used in speech recognition is that a speech signal could be viewed as a piecewise stationary signal or a short-time stationary signal. That is, one could assume in a short-time in the range of 10 milliseconds, speech could be approximated as a stationary process. Speech could thus be thought of as a Markov model for many stochastic processes.

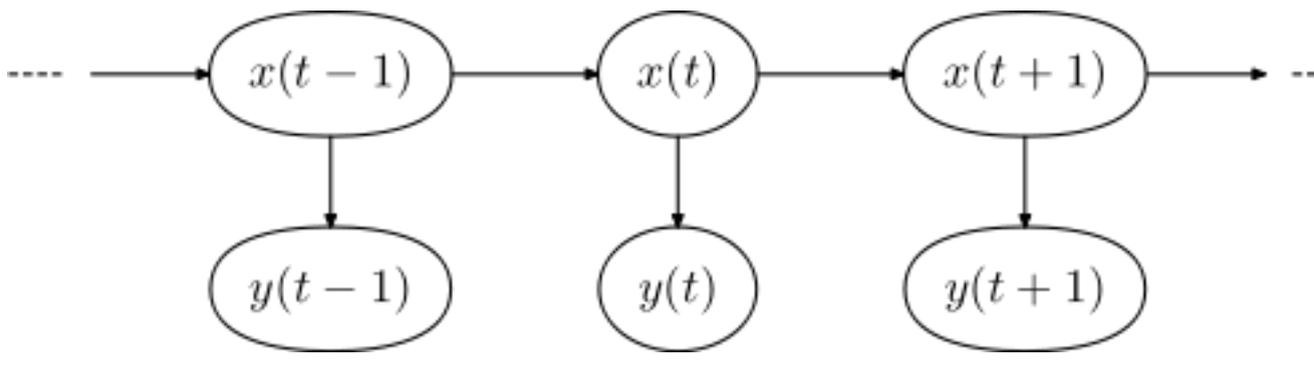


FIGURE 3.1: Architecture of a hidden Markov model

Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use. In speech recognition, the hidden Markov model would output a sequence of n -dimensional real-valued vectors (with n being a small integer, such as 10), outputting one of these every 10 milliseconds. The vectors would consist of cepstral coefficients, which are obtained by taking a Fourier transform of a short time window of speech and decorrelating the spectrum using a cosine transform, then taking the first (most significant) coefficients. The hidden Markov model will tend to have in each state a statistical distribution that is a mixture of diagonal covariance Gaussians which will give a likelihood for each observed vector. Each word, or (for more general speech recognition systems), each phoneme, will have a different output distribution; a hidden Markov model for a sequence of words or phonemes is made by concatenating the individual trained hidden Markov models for the separate words and phonemes.

When such a system is presented with a new utterance, it must compute the most likely source sentence. Decoding of the speech would probably use the Viterbi algorithm to find the best path, and here there is a choice between dynamically creating a hidden Markov model which includes both the acoustic and language model information, or combining it statically beforehand (the finite state transducer, or FST, approach).

In most cases, speech recognition engines require two types of files to recognize speech. They require an acoustic model, which is created by taking audio recordings of speech and their transcriptions (taken from a speech corpus), and 'compiling' them into a statistical representations of the sounds that make up each word (through a process called 'training'). They also require a language model or grammar file. A language model is a file containing the probabilities of sequences of words. A grammar is a much smaller file containing sets of predefined combinations of words. Language models are used for dictation applications, whereas grammars are used in desktop command and control or telephony interactive voice response (IVR) type applications.

3.2 Language Modeling

A statistical language model assigns a probability to a sequence of m words $P(w_1, \dots, w_m)$ by means of a probability distribution. Language modeling is used in many natural language processing applications such as speech recognition, machine translation, part-of-speech tagging, parsing and information retrieval. Statistical language modeling tries to acquire regularities of natural language using corpora processing. Corpora are collections of text, e.g., news articles, scientific papers, transcribed dialogues; they can be considered as informative resources.

In speech recognition and in data compression, such a model tries to capture the properties of a language, and to predict the next word in a speech sequence. N-gram models are thoroughly used in the area. A word sequence is useful to be modeled for a variety of reasons. For example, in many cases one may want to predict the next word in a sequence. The prediction becomes applicable when it can be measured. A probabilistic language model treats words as events, distributing to them a probability mass.

In an n-gram model, the probability $P(w_1, \dots, w_m)$ of observing the sentence w_1, \dots, w_m is approximated as

$$P(w_1, \dots, w_m) = \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1}) \approx \prod_{i=1}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) \quad (3.1)$$

Here, it is assumed that the probability of observing the i_{th} word w_i in the context history of the preceding $i-1$ words can be approximated by the probability of observing it in the shortened context history of the preceding $n-1$ words (n_{th} order Markov property).

The conditional probability can be calculated from n-gram frequency counts:

$$P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \dots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, w_{i-1}, \dots, w_i)} \quad (3.2)$$

For example, in a bigram ($n=2$) language model, the probability of the sentence:

“*I saw the red house*” is approximated as

$$P(I, \text{ saw, the, red, house}) \approx P(I)P(\text{saw}|I)P(\text{the}|\text{saw})P(\text{red}|\text{the})P(\text{house}|\text{red}) \quad (3.3)$$

whereas in a trigram ($n=3$) language model, the approximation is

$$P(I, \text{ saw, the, red, house}) \approx P(I)P(\text{saw}|I)P(\text{the}|I, \text{ saw})P(\text{red}|\text{saw, the})P(\text{house}|\text{the, red}) \quad (3.4)$$

N-gram models have received intensive research since its invention, several enhanced models have been proposed. Here are some typical extensions to traditional one:

Class-based N-gram model

In order to cope with the data sparseness problem, class-based N-gram model was proposed. Instead of dealing with separated words, class-based N-gram estimates parameters for *word classes*. By clustering words into classes, a class-based N-gram model can reduce the model size significantly with the cost of slightly higher perplexity.

Sequence N-gram model

Sequence N-gram is an attempt to extend N-gram models with variable length sequences. A sequence can be a sequence of word, word class, part-of-speech or whatever a sequence of *something* that the modeler believes bearing important grammar information.

3.3 Acoustic Modeling

An acoustic model is created by taking audio recordings of speech, and their text transcriptions, and using software to create statistical representations of the sounds that make up each word.

Audio can be encoded at different sampling rates (i.e. samples per second - the most common being: 8 kHz, 16 kHz, 32 kHz, 44.1 kHz, 48 kHz and 96 kHz), and different bits per sample (the most common being: 8-bits, 16-bits or 32-bits). Speech recognition engines work best if the acoustic model they use was trained with speech audio which was recorded at the same sampling rate/bits per sample as the speech being recognized.

The limiting factor for telephony based speech recognition is the bandwidth at which speech can be transmitted. For example, your standard land-line telephone only has a bandwidth of 64 kbit/s at a sampling rate of 8 kHz and 8-bits per sample (8000 samples per second * 8-bits per sample = 64000 bit/s). Therefore, for telephony based speech recognition, you need acoustic models trained with 8 kHz/8-bit speech audio files.

In the case of Voice over IP, the codec determines the sampling rate/bits per sample of speech transmission. If you use a codec with a higher sampling rate/bits per sample for speech transmission (to improve the sound quality), then your acoustic model must be trained with audio data that matches that sampling rate/bits per sample.

Most speech recognition systems today model sound units, or phonemes, by a sequence of connected Hidden Markov Model (HMM) states. Speech is analyzed at 100 frames per second and spectral parameters are extracted at each time instant to produce a 39-dimensional feature. This feature representation, known as cepstral parameters, efficiently models the spectral content, frame energy as well as the velocity and acceleration of the audio at each time instant. For each analyzed time frame, t , the HMM-based recognizer is assumed to transition from state i to state j with probability a_{ij} and emit an observation symbol (effectively a 39-dimensional feature) with probability density $b_j(o_t)$. Each phoneme is typically modeled using anywhere between 3 to 5 HMM states depending on the recognizer implementation and each state is modeled by a weighted set

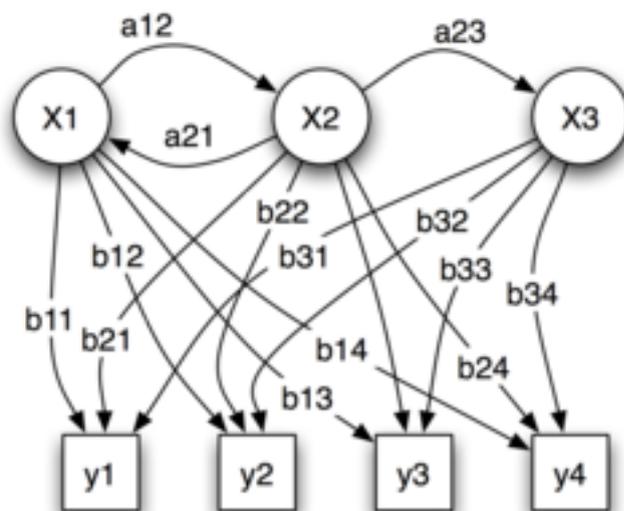


FIGURE 3.2: Probabilistic parameters of a hidden Markov model
 x - states
 y - possible observations
 a - state transition probabilities
 b - output probabilities

of M multivariate Gaussian distributions. Acoustic modeling usually makes the assumption that the feature elements are independent and therefore able to be modeled using a diagonal covariance matrix rather than a full-covariance matrix. In acoustic modeling, there are some techniques to remove the speaker-dependent variations and thus model relevant details in the speech signal more accurately.

Spectral features vary significantly from speaker to speaker due to vocal tract differences. The idea of Incremental Vocal Tract Length Normalization (VTLN) is to warp the frequency axis linearly and thereby transform the speech of different speakers to that of a “generic” speaker. Another method is the Maximum Likelihood Linear Regression (MLLR) based adaptation. MLLR is based on estimation of a linear transformation matrix (or class of matrices) that transform the mean vectors of the system Gaussians based on a set of adaptation data. Finally, Constrained Maximum Likelihood Linear Regression (CMLLR) estimates a linear transform in which the same transform is applied to both the means and variances of the system parameters. CMLLR has the advantage of being applicable to the feature-space rather than the model space. Hence, the computational overhead is reduced to a simple feature-space transform.

Acoustic model training typically consists of three basic steps: feature extraction, Viterbi state-based alignment, and model estimation. The outcome of the training process

is a set of decision-tree state-clustered Hidden Markov Models. The last two steps (alignment, model estimation) are iterated until speech recognition errors are minimized on a development test set. In general, you will need a set of audio files and associated text-based transcriptions along with a pronunciation dictionary. A block diagram of the process is shown in next figure.

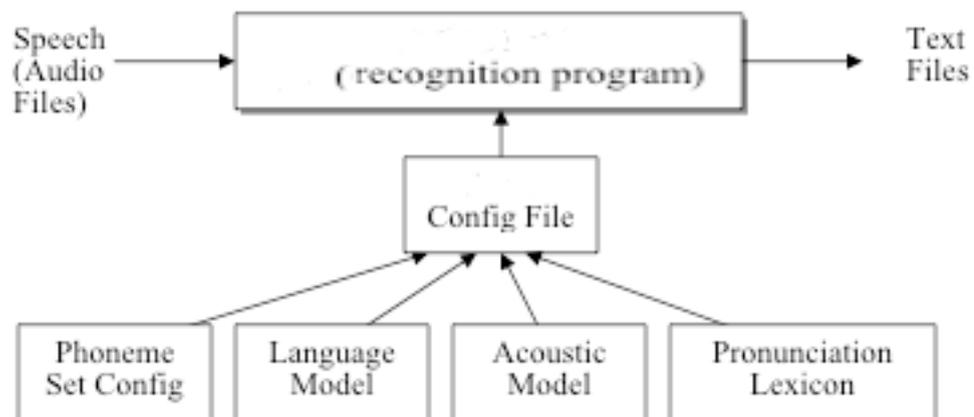


FIGURE 3.3: Example inputs required to perform recognition. A single configuration file is used to setup the recognition job. The configuration file provides the listings of the decoder settings, phoneme set, language model, acoustic model, and pronunciation lexicon.

Chapter 4

Game Platform

4.1 Introduction

In this Chapter, the game platform is described. This educational platform targets towards children of ages between 3 and 6 years old. Theofanis Kannelis(M.Sc) has deployed the platform to explore some characteristics of children computer interaction. Although it has been designed to integrate automatic speech recognition, a wizard of oz component is temporally in use. It hosts 3 multimodal games.

4.2 System Information & Architecture

The system's architecture is modular [4.1](#). Each module is responsible for the completion of a specific task. The synergy of independent modules determines the behavior of the application. The platform is based upon some multimodal games where children can interact with embodied agents using a microphone because children can hardly use keyboard and mouse in preschool ages. Games convey information mainly through speech and animations which can be easily absorbed by children.

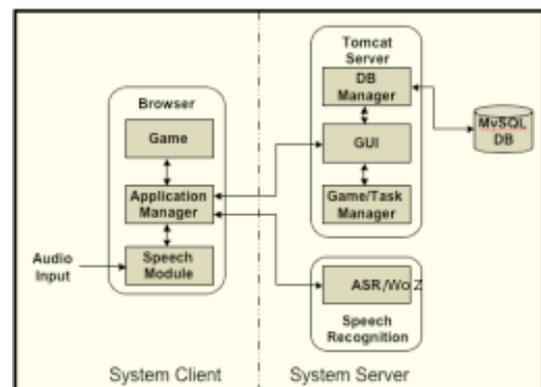


FIGURE 4.1: The Architecture

Apart from being modular, the architecture is conformed to client-server structure. In client's side, there are components mainly related with the front-end. The application manager is the sub-component of client which manages the game and communicate with the server side interchanging audio and text messages. Speech module is responsible to manipulate the user's speech. It will detect voice activity 4.16 and it will notify the system.

On server's side, there are vital components of the back-end. The interface was deployed with java server pages technology. A database stores information per user such as name, age, genre and some features related with the interface configuration. For example, these features may differentiate the background color. Two are the main components in server's side. The first is responsible for speech recognition. It has been designed to easily switch from Wizard of Oz 4.17 mode to realtime automatic speech recognition(ASR). The latter is the manager which controls all the traffic between server and client.

4.3 Games Overview

The main platform's concept is the design of a virtual environment along with a small town 4.2 where children could explore though gaming. Children may attend different games in a variety of locations such as: the theater, the park and the school. Most games include different levels with escalated difficulty. A game may also constitutes from more than one task. A concise description of each game follows.

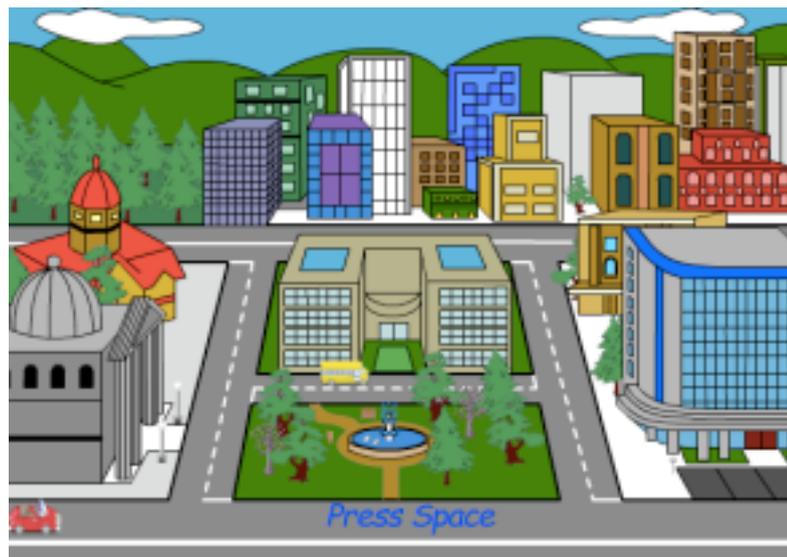


FIGURE 4.2: Town - Games

The shapes¹ 4.3 familiarizes the child with some concepts of geometry. This game is appropriate for ages between 4 and 6 years old. Children can initially select a shape to hear from the agent some basic characteristics. Next, user can play the main recognition game. During the game, a shape appears onto the stage and the user is asked to recognize it. Each time an object is identified correctly, a part of the train at the bottom of the screen is colored. Children must color three different trains in order to finish first level. Each time a train is colored, a reward is given. During next level, an item appears onto the stage and children must recognize the shape that corresponds to the item. At third level, the animated agent picks up a shape, and children must choose the item with the similar shape.

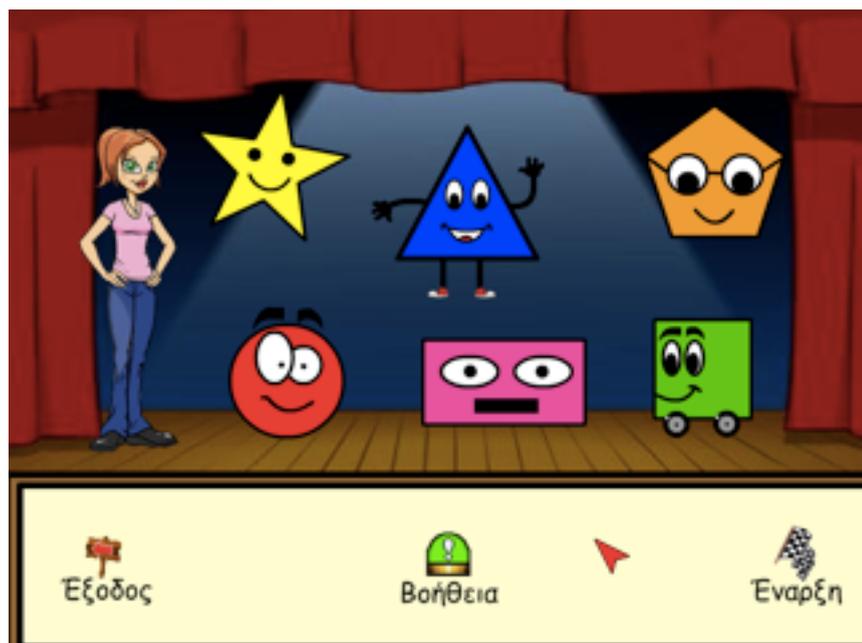


FIGURE 4.3: Shapes Game

¹Designed by Theofanis Kannetis(M.Sc)

The numbers¹ introduces the child with simple mathematics such as the comparison and the addition. This game includes three different tasks handled by the corresponding agents.



FIGURE 4.4: Numbers Game

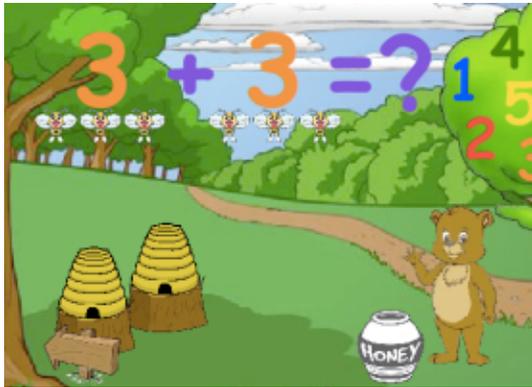


FIGURE 4.5: Task with arithmetics

The little bear character teaches the addition for numbers between 1 and 9. Each time, some equation appears onto the stage but the result omits. The user is then asked to find the missing term and complete the equation. In each right effort, a bee adds honey into the vase. The addition for numbers between 1 and 9 is often introduced in kindergarten.

The little rabbit with the carrots teaches the comparison to children. Some carrots appear inside a bucket and a bunch of other carrots outside of it. The rabbit agent then asks the child where are the more carrots. Children in school ages learn to compare groups of objects evaluating quantities. Pre-school children could also play this task through numbering the objects.



FIGURE 4.6: Task with more or less

In the last task of the game, a squirrel teaches the digits from 1 to 9 to the child. The system picks up a number, then puts it onto the stage and ask the user to identify it. While most children learn to recognize some numbers (1-2) at the age of 3, number recognition mainly adopted by children between 4 and 6 years old.



FIGURE 4.7: Find the number

4.4 The Farm

The Farm is a multimodal game developed in Flash technology [3]. The game is proper for 3-6 years old children. The game's target is to move the animals inside the farm by recognizing their voice. The main character 4.8 is a farmer who interacts with the child.



FIGURE 4.8: The Farmer

4.4.1 Description

First of all there is a welcome screen 4.9 where the child or a supervisor selects the language(Greek-English) and the screen mode(full screen). The user may as well watch a clip which includes some instructions for the microphone's adjustment. In this initial stage, you have to use the mouse.



FIGURE 4.9: Welcome Scene of Game

Next, there is a help mode. During this stage 4.10, the child becomes familiar with animals and the usage of mouse. By “clicking” upon an animal, child watches some kind of animation(e.x. a dog barking and moving its tail). This mode may be useful for children under 5 years old. A supervisor can help the child to learn the mouse device and the animals which will later try to identify. The child can either use speech or the mouse as the main modalities. In case that the child cannot handle the mouse but has the tendency to touch the screen, the supervisor can replace the touches with the mouse clicks (pseudo-touch mode). After the help mode, the main game begins. The game



FIGURE 4.10: The Farm

picks up an animal and plays the corresponding sound. The agent asks from the child to find the animal. In case of successful recognition, the animal walks toward the farm. Otherwise, the farmer reports the wrong answer and the animal keeps being out of the farm. The game ends-up when the child succeeds in putting all the animals inside. The farmer congratulates the user on finishing the task and closes the farm’s door.



FIGURE 4.11: The end of game

4.4.2 Software Analysis

In this section, the game is described in a software-oriented way. First of all, we will discuss about the state 4.12 transition logic of the game. The main states of the game are:

- Help

There are three ways to leave this state and begin the main game:

- Passing over all animals in order to listen at them.
- Waiting inactive for some seconds.
- “Clicking” upon the farmer agent.

- Animal Selection

In this state, the system picks up randomly one of the animals left onto the scene.

- Animal Animation & Recognition

The game plays the animal’s animation and wait for the user’s answer. The agent asks the child to recognize the animal. The process repeats when a timeout passes waiting for the child’s response. Two are the modalities that the game provides to the user:

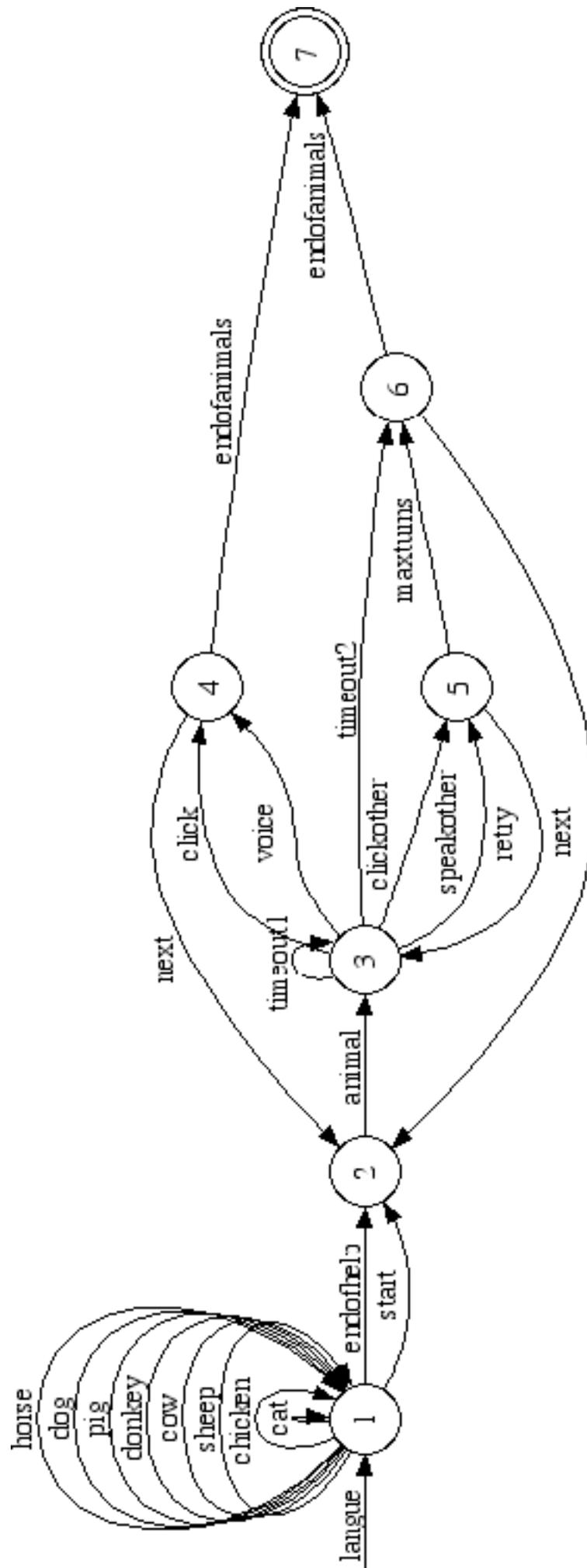
- Mouse. For example you can “click” the cow if you hear *mmmmmm*.
- Speech. You may as well speak out “It’s the cow”.

- Success

In this state, the embodied agent congratulates user on finding the correct answer. Then, the animal walks towards the farm and finally disappears from the scene. The system transits either to the previous state “Animal Selection” choosing the next animal or to “The End” of game in case that animal was the last one.

- Failure

On the other hand, if the child chose a wrong animal, the farmer would mention this fact and then prompt the child to retry. After three efforts, the farmer says the correct (state “Answer”) animal which then disappears from the scene. The system then transits to either the “Animal Selection” state or to “The End”.



The next diagram 4.13 describes a typical example of a game session. The user selects a language and starts the game. He passes through the help mode to the main one where he hears sounds and give some answers. The farmer interacts with the child though some prompts. The procedure repeats and finally the games ends.

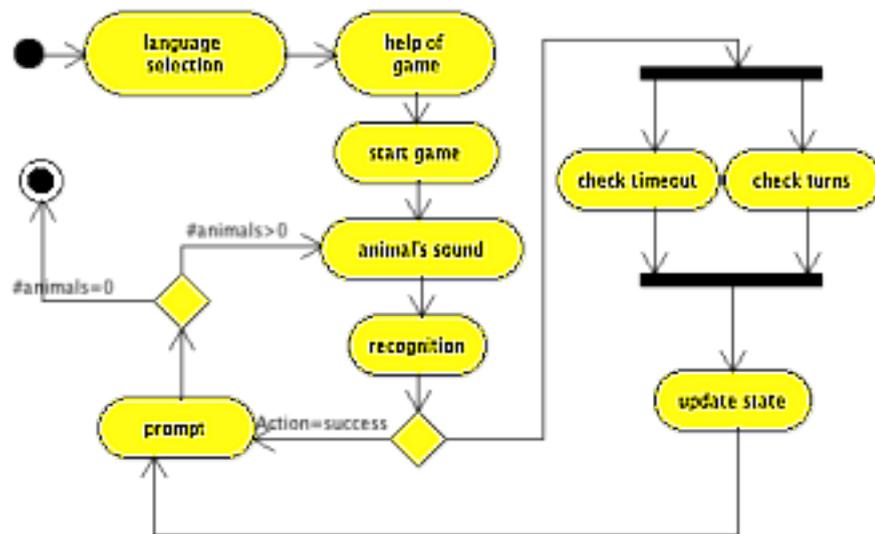


FIGURE 4.13: User Activity Diagram

An insight into the core of the game which includes the most important functions can be approached by the Class Diagram 4.14.

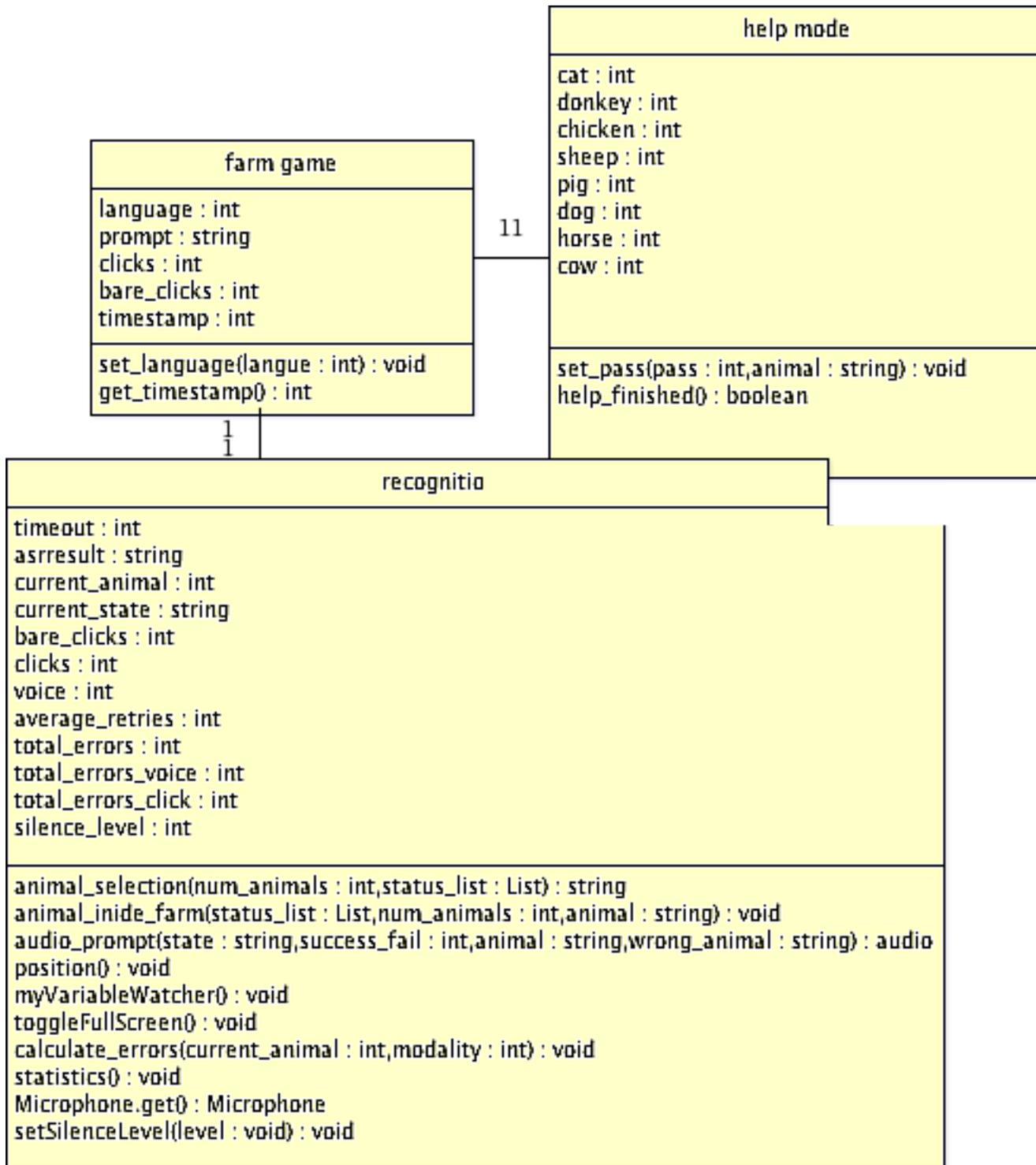


FIGURE 4.14: Class Diagram

4.5 Speech Synthesis

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech synthesizer, and can be implemented in software or hardware. A text-to-speech (TTS) [4.15](#) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely “synthetic” voice output. The quality of a speech synthesizer is judged by its similarity to the human voice, and

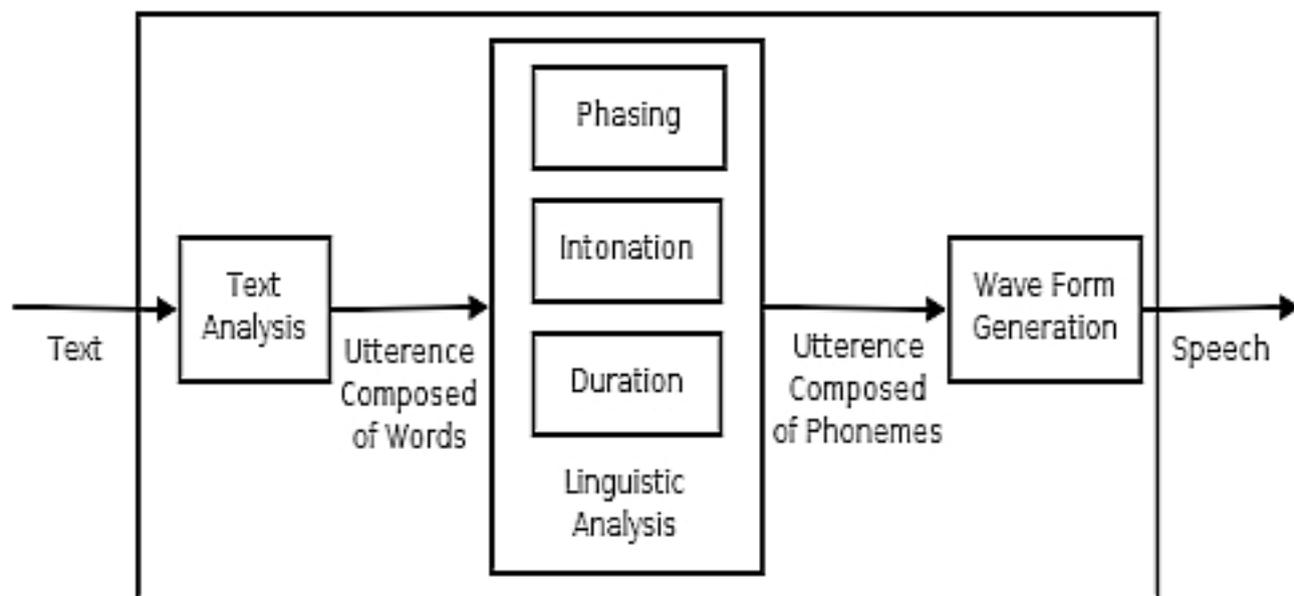


FIGURE 4.15: A Text To Speech System

by its ability to be understood. An intelligible text-to-speech program allows people with visual impairments or reading disabilities to listen to written works on a home computer. Many computer operating systems have included speech synthesizers since the early 1980s.

In the platform, we incorporated pre-recorded audio prompts. These prompts were used by the agents to communicate with the child in a natural way. Some games utilized free TTS software [\[4\]](#) to create these prompts whereas others made use of natural human voice.

4.6 Voice Activity Detection

Voice activity detection 4.16 is an algorithm used in speech processing where the presence or absence of human speech is detected in regions of audio. The process of separating conversational speech from silence, music, noise or other non-speech signals is a function of the activity detector. The main uses of VAD are in speech coding and speech recognition. Voice activity detection(VAD) may not only indicate the presence or absence of speech, but also other aspects of the speech, for example whether the speech is voiced, unvoiced or sustained. Voice activity detection is usually language independent.

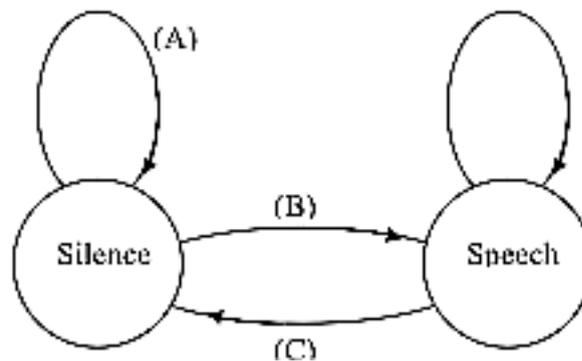


FIGURE 4.16: The Voice Activity Detection

The primary function of a voice activity detector is to provide an indication of the presence of speech in order to facilitate speech processing as well as possibly providing delimiters for the beginning and end of a speech segment. It was first investigated for use on time-assignment speech interpolation (TASI) systems. VAD is an important enabling technology for a variety of speech-based applications. For these purposes, there have been proposed various VAD algorithms that trade off delay, sensitivity, accuracy and computational cost.

For a wide range of applications such as digital mobile radio, Digital Simultaneous Voice and Data (DSVD) or speech storage, it is desirable to provide a discontinuous transmission of speech-coding parameters. Advantages can include lower average power consumption in mobile handsets, higher average bit rate for simultaneous services like data transmission, or a higher capacity on storage chips. However, the improvement depends mainly on the percentage of pauses during speech and the reliability of the VAD used to detect these intervals. On the one hand, it is advantageous to have a low percentage of speech activity. On the other hand clipping, that is the loss of milliseconds of active speech, should be minimized to preserve quality. This is the crucial problem for a VAD algorithm under heavy noise conditions.

Some typical VAD applications are:

- VAD is an integral part of different speech communication systems such as audio conferencing, echo cancellation, speech recognition, speech encoding, and hands-free telephony.
- In the field of multimedia applications, VAD allows simultaneous voice and data applications.
- Similarly, in Universal Mobile Telecommunications Systems (UMTS), it controls and reduces the average bit rate and enhances overall coding quality of speech.
- In cellular radio systems based on Discontinuous Transmission (DTX) mode, VAD is essential for enhancing system capacity by reducing co-channel interference and power consumption in portable digital devices.

One controversial application of VAD is in conjunction with predictive dialers and telemarketing firms. In order to maximize agent productivity, telemarketing firms set up predictive dialers to call more numbers than they have agents available, knowing most calls will end up in either “Ring - No Answer” or answering machines. When a person answers, they typically speak briefly (“Hello”, “Good evening”, etc.) and then there is a brief period of silence. Answering machine messages usually contain 3-15 seconds of continuous speech. By setting VAD parameters correctly, dialers can determine whether a person or a machine answered the call, and if it’s a person, transfer the call to an available agent. If it detects an answering machine, the dialer hangs up.

In our platform, each one game integrate some kind of code which conceives the transition from “speech” mode to “silence” and vice-versa. It then notify the application manager to either start the recording procedure or stop it.

4.7 The WoZ Part

The Wizard of Oz (WoZ) [4.17](#) technique is an experimental evaluation mechanism. It allows the observation of a user operating an apparently fully functioning system whose missing services are supplemented by a hidden wizard. The user is not aware of the presence of the wizard and is led to believe that the computer system is fully operational. The wizard observes the user through a dedicated computer system connected to the observed system over a network. When the user invokes a function that is not available in the observed system, the wizard simulates the effect of the function. Through the observation of users’ behavior, designers can identify users’ needs when accomplishing a particular set of relevant tasks and evaluate the particular interface used to accomplish the tasks.

This technique has primarily been used to study natural language interfaces. With recent advances in interactive media, multimodal user interfaces are becoming popular but our current understanding on how to design such systems is still primitive. In the absence of generalizable theories and models, the WOz technique is an appropriate approach to the identification of sound design solutions.

Most of existing WOz systems have been primarily developed to study the usage of natural languages for retrieval information systems. Telephone information services such as telephone directories, flights or trains information and reservation services, have been an interesting field for experiments . The experimental setup is quite simple: the wizard answers phone calls and pretends callers are talking to an automatic information system. In order to give callers the illusion that they are actually talking to a computer, the wizard's voice is filtered through a distortion system (e.g., a vocoder) that gives the voice a robotic flavour. Questions and answers are tape-recorded for later manual transcription and analysis. Other case studies involve databases or advisory systems interrogation as well as dialogues with expert systems. Most of them aim at collecting vocabulary corpus in order to tune and augment the robustness of spoken or written natural language recognizers.

In our system, the WoZ component was deployed wholly in JAVA and replaced the ASR component which was described above in the platform structure. Each game uses the VAD component to notify the presence of speech to the manager which streams the audio to the Wizard of Oz interface. It then replays the sound to the wizard who chooses the appropriate answer and sends it back to the game. For the synchronization of the various components (game with vad, manager, wizard) to be achieved, the Wizard of Oz application has a text flag (**wait** - **give answer**) which notifies the wizard when to give the answer.

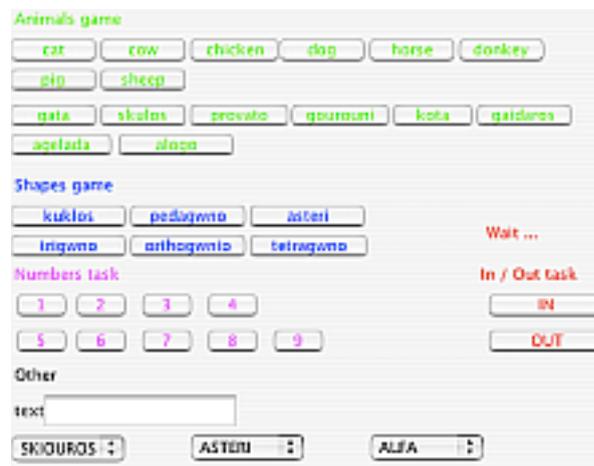


FIGURE 4.17: The Wizard GUI

Chapter 5

Evaluation

5.1 Introduction

Historically, the development of computer interfaces has been a technology-driven phenomenon. New multimodal interfaces are composed of recognition-based technologies that must interpret human speech, gesture, gaze, movement patterns, and other complex natural behaviors.

The performance of a multimodal system can be assessed through human factors analyses and evaluation of the acquired multimodal data. In case that the multimodal environment handles speech recognition, there are several ways of evaluating the speech modality.

5.1.1 Evaluation of Speech Recognition

The performance of speech recognition systems is usually specified in terms of accuracy and speed. Accuracy may be measured in terms of performance accuracy which is usually rated with word error rate (WER), whereas speed is measured with the real time factor. Other measures of accuracy include Single Word Error Rate (SWER) and Command Success Rate (CSR).

Most speech recognition users would tend to agree that dictation machines can achieve very high performance in controlled conditions. There is some confusion, however, over the interchangeability of the terms “speech recognition” and “dictation”.

Commercially available speaker-dependent dictation systems usually require only a short period of training (sometimes also called ‘enrollment’) and may successfully capture continuous speech with a large vocabulary at normal pace with a very high accuracy.

Most commercial companies claim that recognition software can achieve between 98% to 99% accuracy if operated under optimal conditions. ‘Optimal conditions’ usually assume that users:

- have speech characteristics which match the training data,
- can achieve proper speaker adaptation, and
- work in a clean noise environment (e.g. quiet office or laboratory space).

This explains why some users, especially those whose speech is heavily accented, might achieve recognition rates much lower than expected.

Limited vocabulary systems, requiring no training, can recognize a small number of words (for instance, the ten digits) as spoken by most speakers. Such systems are popular for routing incoming phone calls to their destinations in large organizations.

5.1.2 Evaluation of Multimodal Systems

The performance of a multimodal system can be assessed through human factors analyses and evaluation of the acquired multimodal data. The latter requires tools that are able to monitor user input, system feedback, and performance of the multimodal system components. Such tools can bridge the gap between observational data and the complex process of the design and evaluation of multimodal systems. These tools can reveal the properties of different modalities, as well as when users are likely to interact multimodally and how their multimodal input is integrated and synchronized. Moreover, multimodal interaction could be evaluated checking the potential minimization of errors and the improvement of error handling.

Experience has shown that the design and evaluation of multimodal interactive systems poses a complex, multidisciplinary problem. It requires a collaboration between researchers from psychology and cognitive science, up to computer science and artificial intelligence.

Generally speaking, it is very difficult to make sense of the data recorded in multimodal interaction systems. Even if the interaction strategy is designed to constrain the user actions, multimodal interaction appears to offer many alternative ways to approach the goal. This large degree of freedom is especially important in the analysis of interactions with naive subjects, who lack the telepathic knowledge of the system’s expectations that the system designers do have, and that helps tremendously in finding the most efficient interaction strategy and to avoid situations in which the system may not be

robust. In addition, objective data (the input and output of the individual modules in a system, including time stamps attached to actions of the system and the user) form a kind of cascade. Apart from some objective metrics that could depict the interface efficiency, subjective metrics (mainly derived from questionnaires) may demonstrate the user satisfaction. Some typical objective metrics may be:

1. Number of tasks completed successfully
2. Average (number of turns per task/turn duration)
3. Evaluation time
4. ASR WER/CER in case that the system incorporates speech as a modality

In order to analyze the performance of individual modules, for each module its complete set of input and output messages must be considered. For speech and pen input this involves manual annotation of the physical input signals. Speech input must be transcribed verbatim, as well as in the form of the concept values expressed by the words. For pen input, coordinate streams must be annotated with the semantic labels that are relevant in the specific application. To assess the performance of modules that have no direct relations with physical input or output, such as fusion, which receives symbolic input of the speech and pen input processors and passes symbolic data to the dialog action manager, input-output pairs must also be annotated for correctness (or type of error).

5.2 Our Approach

Our methodology targets toward some first feedback from child-computer interaction area. It is beyond saying that the evaluation of the acquired multimodal data and human factors analyses could be further extended in future when the corpora would be greater. In order to evaluate our platform, we took into consideration both objective and subjective criteria.

User Satisfaction

In our platform, user satisfaction was measured upon some subjective metrics. All the children who took part in the WoZ experiment participated in an exit interview [B](#) wherein subjective impressions about the game and the interface were obtained. Sample questions that they were asked include: 1) Did you like talking with the game's characters? 2) Which character did you like the most? 3) Which game was

the best? 4)Did you like the graphics of the games? 5)Did the characters pay attention to you? 6)Did you understand what each character said? 7) Were the acoustics disturbing? **13 children** participated in the WoZ experiment playing the Farm game.

Assessing Multimodality

First of all, we split the total **27 Game Sessions** 5.1 to three parts according to the modality used :

- Speech
- Multimodal Fusion
- Mouse (including pseudo-touch mode ¹)

From the total 27 game sessions, there were 11 where only speech modality was used, 8 where children made use of the mouse and the pseudo-touch mode and 8 where children played using all the possible modalities given.

Sessions		
Speech	Click-Touch	Multimodal
11	8	8

TABLE 5.1: Game Sessions per Modality

Next, we used the data from the above game sessions to assess each modality considering a set of metrics. The objective metrics that we derived from games are:

1. *Time completion of game*
2. *Modality usage*
3. *Errors per Modality*

Taking into consideration the above sessions 5.1, we calculated the mean of the above metrics.

¹where a supervisor replaces the user's touch with mouse clicks

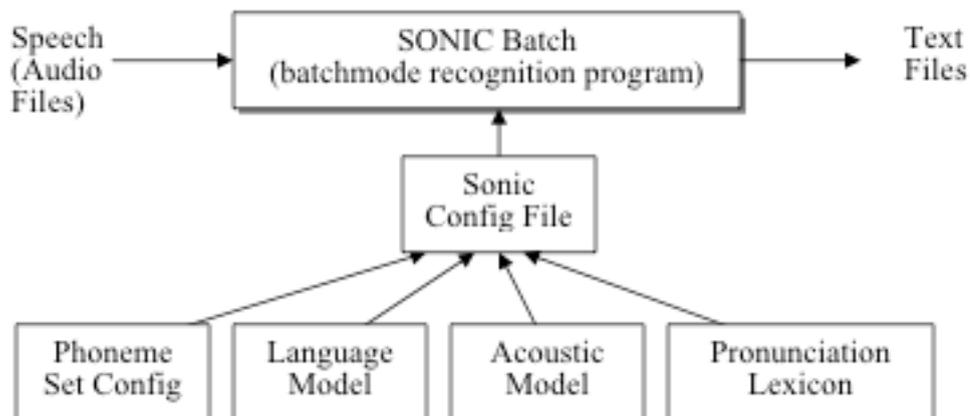
Speech Recognition Performance

Finally, we utilized the total **405 audio 5.2** and the corresponding transcript data from the WoZ experiment to evaluate the performance of an embedded recognizer to the platform using the SONIC [5]. As we have already described in *Chapter 3*, for speech recognition, it is acquired both a language and an acoustic model. We

Current Corpus		
Farm	Shapes	Numbers
96	75	234

TABLE 5.2: Corpus Characteristics

utilized the transcript data to deploy a language model.



1. Procedure:

- (a) Convert corpora (farm game) into sequence of sentences.
- (b) (use CMU/Cambridge Toolkit [6]) Turn the training text into a list of id N-grams (N-grams with each word mapped to an integer ID, which is zero for out of vocabulary words).
- (c) Convert the IDN gram stream into a language model file. This is a Katz backoff trigram language model with witten_bell discounting for smoothing the probabilities
- (d) Finally, we compress the ASCII language model into a binary file format for Sonic

Next, the procedure that we followed using the audio data can be described as:

1. Model Initialization:

- (a) Compress Lexicon

- (b) Build Letter-to-Sound mappings
 - (c) Align data using US English Models + Phoneme mapping
 - (d) Train initial language-dependent models (2/3 of corpus)
2. Model Refinement:
 - (a) Re-align using language-dependent models
 - (b) Retrain 2/3 of corpus
 3. Model Testing: Test the remaining 1/3 of corpus (we used the NIST Sclite Speech Recognition scoring package [7] to compute the word error rate)

We then interchanged the testing corpus with another portion of training data and followed this procedure another two times.

We evaluated the performance of speech recognition by measuring some characteristics such as:

1. Correct Recognitions
2. Substitutions
3. Deletions
4. Insertions

In the next final *Chapter*, we discuss the results from the above three areas.

Chapter 6

Results & Conclusions

6.1 Results & Discussion

In this final *Chapter*, we analyze the results given from the methodology that was described in *Chapter 5*. We have split the analysis into 3 corresponding parts in accordance with the evaluation strategy we followed. Finally, we summarize the conclusions of this thesis and outline interesting directions for future work.

User Satisfaction

Firstly, the children gave very high preference to the speech interface. Moreover, it is an interesting fact that all the children declared that they understood the agent. The usage of TTS did not appear to be a distract for the children. Furthermore, **11 out of 13** children liked an **animated agent 6.1** in comparison with a static one. In the above diagram, you should take into consideration the fact that all the characters apart from the *teacher* incorporate some kind of movement. For example, the farmer moves up and down his arms when talking while the dog moves around its tail when barking. This shows a strong preference towards interactive agents that can moved around and looked as if they were alive.

Further, **7 out of 13** children used the mouse(or the pseudo-touch mode ¹). This virtual touch mode was extensively used in two cases where the users could not speak out their answers to the system. These two little children succeed in playing the farm and completing 3 games each one. The former child **6.2** could not speak whereas the latter was bilingual and cannot handle efficiently the Greek language. The fact that the child becomes accustomed to the virtual-touch modality is illustrated by the decrease of both the errors and the time it takes to finish the game.

¹where a supervisor replaces the user's touch with mouse clicks

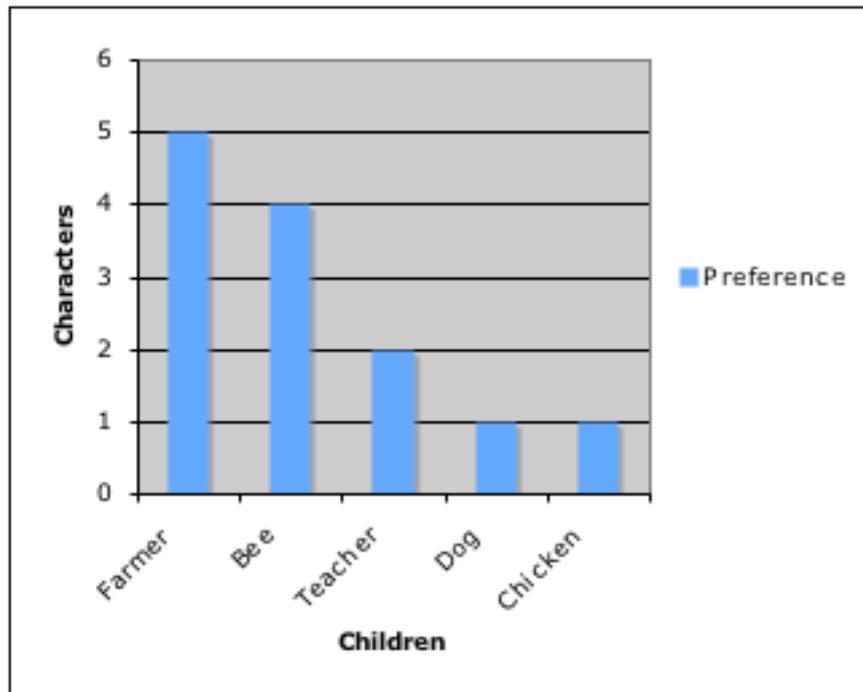


FIGURE 6.1: Preference per Character

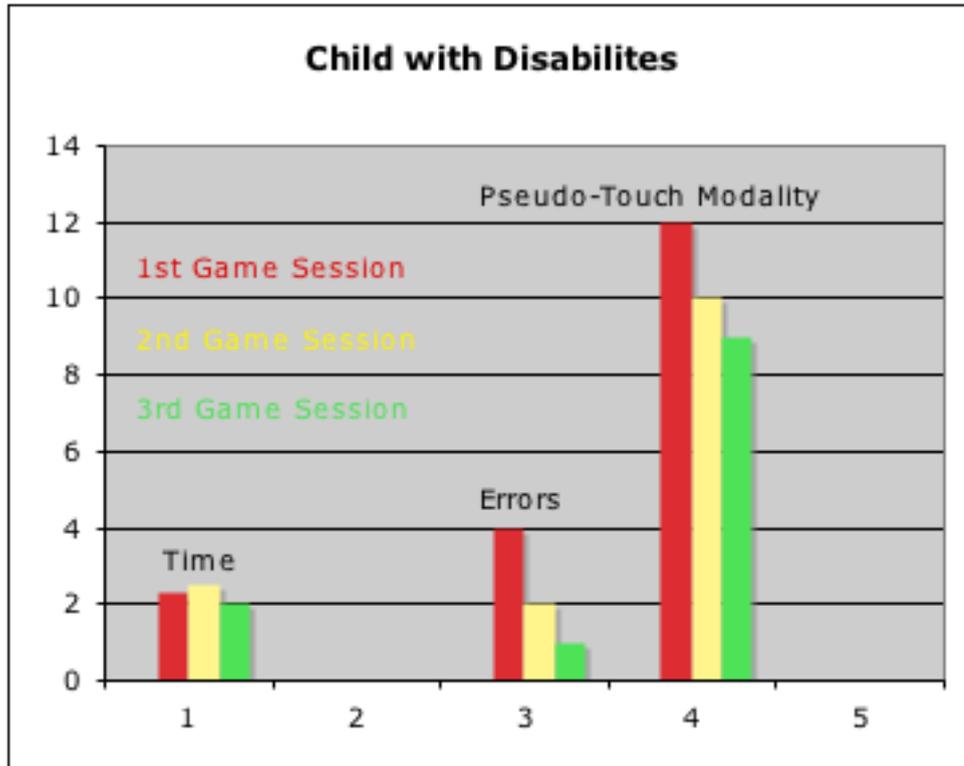


FIGURE 6.2: Child with Disabilities

In the last turn, with 9 actions (touches), the child identify the 8 animals of the farm in almost 184 seconds making only one wrong choice. A similar behavior was observed for the latter child. The flexible choice of input modality (speech, buttons-virtual touch) made the application easy to use even for kids with special needs.

Moreover, about 61% of the children enjoyed the use of a headset microphone. In summary, user experience results were promising for the inclusion of voice, touch and animations as one of the interaction modalities in the design of interactive applications for children. Finally, It is interesting to compare the relation between the number of games won and the ratings. Losing a game had a significant negative effect on the rating of the game. This fact shows the necessity of adapting the game to user level of knowledge. Another factor that was annoying for some kids was the delay in system response when using speech which can be mainly attributed to poor voice activation detection in connection with a noisy environment.

Multimodality

Given the game sessions 5.1, we concluded that the synergy of different modalities reduces the errors that a child makes. The next diagram 6.3 demonstrates that fusion of modalities minimizes the errors. The speech modality is the worst concerning the errors. On the other hand, although multimodality prevails with criteria the errors, there seems to be an overhead on overall time 6.4 that the child spent when playing.

Sessions		
Speech	Click-Touch	Multimodal
11	8	8

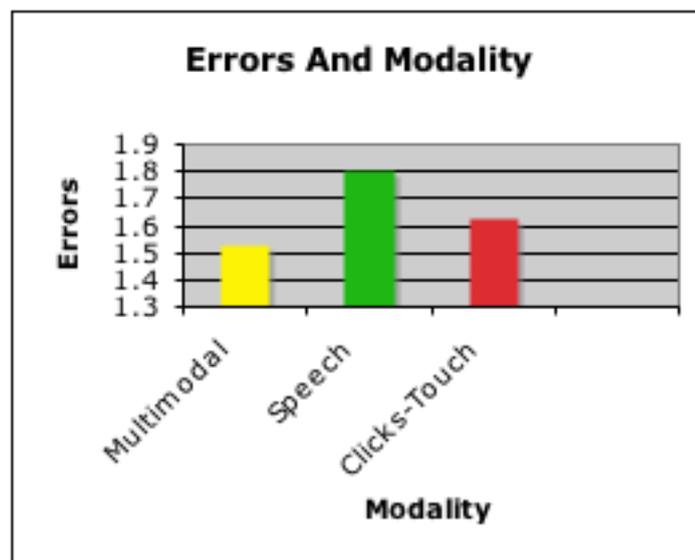


FIGURE 6.3: Errors per Modality

Further, clicks and touch seems to be the best modality concerning the time. Using speech, children consumes on average more time than using clicks-touch. This fact could be attributed to the delay that the voice activity detection component adds. In addition, the transition from speech to silence appears to be much less robust than the transition from silence to speech mode. The audio waveforms clearly depict this observation.

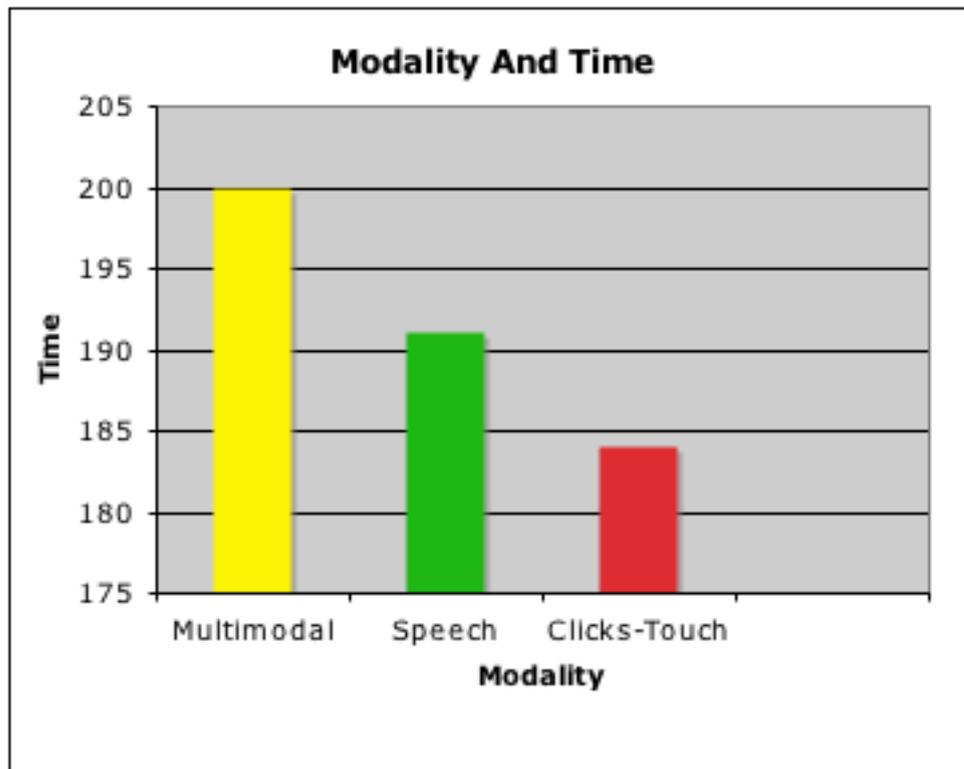


FIGURE 6.4: Time per Modality

Recognition Performance

At first glance, it seems that the recognition does not work properly with the existing acoustic and language model. However, we should take into consideration some factors that maybe justify this performance. First of all the existing corpus is somewhat limited. **405** multimodal data have already be collected by the platform. Furthermore, after thoroughly examination of the audio data, we observed several phenomena which deteriorated the recognition process. We examined the corpus test of the children where a score less than 40% was achieved. In these data, children either **over-articulated** the words or **co-articulated** them. Moreover, some kids were whispering instead of speaking out their answer. Last but not least, there were a lot of audio where the noise covered the speech.

Current Corpus		
Farm	Shapes	Numbers
96	75	234

Test 1									
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err	
child 1	2	2	100.0	0.0	0.0	0.0	0.0	0.0	0.0
child 2	8	8	37.5	50.0	12.5	0.0	62.5	62.5	
child 3	17	17	47.1	35.3	17.6	0.0	52.9	52.9	
Sum/Avg	27	27	48.1	37.0	14.8	0.0	51.9	51.9	
Mean	9.0	9.0	61.5	28.4	10.0	0.0	38.5	38.5	
S.D.	7.5	7.5	33.7	25.7	9.1	0.0	33.7	33.7	
Median	8.0	8.0	47.1	35.3	12.5	0.0	52.9	52.9	

FIGURE 6.5: Evaluation test 1 for recognition

Test 2									
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err	
child 4	4	4	25.0	75.0	0.0	0.0	75.0	75.0	
child 5	10	10	20.0	50.0	30.0	0.0	80.0	80.0	
Sum/Avg	14	14	21.4	57.1	21.4	0.0	78.6	78.6	
Mean	7.0	7.0	22.5	62.5	15.0	0.0	77.5	77.5	
S.D.	4.2	4.2	3.5	17.7	21.2	0.0	3.5	3.5	
Median	7.0	7.0	22.5	62.5	15.0	0.0	77.5	77.5	

FIGURE 6.6: Evaluation test 2 for recognition

Test 3									
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err	
child 6	23	23	30.4	34.8	34.8	0.0	69.6	69.6	
child 7	12	12	66.7	25.0	8.3	0.0	33.3	33.3	
Sum/Avg	35	35	42.9	31.4	25.7	0.0	57.1	57.1	
Mean	17.5	17.5	48.6	29.9	21.6	0.0	51.4	51.4	
S.D.	7.8	7.8	25.6	6.9	18.7	0.0	25.6	25.6	
Median	17.5	17.5	48.6	29.9	21.6	0.0	51.4	51.4	

FIGURE 6.7: Evaluation test 3 for recognition

6.2 Conclusions

Since the system was designed for children users, heavy emphasis was placed on the interface design. Indeed it was found that using animated sequences to communicate information and adding “personality” to the interface significantly improved the user experience. In addition, the flexible choice of input modality (speech, buttons-pseudo touch) made the application easy to use even for little children with disabilities. Moreover, there seems to be a negative relation between multimodality and errors in gaming while it appears to be the fact that overall time increases when a child exploits all the modalities given.

The WoZ experiment in a gaming environment provided data for creating novel language models and understanding strategies for dialog systems. We expect that data acquired will help further conversational child machine interaction technology.

6.3 Future Work

The platform could be improved in many ways. First of all, more modalities could be added in each game. For example, the touch modality might proved to be very interesting for the kids. Some kids had the tendency to touch the screen in order to give their answer to the game.

Moreover, the Wizard of Oz should be replaced with an Automatic Speech Recognition System. This feature would reduce the response time which was disturbing for some children. Firstly, the usage of a grammar instead of language model may increase the system’s performance. Further, Vocal Tract Length Normalization or similar techniques would remove the speaker-dependent variations and thus model relevant details in the speech signal more accurate. Finally, a more sophisticated voice activity detection would further enhance the whole procedure of child-machine interaction through speech.

Furthermore, other features of our system not yet implemented include customizable application content (adaptation of game’s difficulty to child’s abilities). For example, the Farm could be expanded to include another task of matching the parent animal with the child one or the animal and the place where it lives. Finally, a customizable agent personality would provoke more naturalness in the child-machine interaction.

Acknowledgements

First and foremost I want to thank my advisor professor Alexandros Potamianos for his valuable guidance.

Next I want to thank Theofanis Kannetis (M.Sc) who was always there for me. He has developed the internet-based platform in which I worked my diploma thesis. To the staff at the Speech Processing and Dialog Systems laboratory, especially Elias M. Iosif , I send a heartfelt thank-you.

Last and surely not least, I want to acknowledge my wonderful friends and colleagues, some of whom spent hours with me in the laboratory. In particular I want to thank: George Skouma, Isidoro Rodomagoulaki, and Nikola Georgadoni, without whose company and guidance this diploma thesis would have taken longer time. I could not forget to send a big “thank you” to the teachers and the kids from 1st kindergarden of Kouni-pidiana.

It is always impossible to personally thank everyone who has facilitated successful completion of a project. To those of you who I did not specifically name, I also give my thanks for moving me towards my goal.

Appendix A

System Setup

A.1 Introduction

In this appendix, you will find instructions to setup the whole system. First of all, it is required to have a CVS client. It is suggested that you use SmartCVS. Moreover, it is supposed that you have already installed MySQL and Apache-Tomcat in your system.

A.2 Repository Connection

Firstly, it is required to establish a connection with the repository. To access a repository, you will need a valid path and password.



The image shows a configuration window for connecting to a CVS repository. At the top, the 'CVS-Location' is set to 'xt:tmos@orion.telecom.tuc.gr:/home1/users/tmos/repository/'. Below this, there are three radio button options: 'Use CVS-Location as profile name' (which is selected), 'Use this profile name: s@orion.telecom.tuc.gr:/home1/users/tmos/repository/' (with the text in a text box), and 'Use proxy'. Underneath, there are two radio button options for 'Authentication': 'Password' (selected) and 'Public/Private Key'. At the bottom, there is a 'Password:' label followed by a text box containing eight asterisks.

FIGURE A.1: A connection to the repository.

A.3 Checkout Modules

After a successful connection be established, you should checkout certain modules to setup and run the whole system.



FIGURE A.2: Modules of the repository.

A.3.1 Controller Server

Inside this module, there are files associated with the Controller Server component. Most important are:

1. `ControllerServerGUIWOZ.java`
2. `wizardofoz.java`
3. `runcotroller`

You may execute `runcontroller` [A.3](#) for unix-based systems in order to compile and run the controller server along with the wizard graphical interface.



FIGURE A.3: Compile and run Controller with Wizard.

A.3.2 Application Manager

In this module there are three important files:

1. temp.java
2. tuckey
3. createapplet

You may execute createapplet for unix-based systems in order to compile the temp.java, create the applet and finally sign it as safe.

```
koukle mouuuu: ./createapplet
Note: temp.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
added manifest
adding: temp$1.class(in = 546) (out= 391)(deflated 28%)
adding: temp$2.class(in = 3700) (out= 2036)(deflated 44%)
adding: temp.class(in = 4673) (out= 2453)(deflated 47%)
Enter Passphrase for keystore: telecom
```

FIGURE A.4: Compile Application Manager and create applet .

A.3.3 Children Interface

The front-end interface is included in the module. There are two subfolders. The first one is named **children** and the latter **persistence-layer**. The contents of the first subfolder should be copied to the *tomcat-HOME/webapps/ROOT/children* path while persistence-layer's contents should be copied to the *.../WEB-INF/classes/web/children* and *.../web/persistence*

A.4 Scripts Module

Check out scripts module and run `createdatabase.sql` to create the database. Be aware of using the appropriate username and password when connecting to the schema. You may also use certain other modules related with SONIC [5] speech recognizer. In order to use the recognizer you will need:

1. One or more audio files to process
2. A phoneme configuration file which defines the phoneme units in your language
3. A language & acoustic model
4. A lexicon containing word pronunciations for words contained in the language model

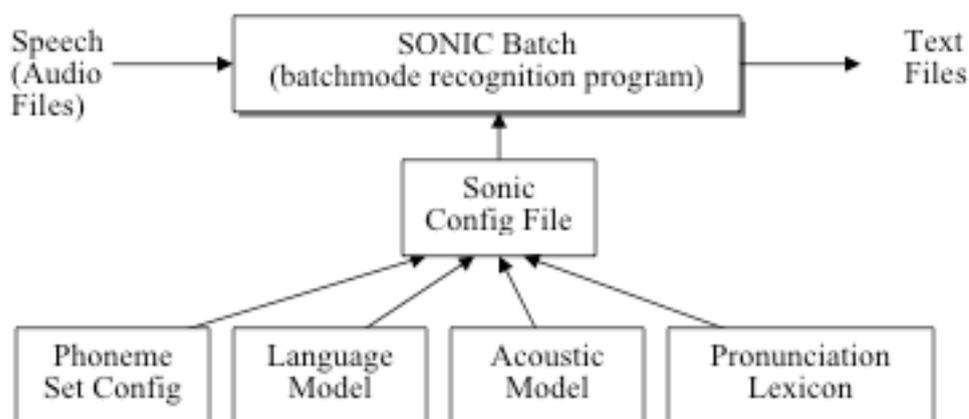


FIGURE A.5: Example inputs required to perform batch-mode recognition in SONIC. A single configuration file is used to setup the recognition job. The SONIC configuration file provides the listings of the decoder settings, phoneme set, language model, acoustic model, and pronunciation lexicon.

A.4.1 Phoneme Configuration File Format

This file defines the phoneme set used by the recognizer. The phoneme configuration file begins with an ASCII header and then lists the valid phoneme symbols to be used by the recognizer.

A.4.2 Pronunciation Lexicon Format

The lexicon contains the pronunciations for the words in the recognition vocabulary. Words are listed followed by their phoneme sequence. Alternate pronunciations are denoted using open parenthesis followed by an integer alternate number followed by a closing parenthesis. Optional normalized log-probabilities can be assigned to pronunciation variants using bracketed expressions.

For speed of access by the recognizer, one can compress an ASCII text version of the lexicon into a binary file. This can be accomplished using the “lex_encode” program provided with SONIC. The program requires the phoneme configuration file (previously described) as input:

```
lex_encode Pp phonemeset.cfg Pi mylex.txt Po mylex.bin
```

```
<numphones> 55
<phonelist>
AA
AE
AH
...
</phonelist>
```

FIGURE A.6: Example of a Phoneme Configuration File

A.4.3 SONIC Configuration File

A configuration file is used to establish the basic settings of the recognizer for decoding. It is an ASCII text file that has a set of parameters followed by arguments. In general, the SONIC configuration file contains the following items,

- Location of the acoustic model to be used during recognition
- Location of the language model or finite state grammar to use
- Location of the pronunciation lexicon
- (optionally) a pointer to a control file containing a list of audio files to process
- Recognizer settings such as search beams, pruning settings, language model scale factors

```

-batch_file      chimp.ctl
-output_file    ../../outdir/chimp.hyp
-log_file       ../../outdir/chimp.log
-acoustic_mod   ../../outdir/models/bin/chimp-gm.mod
-langmod_file   ../../outdir/chimp-lm.bin
-dictionary     wsj-5k.lex
-phone_config   phoneset.cfg
-word_end_beam  80.0
-word_entry_beam 200.0
-state_beam     200.0
-word_trans_penalty -50.0
-state_dur_scale 3.0
-sample_rate    8000.0
-max_active_states 12000
-auto_end_point 1
-end_point_padding 125
-filler_file    chimp.filler
-filler_penalty 0.0

```

FIGURE A.7: SONIC configuration file options

A.4.4 Acoustic Model

The recognizer assumes that the input audio is in 16-bit linear PCM format (no header). The audio files should have their bytes swapped in the native format of the machine running the recognizer. If you are working with Microsoft wav files, you can strip the header from the file using the `soxS` program for Windows or Unix:

`sox infile.wav Pw Ps infile.raw`. In our case, run `./wavToraw.csh` (use `sox` utility [8]) and then `./port-sonic.csh`

1. Model Initialization:

- (a) Compress Lexicon (`./step0-kb.csh`)
- (b) Build Letter-to-Sound mappings
- (c) Align data using US English Models + Phoneme mapping (`./step1-align.csh`)
- (d) Train initial language-dependent models (`./step2-mlf.csh`, `./step3-train.csh`)

2. Model Refinement:

- (a) Re-align using language-dependent models (`./step4-realign.csh`)
- (b) Retrain

3. Model Testing:

- (a) copy games session folders with `.raw`, `.txt`, gender files in the `datatest` folder

- (b) `./create_ctl.csh , ./create_ref.csh`
- (c) adjust `sonic_config.female` and `sonic_config.male`
- (d) replace `phoneset.cfg` and `wsj-5k.lex` with yours
- (e) `./step1-batchmode.csh`
- (f) `./step2-results.csh` (use the NIST Sclite Speech Recognition scoring package [7] to compute the word error rate)

A.4.5 Language Model

`./lmanimals.csh` and then `./lmnumbers.csh`

1. Procedure:

- (a) Convert corpora(farm, numbers, letters) into sequence of sentences.
- (b) (use CMU/Cambridge Toolkit [6]) Turn the training text into a list of id N-grams (N-grams with each word mapped to an integer ID, which is zero for out of vocabulary words).
- (c) Convert the IDN gram stream into a language model file. This is a Katz backoff trigram language model with witten_bell discounting for smoothing the probabilities
- (d) Finally, we compress the ASCII language model into a binary file format for Sonic

A.4.6 Game Metrics

`./prepareplots` extracts information from logfiles of farm-game sessions and calculate some metrics.

```
USER_kostas
bare_clicks 0
clicks 5
voice 5
average_retries 0.25
total_errors_click 0
total_errors_voice 2
answers 0
NEXT_GAME
USER_ariadne
bare_clicks 0
clicks 1
voice 12
```

A.5 Connect as Client

In order to verify that your system is properly set up, connect to the platform as client.

A.5.1 Children Home Page

It is recommended that you use firefox as web browser. Keep in mind that both javascript and java should be activated. Moreover, in case that you have a firewall, it is required that you open TCP port 19999.

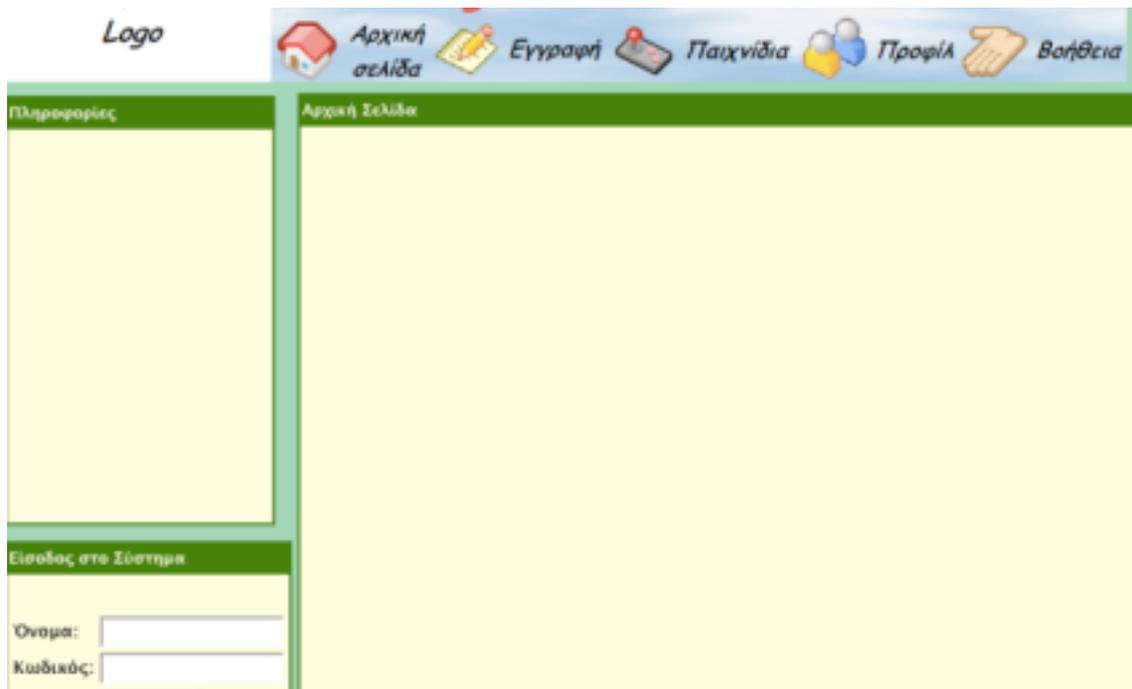


FIGURE A.9: <http://147.27.8.43:8080/children>

A.5.2 Registration Phase

Unless you have already registered to the system, this step is obligatory.



FIGURE A.10: Registration Button

A registration form on a light yellow background. It contains the following fields and labels:

- Όνομα Χρήστη:** Text input field containing "ονομαπαιδιου" with a red asterisk to its right.
- Κωδικός Πρόσβασης:** Text input field containing "***" with a red asterisk to its right.
- Επαλήθευση Κωδικού:** Text input field containing "***" with a red asterisk to its right.
- Ηλικία Παιδιού:** Dropdown menu showing "6" with a red asterisk to its right.
- Φύλο Παιδιού:** Dropdown menu showing "Κορίτσι" with a red asterisk to its right.

Below the fields is a button labeled "Εγγραφή". At the bottom, a red asterisk is followed by the text: "* Τα πεδία που σημειώνονται με αστερίσκο είναι υποχρεωτικά."

FIGURE A.11: Registration Form

A.5.3 Start Playing

After registration, you can choose a game [A.12](#). First of all, you must accept the applet [A.13](#) that runs in the background of the games page as safe. Furthermore, you must allow the flash game to gain access to your microphone [A.16](#).



FIGURE A.12: Games Page

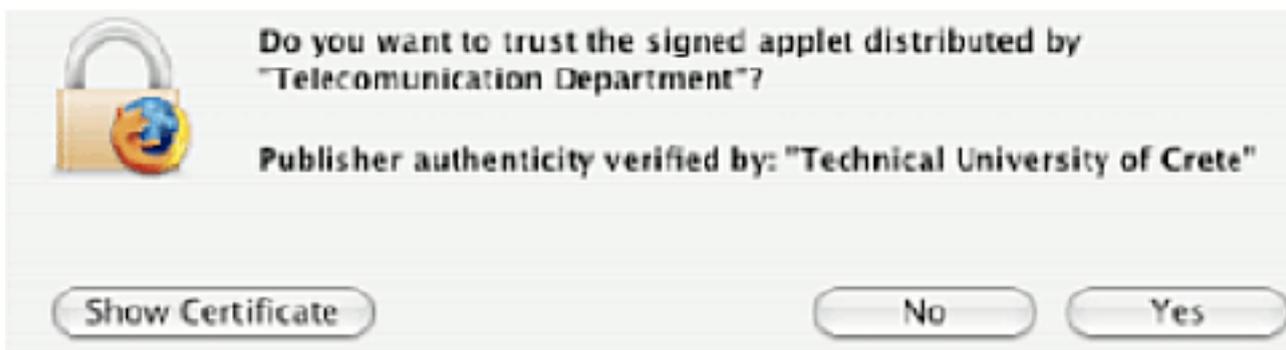


FIGURE A.13: Accept Applet as Safe

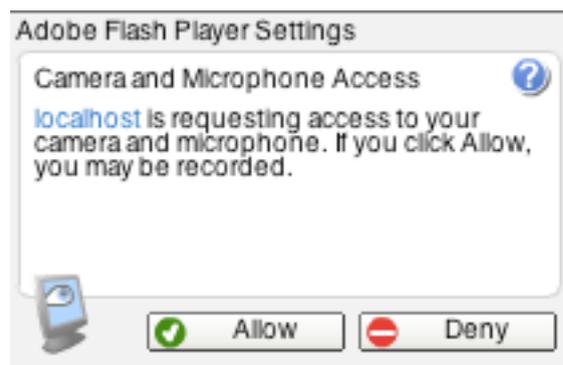


FIGURE A.14: Allow Flash to access Microphone

A.5.4 Microphone Adjustment

This last step is the most crucial. Each user should adjust the microphone carefully so as for the voice activity detection operate properly.

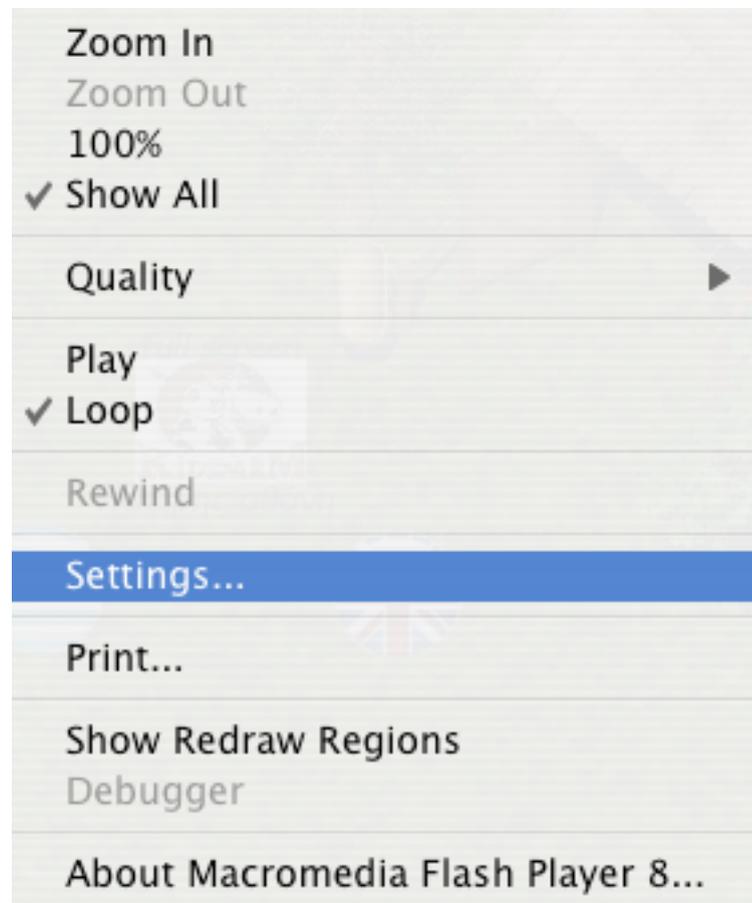


FIGURE A.15: Settings in Flash Game



FIGURE A.16: Microphone



FIGURE A.17: Microphone Adjustment

Appendix B

Questionnaire

1. Did you like talking with the game's characters?

2. Which character did you like the most?

3. Which game was the best?

4. Did you like the graphics of the games?

5. Did the characters pay attention to you?

6. Did you understand what each character said?

7. Were the acoustics disturbing?

8. Would you play again?

9. What did you dislike about the games?

10. Mouse Skill

- Bad
- Good
- Very Good

Bibliography

- [1] Alexandros Potamianos and Shrikanth Narayanan. Robust recognition of children's speech. *IEEE Transactions On Speech And Audio Processing*, 11(6):603–615, 2003.
- [2] Shrikanth Narayanan and Alexandros Potamianos. Creating conversational interfaces for children. *IEEE Transactions On Speech And Audio Processing*, 10(2):65–78, February 2002.
- [3] *Flash MX Tutorials*. Macromedia, Inc, 2002. URL http://www.adobe.com/support/flash/documentation/fl_mx_tutorials.html.
- [4] Loquendo. Text to speech system. URL www.loquendo.com/en/technology/TTS.htm.
- [5] Colorado. *SONIC: The University of Colorado Continuous Speech Recognizer*. University of Colorado. URL http://cslr.colorado.edu/beginweb/speech_recognition/sonic.html.
- [6] CMU. The cmu language modeling toolkit. URL <http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html>.
- [7] NIST. The nist scoring package. URL ftp://jaguar.ncsl.nist.gov/current_docs/sctk/doc/sclite.htm.
- [8] Sox project. The sox utility. URL <http://sox.sourceforge.net/>.