LanguageLand: A Multimodal Conversational Spoken Language Learning System

by

Vivienne C. Lee

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

August 2004

Author..
Department of Electrical Engineering and Computer Science
August 18, 2004

Certified by..
Stephanie Seneff
Principal Research Scientist
Thesis Supervisor

Certified by..
Chao Wang
Research Scientist
Thesis Supervisor

Accepted by...
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

This work is sponsored in part by the Cambridge MIT Institute, and in part by the Defense Language Institute under Air Force contract F19628-00-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

LanguageLand: A Multimodal Conversational Spoken Language Learning System

by

Vivienne C. Lee

Submitted to the
Department of Electrical Engineering and Computer Science

August 18, 2004

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

LanguageLand is a multimodal conversational spoken language learning system whose purpose is to help native English users learn Mandarin Chinese. The system is centered on a game that involves navigation on a simulated map. It consists of an edit and play mode. In edit mode, users can set up the street map with whichever objects (such as a house, a church, a policeman) they see in the toolbar. This can be done through spoken conversation over the telephone or typed text along with mouse clicks. In play mode, users are given a start and end corner and the goal is to get from the start to the end on the map. While the system only responds actively to accurate Mandarin phrases, the user can speak or type in English to obtain Mandarin translations of those English words or phrases. The LanguageLand application is built using Java and Swing. The overall system is constructed using the Galaxy Communicator architecture and existing SLS technologies including Summit for speech recognition, Tina for NL understanding, Genesis for NL generation, and Envoice for speech synthesis.

Thesis Supervisor: Stephanie Seneff
Title: Principal Research Scientist, MIT Spoken Language Systems Group

Thesis Co-Supervisor: Chao Wang
Title: Research Scientist, MIT Spoken Language Systems Group

Acknowledgments

Working on this thesis for the past year has been a pleasure. First of all, I would like to extend my gratitude to Stephanie Seneff, my thesis advisor, for guiding me all along the way and spending countless hours helping me create and debug the natural language component, both with me sitting in her office and even on her spare time at home. I find it hard to catch up sometimes to her fast typing and talking, but it made every work session efficient and worthwhile. I would like to thank her for her patience with my endless stream of emails (which sometimes contained trivial questions), all of which were replied to promptly and fully. I truly did not expect a thesis advisor to be able to spend so much time with her students, but Stephanie has surprised me with her effort to commit time and energy to students like myself.

I would also like to thank Chao Wang, my co-advisor, who has been available for help whenever I needed her. She is one of the few people who have smaller hands than mine, but they are also one of the fastest and most agile pair of hands I have ever encountered. She also spent many hours helping to pull my system together. I am especially grateful to her for putting together my bilingual recognizer for English and Chinese, and for teaching me how to record my own Chinese sentences. Her meticulous notes were an invaluable resource and helped me navigate the SLS systems with little prior knowledge about them. One thing I will always remember is the day when she was on a time constraint to pick up her son from daycare, but nevertheless stayed and helped me until the very last minute.

I could not have gotten a better mentor than Scott Cyphers. He wrote a thorough Communicator Tutorial which served as my orientation to SLS and helped me understand the Galaxy structure and message passing interface. I recall having a lot of problems early on with connecting my Java application to the natural language components, and Scott was always more than willing to let go of whatever he was doing and help me with my questions. In my early stages, I must have asked him for help at least a few times a day. Scott is truly the Java king. I thank him for writing the script to start all servers, which has saved me a lot of time in the past year. I also thank him for spending hours helping me set up a second laptop with the Galaxy system.

I would like to thank Rob Miller and Michael Collins, professors for my User Interface and Machine Learning Approaches for Natural Language Processing classes, respectively, for providing guidance and allowing me to incorporate my thesis into the final projects. I would also like to thank Janet Lai, my partner for the final project in the User Interface class, who helped me build the user interface for LanguageLand from the ground up.

Of course, I would like to thank all the people who have used and tested my system, including students from the User Interface class, my past Chinese and Japanese language professors, and friends. Special thanks to Danwen Chen, a current high school intern at SLS, who helped me test the final phase of LanguageLand.

Thanks also to Jim Glass for responding to my initial email seeking a thesis project, for directing me to Stephanie and Chao, and for giving me this invaluable opportunity. Marcia Davidson has been wonderful and helped me get acquainted with SLS. I am grateful to those who have contributed to the system either directly or indirectly, including Jonathan Lau, Julia Zhang, Jon Yi, John Lee, Lee Hetherington, Mitchell Peabody, and the vast array of past SLS members who helped build the components that I used in the system.

I am extremely thankful to my parents and brothers for giving me constant opportunities, confidence, and encouragement throughout my life and especially during the past five years. Without them, I may not even be a Computer Science major nor a student at MIT, so a Master's thesis would totally be out of the question. They never restricted me from choosing my own major, and just urged me to do what I enjoy.

I would also like to thank Michael Cheng for being by my side day and night during the past year and a half, even if it was from 800 miles away most of the time. Whenever I was stressed out or needed someone to talk to, or just wanted to procrastinate and chat, he was always there for me.

Lastly, I would like to thank the friends that I have made over the years. It is a little sad that so many of them are spread all over the world, but that also means that I will probably have a friend to visit wherever I travel to in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Spoken Language Systems (SLS) group at MIT's Computer Science and Artificial Intelligence Laboratory was formed in 1989 with the goal of enabling communication between people and computers via natural spoken language. Its objective has been to enable computers to communicate like people, to people, through the mediums of speaking and listening. Over the years, the SLS group has developed useful technology to understand and generate sentences. In recent years, the group has also started exploring foreign languages with an emphasis on Mandarin Chinese. For example, MuXing is a telephone-access Mandarin conversational system which enables a user to access weather information (Wang et al. [17]). Furthermore, the spoken language learning system (SLLS) is an interactive, online application which allows teachers to plan language curriculums and students to practice and learn Mandarin Chinese (Lau [10]). The focus of this thesis is to use existing SLS technologies to further enhance the language learning domain. Specifically, the goal is to build an interactive map application called LanguageLand that will engage both teachers and students to learn Mandarin Chinese. LanguageLand differs from SLLS in its extension to the notion of game playing within a more engaging multimodal environment.

1.1 Motivation

The problem to be addressed by this thesis is that users who want to learn foreign languages are bound by human resources and conventional classroom settings.

Interactive and entertaining ways of learning a language are lacking. Although there are computer applications for language learning, most of them are not intelligent—the computer does not actually understand what the user is saying. For example, the "Japanese: The Spoken Language, Multimedia Collection" computer application converses with the user following a preset template synchronized with its corresponding textbook (The University of Maryland University College [16]). The computer speaks a phrase and the user replies. The problem is that there is no indication when the user's pronunciation is wrong, as the user's recording is not checked for accuracy. In fact, even if the user remains silent during his allotted recording time, the computer still continues with its role in the conversation. It becomes quite monotonous when the computer application is just repeating material presented in a textbook, without providing any useful feedback to the user.

The purpose of this thesis is to use existing SLS technologies to build LanguageLand, a multimodal spoken conversational system for language learning. LanguageLand is a sister application to SLLS, an SLS language learning system already in place (Lau [10]). This new system will accept spoken, typed, and mouse inputs, so as to capture the user's attention and to be able to react in a timely and intelligent manner. Instead of following a preset template, it allows the user to roam freely within the constraints of the domain, and will react accordingly by expressing understanding or confusion. Its game-oriented nature engages the user in a real-life map environment similar in concept to a game of Simon Says or of guiding a mouse through a maze.

1.2 Goals

The goals for this thesis project are to create a spoken conversational system for language learning that is more engaging and intelligent than present applications. In particular, it should:

1. Allow users to engage in dynamic conversations
2. Both allow students to learn languages using the system and teachers to track students' progress
3. Be significantly different from SLLS in terms of presentation and usage
4. Incorporate existing SLS technologies seamlessly
5. Have multimodal capabilities with spoken, typed, and mouse inputs
6. Present a dynamic set of vocabulary for continued learning
7. Provide seamless language switching and bilingual spoken input

1.3 Outline

The rest of the thesis is structured as follows. Chapter 2 discusses related work, the Galaxy architecture, and existing SLS technologies for recognition, understanding, generation, and synthesis. Chapter 3 introduces the LanguageLand system by describing in detail a couple of usage scenarios. Chapter 4 explains how the different natural language components were augmented for this thesis. Chapter 5 discusses the LanguageLand application component. Chapter 6 then talks about the evaluation of the system on real users. Finally, Chapter 7 gives a summary of this thesis and presents future work.

Chapter 2

Background

This chapter outlines the precursor applications and components of SLS technology which have been crucial in the formation and usage of LanguageLand. The first section introduces some related projects and describes in more detail the inner workings of SLLS (Spoken Language Learning System), a sister application to LanguageLand. The second part presents the Galaxy architecture, which ties together all the separate technology components. The third section discusses the technology built to recognize and understand sentences in different languages. The final piece explains how new sentences or paraphrases are generated in response to analyzed sentences.

2.1 Related Work

There are currently a number of different interactive language learning applications, including Language Adventure, the DARPA Tactical Language Training Project, and "Japanese: The Spoken Language, Multimedia Collection," as well as SLLS, a system built using SLS technology.

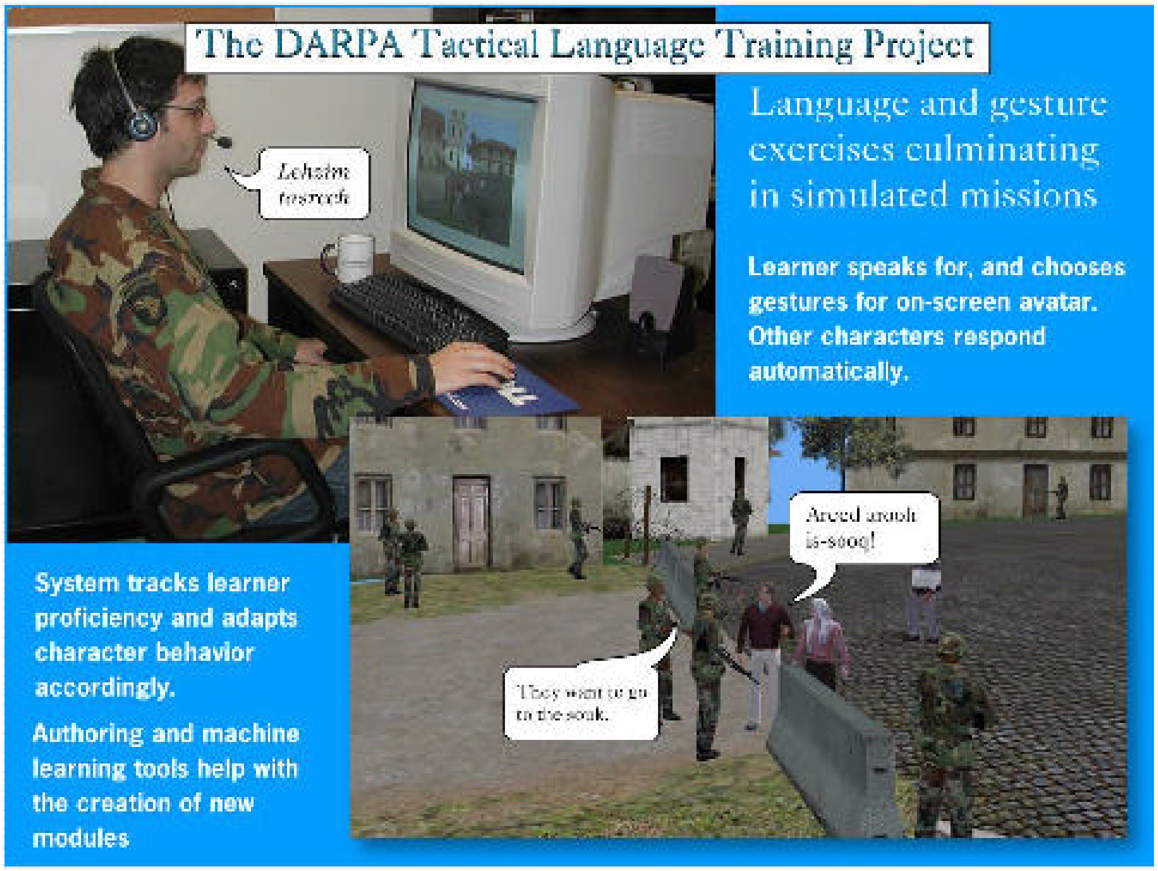
2.1.1 Language Adventure

Language Adventure, a system developed by Instinct, is aimed at teaching a foreign language to children aged six to twelve (Instinct Corporation [5]). It makes children interested by involving them in a story. Some of the exercises involve having children listen to phrases and then record and listen to their own voices speaking the same phrases.

Another exercise has children click on specific things mentioned by the system. The reverse is also true in that users can click on things and find out what they are. Finally, children are also able to create their own animated movie in the foreign language. Language Adventure is also referred to as “tracing paper for speech,” because of the way in which children learn and try to copy things in the program.

2.1.2 DARPA Tactical Language Training Project

The DARPA Tactical Language Training Project is led by Lewis Johnson and is focused on helping military personnel learn subsets of linguistic, gestural, and cultural knowledge that are required to be able to successfully complete missions (Johnson [8]). Its approach involves users learning vocabulary and gestures and then applying them to simulated computer sessions, where they interact with virtual characters using speech. Figure 1 illustrates a screenshot of a simulated session.


The DARPA Tactical Language Training Project

Language and gesture exercises culminating in simulated missions

Learner speaks for, and chooses gestures for on-screen avatar. Other characters respond automatically.

System tracks learner proficiency and adapts character behavior accordingly.

Authoring and machine learning tools help with the creation of new modules

Figure 1: Screenshot of a simulated session in the DARPA Tactical Language Training Project (Johnson [8]).

2.1.3 Japanese: The Spoken Language, Multimedia Collection

"Japanese: The Spoken Language, Multimedia Collection" is a set of CD-ROMs produced by the University of Maryland University College in 1998 and is based on the popular textbook, Japanese: The Spoken Language, Part I by Eleanor Harz Jorden with Mari Noda (The University of Maryland University College [16]). It follows the textbook very closely and has a vocabulary for each lesson. A user goes through a set of grammar and vocabulary words and learns their pronunciations along with their meanings. Then, the user repeatedly listens to a scripted video known as a core conversation until he feels prepared to participate in it. Taking on the role of one of the characters, the user can record and listen to his own voice along with the computer's in the opposite role. Figure 2 shows a screenshot of a core conversation.

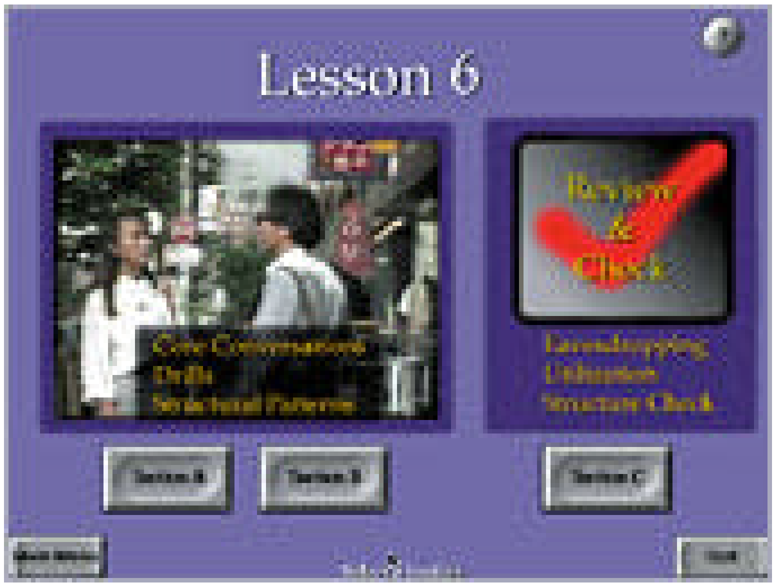


Figure 2: Screenshot of a core conversation of "Japanese: The Spoken Language, Multimedia Collection" (The University of Maryland University College [16]).

2.1.4 SLLS

Using the Galaxy architecture, Jonathan Lau has developed SLLS, an online conversational spoken language learning system which is a predecessor to LanguageLand (Lau [10]). Presented as a Web interface, it allows users wishing to learn Mandarin Chinese to converse using a telephone. The system is broken up into three stages: preparation, conversation, and review. In the preparation stage, users can listen to practice phrases and simulated conversations to get practice and a sense of what types of answers are expected. When they are ready to proceed with the actual conversation, they can instruct the system to call them. The telephone conversation is closely tied to the practice phrases and its content would in theory be generated by a teacher or administrator. While the structure is set, certain words which fall into the same categories are interchangeable and randomly selected by the system. The final stage is review, in which the user can listen to the recorded conversation and see which words the system was able to recognize correctly.

2.1.5 How LanguageLand is Different

LanguageLand is different from the first three systems in one fundamental way. While those systems have set lesson plans, LanguageLand is flexible and allows the user to freely explore his opportunities. Instead of letting the user listen to his own voice, recognition quality is judged by whether the system interprets the user's phrase correctly. Even though LanguageLand is based on the same core technologies as SLLS, it is different from it in three main aspects: domain, system response, and modality. While typical SLLS conversations are focused on everyday topics such as family relationships and sports, LanguageLand's domain is specific to building maps consisting of landmarks and of getting from one place to another on the map. Secondly, LanguageLand not only offers spoken system responses like SLLS, but there is also on-screen feedback, such as movement from one location to another on a map. Finally, LanguageLand is multimodal, allowing users not only to speak to the system but also to be able to blend typing and mouse clicks with speech in conveying an instruction. We anticipate that this will be a more rewarding and engaging exercise for the student.

2.2 Galaxy Architecture

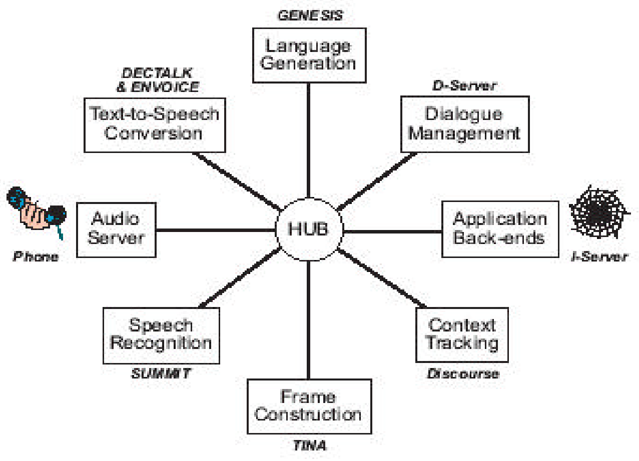


Figure 3: Architecture of Galaxy (Seneff et al. [15]).

LanguageLand uses the SLS group's extensive technology built into the Galaxy architecture (Seneff et al. [15]). As shown in Figure 3, Galaxy is based on an architecture where a central hub mediates communication among a number of specialized servers (Cyphers [4]). This mediation is performed according to a hub script which consists of a set of rules that coordinate the communication.

The servers in the Galaxy architecture generally interact in the following manner. When the user speaks to the system over the telephone, the audio server records the user's sentence and constructs a digital waveform for it. Then, the speech recognition server (Summit) component determines a set of hypothesized sentences and passes them to the frame construction (Tina) server, which selects the most plausible hypothesis and creates a semantic frame representation encoding the meaning. The context tracking server alleviates any ambiguity in accordance with the appropriate context. The application backend then receives the frame and in some cases passes back out information as a response frame. This semantic frame is sent to the language generation (Genesis) server, which generates a sentence accordingly. Finally, the text-to-speech conversion server (Dectalk or Envoice) converts the sentence into speech.

Of special importance to LanguageLand are the audio server (telephone), speech recognition (Summit), frame construction (Tina), application backend (LanguageLand), language generation (Genesis), and text-to-speech conversion (Dectalk and Envoice) server components of Galaxy. Several of these servers are described in the following sections.

2.3 Recognition and Understanding

2.3.1 Summit

The Summit server is the speech recognizer component of the Galaxy architecture. Its basic approach to recognition is to use acoustic-phonetic knowledge to detect landmarks as critical points in time for feature extraction, and to score the regions around the landmarks against a set of diphone acoustic models (Zue et al. [20]). There are three main components to Summit: the first one takes the speech signal as input and outputs an acoustic-phonetic representation, the second one takes a set of baseform pronunciations and transforms it into an expanded lexical network based on a set of phonological rules, and the last one matches the expanded lexicon against the acoustic-phonetic representation to produce a set of hypothesized sentences, guided by a statistical language model.

This thesis exploits two special capabilities of Summit: 1) dynamic adjustment of the vocabulary on-the-fly, and 2) combining recognizers for two different languages within a single search space.

The first of these capabilities can be utilized when there is a need to specialize the vocabulary in order to improve recognition quality (Chung et al. [3]). For example, the restaurant domain for Massachusetts alone has thousands of restaurant names. If Summit were licensed to recognize all of those names, there would be a lot of ambiguity and a higher error in figuring out which restaurant a user actually means to say. By representing restaurant names in a dynamic class in the speech recognizer and the natural language grammar, the system can appropriately swap in different sets of names depending on context. For example, if the city of Cambridge is of interest, then the set of restaurants in Cambridge is swapped in, while restaurants in all other cities of Massachusetts are voided from recognition. As a result, confusion between similar-sounding restaurants located in different cities is avoided, and search results are vastly improved.

Summit's second capability is in creating a multi-lingual recognizer by combining recognizers for different languages (Hazen et al. [7]). This allows the recognizer to automatically switch between different languages without the user having to specify in advance which one they are speaking. The way Summit performs this is by running each language's recognizer component in parallel, thus outputting potentially a mixed language N-best list. The only issue is that, since each recognizer uses its own set of acoustic models, there may be a bias in the scores, causing those languages that produce higher scores on average to be inappropriately favored. This problem is alleviated by normalizing the acoustic model scores using a mathematical equation (Zue et al. [20]). Tests for a bilingual recognizer for the weather domain that handles both English and Japanese have shown that the accuracy level for the multi-lingual recognizer is only slightly lower than that of separate recognizers. The results are reproduced here in Figure 4, which show an average error rate degradation of 4% (Hazen et al. [7]).

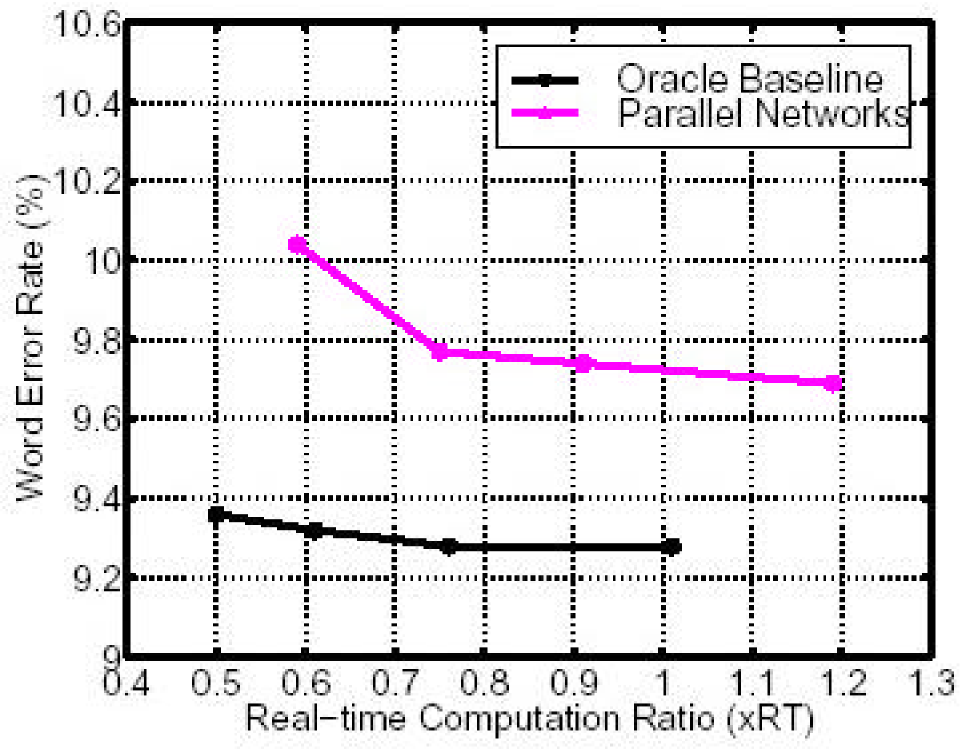


Figure 4: Bilingual (English/Japanese) recognition results for the weather domain comparing parallel recognizers to a single recognizer which knows the language beforehand (Hazen et al. [7]).

Further analysis has shown that this accuracy degradation is mainly due to errors in identifying the appropriate language, which in this experiment occurred at a rate of 1.25%. These errors usually stemmed from spoken phrases which consisted of only one or two words.

2.3.2 Tina

Tina, the frame construction module, serves as an interpretive interface in the Galaxy architecture to interpret the recognizer outputs and/or typed inputs at a GUI interface (Seneff [14]). In terms of speech input, because of the uncertainty and variation in speech, a number of different possibilities are given by the recognizer. Tina scores the N-best hypotheses from the recognizer by parsing them into a meaning representation, and chooses the highest scoring hypothesis to represent the utterance in future processing.

With Tina, one can specify syntactic and semantic rules for the parse tree generation. In the parse tree for a sentence, most of the syntactic rules are contained in the upper nodes while the semantic rules are associated with nodes near the leaves. For instance, a sample sentence “go straight” parsed with the LanguageLand MapGame domain version of Tina is shown in Figure 5.

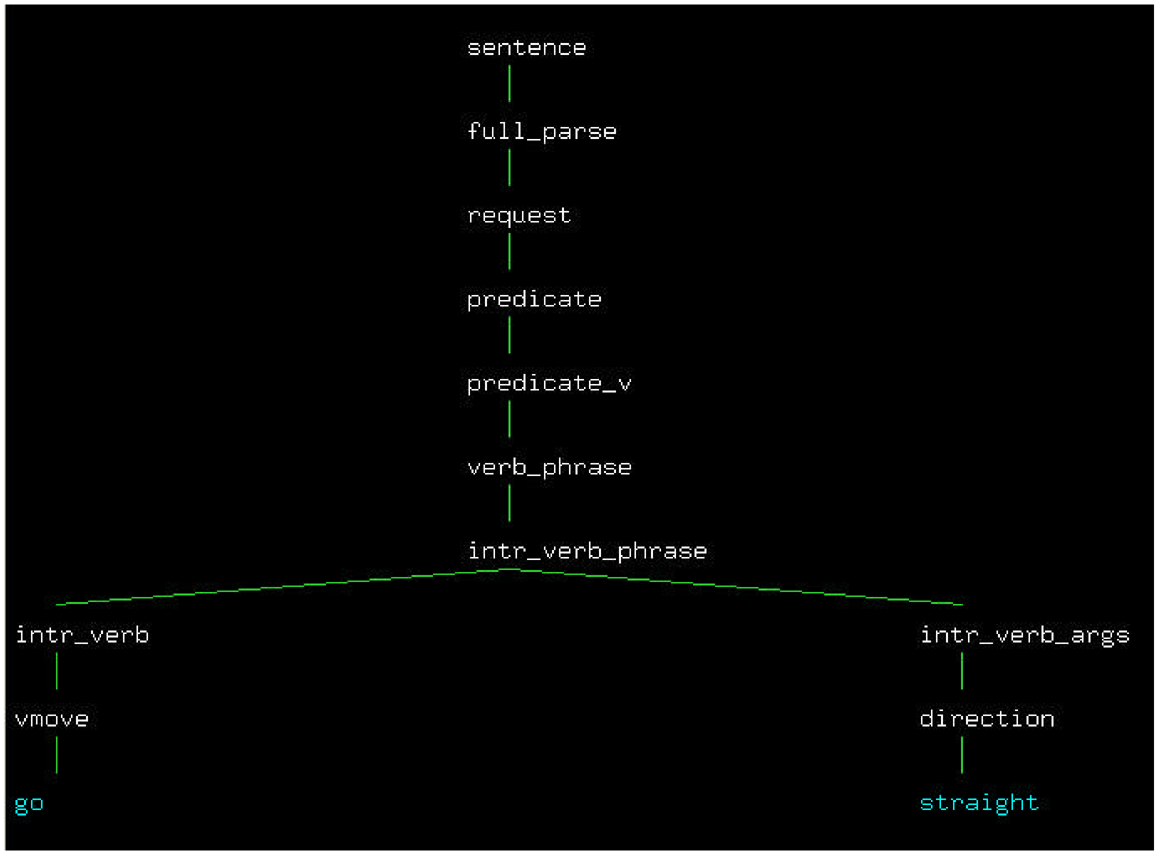


Figure 5: Parse Tree for "go straight."

A novelty of Tina's interpretation of the rules is that it decomposes all rules into sibling-sibling transition probabilities, such that the entire parse space can be viewed as a hierarchy of bigrams. Thus, in forming a parse tree, the relationships of importance are parents, children, and siblings. Sibling pair probabilities are generated through frequency counts on parse trees in a training corpus. When the Tina system is used with a recognizer counterpart for speech input, the recognizer will give it multiple input possibilities. A word graph is constructed from the N-best list and a Viterbi search derives N-best parse trees, taking into account both acoustic and linguistic scores.

Once a successful sentence is generated, the parse tree is also already created. The next step is to derive the meaning representation from this tree. This is encoded as a special object type in Galaxy known as a semantic frame. There are three types of semantic frames: clauses, predicates, and topics. Frames can also contain other frames in a recursive manner, and there is only one frame associated with a particular sentence. The way it is generated is by passing a frame under construction along from node to node, top down and left-to-right, in the completed parse tree for that sentence. Each node then has the opportunity to modify the frame according to the specifications based on that node's class-wide identity. The path of traversal is identical to that when the parse tree was generated. For example, the semantic frame generated for the parse tree of "go straight" in Figure 5 is located in Figure 6.

```
{c request
  :domain "MapGame"
  :sro "go straight"
  :pred {p go
        :pred {p direction
              :topic "straight" } } }
```

Figure 6: Semantic meaning representation frame for "go straight."

LanguageLand's domain, called MapGame, deals specifically with parsing sentences relating to directions on getting from one location to another, as well as on placing different landmarks on the map. The English grammar for LanguageLand was built on top of a pre-existing generic syntax-based grammar specific to database-query domains. Once a small set of semantic rules was written for the LanguageLand domain, they were automatically incorporated into the probabilistic networks of the whole system.

Phrasebook and Restaurants are examples of other domains that have been built on top of the generic grammar (Chung et al. [3]).

2.3.3 Grammar Induction

A resource that was important for preparing the Chinese grammar LanguageLand is a grammar induction system that is able to induce a grammar for a new language given a pre-existing translation system from English to that language (Lee [11]). This module was previously used for an English-French system in which the French grammar was induced. The user provided a set of English training sentences, grammar rules to create the semantic frames, and generation rules to translate the sentences into French, and the system was able to work backwards and create a set of grammar rules to construct semantic frames from the French sentences. Because English and French have very similar grammar structures, the simulation was close to perfect and only needed minor modifications.

2.4 Generation and Synthesis

2.4.1 Genesis

Genesis is the language generation component of the Galaxy dialogue system (Baptist [1] and Baptist et al. [2]). The first version of this system was developed in the 1990s. However, over the years, as domain experts and other users requested more features, it became increasingly clear that the system was insufficient. In response, the SLS group at MIT revamped the generation component with three key goals in mind: generality, consistency, and simplicity—core values reflected in the components and structure of the current version of Genesis.

Genesis assumes many different roles as the language generation component; it can perform paraphrase generation, formal-language generation, and response generation. (The first two are of most relevance to the LanguageLand system.) Additionally, Genesis can take in a number of different types of meaning representations. One such meaning representation was previously discussed in the Tina section: the hierarchical linguistic structure, also known as a semantic frame. Genesis can also receive simpler structures made up of key-value bindings, much like SQL entries.

This generation component has been designed to work for a number of different domains. (As an example, the LanguageLand application domain is MapGame.) Each domain must specify parameters for Genesis to be able to use and generate domain-specific sentences. The architecture is comprised of a code kernel written in C and a linguistic catalog, as shown in Figure 7 (Baptist et al. [2]). This code kernel takes in a meaning representation as input, along with domain-specific information such as the language in which to translate. Such languages include natural languages (i.e. English, Chinese, Japanese, Spanish) as well as formal languages (i.e. SQL, html). Then, using the particular domain's linguistic catalog, the kernel system is able to generate and output an appropriate string.

There are three main parameter files in each domain's linguistic catalog: the grammar file, the vocabulary file (aka lexicon), and the rewrite rules file. The grammar file specifies the rules that the kernel will use to generate, for example, the paraphrase string.

The vocabulary file specifies semantic and contextually relevant replacements for substrings. The rules file specifies syntax replacements like spaces and contractions. These three files are used together for sentence generation.

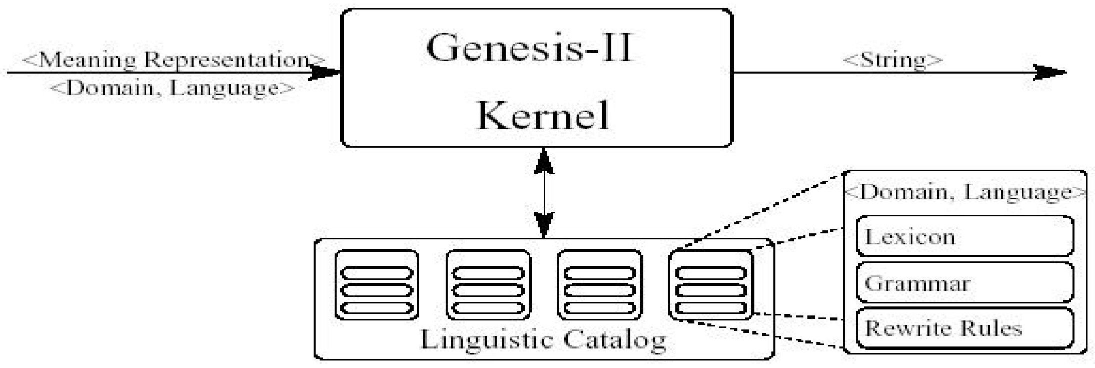


Figure 7: System architecture for Genesis (Baptist et al. [2]).

Once the paraphrase string is formed and returned by the kernel, the hub script for the LanguageLand MapGame domain then issues a command to generate an eform of the paraphrase string. An eform is basically a simplified encoding of the same information, providing key-value bindings to be retrieved by the programmer using a common API. After parsing the eform, the LanguageLand system performs an action or displays the reply string provided. Figure 8 depicts a simple eform constructed for the phrase "go straight," with "direction" being the key and "straight" being the value.

```
{c eform :direction "straight"}
```

Figure 8: Eform for the phrase, "go straight."

2.4.2 Dectalk & Envoice

For the text-to-speech conversion required to enable system responses to spoken user input, Dectalk is used for English and Envoice is used for Chinese in the LanguageLand system. Dectalk is an off-the-shelf commercial product whereas Envoice, a concatenative speech synthesizer, was developed by the SLS group (Yi et al. [19]).

Envoice is used in conjunction with Genesis to convert meaning representations into speech waveforms (Yi et al. [18]). Its primary goal is to concatenate different segments of waveforms from a set of pre-recorded utterances. Its knowledge/search space is represented as a finite-state transducer (FST) such that, given an input sequence, a graph of output sequences can be generated (Yi et al. [19]). The generation rules follow the FST topology, directed by its states, arcs, and corresponding weights. Because using longer-length units provides better synthesis quality, the system carefully tries to maximize the size and contiguity of different segments. When deemed necessary, it will backoff to shorter units (Yi et al. [18]).

Chapter 3

Usage Scenarios

In this chapter, the LanguageLand system is introduced from two different perspectives. The first section presents a detailed experience of how a student might use the system, while the second section approaches it from a teacher's point of view.

3.1 Student

LanguageLand can be used by anyone who has the Java Virtual Machine installed on their OS and has the appropriate connection settings to the SLS servers. The following section describes the interaction between Michael, a fictitious first year student learning Mandarin Chinese, and LanguageLand, to illustrate how a normal student user might use the system.

Michael is in his second term of Chinese and has recently started learning about real-life applications such as navigating the streets of China and shopping for groceries. Though having already learned a lot of vocabulary in the past, he has not had much experience stringing them into real sentences. His teacher introduces his whole class to LanguageLand, a multi-modal application that will supposedly apply their current knowledge to real-life situations and also allow them to learn new vocabulary. A bit hesitant about his speaking skills, Michael decides to give it a try anyway. He understands that everything he performs may possibly be monitored by his teacher, as the application supports teacher feedback.

Michael gets help from an administrator in downloading the application and setting up appropriate server connections to SLS. He then opens an email from his teacher which contains his password. His username is just his school email address. He starts up the application and inputs this information into the login screen. After clicking OK, a "LanguageLand Instructions" screen pops up. He reads it briefly then clicks OK, unveiling the welcome screen that shows three options. He decides to create a new game and names it "My first game."

An "Edit Game Instructions" screen then pops up describing how to edit the game. A bit impatient, he closes it after a quick glance and decides to venture on his own. He sees that there are a collection of pictures on the left and a map on the right. Since the instructions mentioned "Edit Game," he assumes he should be doing some sort of editing. He left-clicks on what looks like lab flasks and then clicks on one of the map tiles, causing a red outline to appear around the tile. He clicks it again and the flasks appear in that tile. Things are starting to make sense. After placing a couple more items onto the map, he notices an "Instruction" text area, where presumably he can enter text. He clicks on the animal on the left which looks like either a hamster or a mouse and then types in, "What is this?," clicks enter, and notices that in the "System Interpretation:" text area right below, it says "zhe4 shi4 shen2 me5." Since Michael is a whiz at pinyin and is also a quick learner, he immediately realizes that the system just did a translation for him into Chinese pinyin. Michael then types in "zhe4 shi4 shen2 me5" and the system responds with "lao3 shu3." One of his guesses was right--it is indeed a mouse. Michael decides to explore more on his own. He clicks on another tile on the map and types in "fang4 lao3 shu3 zai4 zhe4 li3." A picture of the mouse immediately shows up on that tile. After playing around with the edit mode a little more, he decides to try to play a game.

Michael clicks on the Play item in the Edit menu, which pulls up a new screen along with instructions for the play mode. After a minute he closes the “help” screen, noticing a human stick figure on the corner of the map. He suddenly remembers that his teacher had mentioned he could have the system call him by typing "call me at” followed by the phone number. He types in "call me at 7707676578.” After a few seconds, his phone rings and he picks up right away. The system says “Wecome” followed by a beep. Michael tries his luck with “go south.” The system responds with “nan2 zou3,” which also shows up on his screen in the “System interpretation” text area. He then says “nan2 zou3” and again he hears and sees “nan2 zou3.” This time, he also sees a red line and a green arrow on the screen, mapping his movement from the starting position down one block. Michael says “you4 zhuan3,” hears and sees “you4 zhuan3,” and notices that the green arrow rotated clockwise 90 degrees, indicating a right turn. Having had enough for the time being, he says “good-bye” and hangs up the phone.

He checks out the menu items for fun and notices a “View My Progress” item in the “File” menu. When he clicks it, a new screen pops up and he notices a log of his utterances. He also notices a Feedback area at the bottom, where presumably his teacher might leave him comments. Having had enough for one session, he exits out of the application.

3.2 Teacher

Like Michael, Zhang Laoshi is also a fictitious character, but she plays the role of a language professor teaching Level 1 Mandarin Chinese. Having recently acquired this application from the SLS group, Zhang Laoshi has already tried it out once and is familiar with its features.

She logs in with her password and chooses “View Students Progress” from the welcome menu. While teachers also have the ability to create and play games, her main goal today is to set up accounts for the remaining students and also to check progress on some of her existing students. The Students’ Progress screen pops up, showing the list of students on the left, including Michael Cheng, an excellent Level 1 student. She clicks on his name and the right panel shows his game statistics. She notices that there are some English phrases followed by their direct Chinese counterparts, and then some stand-alone Chinese phrases, and realizes Michael probably typed some sentences in English initially to test out the system. Pleased by his grasp of the grammar shown by the sentences, she types “good job on the first try!” in the subject field and the following in the comment field: “Try to limit the English only to when absolutely necessary. I also encourage you to try out different expressions and see how the system responds to them.” She then clicks on “Submit Feedback” and receives a “Feedback Submitted” confirmation window. To make sure it got put in correctly, she clicks on the “View My Feedback to Student” link and sees the message she just typed in.

Zhang Laoshi now attempts to add the remaining students into the system. She clicks on the “Add New Student” button on the left and fills out the First name, Last name, and Email address fields before clicking OK. She proceeds in repeating that routine for the rest of the students. Knowing that the default password is always just the first letter of the first name and the first letter of the last name, she sends out emails to those students informing them of that. Zhang Laoshi then exits out of the application and begins preparing material for her language class the following day.

Chapter 4

Natural Language Technology Development

This chapter explains how each of the different SLS technologies was used in the system and how they had to be modified to be compatible with the LanguageLand domain. The first section describes components which were used for recognition and understanding. The second part discusses the work done for generation and synthesis.

The main usage of these components is to parse and create meaning representations for both English and Chinese sentences, and then either paraphrase or generate key-value bindings from the meaning representations. Our first step was creating the grammar rules for paraphrasing English into Chinese. Using those rules, we were able make use of grammar induction to expedite the process of constructing a grammar to parse Chinese inputs (Lee [11]). The final step was writing rules to paraphrase Chinese back into Chinese and also to create key-value bindings for Chinese sentences.

4.1 Recognition and Understanding

4.1.1 Summit

As mentioned in Chapter 2, Section 3, Part 3, Summit is the speech recognizer that is used by the LanguageLand system (Zue et al. [20]). To configure a recognizer for Summit, one needs to provide it with a lexicon, a statistical language model (i.e., a class n-gram), and training sentences. Both the lexicon and language model were derived automatically from the grammar, but we had to manually prepare training sentences. One of the subgoals of this thesis is to explore the use of two recently introduced features of Summit: dynamic vocabulary and multilingual recognizer configurations.

In the English-only version of the LanguageLand system, a dynamic vocabulary recognizer has been configured in order to improve recognition performance while permitting variety in the number of different street names and landmarks that are supported (Chung et al. [3]). Every time a user enters a new game, a pseudorandom subset of the available landmarks and street names is selected, and these become immediately accessible to the recognizer. In this version, the language to be learned is English, which is also the only language the system is equipped to respond to. To implement this capability we established two dynamic classes: landmarks and street names. To incorporate these classes, we created a set of training sentences with special dynamic markers which serve as placeholders for words that fall within those classes. Figure 9 is an example of a sentence with dynamic markers in place.

turn right when you pass the <dynlandmark> pig </dynlandmark> and then turn left when you reach <dynstreet> prince </dynstreet> street.

Figure 9: A training sentence with landmark and street name dynamic markers.

These dynamic markers ensure that the sentences are not specifically training the words "pig" and "prince." Instead, their respective landmark and street classes are stand-ins. Correspondingly, the natural language grammar was modified to include these two dynamic classes. The system is flexible to swap in different sets of words for both landmarks and street names, thus decreasing the number of words active at any given time and improving recognition performance.

The other feature we exploited in Summit is the ability to support seamless language-switching by packaging up both a Chinese and an English recognizer within a common search space (Hazen et al. [7]). The process to construct the Chinese recognizer was essentially identical to that of building the English one. We created a set of Chinese sentences and trained the language model from them. Then, we fused those two recognizers by configuring them in parallel. Summit automatically determines an N-best list of the most probable hypotheses generated by both the English and Chinese recognizers, which thus compete in a common search.

It turns out that combining dynamic classes with multilinguality has additional issues with respect to distinguishing the phonological rule applications with respect to language. Due to this complication, we were unable to combine dynamic vocabulary with seamless language switching. Thus, the multilingual system does not swap in different sets of words for the "landmark" and "street_name" classes, but instead always licenses the full set, for both English and Chinese. While recognition precision is sacrificed, we felt that it was more important for the system to be bilingual.

4.1.2 Tina

Once the NL System, Tina, receives an N-best list from the Summit speech recognizer or a typed sentence from the LanguageLand GUI interface, it creates a parse tree and a corresponding semantic frame (Seneff [14]). For the English-only LanguageLand

MapGame domain, only a few changes were made to the pre-existing generic grammar. These changes were necessary for generating the parse trees, though rules for creating the semantic frames were not modified.

An example of a modification for the English-only version of the LanguageLand system is that new rules were added to the existing generic grammar in order to support the dynamic street name and dynamic landmark markers. Also, additions were made to some of the rules like “street_name.” There, we added about 113 street names. We made sure to pick words that have clear translations in Chinese, so that the user can benefit from learning new street name vocabulary in the foreign language. Examples of some street names used are “Prince,” “Bridge,” and “Coral.” The “street_type” rule had to be augmented as well. Most of the existing terminal nodes, including “parkway,” “highway,” and “boulevard,” are not needed for the LanguageLand domain because we decided to shrink the vocabulary in order not to overwhelm the language student. But it was not necessary to remove them from the rules because our system is not hindered by their existence. We did, however, add new terminal nodes for the “street_type” rule such as “rd,” which serves as shorthand for road. Thus, if a user types in “park rd,” “rd” will be identified as a “street_type.” Figure 10 shows the parse tree which results from the phrase, “park rd.” The dynamic markers are automatically inserted in a pre-processing step.

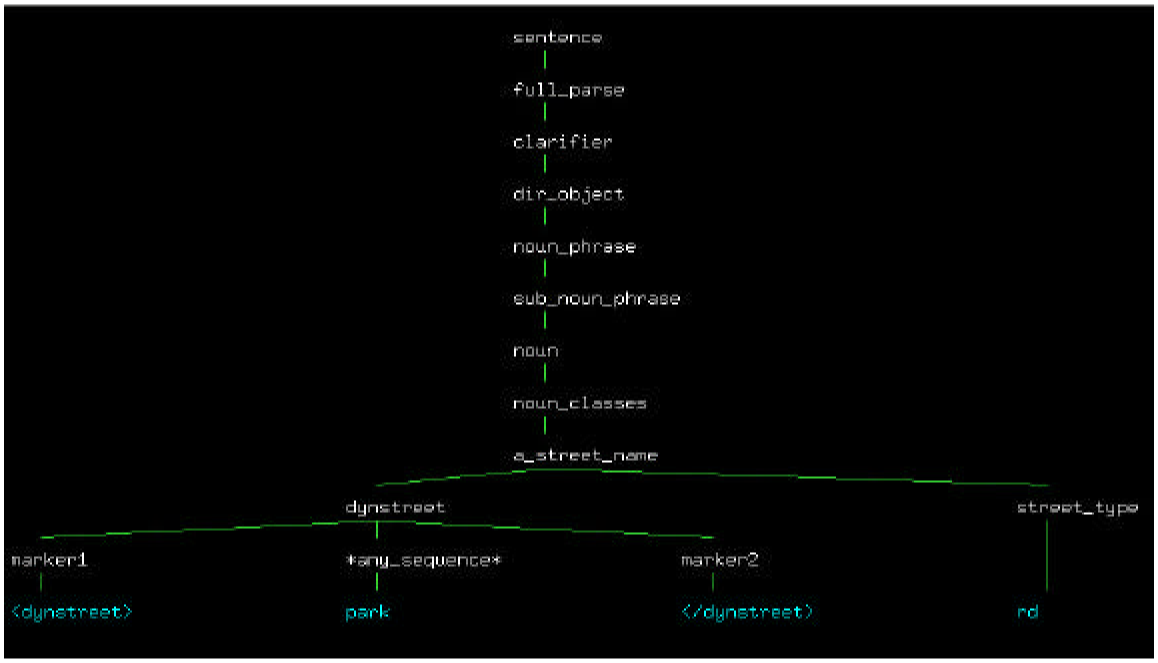


Figure 10: Parse tree for the phrase, "park rd."

The "a_street_name" rule is exercised here, as can be seen fom the "dynstreet" and "street_type" children nodes. The terminal nodes for "dynstreet" includes the markers for dynamic classes, which is yet another rule that was added. Also, the "rd" terminal node is a result of the new rule for "street_type."

An interesting problem we ran into was the misinterpretations of conjunctions in sentences. For example, the sentence "go north until you pass the pig and then turn right" can be interpreted as either "go north until you pass the pig and then turn right" or "go north until you pass the pig and then turn right." The second interpretation is what we wanted, but the system would sometimes misparse ambiguously as the first option. The problem was that the phrase after the "and" would try to hide under the adverbial clause that begins with "until," thus resulting in the first parse. We tried to fix this problem by adding markers in training sentences to explicitly end the adverbial clauses before the "and." We also tried populating the training sentences with lots of marker-augmented examples, hoping that the system would learn to correct itself. Although this method worked for some novel inputs, it did not work for all of them. We eventually resorted to restructuring the semantic frames such that the compound predicate is only possible in the main clause, thus avoiding the confusion of multiple parses and the resulting selection of a wrong parse.

4.1.3 Grammar Induction

While the English grammar for LanguageLand was built simply by extending the existing generic grammar, the Chinese grammar was created using a software package that was the product of a recent thesis written by John Lee in the SLS group (Lee [11]). For that work, we provided a set of English training sentences and used the software to induce a preliminary set of grammar rules for the Chinese domain. The system uses the existing English grammars to obtain semantic frames for the English sentences, and then the Chinese translations. Throughout the whole process of translating and reordering words, the system keeps track of where each Chinese word came from in the frame and aligns it with its counterpart translation from the English sentence. It is thus able to recreate semantic frames from the Chinese phrases and form the preliminary Chinese grammar.

While the rules produced were not comprehensive, they provided a good starting ground and made it a lot easier to extend and modify the rules. The program had previously been tested successfully with inducing French, which is very similar to English. However, we ran into some interesting difficulties because the grammar structure for Chinese is quite different from that of English.

One of the problems was recursion. Some of the rules generated were left recursive such that they would loop around and revisit the same rule in an infinite cycle. The system was able to recognize such recursion and identify the offending rules, which allowed us to reconfigure the grammar to avoid it.

Another problem was that, similar to the aforementioned English grammar rules, the Chinese grammar rules generated by the training sentences did not include all possible terminal nodes, and thus we had to manually add them. For example, the direction rule shown in Figure 11 was modified to include all the possible directions (in pinyin) that were not included in the training sentences.

```
.direction
#zhi2
#xi1
#dong1
#zuo3
#nan2
#you4
#bei3
#hou4
#shang4
#xia4
#yi1 #ge5 #quan1
#qian2
#guo4 #lai2
```

Figure 11: Modified rule for direction.

Other examples of changes include fixing incorrect rules such as the following in Figure 12.

```
.at
;;#rang2 #hou4
#zhi2 #dao4
#zai4
```

Figure 12: Modified rule for direction.

The first line in the "at" rule, "#rang2 #hou4" (which means "and then") was commented out because the program had mistaken it as being a child of the "at" node. In fact, "#rang2 #hou4" already exists under the "and" rule. This correction fixed the following sentence and many others, which previously had faulty parse trees: "zhi2 zou3 rang2 hou4 zai4 bei3 jie1 you4 zhuan3." Figure 13 depicts the parse tree for this sentence. As can be seen, the "and" node contains the terminal nodes "rang2" and "hou4."

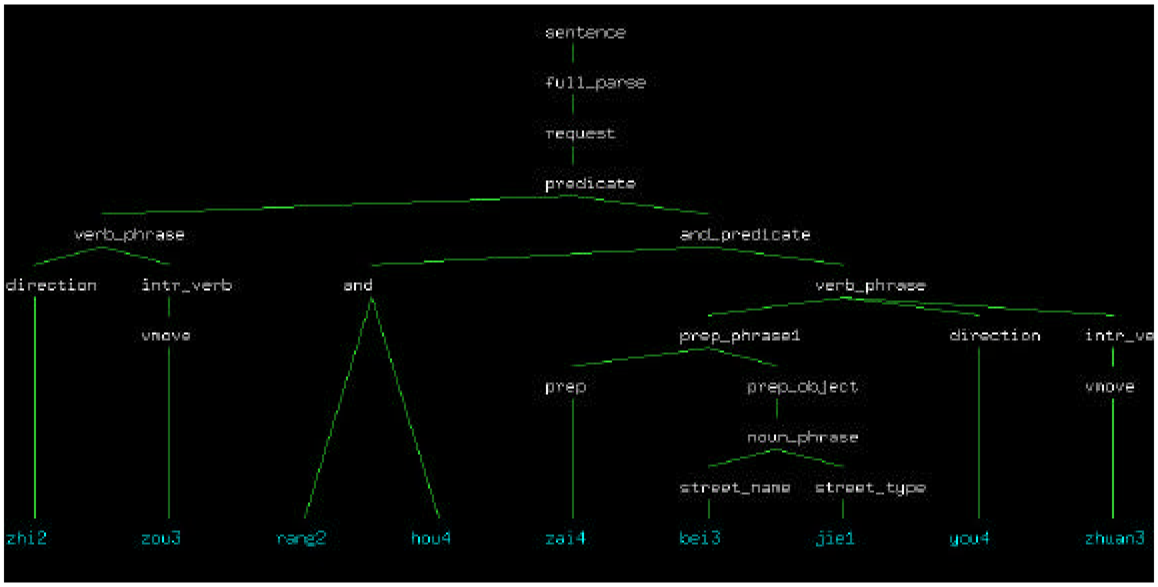


Figure 13: Parse tree for "zhi2 zou3 rang2 hou4 zai4 bei3 jie1 you4 zhuan3."

For the Chinese grammar, modifications also needed to be made to the “action lists” file for creating semantic frames from the parse trees. One such example was adding “and_street” to the “and” rule, which made it possible to have sentences like “fang4 zhe4 ge5 zai4 gong1 yuan2 lu4 he2 shan1 hu2 jie1 de5 jie1 tou2.” The meaning representation is shown in Figure 14.

```
{c request
   :domain "MapGame"
   :pred {p place
            :topic {q pronoun
                      :quantifier "this" }
            :pred {p at
                     :topic {q a_location
                               :name "corner"
                               :pred {p of
                                        :topic {q street_name
                                                  :name "gong1 yuan2"
                                                  :street_type "road"
                                                  :and {q street_name
                                                         :name "coral"
                                                         :street_type "street" } } } } } } }
```

Figure 14: Semantic frame for “fang4 zhe4 ge5 zai4 gong1 yuan2 lu4 he2 shan1 hu2 jie1 de5 jie1 tou2” with modification to the “and” rule.

Without that modification to the “and” rule, the semantic frame would look like the one in Figure 15. As can be seen, the “street_name” topic in Figure 15 contains only one of the two streets.

```
{c request
   :domain "MapGame"
   :pred {p place
            :topic {q pronoun
                      :quantifier "this" }
:pred {p at
        :topic {q a_location
                :name "corner"
                :pred {p of
                        :topic {q street_name
                                :name "coral"
                                :street_type "lu4_jie1" } } } } } }
```

Figure 15: Semantic frame for "fang4 zhe4 ge5 zai4 gong1 yuan2 lu4 he2 shan1 hu2 jie1 de5 jie1 tou2" without modification to the "and" rule.

4.2 Generation and Synthesis

4.2.1 Genesis

From the semantic frame constructed by Tina, the language generation system can assemble a paraphrase, a set of key-value bindings, or a response (Baptist [1] and Baptist et al. [2]). Of special significance to the LanguageLand system are the first two, in constructing both the English-only as well as the bilingual version of the system. Genesis requires a linguistic catalog comprised of a grammar file, vocabulary file, and rewrite rules file, which we created for the LanguageLand MapGame domain separately for each of the generation languages.

For the English-only version of the LanguageLand system, we were interested in performing English-to-English paraphrasing as well as English-to-key-value bindings. The English-to-English paraphrasing we wanted was basically just a mirror copy of the input sentence. For example, for the sentence "go north until you reach prince street," Tina generates a parse tree and a corresponding semantic frame. The resulting frame is shown in Figure 16.

```
{c request
   :domain "MapGame"
   :pred {p go
          :pred {p direction
                 :topic "north" }
          :adv_clause {c statement
                              :conjn "until"
                              :topic {q pronoun
                                       :name "you" }
                              :pred {p reach
                                       :topic {q street_name
                                                :name "prince"
                                                :street_type "street" } } } } }
```

Figure 16: Semantic frame for "go north until you reach prince street."

The Genesis English paraphrase rules then reconstruct an English paraphrase from this semantic frame. For example, Figure 17 shows the "street_name" rule, which strips out the "name" and "street_type" bindings from the "street_name" topic frame, thus completing the "prince street" portion of the English paraphrase. The ":pred" rule covers any subpredicates within "street_name," such as "in Boston," and the ">do_and" rule will handle any compound NPs.

```
street_name          :name :street_type :pred >do_and
```

Figure 17: Generation rule for "street_name" topic in English paraphrasing.

Because the generic rules covered ground for most of our domain's English sentences, little modification was necessary. Rules for key-value bindings, however, did not exist and had to be created from scratch. After careful consideration of the LanguageLand application and how it would interface with the natural language components, we decided to have key-value pairings such as the ones shown in Table 1.

Key	**Value**
Direction	North, south, east, west, special, straight, back
Block	1, 2, 3, …
When	Reach, pass
Turn	Right, left, around
Clear	This, that, history, map
What	This
Landmark	<landmark>
Put	This, that, <landmark>
Location	Here, there
Street_name	<street_name>
Street_type	Street, avenue, road
Street1	<street_name>
Street2	<street_name>
Street_type1	Street, avenue, road
Street_type2	Street, avenue, road

Table 1: Key-value bindings.

If the input is "go north until you reach prince street," the corresponding set of key-value bindings is: "direction: north when: reach street_name: prince street_type: street," whereas if the input is "go north until you pass prince street," the set of key-value bindings is: "direction: north when: pass street_name: prince street_type: street." In both cases, the direction is "north" and the street of interest is "prince street." The difference is that, in the first case, the "when" key is bound to "reach" whereas in the latter case it is bound to "pass." An example of another sentence is "put the pig at the corner of prince and south street," which results in "put: pig street1: prince street2: south street_type2: street." This key-value binding informs the system to put the pig at the corner of the two specified streets. We also added the capability of having multiple instructions. For example, if the user says "go north for a block and then turn right," we want a sequence of two different sets of key-value bindings. The following is the template we came up with: "seq: <start> direction: north block: 1 <and> turn: right <end>." A more in-depth explanation of these key-value bindings and how they are dealt with in the LanguageLand application is discussed in Chapter 5.

In moving from the English-only system to the bilingual system, a few changes came into play. We no longer cared about the English-to-key-value bindings or English-to-English paraphrases. If a user speaks/types English, the paraphrase is in Chinese. If a user speaks/types Chinese, we want both the key-value bindings and the paraphrase in Chinese. It was pretty easy to make the modifications for the key-value bindings because for the most part they were identical to those in the English language. Since the key-value bindings are just used for backend purposes, we could have "direction" as a key instead of requiring “fang1 xiang4,” the Chinese translation of “direction.” Therefore, as long as we have the semantic frames from a Chinese phrase, we can use the same rules in creating the key-value bindings from English. We just had to add a couple of new rules because of slight differences in the semantic frame structure for Chinese.

In terms of the Chinese paraphrasing from both English and Chinese, we wrote one set of rules that was shared by both languages. It was somewhat tricky because of occasional differences in structures of the semantic frames.

4.2.2 Dectalk & Envoice

As mentioned in the Background section, the LanguageLand system uses Dectalk, an off-the-shelf product, for English synthesis. However, for Chinese synthesis, LanguageLand uses Envoice, which is more flexible and is specifically personalized for the Chinese LanguageLand domain (Yi et al. [18] and Yi et al. [19]). As a preparation for Envoice to be used, a large number of Chinese sentences and phrases were recorded. The longer the sentences are, the better, but due to the variation of word classes not all sentence combinations were recorded. For example, LanguageLand has about a hundred street names, and it is virtually impossible to create sentences for each of those street names. Thus, street names were recorded separately. However, phrases like “go north” and “go south” were recorded fully since there are not that many different directions (north, south, west, east, straight, backwards, etc.), and there is only one appropriate way to say “go” in Chinese. Envoice selects and concatenates different waveform segments that are prerecorded, giving preference to larger recorded sections. Concatenation can occur at the phrase, word, or sub-word level, chosen in that hierarchy in accordance to what is available.

The beauty with Envoice is that even if a sentence cannot be perfectly constructed from existing phrases, as long as phonetic recordings exist for each sub-word, Envo ice can pull pieces from different phrases. For example, “gong1 yuan2 jie1” (Park Street) and “shan1 hu2 da4 dao4” (Coral Avenue) were recorded. If we want the system to say “Park Avenue” in Chinese, Envoice can simply pull “gong1 yuan2” from the first recording and “da4 dao4” from the second recording, in order to produce “gong1 yuan2 da4 dao4.” Of course, if “gong1 yuan2 da4 dao4” was itself a recorded phrase, then that would be used instead, because Envoice favors longer recordings.

Chapter 5

LanguageLand Application Development

This chapter covers the meat of the thesis work and focuses purely on the Java and Swing application end (Lai et al. [9]). It is broken up into seven subsections. The first section talks about the help structure for both using the application and helping a student learn the language. The second section describes the process in getting started. The third and fourth parts describe the edit and play modes, respectively. The fifth and sixth sections detail how the teacher views his students' progress as well as how students view their own progress. The last section discusses system integration with the natural language components and the communication that takes place.

5.1 Help Structure

There are three different ways teachers and students can get help in this application. The first way is through the instructions screen. The first time a user comes across a crucial component of the application in a particular user session, an instructions screen pops up explaining how to get started and what everything in this stage stands for. There is a checkbox called "Do not show this message again" on that screen. If the user checks this box, then for all subsequent sessions this particular instructions screen will never pop up again. The user can always change this setting by toggling the checkbox in the Help menu under the item called "Show Instructions on Start." This is the same structure used for all instructions screens. Even if the user opts not to have one instructions screen show, all other instruction screens will still pop up unless the user checks those as well.

If a user does not want the instruction screen to pop up automatically but still wants to view it, or if a user closes the screen but wants to view it again, he can click on the "Instructions" menu item under the "Help" menu, which will pop up the instructions screen. Note that the toggle button setup for the instructions screen is customized individually so one user's settings will have no effect on someone else's.

Another way the user, and in particular a student, can get help is by clicking on the "Examples" menu item in the "Help" menu for both the "Play" and "Edit" modes. Since these two modes are where the user will be learning the foreign language, we felt it was important to let the user know what types of sentences can be said. This screen does not pop up automatically upon startup because we know there are students who may want to brave the system without getting any clues. Plus, the user can actively open it when necessary.

The last way a user can get help while using the application exists only in the "Play" mode, the reason being this is the mode in which users may need hints to proceed in the game. There is a "HINT" button in the play mode which users can click on if they are stuck in the game. The hints are staggered into two tiers. The first time the user clicks it, the hint is rather vague. If the user clicks it again without having made any progress in the game, the hint is more explicit. This format urges users to try their best attempt before asking for help again.

The different help mechanisms mentioned will be elaborated in subsequent sections of this chapter.

5.2 Getting Started

The first step in getting started is logging in. Users can do so by entering their email addresses and assigned passwords, as shown in Figure 18.

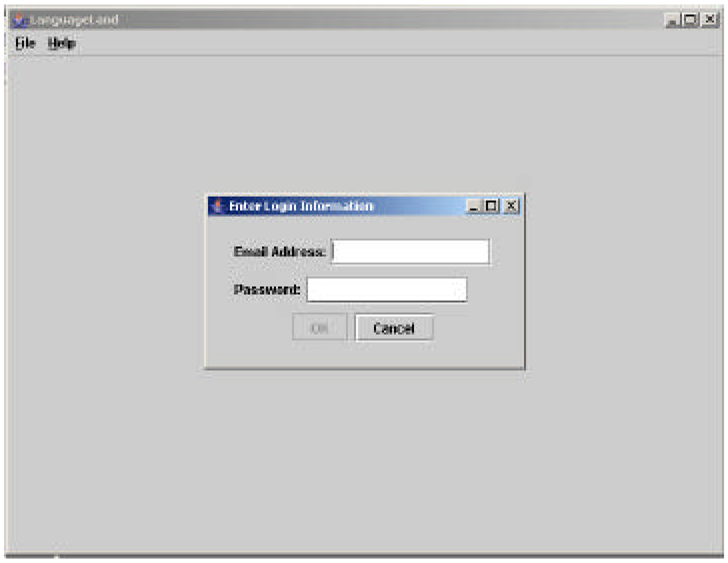
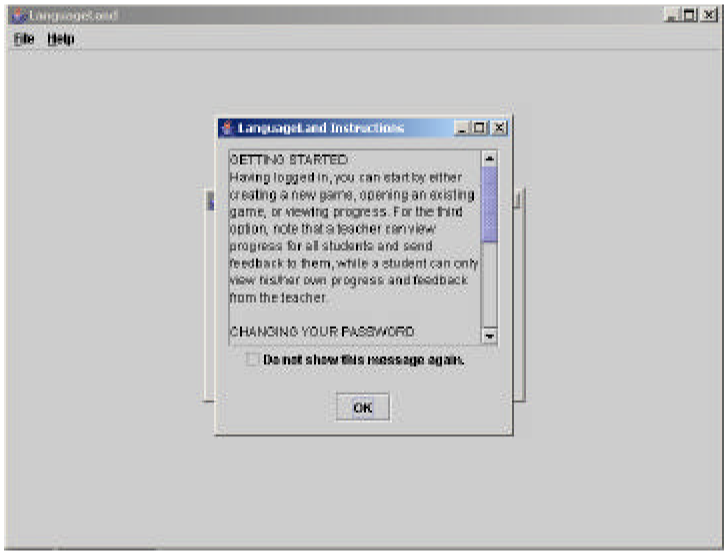


Figure 18: Login screen.

After successfully logging in, a set of instructions pop up that describe the Langua geLand system and what features are available. This screen is shown in Figure 19.



GETTING STARTED
Having logged in, you can start by either creating a new game, opening an existing game, or viewing progress. For the third option, note that a teacher can view progress for all students and send feedback to them, while a student can only view his/her own progress and feedback from the teacher.

CHANGING YOUR PASSWORD

Figure 19: Instructions for getting started.

As described in the "Help Structure" section, the user can opt not to have this instructions screen show again. After closing the screen, the user sees the welcome screen. For teachers, the screen is as shown in Figure 20, while for students it is like Figure 21.

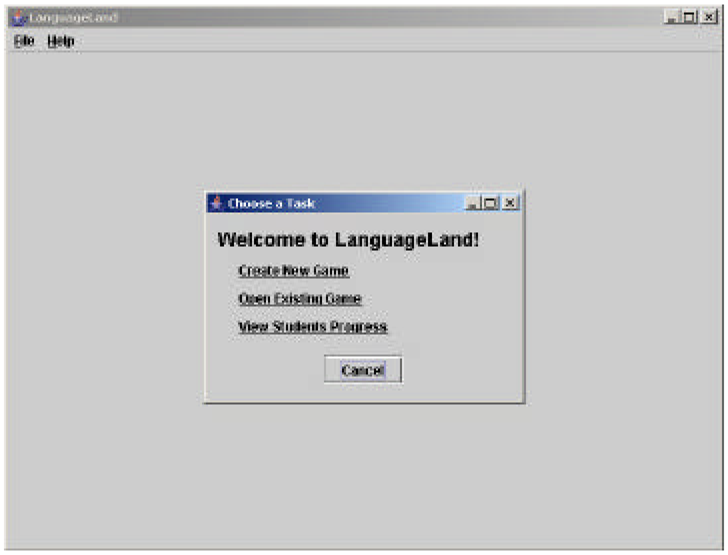


Figure 20: Welcome screen for teachers.

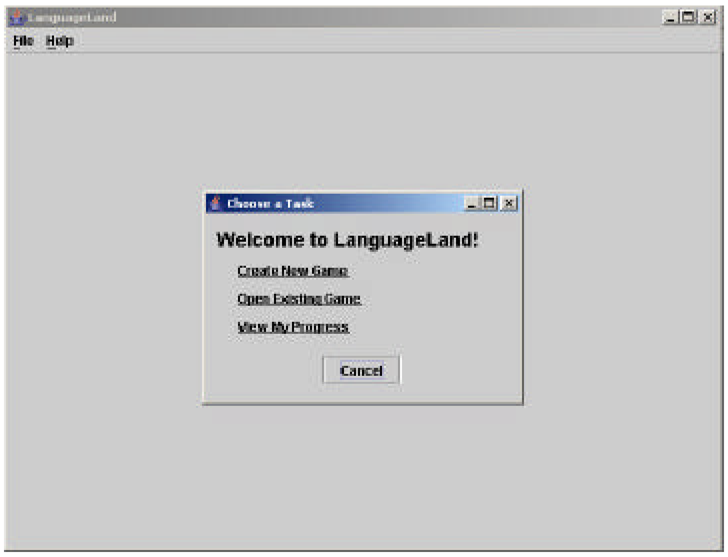


Figure 21: Welcome screen for students.

In the welcome screen, there are three options: create new game, open existing game, and view students' progress (for teachers) or view my progress (for students). If the user selects "Create New Game," a new screen opens up that allows the user to either specify a game name or accept the default name. The default game name is labeled "New Game (#)," where the "#" is incremented from 0 upwards so that all game names are unique. Figure 22 shows the game name screen.

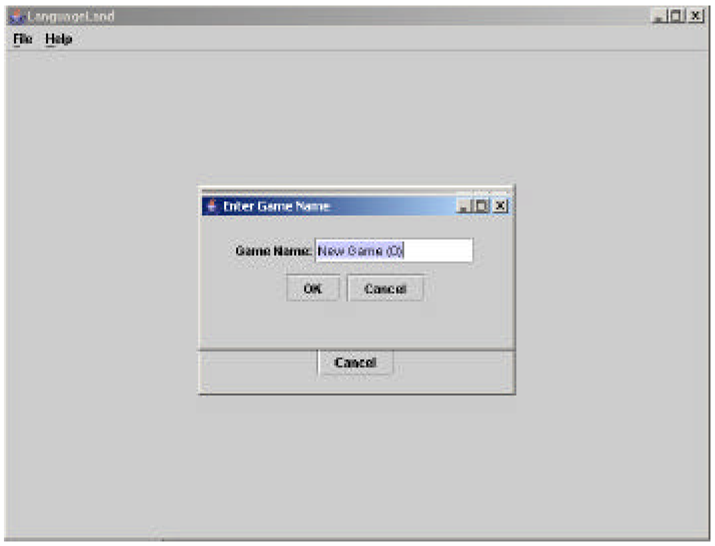


Figure 22: Game name screen.

Clicking "OK" on this screen leads to a new game in edit mode, which will be described in the "Edit Mode" section. If instead of choosing to create a new game, the user opens an existing game, a game selection screen pops up, as depicted in Figure 23

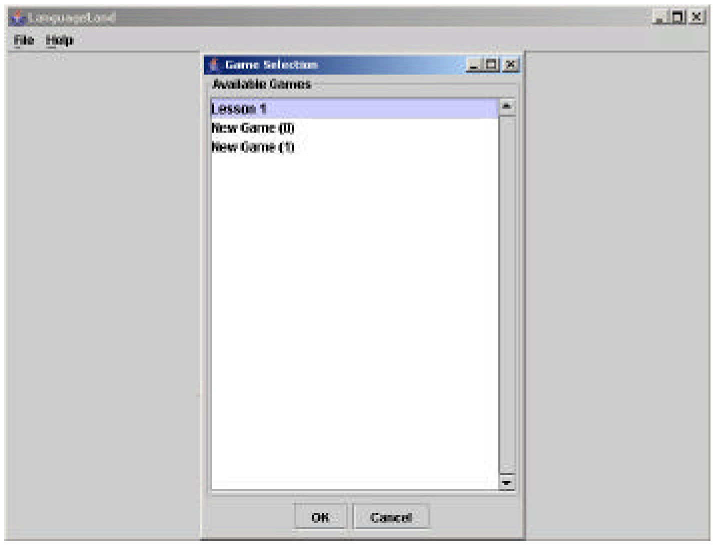


Figure 23: Game selection screen.

In the game selection screen, the user can choose to open any existing game. Since all game names are required to be unique, there is no room for ambiguity here. If instead of opening an existing game, the user had chosen to view progress, the corresponding progress screen would open up. This screen is described in the teacher and student progress view sections.

Finally, if the user closed the welcome screen without selecting any of the three options, all that is left are the “File” and “Help” menus. The “File” menu, as revealed in Figure 24, has all three options that were available in the welcome screen.

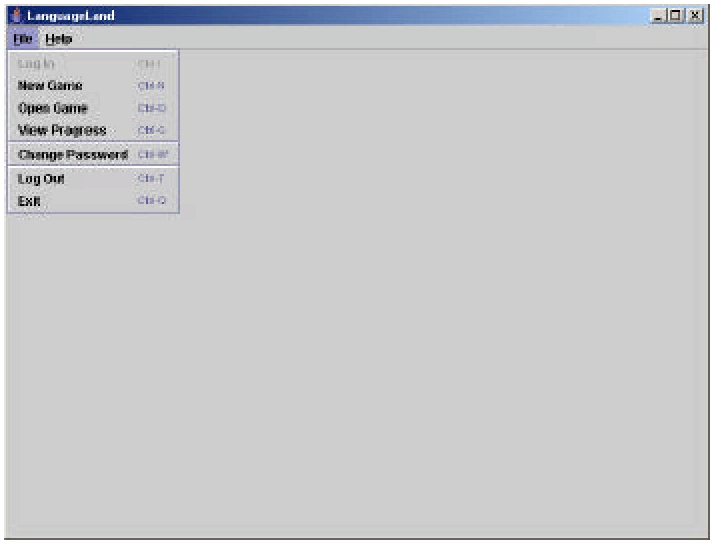


Figure 24: File menu in logged-in state.

The “Log in” option is disabled since the user is already logged in. It also has options to log out and to exit the application. If the user clicks on “Exit,” the whole application closes. If the user clicks “Log Out,” the user is logged out of the current session. Menu items in the logged-out state are described below. If the user clicks on “Change

Password," the screen in Figure 25 shows. The user just has to type in his old password once and new password twice in order to change it.

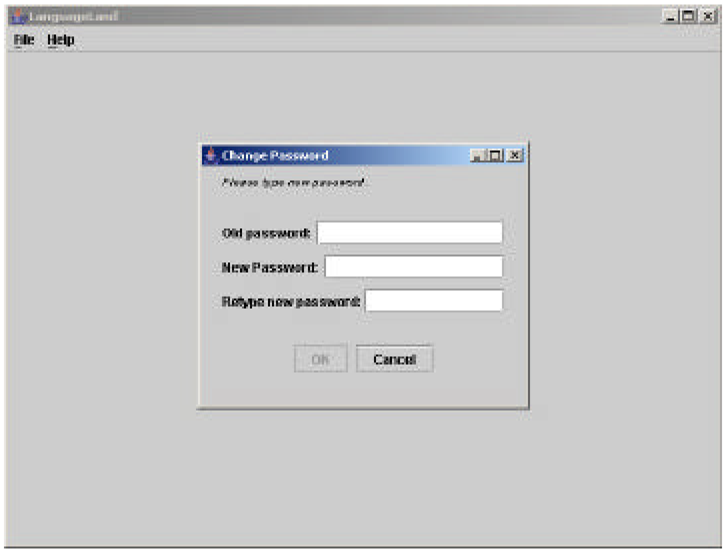


Figure 25: Change Password option in logged-in state.

The "Help" menu in the logged-in state has options to show the "About" and "Instructions" screens, as well as a toggle checkbox for showing the instructions on start. This is shown in Figure 26.

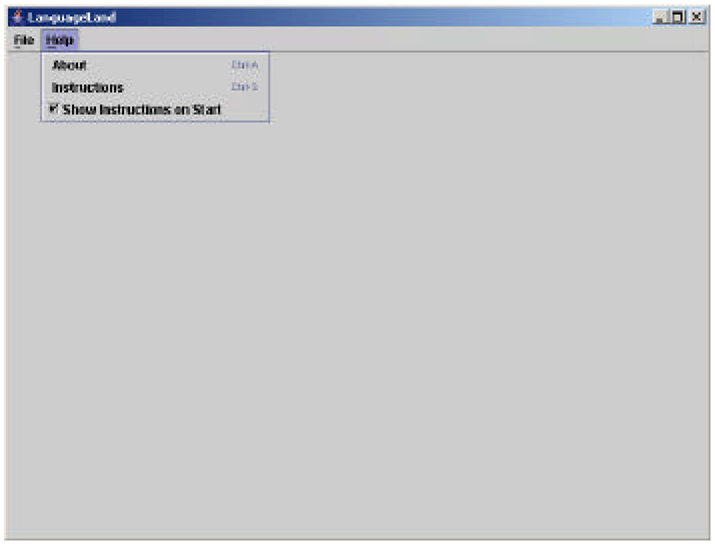
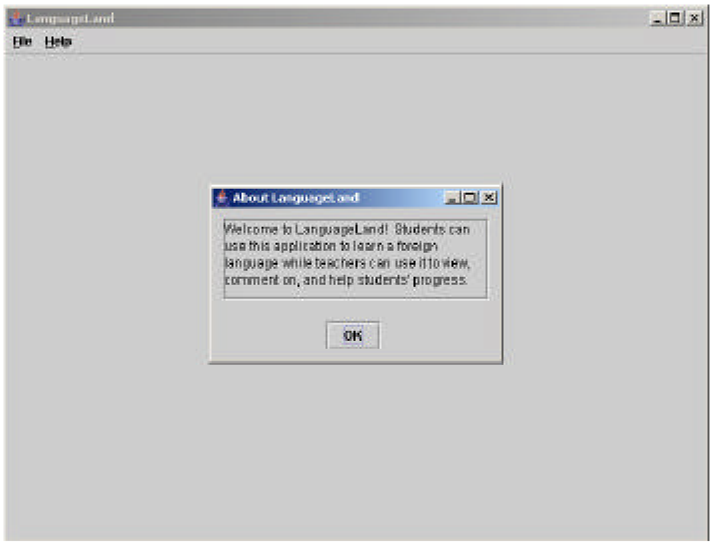


Figure 26: Help menu in logged-in state.

If the user clicks on "About," the screen in Figure 27 pops up.



Welcome to LanguageLand! Students can use this application to learn a foreign language while teachers can use it to view, comment on, and help students' progress.

Figure 27: "About" screen in logged-in state.

As described in the “Help Structure” section, if the user clicks on “Instructions” the instructions screen pops up. There is also the option to have the instructions screen always or never pop up when the user logs in.

In the logged-out state, most of the items in the “File” menu are disabled. In fact, the only available options are to log in or to exit the application, as can be seen in Figure 28.

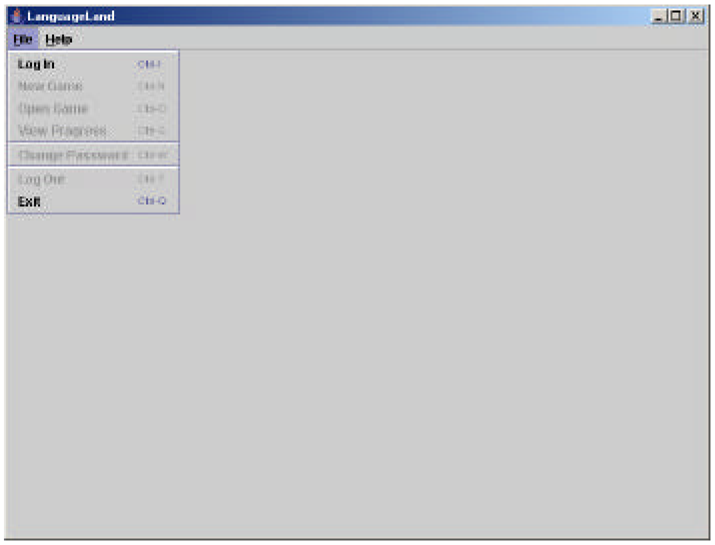


Figure 28: File menu in logged-out state.

The “Help” menu, shown in Figure 29, is the same as in the logged-in state except there is no option to check the box for whether or not the instructions screen should show on start. This is because no one is logged in yet and there would be no one to associate that setting with.

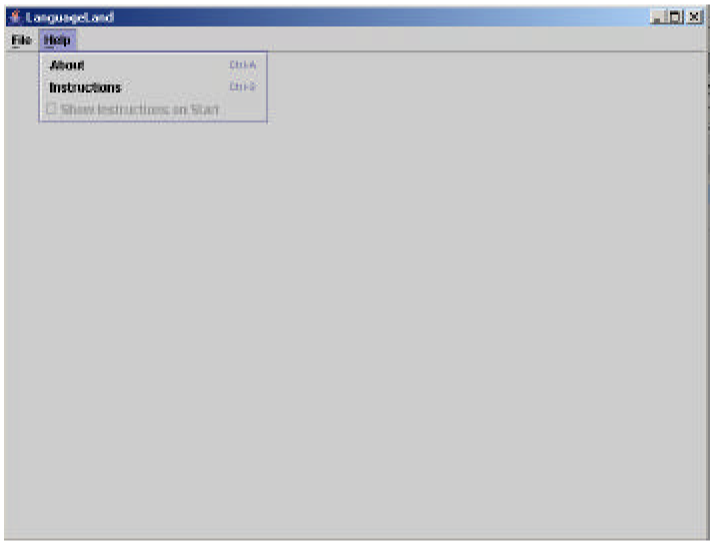
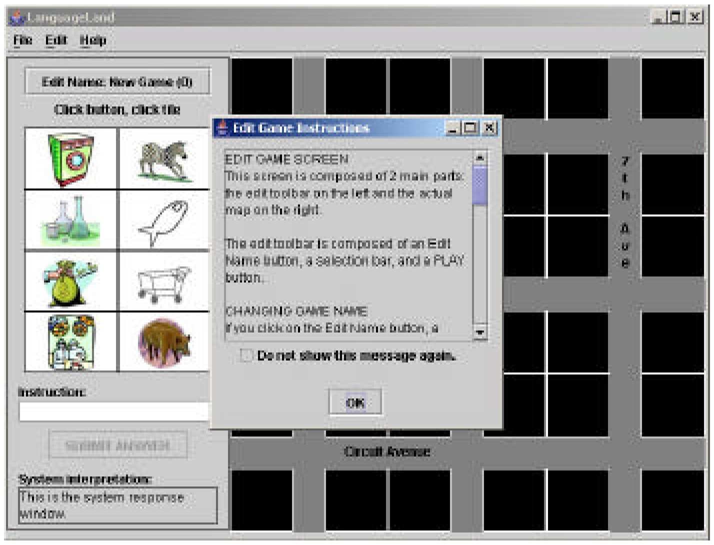


Figure 29: Help menu in logged-out state.

5.3 Edit Mode

When a user creates a new game, it automatically opens up in edit mode with the default instructions, which looks like the screenshot in Figure 30.



EDIT GAME SCREEN
This screen is composed of 2 main parts: the edit toolbar on the left and the actual map on the right.

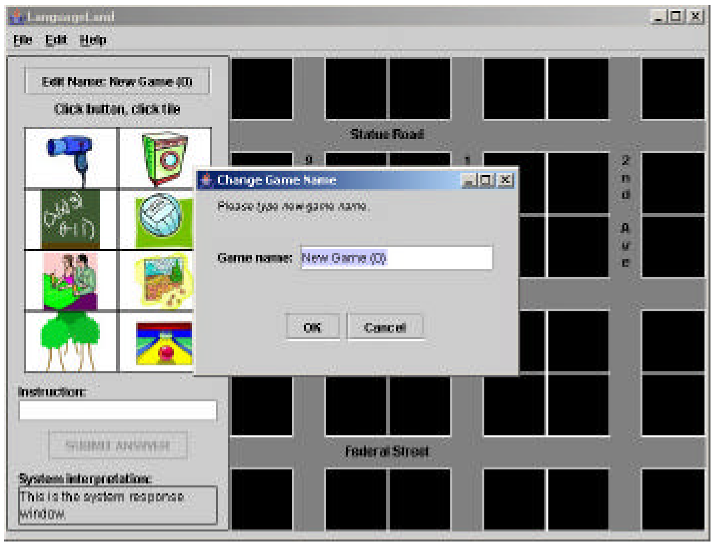
The edit toolbar is composed of an Edit Name button, a selection bar, and a PLAY button.

CHANGING GAME NAME
If you click on the Edit Name button, a

This is the system response window.

Figure 30: Instructions screen in edit mode.

On the left is an "Edit Name" button with the current game's name next to it. Users can click on it to change the game name, as shown in Figure 31.



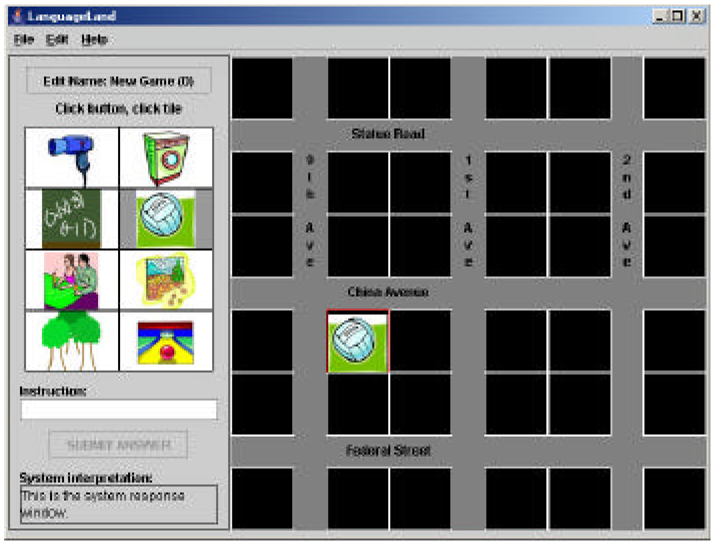
Please type new game name.

Game name: New Game (0)

This is the system response window.

Figure 31: Change game name screen in edit mode.

Underneath the "Edit Name" button are a series of eight buttons which contain the landmarks pertinent to the current game. Note that this dynamic set of landmarks is randomly picked for each game, so that the user can learn more vocabulary words. There are a total of 68 different landmarks. A few were drawn using Adobe Photoshop, but the rest were stripped from various Web pages offering free pictures. We tried to pick pictures that would resemble a venue, object, or something of learning value. Some examples are: washer, volleyball, and bowling alley. On the right is the game map, with dynamic street names which are also randomly picked. As mentioned in Chapter 4, there are a total of 113 streets, all of which have translations in both English and Chinese. In the edit mode, users can get by with just their mouse by clicking on landmarks on the left and then double-clicking on map blocks on the right. One click on a block causes a red outline to appear around that block, while another click places the currently selected landmark onto the block. A right-click on a block erases the current landmark on it, if there exists one. Or, users can choose to place new landmarks over old ones directly. Figure 32 shows an example of a volleyball being placed on a block.



This is the system response window.

Figure 32: Placing item on map in edit mode.

On the left, below the landmarks, is the "Instruction" textbox, in which users can type in commands. Right below that is the button to submit commands, and below that is the "System Interpretation" window, in which translations and responses are shown. If, for example, the user types in "what is this?" and submits it, the "System Interpretation" window will have the Chinese translation, "zhe4 shi4 shen2 me5." If the user then types in "zhe4 shi4 shen2 me5," the "System Interpretation" will show the Chinese name of the landmark currently selected. Figure 33 shows an example of asking the system "zhe4 shi4 shen2 me5" when the bowling alley is selected. The "System Interpretation" window shows "bao3 ling2 qiu2 di4," the Chinese name for bowling alley.

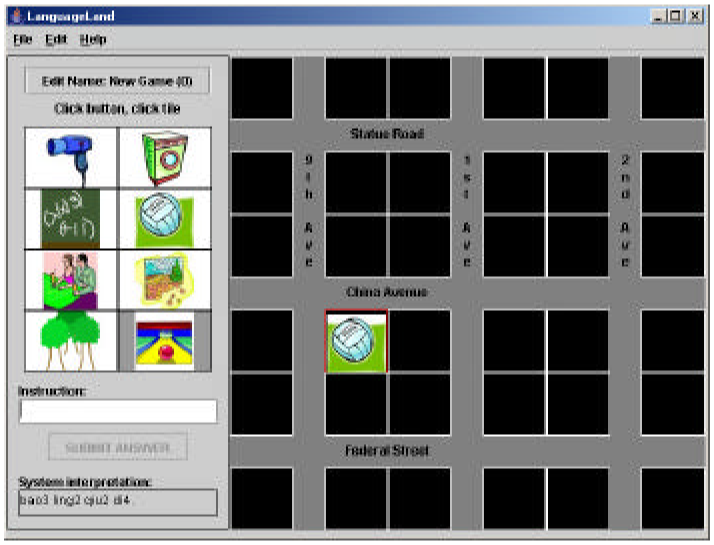


Figure 33: Asking the system "zhe4 shi4 shen2 me5" when the bowling alley is selected in edit mode.

The "File" menu for the edit mode is the same as that of the welcome screen. The "Edit" menu has only one item, "Play." Clicking on it will transfer the user to the play mode of this game, but only if there exists at least one item on the map. Otherwise, there is no game to play. This "Edit" menu is shown in Figure 34.

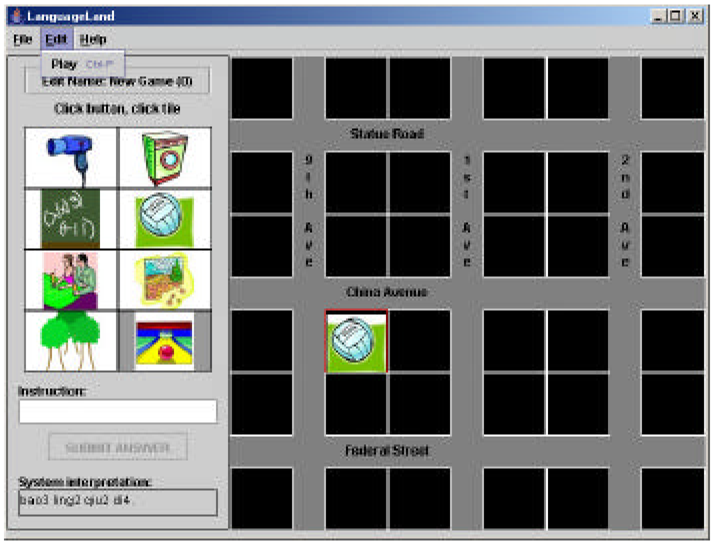


Figure 34: Edit menu in edit mode.

The "Help" menu, shown in Figure 35, has the "Instructions" item and "Show Instructions on Start" checkbox, which function just like those in the welcome screen.

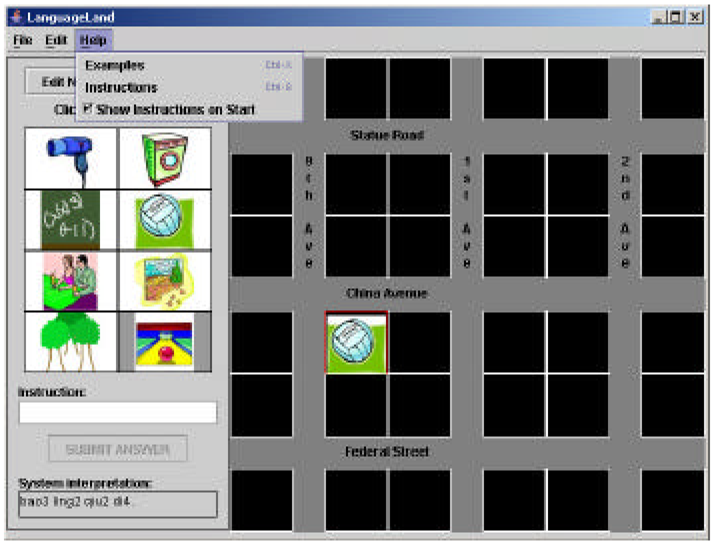
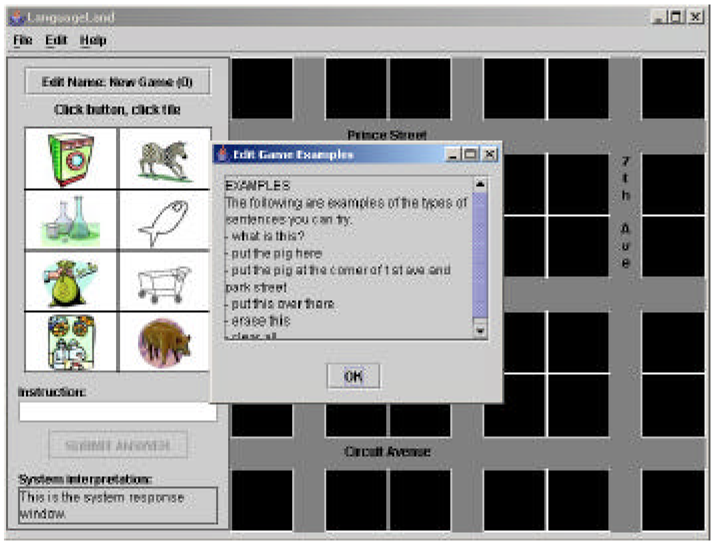


Figure 35: Help menu in edit mode.

There is also an "Examples" item, which opens up a dialogue box that shows examples of the types of sentences the user can try. This is shown in Figure 36.



EXAMPLES
The following are examples of the types of sentences you can try.
- what is this?
- put the pig here
- put the pig at the corner of 1st ave and park street
- put this over there
- erase this
- clear all

This is the system response window.

Figure 36: Examples screen in edit mode.

The key-value bindings table in Chapter 4 is condensed in Table 2 into just the entries which are pertinent to the edit mode.

Key	**Value**
Clear	This, that, history, map
What	This
Landmark	<landmark>
Put	This, that, <landmark>
Location	Here, there
Street1	<street_name>
Street2	<street_name>
Street_type1	Street, avenue, road
Street_type2	Street, avenue, road

Table 2: Key-value bindings in edit mode.

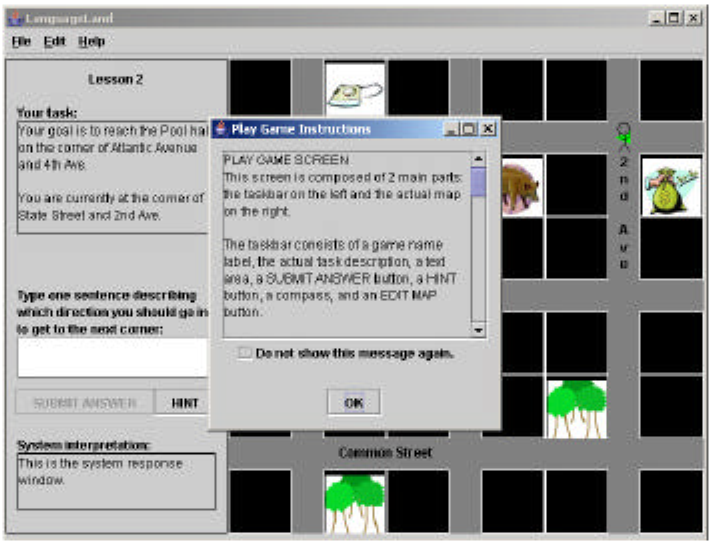
Sentences like “ca1 diao4 zhe4 ge5” (clear this) give the key “clear” and the value “this,” causing the currently selected block on the map to be erased. The multimodal nature of this and other commands allows the user to both type and click with the mouse. If the value is “history” or “map,” all blocks on the map are erased. If a user says “zhe4 shi4 shen2 me5” (what is this?), the key is “what” and the value is “this.” The system determines which of the landmarks on the left are selected and sends a message with the landmark’s English name to the natural language backend, which then passes back out the Chinese translation to be shown in the “System Interpretation” text area. If the user types the landmark’s English name into the window directly, the system will also pass back out the Chinese translation of the landmark.

A “fang4 zhe4 ge5 zai4 zhe4 li3” (put this here) sentence will give the key “put” and the value “this,” along with the key “where” and the value “here.” The system responds by placing the currently selected landmark onto the currently selected map block. The user could also say “fang4 zhu1 zai4 shan1 hu2 jie1 he2 gong1 yuan2 da4 dao4 de5 jie1 tou2” (put the pig at the corner of coral street and park avenue), and the corresponding key-value pairs are: “put” and “pig,” “street1” and “coral,” “street_type1” and “street,” “street2” and “park,” “street_type2” and “avenue.” The system places the landmark pig at the specified corner block on the map.

Note that not only can the user type to the system, but she can also speak to it on the telephone. By typing “call me at #” (where “#” stands for a 10-digit phone number including the area code), the user prompts the system to call that number. With both typing and speaking, if the user communicates in English, the system responds with a Chinese translation, both in the "System Interpretation" window and also over the phone, if spoken. If the user communicates in Chinese, the system performs the command and also repeats what the user says as confirmation.

5.4 Play Mode

The play mode for a game can be accessed either via the edit mode (after a user has set up the map) or by opening an existing game that is already in play mode. The first time it is opened in a specific session, the instructions screen pops up as in Figure 37.



Your goal is to reach the Pool hal on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2nd Ave.

Type one sentence describing which direction you should go in to get to the next corner:

System interpretation:

This is the system response window.

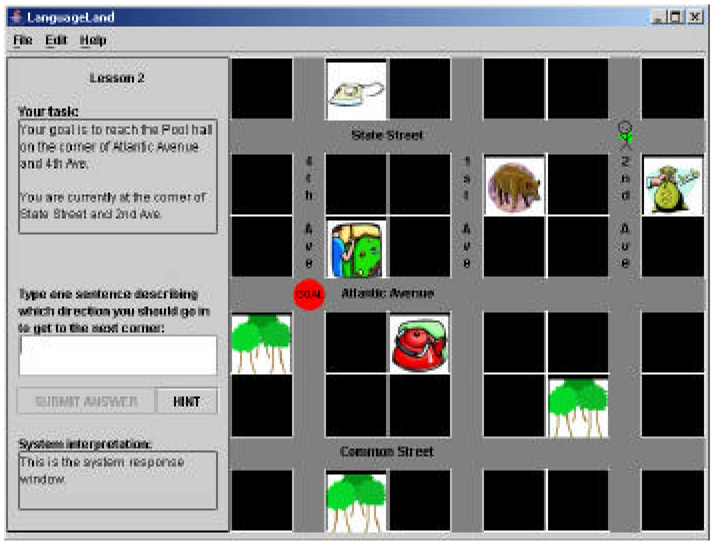
PLAY GAME SCREEN
This screen is composed of 2 main parts: the taskbar on the left and the actual map on the right.

The taskbar consists of a game name label, the actual task description, a text area, a SUBMIT ANSWER button, a HINT button, a compass, and an EDIT MAP button.

Do not show this message again.

Figure 37: Instructions screen in play mode.

Like in all other stages, the user can choose to not have this instruction screen show up anymore in subsequent sessions. Figure 38 shows the play mode right after a game starts up.



Your goal is to reach the Pool hall on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2nd Ave.

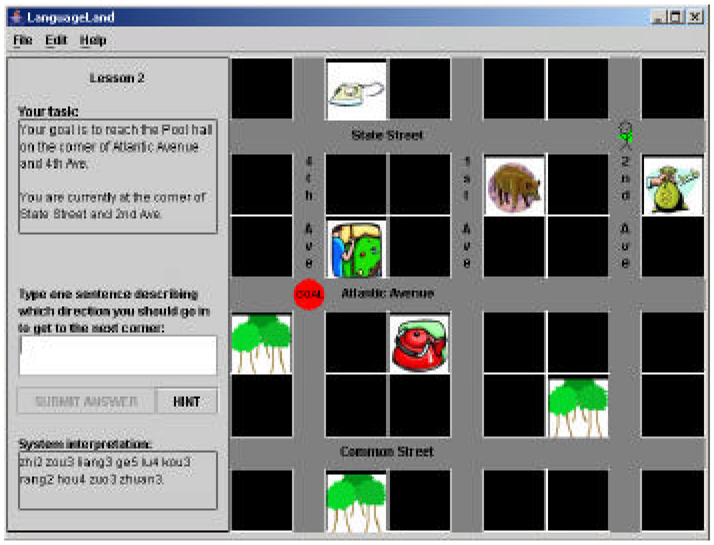
Type one sentence describing which direction you should go in to get to the next corner:

This is the system response window.

Figure 38: Play mode at startup.

In play mode, one of the landmarks on the map (in this case, the pool hall) is randomly chosen and its corner block becomes the goal, as marked by the red, filled-in circle with the word "GOAL" written across it. The text area on the upper left part of the screen details the task at hand. The starting point, marked by the human stick figure and the green direction arrow, is somewhat randomized but not completely. There are nine corner blocks on the map. If the goal is on one of the corner blocks on the top-left, top-right, bottom-left, or bottom-right, then the starting point is always the opposite corner block. But, for example, if the goal is in the center corner block of the map, then the starting point is any corner block but the middle one. A similar algorithm holds for the other corner blocks. The key motives are to have randomization and to make sure the starting point and goal are not too close to each other, if possible. Also note that the direction for the green arrow (marked by the tip of the triangle) on the starting point is randomized with uniform probability of facing north, south, east, or west. The reason we wanted this probability is so students can practice reorienting themselves at the beginning of a game.

The structure for typing and speaking is the same as in edit mode. That is, if a user communicates in English, the system responds with the Chinese translation of the sentence. If the user communicates in Chinese, the system paraphrases the same sentence in Chinese and also performs the command. For example, if the user submits "walk straight for 2 blocks and then turn left," the "System Interpretation" text area will present "zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3," as shown in Figure 39.



Your task:

Your goal is to reach the Pool hall on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2nd Ave.

Type one sentence describing which direction you should go in to get to the next corner:

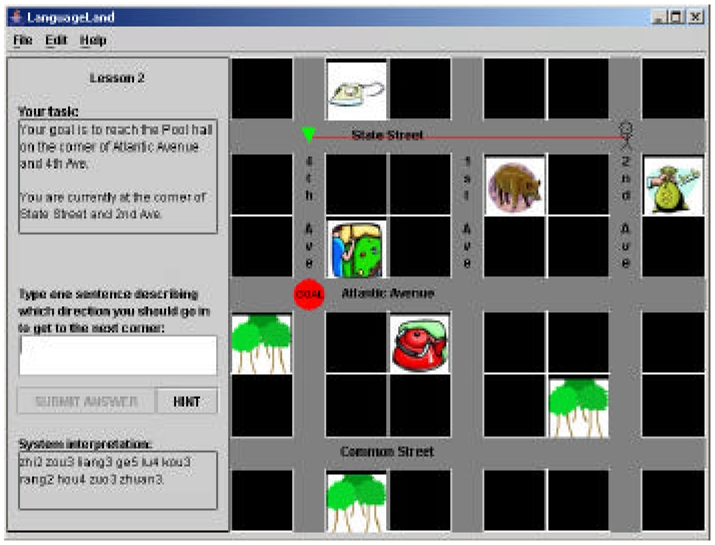
System interpretation:

zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3

Figure 39: After typing "walk straight for 2 blocks and then turn left" in play mode.

If the user then submits the translated Chinese sentence, the system paraphrases the sentence in Chinese and performs the action of going straight for two blocks and turning left. In Figure 40, the red line shows the walking and the green arrow shows the left turn.

While the green arrow is allowed to rotate freely without leaving a trail, red lines drawn on the screen are never erased, thus showing a user's history.



Lesson 2

Your task:

Your goal is to reach the Pool hall on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2nd Ave.

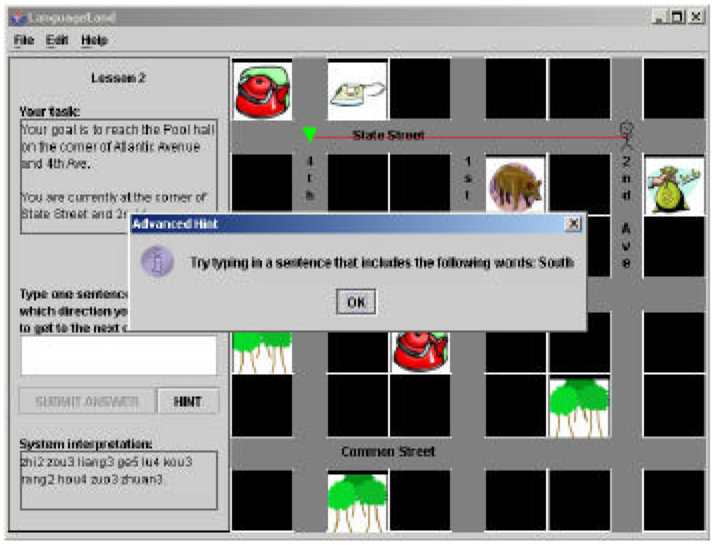
Type one sentence describing which direction you should go in to get to the next corner:

System interpretation:

zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3

Figure 40: After typing "zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3" in play mode.

If the user is stuck, she can click on the "HINT" button which will give a vague hint as to which direction to proceed in. If the user clicks on the button again without having made any progress, the hint becomes a little more detailed. Currently, there are only two hint levels. Figure 41 shows an example of an advanced (second level) hint.



Lesson 2

Your task:

Your goal is to reach the Pool hall on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2n

Try typing in a sentence that includes the following words: South

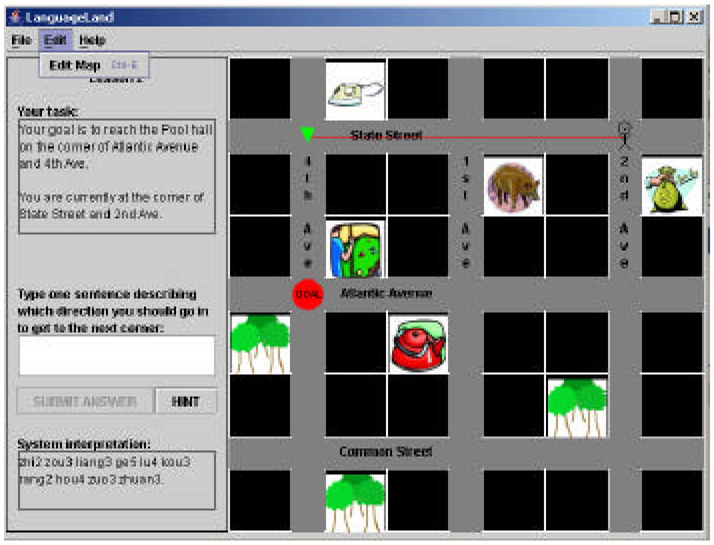
Type one sentenc which direction yo to get to the next c

System interpretation:

zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3.

Figure 41: Advanced hint in play mode.

The "File" menu for the play mode is the same as in the edit mode. The "Edit" menu shown in Figure 42 has an "Edit Map" item. Clicking on it brings the user back to the edit mode of the current game, from which the user can make changes to the existing map.



Your task:

Your goal is to reach the Pool hall on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2nd Ave.

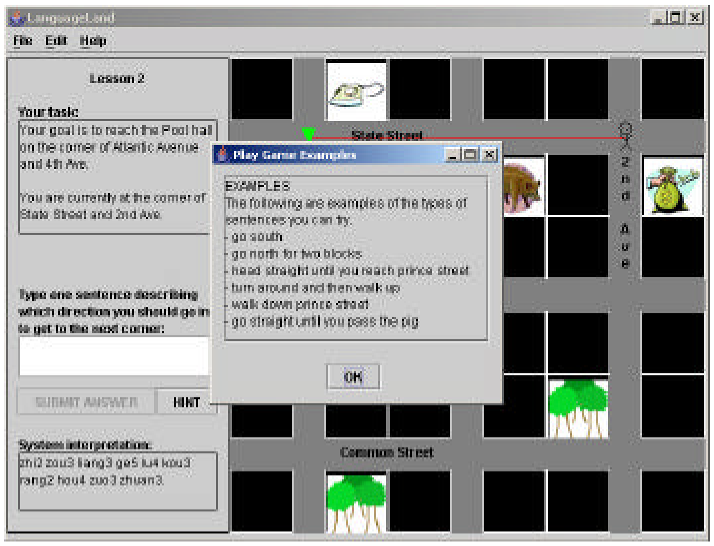
Type one sentence describing which direction you should go in to get to the next corner:

System interpretation:

zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3.

Figure 42: Edit menu in play mode.

The “Help” menu in the play mode is the same as that of the edit mode, except that clicking on the “Examples” item brings up a different screen. This screen for the play mode is depicted in Figure 43.



Your goal is to reach the Pool hall on the corner of Atlantic Avenue and 4th Ave.

You are currently at the corner of State Street and 2nd Ave.

zhi2 zou3 liang3 ge5 lu4 kou3 rang2 hou4 zuo3 zhuan3.

EXAMPLES
The following are examples of the types of sentences you can try:
- go south
- go north for two blocks
- head straight until you reach prince street
- turn around and then walk up
- walk down prince street
- go straight until you pass the pig

Figure 43: Examples screen in play mode.

The key-value bindings table in Chapter 4 is reduced to entries relating to the play mode, as shown in Table 3.

Key	**Value**
Direction	North, south, east, west, special, straight, back
Block	1, 2, 3, …
When	Reach, pass

Turn	Right, left, around
Landmark	<landmark>
Street_name	<street_name>
Street_type	Street, avenue, road

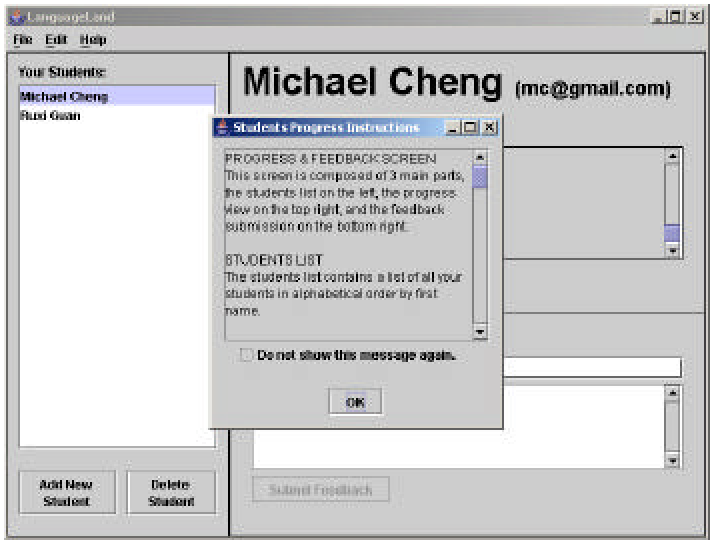
Table 3: Key-value bindings in play mode.

Sentences such as “wang3 nan2 zou3” (go south) have the key “direction” and value “south,” and instructs the system to go south for a block. If the user says “zai4 wang2 zi3 jie1 shang4 zou3” (go down Prince Street), the bindings are: key “direction” and value “special,” key “street_name” and value “prince,” and key “street_type” and value “street.” Assuming that one of the current cross streets is Prince Street, the system walks one block on Prince Street. The direction of traversal depends on the green arrow. If it is already aligned along Prince Street, it goes in that direction. Otherwise, the direction is randomly picked to align with Prince Street. A sentence like “xia4 zou3 liang3 ge5 lu4 kou3” (walk down for two blocks) provides key “direction” and value “south,” and key “block” and value “2.” The sentence “zuo3 zhuan3” (turn left) gives key “turn” and value “left,” which causes a left turn. The user can also combine instructions such as asking the system to walk in a certain direction until it reaches a landmark and then turn right at a particular cross street. For example, the sentence “wang3 xi1 zou3 zhi2 dao4 zhu1 rang2 hou4 zai4 wang2 zi3 jie1 you4 zhuan3” results in the following key-value frame: “seq: <start> direction: west when: reach landmark: pig <and> turn: right when: reach street_name: prince street_type: street <end>.” Notice the sequence “<start>,” “<and>,” and “<end>” markers separating the two instructions. The key “when” and value “reach” tell the system to wait until the corresponding landmark or street is reached before stopping or turning.

Synonymous to the edit mode, the play mode also supports both typed and spoken input, and the user can submit a “call me at #” phrase in order to be called by the system at a specific number.

5.5 Teacher Progress View

Teachers can examine their students’ performance via the teacher progress view portion of the system. The first time a teacher opens it, an instructions dialogue opens up. Just like in all other cases, the user can choose to not have this open up in future sessions. This screen is shown in Figure 44.

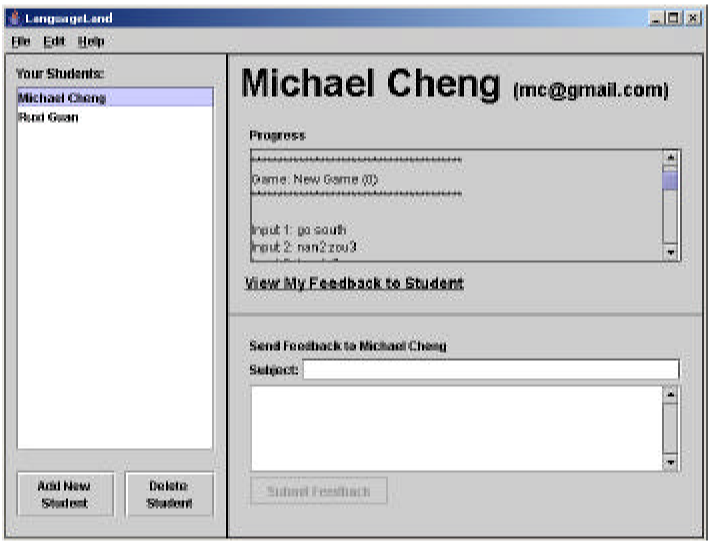


PROGRESS & FEEDBACK SCREEN
This screen is composed of 3 main parts, the students list on the left, the progress view on the top right, and the feedback submission on the bottom right.

STUDENTS LIST
The students list contains a list of all your students in alphabetical order by first name.

Figure 44: Instructions screen in teacher progress view.

To the left of the teacher progress view is a list of existing students, ordered alphabetically by first name. The student at the top of the list is selected by default when this screen is opened. Clicking on a particular student will pull up that student's data on the right. In Figure 45, Michael Cheng's data is shown on the right.

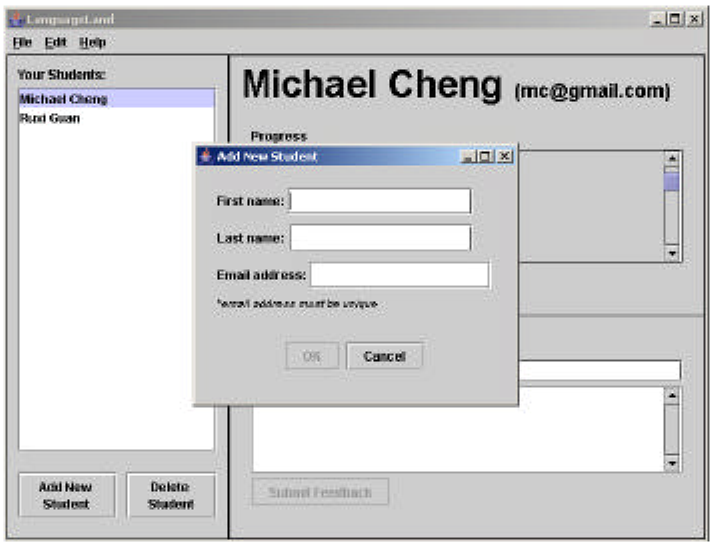


```
**********************************
Game: New Game (0)
**********************************


Input 1: go south
Input 2: nan2zou3
```

Figure 45: Michael Cheng's data in teacher progress view.

The teacher can add new students to the database at any time. The only requirement is that each student's email address is unique, as that becomes the login ID. It is okay to have students with the same first and last names. The dialogue for adding a new student is depicted in Figure 46.

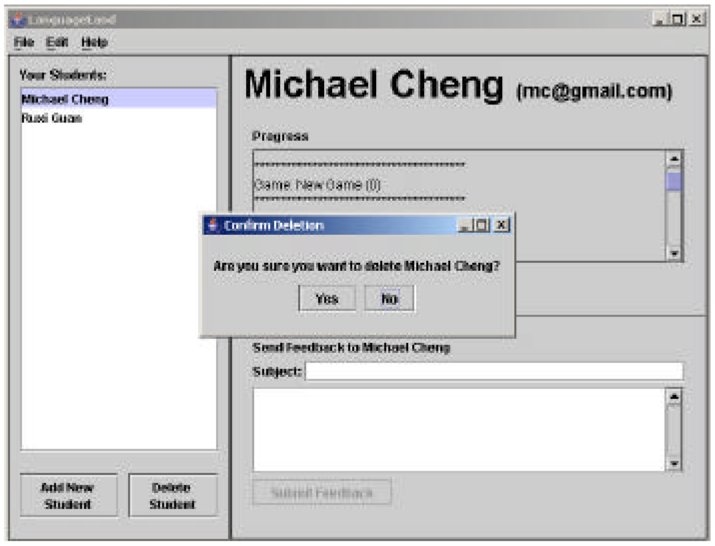


Michael Cheng (mc@gmail.com)

*email address must be unique

Figure 46: Adding a new student in teacher progress view.

The teacher can also delete students from the database. To ensure that the user actually wants to perform the deletion, there is a confirmation dialogue which is shown in Figure 47. By default, "No" is selected to further prevent unwanted deletions.



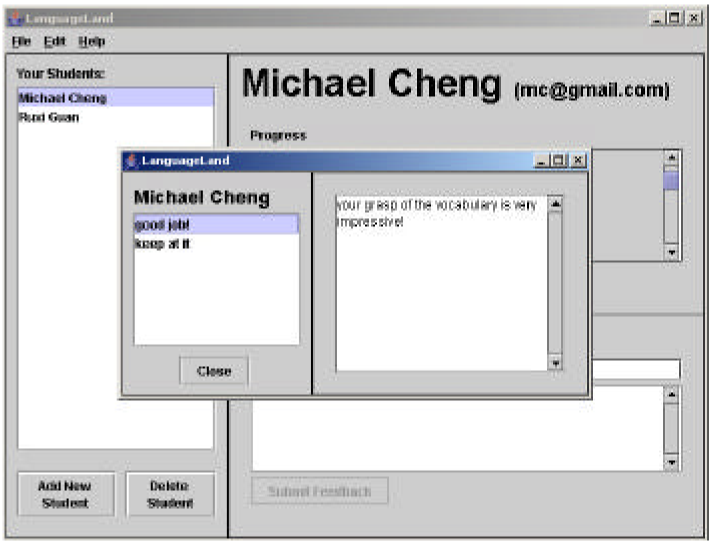
Michael Cheng (mc@gmail.com)

```
****************************************
Game: New Game (0)
****************************************
```

Are you sure you want to delete Michael Cheng?

Figure 47: Deleting a student in teacher progress view.

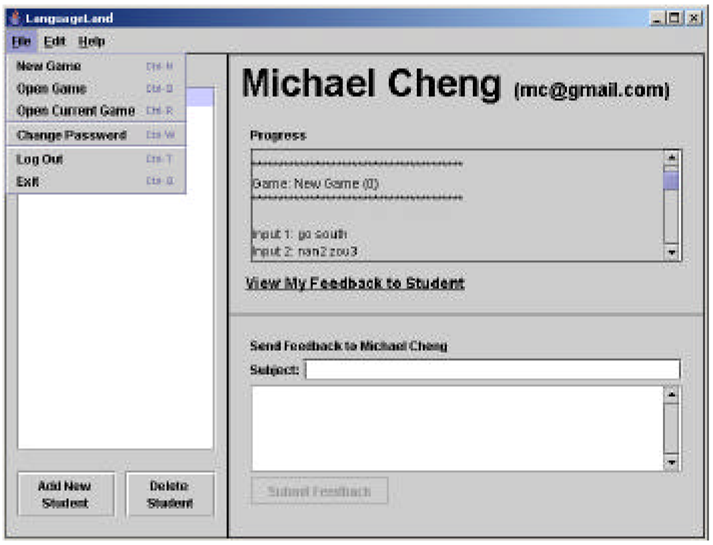
To the right of the screen is a "Progress" text area which shows progress made by a particular student in all his games. Since students don't have an option to delete the history of their actions, this list is comprehensive and allows the teacher to view students' mistakes as well as accomplishments. There is also a form underneath where teachers can write and submit feedback to the student. If the teacher clicks on the "View My Feedback to Student" link (which can be viewed in Figure 45) on the right of the screen, the feedback dialogue pops up with all written entries. Feedback entries are ordered alphabetically by the subject title, thus requiring that all feedbacks have unique subjects. Teachers may want to include dates or game names at the beginning of subject titles for easier identification. The feedback at the top of the list is selected by default. Clicking on one will pull up that feedback's body on the right. Note that there is no option to delete feedback as they are meant to be comprehensive. This feedback dialogue is shown in Figure 48.



your grasp of the vocabulary is very impressive!

Figure 48: Viewing feedback to student in teacher progress view.

The “File” menu is similar to the ones in the other screens, except that the third menu item, “View Progress,” is replaced by “Open Current Game.” Clicking on it just pulls up the most recently opened game, regardless of whether it is in edit or play mode. Figure 49 shows the “File” menu.



Michael Cheng (mc@gmail.com)

Progress

```
**********************************
Game: New Game (0)
**********************************


Input 1: go south
Input 2: nan2 zou3
```

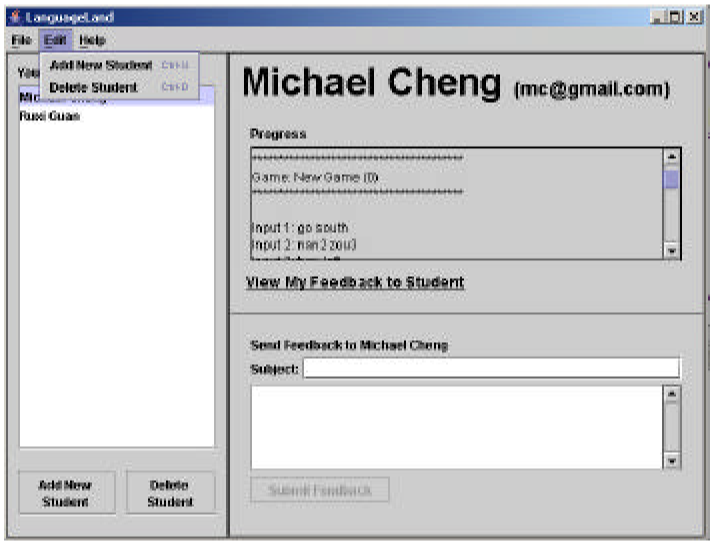
View My Feedback to Student

Send Feedback to Michael Cheng

Subject:

Figure 49: File menu in teacher progress view.

The “Edit” menu, as shown in Figure 50, contains two items: “Add New Student” and “Delete Student.” These items correspond to the two buttons on the lower left of the screen.

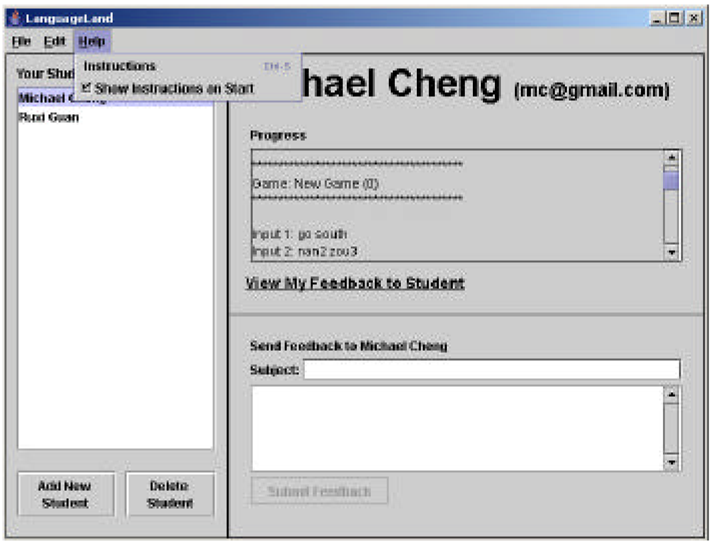


```
Game: New Game (0)

Input 1: go south
Input 2: nan2 zou3
```

Figure 50: Edit menu in teacher progress view.

The “Help” menu, as shown in Figure 51, is similar to the ones in the edit and play modes. The only difference is that the “Examples” menu item is removed, since there is no language learning component in the progress view.



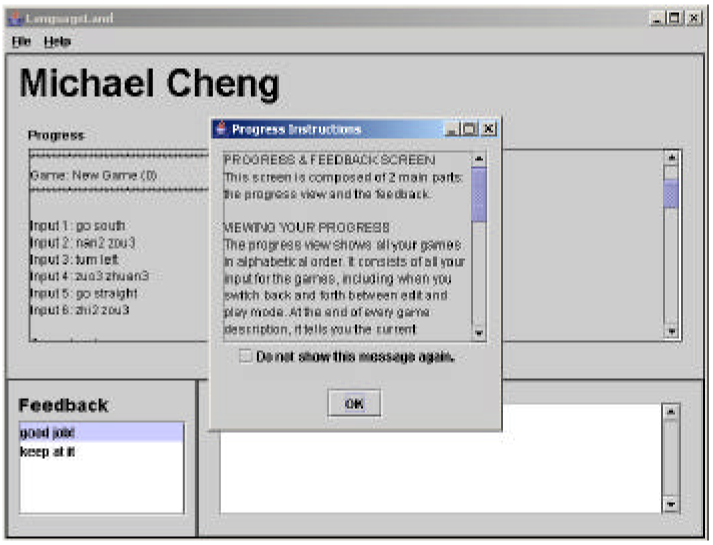
```
Game: New Game (0)

Input 1: go south
Input 2: nan2 zou3
```

Figure 51: Help menu in teacher progress view.

5.6 Student Progress View

Students have a similar progress view, except that they can only view their own data. When it is first opened, the instructions screen depicted in Figure 52 pops up as usual. There is an option to have it never show upon startup again.



Game: New Game (0)

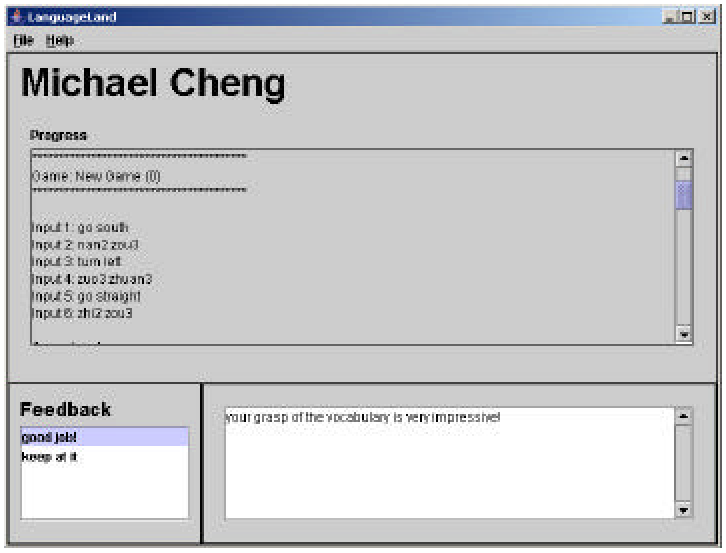
Input 1 : go south
Input 2 : nan2 zou3
Input 3 : turn left
Input 4 : zuo3 zhuan3
Input 5 : go straight
Input 6 : zhi2 zou3

PROGRESS & FEEDBACK SCREEN
This screen is composed of 2 main parts: the progress view and the feedback.

VIEWING YOUR PROGRESS
The progress view shows all your games in alphabetical order. It consists of all your input for the games, including when you switch back and forth between edit and play mode. At the end of every game description, it tells you the current

Figure 52: Instructions screen in student progress view.

The main progress view, shown in Figure 53, contains the student's data. At the top is a list of all progress made so far in all games. There is no option for a student to erase history of past actions because we do not want students to erase mistakes and simply show the good parts. On the bottom of the screen is the feedback area where a student can view everything a teacher has written to him. The feedback entries are ordered alphabetically by subject title, with the one at the top of the list selected by default. Clicking on an entry opens up its feedback body to the right.



```
****************************************
Game: New Game (0)
****************************************


Input 1: go south
Input 2: nan2 zou3
Input 3: turn left
Input 4: zuo3 zhuan3
Input 5: go straight
Input 6: zhi2 zou3
```

```
good job!
keep at it
```

```
your grasp of the vocabulary is very impressive!
```

Figure 53: Michael Cheng's student progress view.

The "File" and "Help" menus are identical to the ones in the teacher progress view. They can be viewed in Figure 49 and Figure 51, respectively.

5.7 System Integration and Communication

A crucial part of the LanguageLand application is its connection to and communication with the natural language components (Cyphers [4]). This communication is facilitated by a central hub. We wrote a script to run a hub and the following servers in the backend: NL, Discourse, MapGame Recognizer, Waveform, and Envoice. The hub communicates with each of the servers on a different port. It also leaves a port open for communication with the LanguageLand application and waits for a connection to be established. The following sections describe the Connection Manager's role in the LanguageLand application, sending dynamic vocabulary, sending messages, and receiving messages.

5.7.1 Connection Manager

When the LanguageLand application first starts up, it creates a hub client server, the Connection Manager, which connects with the hub and waits for messages on the specified port. It also establishes a unique session ID (based on the system time in milliseconds) and an increment ID that starts off at 0 and is manually incremented by 1 every time the application sends a message to the hub. This ensures that messages do not get mixed up.

5.7.2 Sending Dynamic Vocabulary

The dynamic vocabulary only applies for the English-only version of the system. Every time a new game is opened, the application creates a new frame that contains a list of two clauses. The clauses contain the list of landmarks and street names, respectively, for the current game. The system then sends the frame to the hub, thus ensuring that the recognizer only deals with this subset of landmarks and street names.

5.7.3 Sending Messages

Normal messages are sent similarly. When the user is in edit or play mode and submits a message, a new frame is created which contains the input string and utterance ID. The system then increments the utterance ID before sending off the message to the hub. This process is asynchronous in that the system will not wait for a reply message. Thus, a user could potentially submit two strings before receiving a reply for the first one. This ensures that the system is not blocked by slow network connectivity.

5.7.4 Receiving Messages

The process of receiving messages is also asynchronous. When the system receives a message through the Connection Manager, it first looks for two string properties in the message frame: “hub_session_language” and “hub_session_learn_language.” The first one specifies the language the user typed or spoke in (either English or Chinese). The second one is the language to be learned (Chinese). If the first language is English but the second language is Chinese, then our system just displays the translated phrase (from English to Chinese) on the screen. If both languages are Chinese, however, then our system also performs the command, which is stripped from the key-value bindings in the eform. The reason we have both the “hub_session_language” and the “hub_session_learn_language” parameters is for flexibility in extending the system, in case we want to change things such as having the learned language be English instead of Chinese. It is important to note that our system distinguishes between receiving messages in the edit versus play mode, since the set of recognized key-value bindings is different. This division occurs right after the message is received.

Chapter 6

Evaluation

This chapter discusses evaluation of the LanguageLand system, both for its user interface and for its natural language component.

6.1 User Interface

The user interface of the LanguageLand system went through several iterations, following the waterfall model, which is a software development model introduced by W. W. Royce in 1970 (Royce [13]). During each iteration, multiple users tested the interface based on several sets of standard criteria, such as Jakob Nielsen's ten usability heuristics (Molich et al. [12]). The system started out as a paper prototype. The edit mode for the prototype is shown in Figure 54.

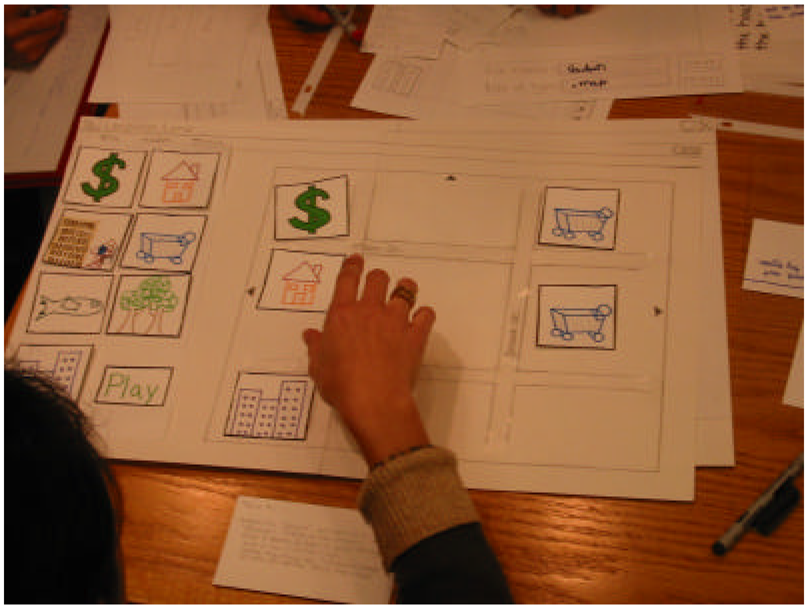


Figure 54: Edit mode in paper prototype.

Figure 55 shows the prototype for the teacher progress view.

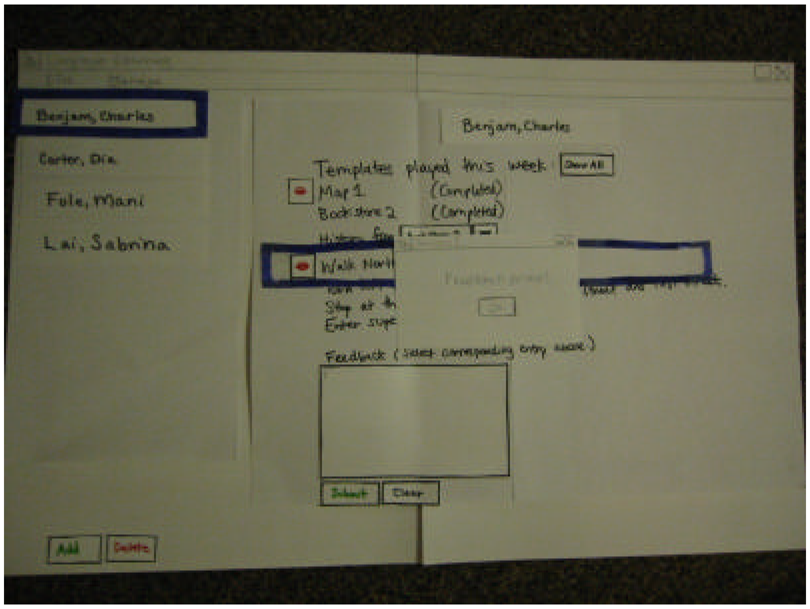


Figure 55: Teacher progress view in paper prototype.

Creating a paper prototype before delving into the coding process proved helpful because the prototype takes less time to make and we were able to adapt to and fix things on the fly during user testing. Also, users see that the system is still in the "draft" stage and are more willing to offer suggestions and criticism. After gathering results from the paper prototype, we proceeded in building an initial version of the system using Visual Basic, which is mostly used for storybook applications because it does not require extensive coding. This version of the system had a horizontal look and feel, meaning most of the screens looked complete but there was not much depth to the available actions. After completing another round of usability testing, we began to develop the actual system using Java.

The Java version of the system also went through user testing, and we incorporated many of the changes suggested by users. Some of these changes include providing help to the user. In response, we added instruction dialogues, examples, and a hint function. There were also a lot of minor suggestions such as choosing colors that blend well and also stand out. Another addition is the green arrow, which we added to show users their current orientation. Overall, we believe the extensive testing and redesigning of the user interface makes the system comprehensive and easy to use.

6.2 Natural Language Component

Though not as formally, the natural language component went through several iterations as well. The system started with a simple, hardwired artificial intelligence which only understood the four cardinal directions: north, south, west, and east. We then attempted to add other simple commands such as “go north” and “turn left.” Eventually, we incorporated the natural language components to form an actual “intelligent” system which understands and processes user input. We built the grammar for the natural language bit by bit. After creating an initial version, we continued testing the system and writing grammar rules for new sentences to include. In fact, we currently have about 189 English training sentences and 245 Chinese training sentences. The process was somewhat challenging because rules for new sentences would sometimes break old ones, requiring us to come up with inventive ways to incorporate both old and new sentences.

The actual testing of the complete system was bound by time constraints and we were only able to test with one user. Another setback was that because of the removed dynamic capability of the bilingual recognizer, the system had to be able to recognize all 113 streets and 68 landmarks. As a result, the recognizer is not very robust. For example, one time the system heard "Boston" when the user did not say anything resembling that. Thus, we decided that, because the recognizer was still premature, for evaluation purposes we would only test typed input. Spoken input is extremely important as well, but it will have to wait until a version of the multilingual recognizer is created that supports dynamic vocabulary.

Our test involved a native Chinese speaker. Because of her strong ability in Chinese, she attempted most of her sentences in Chinese (typed pinyin). This proved to be somewhat of a setback because some phrases have variations in Chinese that we had not accounted for. That is, if she had communicated in English, there would have been a higher probability of a successful parse, during which the system would respond with the correct Chinese translation that she could then type to the system. However, because she often wrote in Chinese directly, a lot of the sentences failed to parse properly. In fact, we learned that our system does not adequately support slang and shortcuts. For example, our tester informed us that in a real-life situation, she would never say "zhi2 zou3 zhi2 dao4 gao1 jie1" (go straight until you reach High Street), a sentence which is supported by the LanguageLand system. Instead, she would probably say something like "gao1 jie1 zhi2 zou3," which is not quite grammatically correct but makes sense to native speakers. A tradeoff we made with our system was to avoid supporting informal language, understanding that it would limit the robustness of the system but would work better with our grammar structure and would allow students to learn proper Chinese. But in dealing with proper sentences, our system still allows some variability. For example, the sentence “zhi2 zou3 dao4 gao1 jie1” works, even though the “zhi2” in front of “dao4” is stripped out. While this sentence is comparatively not as grammatically complete as the initial version, “zhi2 zou3 zhi2 dao4 gao1 jie1,” our system still supports it. However, we choose not to support sentences which have completely incorrect grammar structures, a decision which may change in a future version of the system. Overall, the evaluation helped us understand the limits of the system and why it failed for particular sentences. Fortunately, we were able to incorporate many of the suggestions raised by the tester. For example, we learned that “cha1 kou3” is another way of saying “corner,” and “zuo3 guai3” is another way of saying “turn left.”

Due to time constraints, we were unable to subject the system to the series of data collection/system expansion cycles that will be necessary before it can actually be fielded in the classroom.

Chapter 7

Conclusion

This chapter summarizes the thesis and discusses future work covering both natural language and application development.

7.1 Summary

The goal of this thesis is to develop an intelligent, multimodal, and bilingual language learning system in the map game domain which allows English-speaking students to learn Mandarin Chinese. We have accomplished this goal in three phases. The first phase was building the application user interface with a stand-in artificial intelligence component. The second phase was creating and augmenting grammar rules and catalogs using SLS's extensive set of natural language components. The third phase was replacing the limited AI with the natural language components and providing an appropriate communication structure. All three phases were based on an iterative approach in which we repeatedly gathered feedback and suggestions and rolled them into the next version of the system. Overall, we feel that we have accomplished the goals we set out for ourselves in the beginning, although a lot of new work remains for the future.

7.2 Future Work

7.2.1 Extending Application Development

In the process of building this first version of the LanguageLand application, we came across a variety of different ideas for extending the application in the future. For example, it is currently limited in its scope of commands. While the user can ask the system to erase a particular map block or even all the blocks on the map, there is no way to selectively erase items. A user cannot ask the system to erase all mice on the map, or to erase a particular house which is on the corner of two specified streets. The system can deal with commands such as turning right at or after a specific landmark or street; but, the system cannot deal with more general objects such as a "corner" or a "building." For example, a user cannot ask to turn left at the "next corner." It is our hopes that future development can include expanding the system command grasp.

Another suggestion is to make the map bigger so that street corners are not at most two blocks from each other horizontally and vertically. If it will not fit on the screen, scrollbars could be implemented to allow the user to navigate the map. A bigger map would allow the user to form more elaborate sentences, perhaps involving conjunctions or traveling multiple blocks, without reaching the goal too soon.

A problem with streets right now is that the vertical streets have to have shorter names because otherwise their names are blocked from view. We have temporarily solved the problem by naming those streets things like "1st Ave," but a new graphical user interface replacing the street interface could remove this requirement.

While the application currently takes English and Chinese pinyin as typed input, it cannot accept actual Chinese characters. Due to time constraints, we did not explore whether

Java applications can support Chinese characters such as Big 5, but hopefully it can be added in the future.

Some more high-level suggestions include things like providing simulated conversations as examples to follow and allowing users to record, listen to, and get ratings on their own spoken sentences, similar to what SLLS provides. This may allow students to compare their own voices to those of the system, thus having a better idea of how good their pronunciations are.

At present, the LanguageLand system is limited to the map game domain. However, we see potential for eventually extending it to other real-life situations such as buying items from a store. In that scenario, setting up the store shelves with products might be the edit mode while purchasing items would be the actual play mode, the task being to buy a specific number of different things.

It would be nice to make it easier for teachers and administrators to add new vocabulary. Currently, landmark pictures are hand-picked and are all resized to fit within the buttons. Moreover, each picture has to be labeled with the actual vocabulary name because that is how the application extracts the picture. The name of the picture also has to be added to a global list of landmarks. The same goes for new streets, although it is a little easier since streets do not have corresponding pictures. If we can automate this process and minimize the number of steps it takes to build new vocabulary, the system would be easier to maintain and expand.

Finally, a nice feature would be to allow teachers to choose and modify different sets of vocabulary. While vocabulary can still be chosen randomly, there may be some which are more fitting for a particular lesson plan.

7.2.2 Extending Natural Language Development

As for the natural language components, the most important feature we need is for Summit to allow both a bilingual recognizer as well as dynamic vocabulary. Once that feature is enabled and our system can handle both things, the recognizer should be robust enough for evaluation and usage.

The second improvement, just like for the application development, is expanding the domain and writing grammar to support a lot more sentences. This work is somewhat coupled with extending the application's capability to deal with more complicated sentences, because we have to come up with a common interface for the key-value bindings.

Aside from adding new sentences, we would also like to figure out a reliable way to fix misparsed sentences. For some of our previously misparsed sentences, we had tried but later gave up implementing markers to explicitly end adverbial clauses. We ended up changing the structures of the parse trees instead. It would certainly be ideal to figure out why those markers worked for some but not all sentences, and to get it to work in case we run into similar misparses in the future.

As mentioned in the section on application improvements, the system does not support Chinese characters yet. The same goes for the natural language components. We currently have rules for pinyin but not Big 5 Chinese. This feature should not be hard to implement, once we know for sure that the application can support typed input in Big 5.

Furthermore, we'd like to make the system bilingual in all senses of the word. Not only should it let English-speaking students learn Mandarin Chinese, but it should also let Mandarin speakers learn English. We already have rules that retrieve key-value bindings from English sentences and also perform English-to-English paraphrasing, although they may need some minor modifications. The only major missing piece is the grammar rules which translate Chinese phrases back into English. Once we have that, the system should be fully reversible between English and Chinese.

Finally, if we are ambitious enough to add other domains to the LanguageLand system, such as buying items in a store, we would have to create new grammar from the ground up, just as we did for the map game domain.

Bibliography

[1] L. Baptist, "GENESIS-II: A Language Generation Module for Conversational Systems," SM. Thesis, MIT, 2000.

[2] L. Baptist and S. Seneff, "GENESIS-II: A Versatile System for Language Generation in Conversational System Applications," in Proc. ICSLP, Beijing, China, 2000.

[3] G. Chung, S. Seneff, C. Wang and I. Hetherington, "A Dynamic Vocabulary Spoken Dialogue Interface," submitted to Proc. ICSLP, Jeju Island, Korea, October, 2004.

[4] S. Cyphers, *"Communicator Tutorial,"* Spoken Language Systems, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, October 30, 2003.

[5] Instinct Corporation with Stanford University researchers, Language Adventure CD-ROM, http://www.magictheatre.com/la-products.htm, 2003.

[6] Google, Google search for spoken computer-aided language learning systems, http://www.google.com/, search terms: computer-aided, language, learning.

[7] T. J. Hazen, I. L. Hetherington, and A. Park, "FST-Based Recognition Techniques for Multi-Lingual and Multi-Domain Spontaneous Speech," in Proc. EUROSPEECH, pp. 1591—1594, 2001.

[8] L. Johnson, The DARPA Tactical Language Training Project, http://www.isi.edu/isd/carte/proj_tactlang/tactical_lang_overview.pdf.

[9] J. Lai, V. Lee, “LanguageLand: A Foreign Language Teaching Application,” MIT 6.893 Final Project Report, 2003.

[10] T. J. Lau, “SLLS: An Online Conversational Spoken Language Learning System,” M.Eng. thesis, MIT Department of Electrical Engineering and Computer Science, May 2003.

[11] J. Lee, “Translingual Understanding Grammar Induction,” MIT Computer Science and Artificial Intelligence Laboratory, 2004.

[12] R. Molich and J. Nielsen (1990), Ed. J. Nielsen (1994), “Ten Usability Heuristics.”

[13] W. W. Royce, “Managing the Development of Large Software Systems,” IEEE, 1970.

[14] S. Seneff, “TINA: A Natural Language System for Spoken Language Applications,” in *Computational Linguistics*, vol. 18, no. 1, pp. 61--86, 1992.

[15] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, “Galaxy-II: A Reference Architecture for Conversational System Development,” in Proc. ICSLP, Sydney, Australia, November 1998.

[16] The University of Maryland University College, Japanese: The Spoken Language, Multimedia Collection, http://www.learner.org/resources/series149.html, 1998.

[17] C. Wang, S. Cyphers, X. Mou, J. Polifroni, S. Seneff, J. Yi and V. Zue, "Muxing: A Telephone-Access Mandarin Conversational System," in Proc. 6th International Conference on Spoken Language Processing, Beijing, China October 2000.

[18] J. Yi and J. Glass, "Natural-Sounding Speech Synthesis Using Variable-Length Units," in Proc. ICSLP, Sydney, Australia, November 1998.

[19] J. Yi, J. Glass and L. Hetherington, "A Flexible, Scalable Finite-State Transducer Architecture for Corpus-Based Concatenative Speech Synthesis," in Proc. ICSLP, Beijing, China, October 2000.

[20] V. Zue, J. Glass, D. Goodine, M. Phillips, and S. Seneff, “The SUMMIT Speech Recognition System: Phonological Modelling and Lexical Access,” in Proc. ICASSP, Albuquerque, NM, April, 1990.