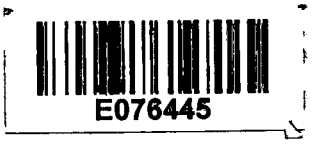
SPEECH COMMAND GAME ON ANDROID



EAKAWAT TANTAMJARIK

PHASUPONG ASAVAVICHIENJINDA

A SPECIAL PROJECT SUMMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2011

Thesis Title	Speech Command Game on Android
Student	Mr.Eakawat Tantamjarik
	Mr.Phasupong Asavavichienjinda
Degree	Bachelor of Science
Program	Computer Science
Year	2011
Thesis Advisor	Asst.Prof.Dr.Korakot Prachumrak

ABSTRACT

Generally, game applications on the mobile devices can only use touch screen to control everything, sometimes it is very difficult to touch on the right place where you want, if you have small screen. Our idea is to create a pet game which focuses on alternative method of interaction with the game's input to increase the efficiency in mobile game. The objective of this special project is to develop a game with speech recognition system by using CMU Sphinx and Eclipse as tools to develop. Android OS has been chosen to develop this project and to provide capability to recognize the speech and convert the input audio into text. It also enables players to perform commands such as "sleep, exercise, food" to a pet by providing voice input. Although, it is the alternative way of pet games, players will meet the new experience of it.

ACKNOWLEDGEMENT

This thesis would not have been possible unless we got help from many people. We would like to express our gratitude to our supervisor, Asst.Prof.Dr.Korakot Prachumrak who always offers invaluable assistance, support, and guidance during the period of our project. We also wish to express our gratitude to Assoc.Prof.Dr.Jeeraporn Werapun and Dr.Rungrat Wiangsripanawan for being committee. Special thanks to all our friends for will power and for sharing the literatures and invaluable knowledge.

Eakawat Tantamjarik

Phasupong Asavavichienjinda

Table of Contents

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

Table of Contents (cont.)

List of Figures

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

List of Figures (cont.)

Chapter 1

Introduction

1.1 Rational and Research Motivation

Generally, game applications on the mobile devices can only use touch screens to control everything, sometimes it is very difficult to touch on the right place that you want, if you have small screen. Our idea is to create game on Android with mobile platform constraint with multimodal interaction such as speech commanding other than only touching the screen. The purpose of this project is to optimize game interface to support both touch screens and spoken interaction together without switching mode.

Eclipse and Pocketsphinx toolkits are the tools that we use for the development of this special project. Eclipse is a tool for developing an application on Andriod operating system by using JAVA programming language and use Pocketsphinx toolkit to develop speech recognition system.

1.2 Objective

The objective of this special project is to play game on mobile device by the touch screens and speech commanding. This special project provides more efficient way to play game and more comfortable for the users to play by their voice.

1.3 Scope

1) This game is able to use speech to control game application
2) Developing friendly user interface
3) Produce module that can translate speech input into text.
4) Develop game which connects with build-in microphone.

1.4 Organization

This special project consists of 5 chapters.

Chapter 1: Introduction

Chapter 2: Background of the special project

Chapter 3: Program work

Chapter 4: Experiment of this special project

Chapter 5: Conclusion of this special project

1.5 Stage of the special project

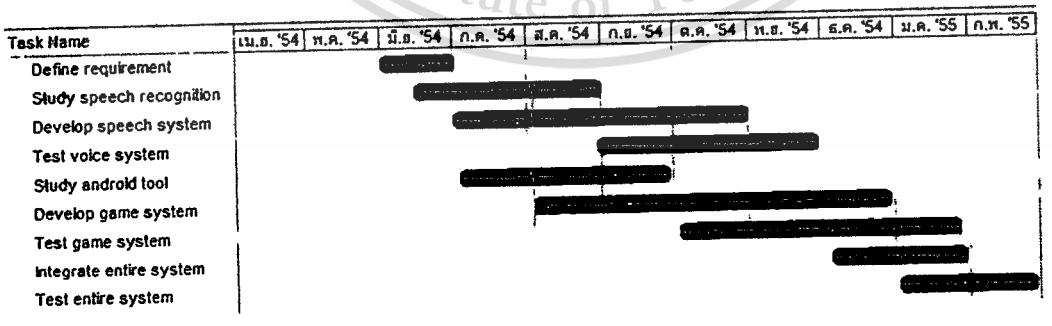


Figure 1.1 Schedule of this special project.

Chapter 2

Background

2.1 What is an Android?

Android is the fastest growing mobile operating system (OS).Together with over 30 smartphones introduced in the last year and over 10,000 applications being added in Android market every month, Android structure have high flexibility enough to install on any devices hardware, including the mobile phone.

Netbooks have always been a natural platform to adopt Android, but the flexibility of Android operating system has lead its growth further into televisions and even in mobiles. Many of the world's largest corporations ensure a presence of Android by creating and providing many compatible services. Android developers have many opportunities, and relevant apps reaching more people than ever before, these increase the satisfaction of creating a relevant app.

Figure 2.1 Logo of Android Operating System.

2.2 Overview of Android

Android operating system (OS) has come a long way since the first announcement of the Open Handset Alliance in late 2007. The idea of an open source OS for embedded systems was not new at that time, but Google was strongly supporting it and definitely help pushing Android to the front world in just few years.

Many wireless carriers in multiple countries across various communication protocols have one or more Android phones available. Other embedded devices, such as tablets, net-books, televisions, set-top boxes, and even automobiles, have also adopted the Android OS.

2.3 The Evolution of Android

Google saw a large growth of Internet usage and searching in mobile devices then acquired Android, Inc., in 2005 to focus its development on a mobile device platform. After that Apple introduced the iPhone in 2007 with some innovative ideas including multitouch and an open market for applications. That makes Android was quickly adapted to include these features and offered definite distinctions such as more control for developers and multitasking. In addition, Android incorporates enterprise requirements such as exchange support, remote wipe, and Virtual Private Network (VPN) support, to go after the enterprise market that Research In Motion (RIM) has developed and held very well with its Blackberry mobile phones.

Device diversity and quick adaptation both have helped Android grows its user base, but it comes with potential challenges for developers. Applications need to support multiple screen sizes, resolution ratios, keyboards, hardware sensors, OS versions, wireless data rates, and system

configurations. Each of those can lead to different and unpredictable behavior, but testing applications to all environments is an impossible task.

Therefore, Android has been constructed some abilities to ensure as uniform an experience across platforms as possible. By abstracting the hardware differences, Android OS tries to prevent applications crash from device-specific modifications while providing the flexibility to adjust aspects as users' need. Future-proofing of applications to the introduction of new hardware platforms and OS updates are also a consideration. This mostly works as long as the developer is well aware of this systematic approach.

Nowadays, as with any embedded platform, wide testing of applications is required. Google provides assistance to third-party developers in many forms as Android Development Tool (ADT) plug-ins for Eclipse (also known as standalone tools) including real-time logging capabilities, a realistic emulator that runs native ARM code, and in-field error reports from users to developers of Android Market applications.

2.4 The Characteristic of Android

Android has some interesting characteristics. Android is an embedded OS that came from the Linux kernel for core system services, but it is not embedded Linux. For example, standard Linux utilities such as X-windows and GNU C libraries are not supported. Writing applications for Android makes use of the Java framework, but it is not Java. Standard Java libraries such as Swing are not supported. Other libraries such as Timer are not implemented; they have been replaced by Android's own libraries, which are optimized for usage in a resource constraint and embedded environment.

The Android OS is open source, which means developers can view and use any of the system source code. This source code is one of the first resources for seeing examples of Android code and it helps easy to the usage when documentation is lacking. This also means developers can utilize the system in the same way as any core application and can swap out some system components for their own components. However, Android devices contain some proprietary software that cannot access to developers (such as Global Positioning System (GPS) navigation).

A final characteristic of Android OS is that Google is also supporting Chrome OS. Android OS is built for embedded platforms, and Chrome OS is built for cloud-based platforms. However, which is the best choice for embedded devices that live in the cloud? The answer is Netbooks which fill the gap between smart phones and laptop computers and could likely go either way. Android has utilized the cloud more than previously and Google also supports a web-based market, so Chrome OS enjoys the same developer leverage that Android currently has.

2.5 Features of Android

2.5.1 Multiprocess and App Widgets

The Android OS does not restrict the processor to a single application at a time. The system manages priorities of applications and threads within a single application. This is the benefit that background tasks can be run while a user is interacting with the device in a foreground process. For example, while a user is playing a game, a background process can check stock market prices and trigger an alert if necessary.

App Widgets are mini applications that can be embedded in other applications such as the Home screen. They can also process events such as starting a music stream or updating the outside temperature while other applications are still running.

Figure 2.2 Example of widget on Android phone.

The expert user will gain benefit from multi-processing. However, user should aware of device's battery that is being consumed by multi-processing because it will run out very fast.

2.5.2 Touch, Gestures, and Multitouch

The touchscreen is an intuitive user interface for a hand-held device. After a finger touches the screen, drags and flings are natural ways to interact with graphics. Multi-touch technology provides a way to track more than one finger at the same time. It is often used for zooming or rotating a view.

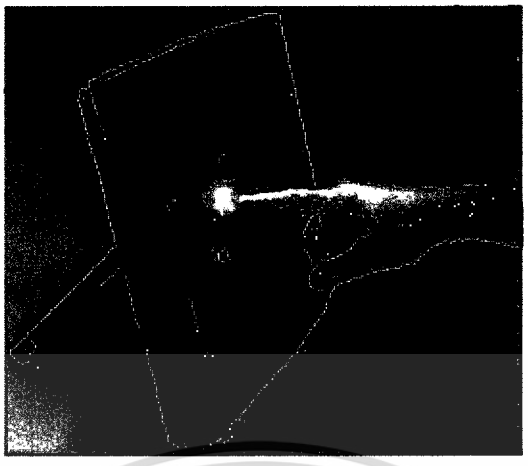


Figure 2.3 Touch Screen Device.

Some touch events are available transparently to the developer without the need to implement their detailed behaviors. Custom gestures can be defined as needed. It is important to try to maintain a consistent usage of touch events compared to other applications.

2.5.3 Hard and Soft Keyboards

One feature on a pocket device that excites users is whether it has a physical (also called hard) keyboard or software (also called soft) keyboard. The tactile feedback and definite placement of keys are provided by a hard keyboard tends to make typing much faster for some people, whereas other people prefer the modern design and convenience offered by a soft keyboard. With the large variety of Android devices available, either type can be found. A side effect for developers is the need to support both. One downside of a soft keyboard is a portion of the screen needs to be dedicated to the input. This needs to be considered and tested for any user interface (UI) layout.

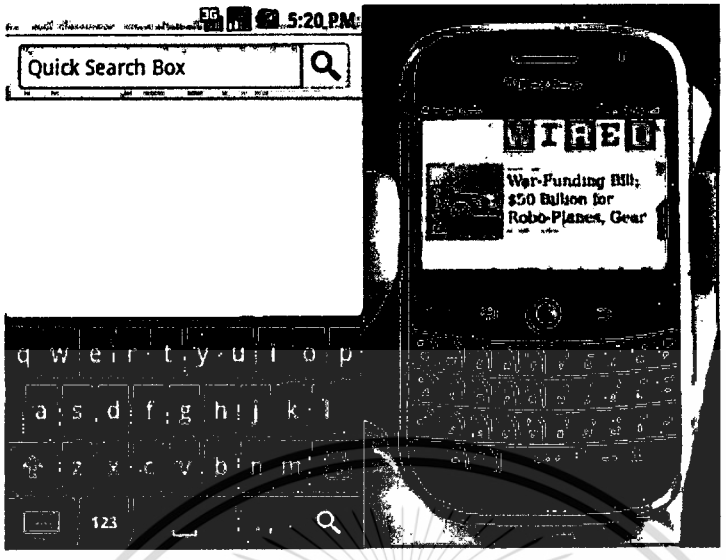


Figure 2.4 Soft Keyboard (Left) and Hard Keyboard (Right).

2.6 Basic Knowledge in Developing Android Application

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android Software Development Kit (Android SDK) provides the tools and Application Programming Interfaces(APIs) necessarily to begin developing applications on the Android platform using the Java programming language.

2.6.1 Android Software Development Kit

The Android software development kit (SDK) includes everything you need to start developing, testing, and debugging Android applications. The elements that include in the Android SDK are:

- **The Android APIs**: The core of the SDK is the Android API libraries that provide accessibility to the Android stack. These are the same libraries used at Google to create native Android applications.

- **Development Tools**: To turn Android source code into executable Android applications, the SDK includes several development tools that let you compile and debug your applications.
- **The Android Emulator**: The Android Emulator is a fully interactive Android device emulator includes many features with several alternative skins. Using the emulator, you can see how your applications will look and behave on real Android devices.
- **Full Documentation:** The SDK includes wide code-level reference information detailing exactly what is included in each package and class and how to use them. In addition to the code documentation, Android's reference documentation explains how to get started and gives some detailed explanations of the fundamentals behind Android development.
- **Sample Code:** The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available using Android as well as simple programs that highlight how to use individual API features.

Therefore, Android released a special plug-in that simplifies project creation and comfortable to integrate Eclipse with the Android Emulator and debugging tools.

2.6.2 Android Software Stack

The Android Software Stack is composed of the elements as shown in figure 2.5.

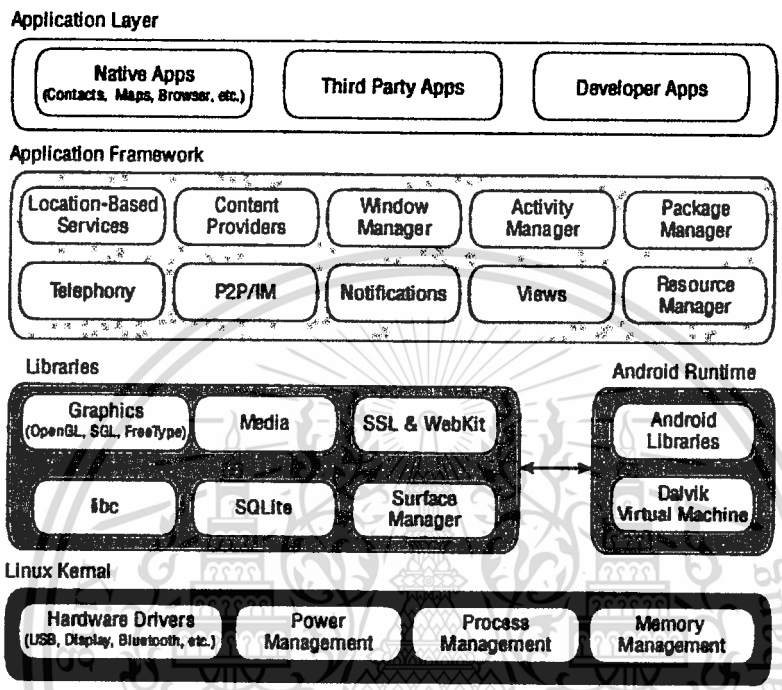


Figure 2.5 The elements in Android Software Stack

Linux Kernel - This is the core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.

Libraries - Running on top of the kernel, Android includes various C/C++ core libraries such as libc and Secure Sockets Layer (SSL), as well as:

- A media library for playback of audio and video media.
- A Surface Manager to provide display management.
- Graphics libraries that include SGL and OpenGL for 2D and 3D graphics.

- SQLite for native database support.
- SSL and WebKit for implementing web browser and Internet security.

Android Run Time – This makes an Android phone become more than a mobile Linux implementation. Including the core libraries and the Dalvik virtual machine, the Android run time is the engine that powers up your applications and forms the basis for the application framework.

- **Core Libraries**: While Android development is done in Java, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries as well as the Android specific libraries.
- **Dalvik Virtual Machine:** Dalvik is a register based virtual machine that has been optimized to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.

Application Framework - The application framework provides the classes to create Android applications. It also provides a generic abstraction for accessing and managing the user interfaces and application resources.

Application Layer - All applications both native and third party are built on the application layer using the same API libraries. The application layer runs within the Android runtime using the classes and services made available from the application framework.

2.6.3 Dalvik Virtual Machine

Dalvik is the process virtual machine (VM) in Google's Android operating system. It is the software that runs the apps on Android phones. Dalvik is an integral part of Android, which is typically used on mobile devices such as mobile phones, tablet,

computers and netbooks. Programs are commonly written in a dialect of Java and compiled to bytecode. Then they are converted from Java Virtual Machine-compatible .class files to Dalvik-compatible .dex (Dalvik Executable) files before installed on a device. The compact Dalvik Executable format is designed to be suitable for systems that are constrained in terms of memory and processor speed.

Dalvik, like the rest of Android up to version 2.3, is open-source software. It was originally written by Dan Bornstein, who named it after the fishing village of Dalvík in Eyjafjörður, Iceland, where some of his ancestors lived.

One of the key elements of Android is the Dalvik virtual machine. Rather than using a traditional Java virtual machine (VM) such as Java ME (Java Mobile Edition), Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device.

The Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, processing, and memory management. It's also possible to write C/C++ applications that run directly on the underlying Linux OS.

All Android hardware and system services are managed to use Dalvik as a middle tier. By using a VM to host application execution, developers have an abstraction layer that ensures they do not have to worry about a particular hardware implementation.

The Dalvik VM executes Dalvik executable files, a format optimized to ensure minimal memory usage. The .dex executables are created by transforming Java language compiled classes using the tools provided within the SDK.

2.6.4 Android Application Architecture

Android's architecture introduces the concept of reuse component that allows developers to publish and share activities, services, and data with other applications with access managed by the security restrictions that were put in place.

The same mechanism that lets developers produce a replacement contact manager or phone dialer can let you expose your application components to let other developers create new UI front ends and functionality extensions, or otherwise build on them.

The following application services are the architectural foundations of all Android applications, providing the framework:

Activity Manager - Controls the life cycle of your activities, including management of the activity stack.

Views - Are used for constructing the user interfaces for your activities.

Notification Manager - Provides a consistent and non-intrusive mechanism for signaling your users.

Content Providers - Lets your applications share data between applications.

Resource Manager - Supports non-code resources like strings and graphics from external resources.

2.6.5 Basic Android Libraries

Android offers a number of APIs for developing applications. The following list of core APIs should provide an example of what is available. All Android devices will offer support for at least these APIs:

android.util - The core utility package contains low-level classes like specialized containers, string formatters, and XML parsing utilities.

android.os - The operating system package provides access to basic operating system services like message passing, interprocess communication, clock functions, and debugging.

android.graphics - The graphics API supplies the low-level graphics classes that support canvases, colors, and drawing primitives, and lets you draw on canvases.

android.text - The text processing tools for displaying and parsing text.

android.database - Supplies the low-level classes required for handling cursors when working with databases.

android.content - The content API is used for managing data and publishing by providing services for dealing with resources, content providers, and packages.

android.view - View is the core user interface class. All user interface elements are constructed using series of Views to provide the user interaction components.

android.widget - Built on the View package, the widget classes are the "here is one we created earlier" user interface elements for user to use in their applications. They include lists, buttons, and layouts.

com.google.android.maps - A high-level API that provides access to native map controls that can be used within the application. Includes the MapView control as well as the Overlay and MapController classes are used for annotating and controlling embedded maps.

android.app - A high-level package that provides access to the application model. The application package includes the Activity and Service APIs that form the basis for all Android applications.

android.provider - To ease developer access to certain standard Content Providers such as the contacts database, the Provider package offers classes to provide access to standard database and all Android distributions.

android.telephony - The telephony APIs give you the ability to directly interact with the device's phone stack. This library is letting developers make, receive, and monitor phone calls, phone status, and SMS messages.

android.webkit - The WebKit package features APIs for working with Web-based content, including a WebView control for embedding browsers in every activities and a cookie manager.

In addition to the Android APIs, the Android stack includes a set of C/C++ libraries that are exposed through the application framework. These libraries include:

OpenGL - The library for supporting 3D graphics based on the Open GL ES 1.0 API.

FreeType - Support for bitmap and vector font rendering.

SGL - The core library are used for providing a 2D graphics engine.

libc - The standard C library optimized for Linux-based embedded devices.

SQLite - The lightweight relation database engine is for storing application data.

SSL - Support for using the Secure Sockets Layer cryptographic protocol is for secure Internet communications.

android.location - The location-based services API gives applications accessibility to the device's current physical location. Location-based services provide generic access to location information using whatever position-fixing hardware or technology is available on the device.

android.media - The media APIs provide to support for playback and recording of audio and video media files, including streamed media.

android.opengl - Android offers a powerful 3D rendering engine using the OpenGL ES API that let developer to create dynamic 3D user interfaces for their applications.

android.hardware - Where available, the hardware API exposes sensor hardware including the camera, accelerometer, and compass sensors.

android.bluetooth, android.net.wifi , and android.telephony - Android also provides low-level access to the hardware platform including Bluetooth, Wi-Fi, and telephony hardware.

2.7 Android and Speech Recognition

In Android Libraries, for Android SDK newer version since Android 2.1, Google has introduced with its own speech recognition that is pre-installed on many Android devices with using Google's Voice Search Application. But, because to recognize every words need very huge data storage that is not suitable for mobile devices. So Google solve this problem by using computational and data storage on the cloud to give this service. However, to use this service needs a fast Internet connection that is the big problem right now.

2.8 Speech Recognition

Speech recognition (also known as automatic speech recognition or computer speech recognition) is a task to convert spoken words into text.

The term of "voice recognition" refers to recognition systems that must be trained to a specific speaker as in the case for most desktop recognition software. However, "speech recognition" is a solution that refers to technology that can recognize speech without being targeted at single speaker means that speech recognition software can recognize speech from other speakers and the accuracy to recognize speech doesn't drop down.

2.8.1 History

The first speech recognizer appeared in 1952 and consisted of a device for the recognition of single spoken digits. Another early device was the IBM Shoebox, exhibited at the 1964 New York World's Fair.

One of the most well-known histories for the commercial application of speech recognition is in the United States. This application has been in health care and in particular the work of the medical transcriptionist (MT) which is an allied health profession deals in the process of transcription. This application task is to convert voice-

recorded reports to text documents by physicians and/or other healthcare professionals, into text format. According to industry experts speech recognition (SR) was sold as a way to completely eliminate transcription rather than make the transcription process more efficient because the accuracy at that time is very low and was not accepted. It was also the case that SR at that time was incomplete. Additionally, to be used effectively, it required changing in the ways of physicians worked and documented the data, which most of them were unwilling to do. However, the biggest limitation to speech recognition automating transcription is the nature of dictation. This nature needs highly interpretive and often requires precision judgment that may be provided by a real human but not yet by an automated system at that time. Another limitation is the time that required by the user and/or system provider to train the software because ever in human it needs time approximately 2 years to remember the sound and understand the meaning. So in computer, 2 years training gives unacceptable accuracy.

2.8.2 Speech Recognition Application Domains

A distinction in automatic speech recognition (ASR) is often made between "artificial syntax systems", which are usually domain-specific, and "natural language processing", which is usually language-specific. Each of these types of application presents its own particular goals and challenges.

2.8.2.1 Health Care

In the health care domain, even in the beginning of speech recognition technologies, medical transcriptionists (MTs) have not yet become out of date. The services are provided may be need to redistributed rather than replaced. Speech recognition can be implemented in both front-end and back-end of the medical documentation process.

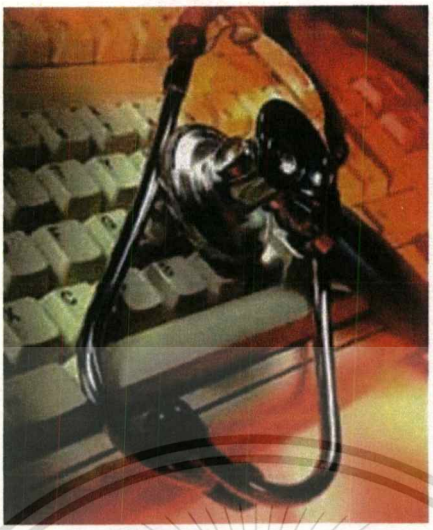


Figure 2.6 Medical Transcription

For example, front-end provides dictate into speech recognition engine that display the recognized word after they are spoken and dictator is responsible for editing and signing off on the document.

2.8.2.2 Military

They have implemented speech recognition in many vehicles and military base because it helps multitasking become easier at the same time.

For example, in aircraft, it needs 2 people to control; one controls the aircraft and the other one sets functions. However, if speech recognition was implemented to the aircraft, it would not need to have two pilots because it can be set by using a pilot speech.

Figure 2.7 U.S. fighter F-16 VISTA aircraft.

2.9 Basic concepts of speech

Speech is a complex phenomenon. It is easy to use but very hard to understand how it is produced. Normally, people only know that speech was built with words, and each word consists of phones but the reality is very different. Speech is a dynamic process that each part is unclearly distinguished look like figure 2.8.

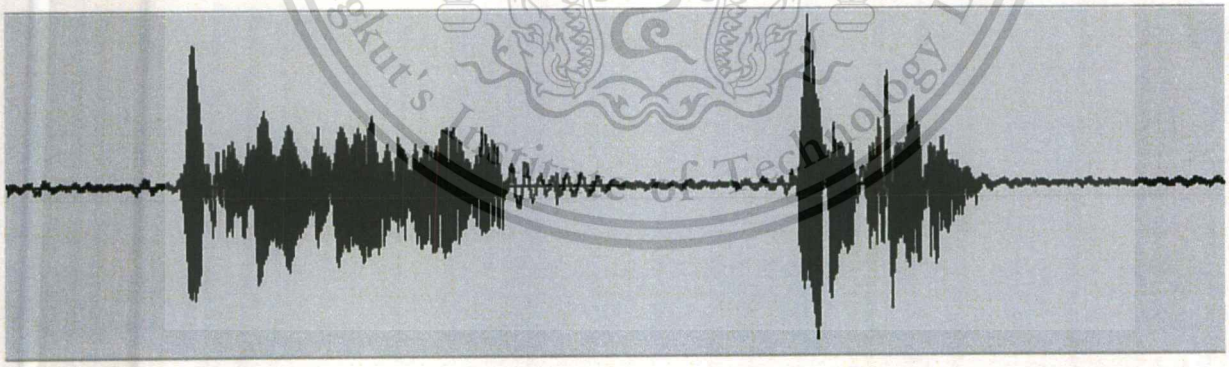


Figure 2.8 Dynamic waves in speech.

In figure 2.8 you can't hypothesis that with the equal amplitude wave, it has got to be the same word because it has many contexts that need to be considered.

This reason makes all modern descriptions of speech using the mathematics like probabilistic to solve problems. That means in wave sounds there are no certain boundaries between each phone or each word. It makes speech recognition application are never 100% correct.

2.10 Speech Recognition Process

Speech Recognition basically means that talking to a computer and having it recognize what we are saying. This process working as a pipeline that converts PCM (Pulse Code Modulation) digital audio from a sound card into recognized speech. The elements in the pipeline are:

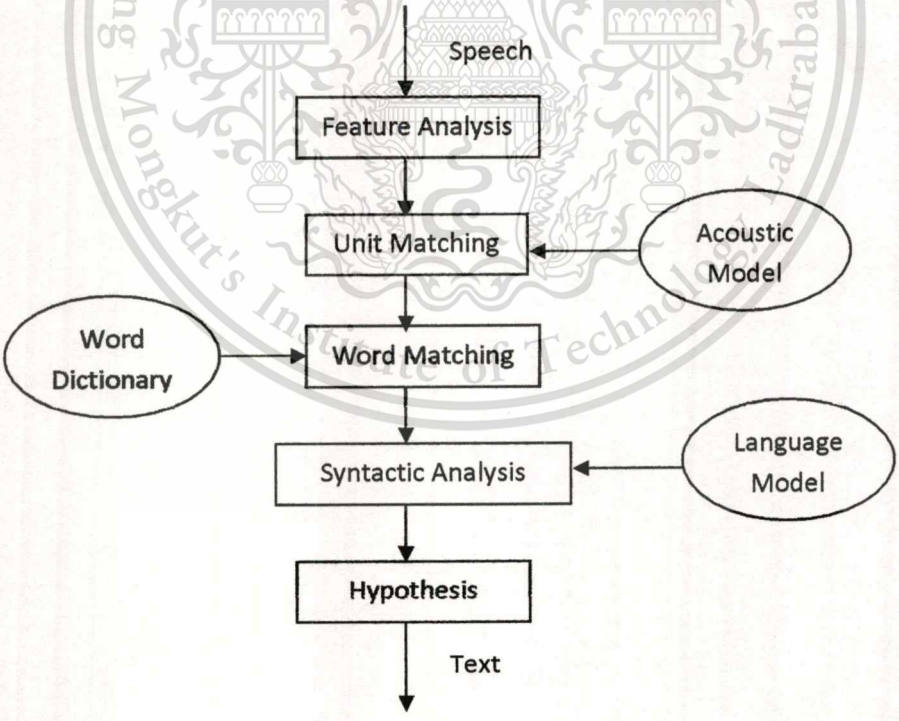


Figure 2.9 Block diagram of speech recognizer.

2.10.1 Feature analysis

The input of the speech recognizer is in the form of waves and amplitudes that have sample rate at about 16000 times per second depend on the devices which are used for recording the sound (e.g. normally 16000 but in mobile device usually record sound at sample rate 8000). But audio in this form cannot be used for speech recognizer. It needs to use Fast-Fourier transformations to produce graphs of frequency components that describe the sound heard in a millisecond and for every graph producing a vector, called the "feature vector".

2.10.2 Unit Matching System

This system provides likelihoods of matches for input speech by comparing it with inventory of speech recognition units of this speech recognizer. These units can be phones or syllables. Each unit is characterized by Hidden Markov Model (HMM) which parameters are estimated by training sets of speech data.

2.10.2.1 Hidden Markov Model

Modern general-purpose speech recognition systems are based on Hidden Markov Models (HMMs). These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time-scales (e.g., 10 milliseconds), speech can be approximated

as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes.

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

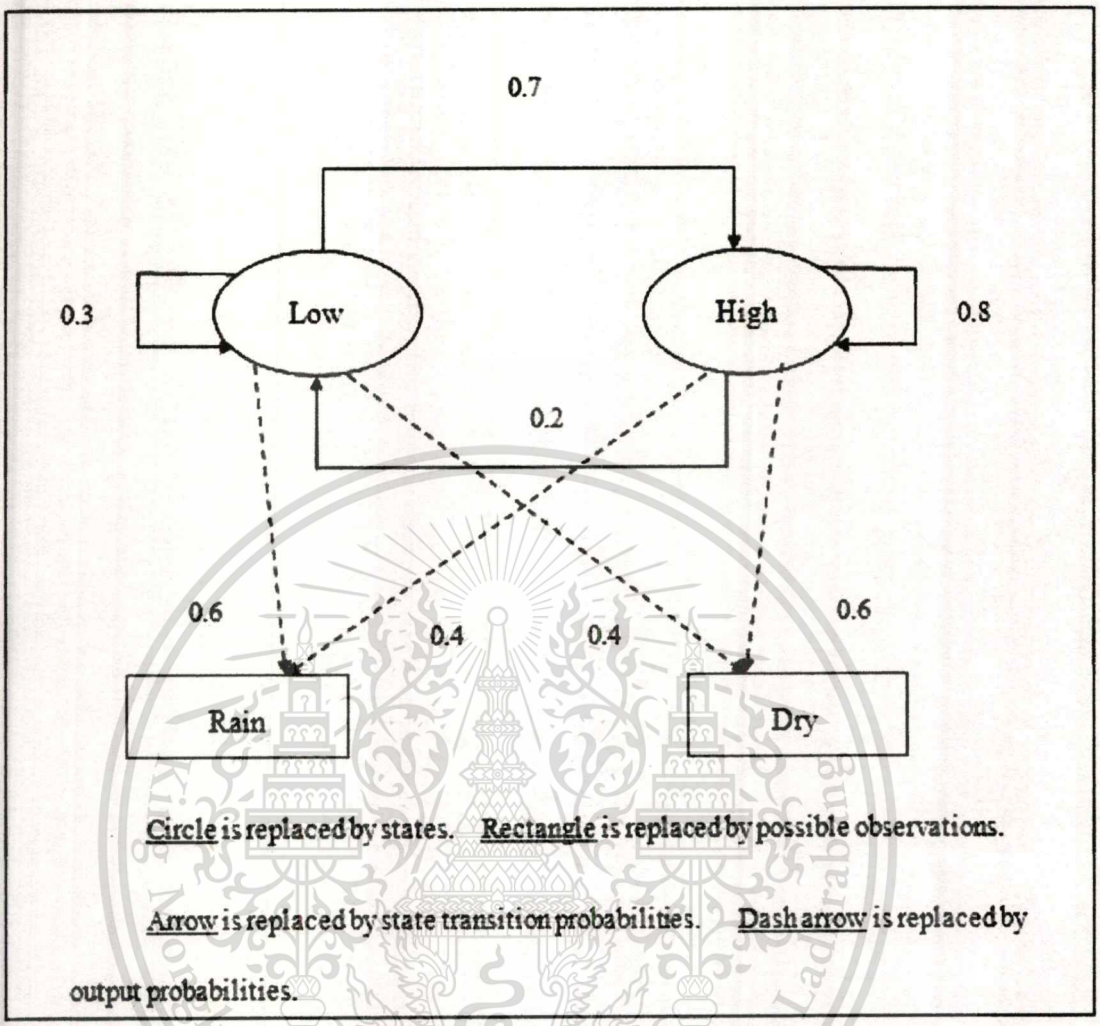


Figure 2.10 The example of Hidden Markov Model.

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

A possible improvement to decoding is to keep a set of good candidates instead of just keeping the best candidate, and to use a better scoring function (rescoring) to rate these good candidates so we may pick the best one according to this refined score. The set of candidates can be kept either as a list (the N-best list approach) or as a subset of the models (a lattice). Rescoring is usually done by trying to minimize the Bayes risk. Instead of taking the source sentence with maximal probability, we try to take the sentence that minimizes the expectancy of a given loss function with regards to all possible transcriptions (i.e., we take the sentence that minimizes the average distance to other possible sentences weighted by their estimated probability). The loss function is the Levenshtein distance, although it can be different distances for specific tasks. The set of possible transcriptions is pruned to maintain the tractability. Efficient algorithms have been devised to rescore lattices represented as weighted finite state transducers with edited distances represented themselves as a finite state transducer verifying certain assumptions.

2.10.3 Word Matching System

This function is used for constraining the unit matching system by letting them follow search paths that only appear in the dictionary.

2.10.4 Syntactic Analysis

Applying a grammar to the system is a task of this process. So the speech recognizer can know what phones or words are expected. This process is also placed

constraints on the search sequence of unit matching system. A grammar can be anything from context-free grammar to full-blown English.

2.10.5 Hypothesis

This task is used for figuring out which phonemes are spoken. This step is quiet hard because different words, different persons produce different sound. Also the background noises from microphone make the recognizer hear a different vector. Because of those problems a probability analysis is used in this process. A hypothesis is formed based on this analysis. A speech recognizer works by hypothesizing many different states at once. Each state contains a phoneme with a history of previous phonemes. The hypothesized state that has the highest score is used as the final recognition result.

2.11 Recognition Process

The common way to recognize speech follows by taking waveform then splits silent sound out of utterances and tries to recognize what is being said by those utterances. In order to do that it needs to take all possible combinations of words and tries to match those words with audio then choose the best matching combination. However, there are three important things to consider.

2.11.1 Concept of features.

Since the speech waves are large and have many parameters so they need to be optimized by calculating numbers of speech that were divided into frames. For extracting

39 numbers that represent the speech for each frames of length typically 10 milliseconds is called feature vector.

2.11.2 Concept of model.

Model shows some mathematical objects that include common attributes of the spoken words. According to speech there are three models used in speech recognition to do the match.

- Acoustic Model contains acoustic properties (most probably feature vectors) for each phone.

- Phonetic dictionary (or Pronunciation Model) contains a mapping from words to phones.

- Language Model (or Grammar Model) is used for restricted word search.

2.11.3 The matching process.

It needs a very huge time to compare all feature vectors with all models because the search is often used many methods to optimize the time such as creating word network.

2.12 CMU (Carnegie Mellon University) Sphinx Toolkit

CMU sphinx toolkit is one of the most popular speech recognition applications for Linux and it can correctly capture words. It also gives the developers the ability to build speech systems, interact with voices and build something unique and useful.

Sphinx originally started at CMU and has recently been released as an open source. This is a fairly large program that includes a lot of tools and information. It is still "in development", but includes trainers, recognizers, acoustic models, language models, and some limited documentation.

CMUSphinx toolkit is a leading speech recognition toolkit with various tools used to build speech applications. CMU Sphinx toolkit has a number of packages for different tasks and applications, here is the list:

- Pocketsphinx — lightweight recognizer library written in C.
- Sphinxbase — support library required by Pocketsphinx
- Sphinx4 — adjustable, modifiable recognizer written in Java
- CMUclmtk — language model tools
- Sphinxtrain — acoustic model training tools
- Sphinx3 — decoder for speech recognition research written in C.

In this special project, we used pocketsphinx because it supports for more portable devices.

2.13 Setting Up Sphinx Toolkit

2.13.1 Installing Cygwin

See Appendix A.

2.13.2 Introduction of tool in CMU Sphinx toolkit

PocketSphinx is a library that depends on another library called SphinxBase which provides common functionality across all CMUSphinx projects. To install Pocketsphinx, you need to install both Pocketsphinx and Sphinxbase.

Pocketsphinx is used for decoding the speech input and translate it into text by using language model and acoustic model generated from SphinxTrain and cmuclmtk.

Sphinxbase is the support libraries that used by Pocketsphinx. You need to install sphinxbase first in order to install pocketsphinx.

SphinxTrain is a tool that used for creating acoustic model.

cmuclmtk is a tool that used for creating language model.

2.13.3 Installing Sphinx toolkit

See Appendix B.

2.14 What is OpenGL ES?

OpenGL ES is an industry standard for (3D) graphics programming. It is especially targeted at mobile phone and embedded devices. It is maintained by the Khronos Group, which is a group of companies including ATI, NVIDIA, and Intel, who works together to define and extend the standard.

Speaking of the standards, there are currently three versions of OpenGL ES: 1.0, 1.1, and 2.0. All Android devices support OpenGL ES 1.0 and most also support 1.1, which adds some new features to the 1.0 specification. OpenGL ES 2.0, however, breaks compatibility with the 1.x versions. It can be used either 1.x or 2.0. The reason for this is that the 1.x versions use a programming model called fixed-function pipeline, while 2.0 provides programmatically define parts of the rendering pipeline via so-called shaders.

Many of the second-generation devices already support OpenGL ES 2.0; however, the Java bindings are currently not usable for OpenGL ES 2.0 (unless you target the new Android 2.3 version). OpenGL ES 1.x is more than good enough for most games.

OpenGL ES is an API that comes in the form of a set of C header files provided by the Khronos group, along with a very detailed specification of how the API defined in those headers. This includes things such as how pixels and lines have to be rendered. Hardware manufacturers then take this specification and implement it for their GPU on top of the GPU driver.

Chapter 3

Design

3.1 Game Design

3.1.1 Game Storyboard

First when player starts new game. Player needs to set the time for pet to sleep. Then an egg will appear on the screen, this state is called "Egg State". The player task in this state is to keep the temperature of egg balance by using cooler for decreasing the temperature or heater for increasing the temperature. Until several times pass, the pet will hatch and go to "Pet State". From then on, the player is given the task of raising the pet to good health throughout its life and attending to its needs, such as feeding it, playing games to make it happy, cleaning up its excrement, returning it to proper health with medicine if it gets sick, and says good night when it goes to bed. If the pet is not taken care well enough, it will soon result in the death. The game will end if the pet die such as sick, hungry for long time, or run out of stamina.

3.1.2 Character Design

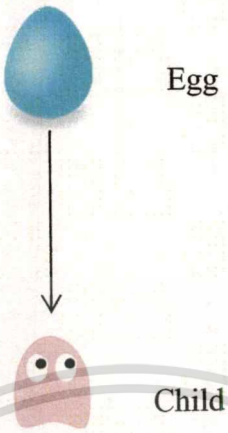


Figure 3.1 Pet Evolution.

3.2 Game Interface

3.2.1 Start Screen

Start screen has 3 buttons to select "New Game" for first time play, "Load" when you want to resume progress you have played and "Instruction" shows tutorial how to play this game.

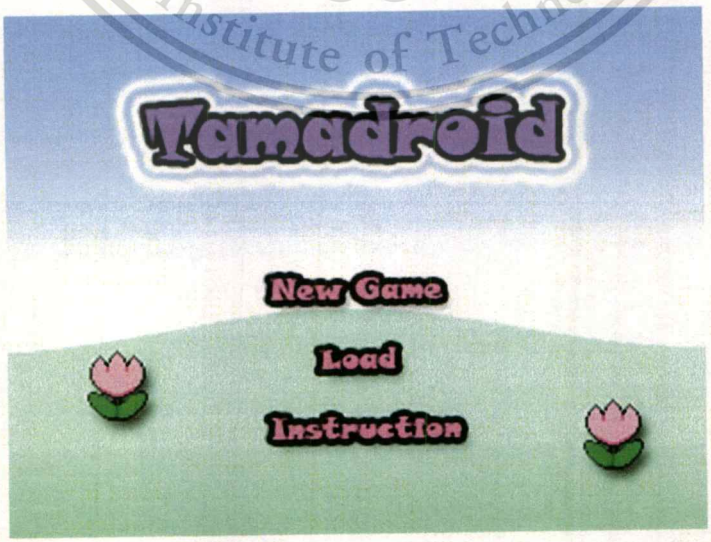


Figure 3.2 Main Manu.

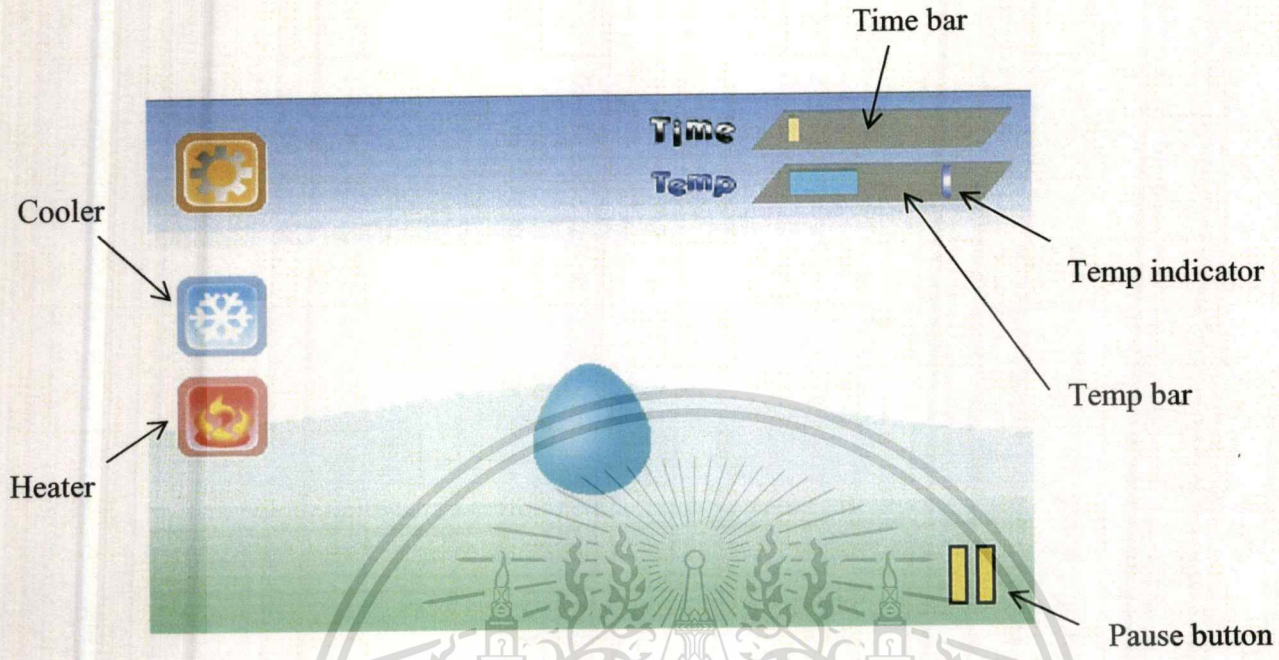


Figure 3.3 "Egg State" Interface.

- Cooler icon reduces temperature.
- Heater icon increases temperature.
- Time bar shows time when egg will be hatched.
- Temp indicator indicates that which temperature the egg will satisfy.
- Temp bar shows currently temperature.

3.2.3 Pet State Screen

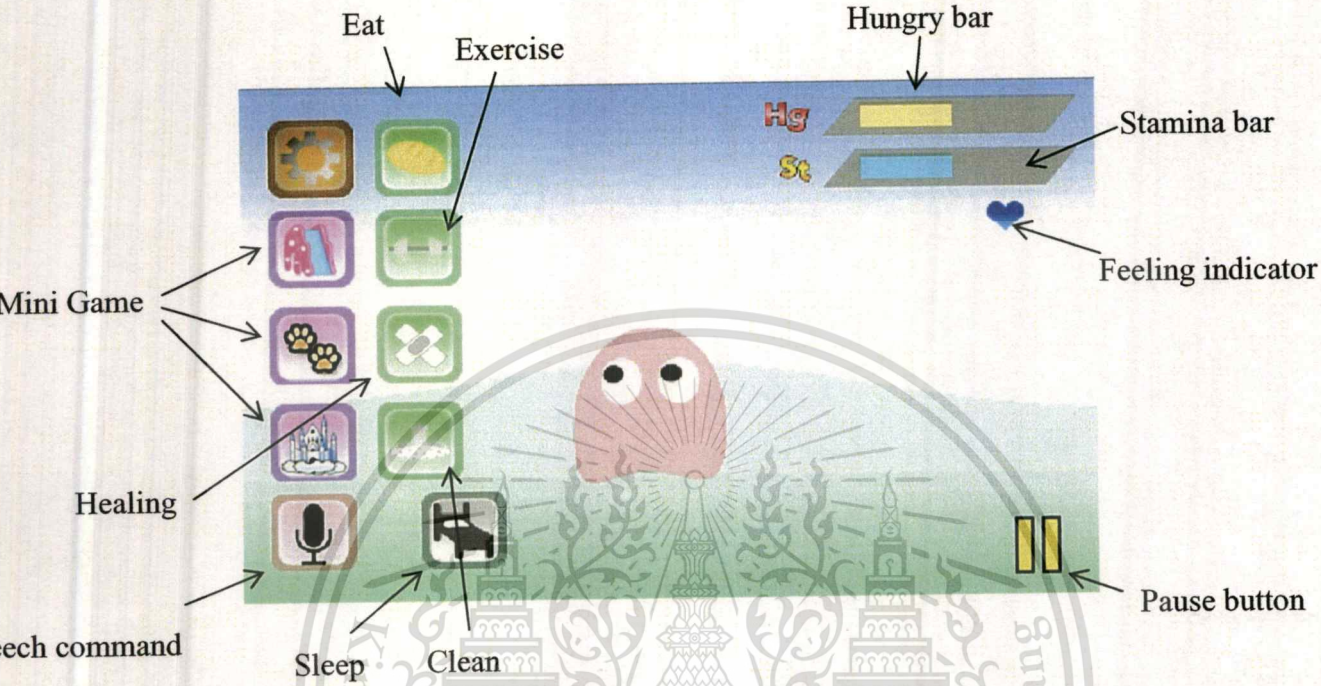


Figure 3.4 "Pet State" Interface.

- Clean icon when pet is released its excrement, use this command to clean it but if player leaves excrement for too long, it will decrease Feeling indicator and sometimes makes the Pet sick.
- Exercise icon uses this command to keep the Pet healthy. This command will directly increase the Stamina bar.
- Healing icon when pet is sick, used this command to cure it. If player don't do it in time, the pet will die.
- Eat icon this command provides food for the pet to eat and directly increase Hungry bar.

- Sleep icon uses this command to turn the light on or off.
- Mini Game win mini game to increase Feeling indicator and some healthy.
- Stamina bar this bar tells healthy of the pet.
- Hungry bar this bar tells the time that the pet will be hungry.
- Feeling indicator this icon tells how well being the pet has.

3.2.4 Mini Game

3.2.4.1 Track Game

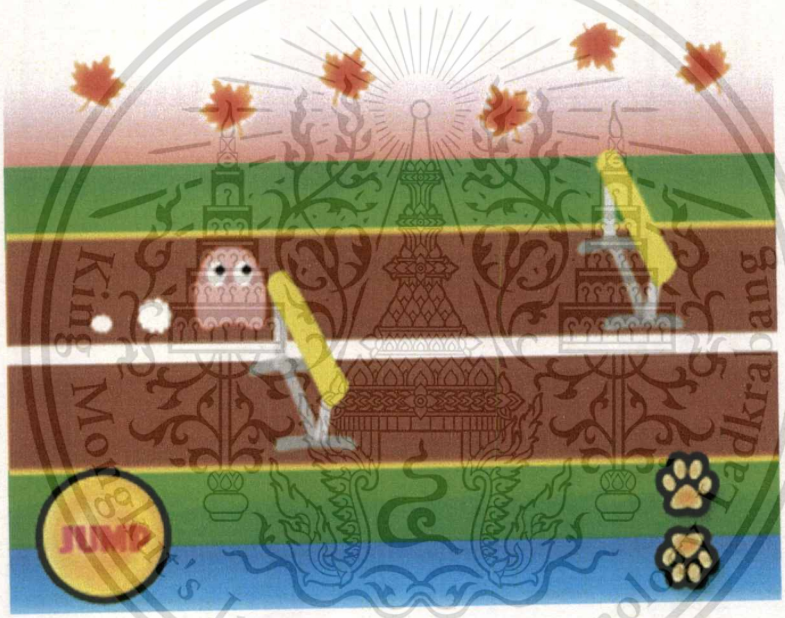


Figure 3.5 Mini Game Track.

There is a button in this game which player will need to hold and speak "jump" to let the pet jump. Pet can jump across an obstacle or tend to move to other lane to evade. Objective is to enter the goal without hitting the obstacle.

3.2.4.2 Snake Game

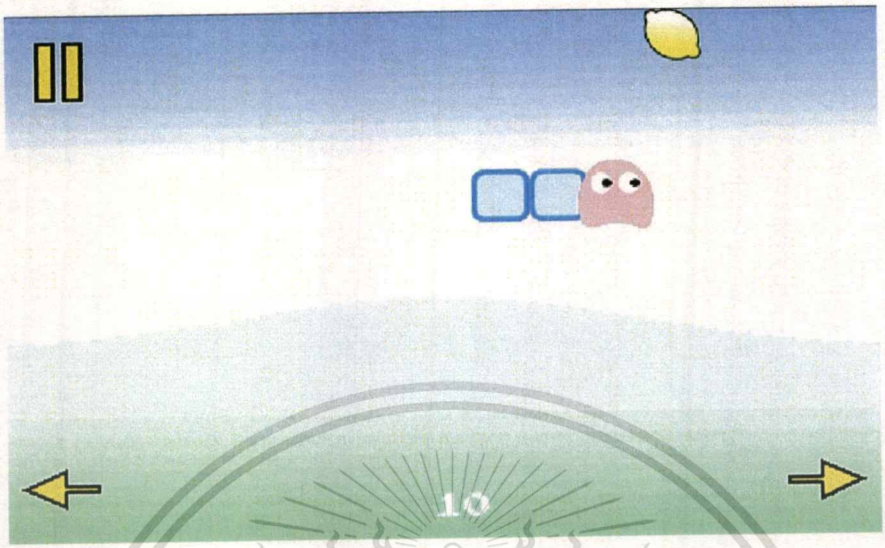


Figure 3.6 Mini Game Snake.

In this game, the task is to let the pet eat as much as possible by eating the fruits that will randomly place on the screen. Each fruits will increase the pet length by one block. Left and Right buttons are here to let the head of pet move to different direction. The game end when the head eat its own block.

3.2.4.3 Jump Game

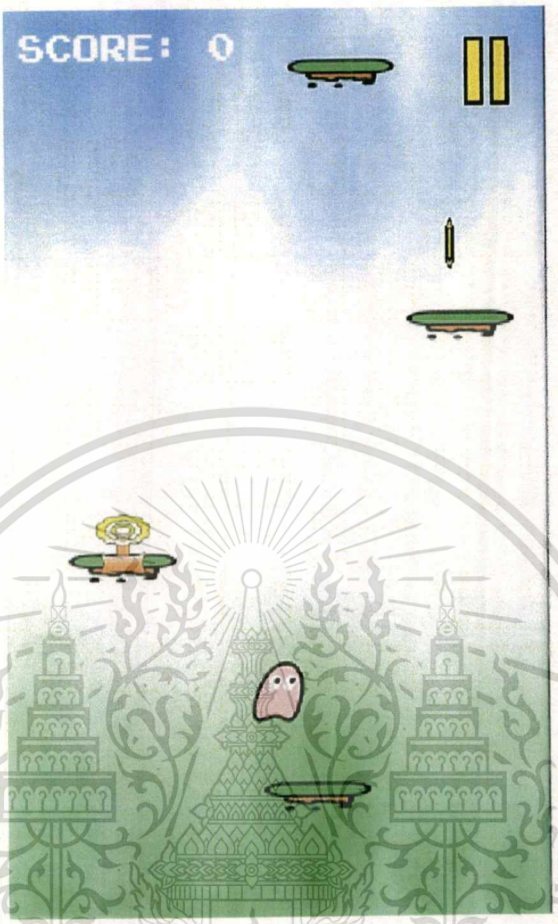


Figure 3.7 Mini Game Jump.

The objective is to reach the castle at the top between the ways up there will be many things that can help the pet go further or obstruct the pet. Collect as many coins as possible to get more scores.

3.3 Creating your own speech model

There are two steps to create speech model. First is creating language model by using "cmuclmtk tool". Then create acoustic model by using "sphinxtrain tool". However, to create acoustic model, it needs language model file. Therefore, it needs to have language model first.

3.3.1 Creating Language Model

3.3.1.1 Prepare the text file that is used for generating language model. This text file have to be normalized and uses tags <s> and </s> at the start and end for each sentences.

Format for creating your text file:

Start with <s> press one space bar and begin your sentence. After finishing your sentence then press another one space bar and end with </s>

<s> your sentences </s>

<s> your another sentences </s>

...

```
<s> left </s>
<s> right </s>
<s> up </s>
<s> down </s>
<s> forward </s>
<s> sit </s>
<s> walk </s>
```

Figure 3.8 example of text file prepared for this project.

As shown in figure 3.8, this special project uses command for only one word. Therefore each sentence will have only a word.

3.3.1.2 Save your prepared text file to where ever you want.

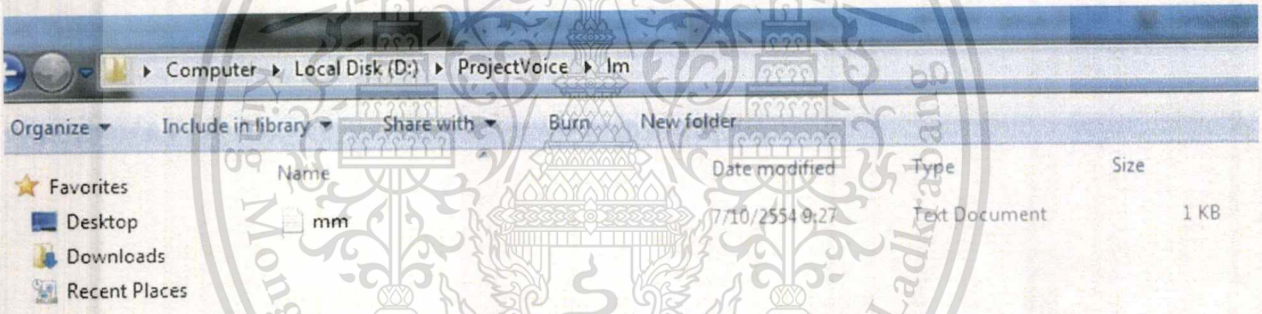


Figure 3.9 save location for text file.

3.3.1.3 Then open Cygwin and "cd" to location you save your prepared text file by using command "cd /cygdrive/where your save text file is"

For this project, is saved at D > ProjectVoice > lm

So use command

"cd /cygdrive/d/projectvoice/lm"

Figure 3.10 how to go to directory that save prepared text file.

3.3.1.4 Type "text2wfreq <NameOfYourText.txt> NameOfYourText.wfreq".

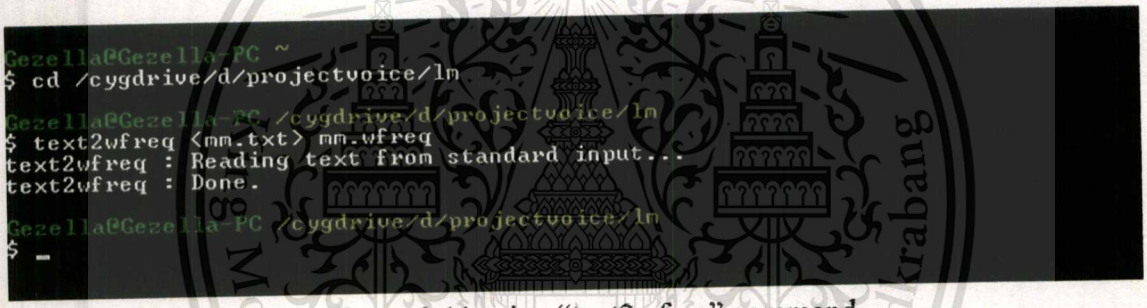


Figure 3.11 using "text2wfreq" command.

The process will generate ".wfreq" file when this process is done.

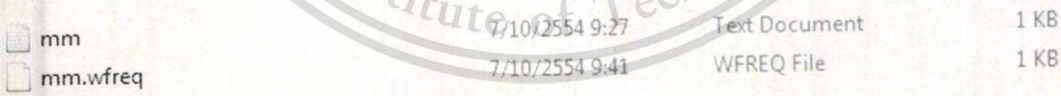


Figure 3.12 output file from "text2wfreq" command.

3.3.1.5 Type “wfreq2vocab <NameOfYourText.wfreq> NameOfYourText.vocab”

```
Gezella@Gezella-PC /cygdrive/d/projectvoice/lm
$ wfreq2vocab <mm.wfreq> mm.vocab
wfreq2vocab : Will generate a vocabulary containing the most
              frequent 20000 words. Reading wfreq stream from stdin...
wfreq2vocab : Done.

Gezella@Gezella-PC /cygdrive/d/projectvoice/lm
$ _
```

mm	7/10/2554 9:27	Text Document	1 KB
mm	7/10/2554 9:50	VOCAB File	1 KB
mm.wfreq	7/10/2554 9:41	WFREQ File	1 KB

Figure 3.13 using “wfreq2vocab” command and output from this command.

3.3.1.6 Type “text2idngram –vocab NameOfYourText.vocab

–idngram NameOfYourText.idngram < NameOfYourText.txt”

```
Gezella@Gezella-PC /cygdrive/d/projectvoice/lm
$ text2idngram -vocab mm.vocab -idngram mm.idngram < mm.txt
text2idngram
Vocab                   : mm.vocab
Output idngram          : mm.idngram
N-gram buffer size      : 100
Hash table size         : 2000000
Temp directory          : cmuclmtk-wr3ES6
Max open files          : 20
FOF size                : 10
n                       : 3
```

mm.idngram	7/10/2554 9:57	IDNGRAM File	1 KB
mm	7/10/2554 9:27	Text Document	1 KB
mm	7/10/2554 9:50	VOCAB File	1 KB
mm.wfreq	7/10/2554 9:41	WFREQ File	1 KB

Figure 3.14 using “text2idngram” command and output from this command.

3.3.1.7 Type “idngram2lm –vocab_type 0 –idngram NameOfYourText.idngram -vocab NameOfYourText.vocab –arpa NameOfYourText.arpa”

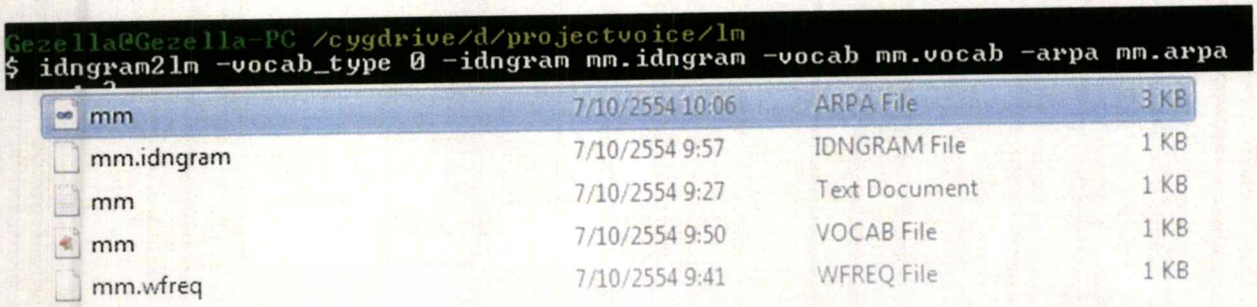


Figure 3.15 using “idngram2lm” command and output from this command.

Now finish generating language model file. However, “.arpa” format is not efficient enough to use for pocketsphinx. It needs to change to “.DMP”.

3.3.1.8 Change language model from “.arpa” to “.DMP”. Type

“sphinx_lm_convert –i NameOfYourText.arpa –o NameOfYourText.lm.DMP”

Figure 3.16 how to convert language model format.

3.3.2 Creating Acoustic Model

Creating acoustic model is also known as "Train the model".

The trainer learns the parameters of the models of the sound units using a set of example (sample) speech signals. This is called a "training database".

The database contains information require extracting statistics from the speech in form of the acoustic model.

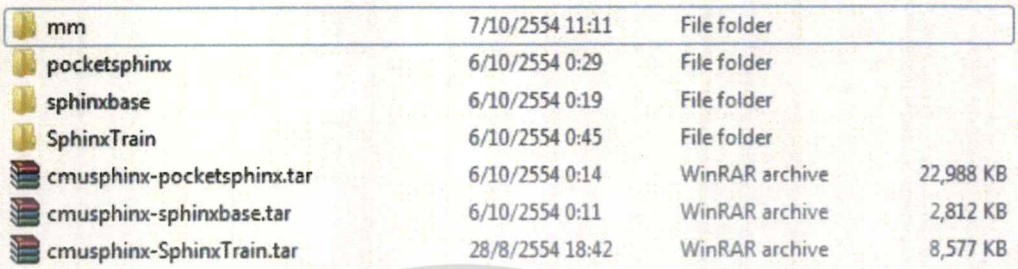
The trainer needs to specify which sound files the users want it to learn in users' training database. This information is provided to the trainer through a file called the transcript file.

Then, the trainer looks into a dictionary which maps every word to a sequence of sound units and derives the sequence of sound units associated with each signals.

In addition to the sound units, users have to create a set of transcripts for the database (in a single file) and two dictionaries, one in which legitimate words in the language are mapped sequences of sound files, and another in which non-speech sounds are mapped corresponding non-speech (for example silence sound (<s>) map to SIL). For sphinxtrain tool will refer these dictionaries to the former as the language dictionary and the latter as the filler dictionary.

After training, it needs to run the decoder to check the training results. The Decoder takes a model, tests part of the database and estimates the quality (WER) of the model.

3.3.2.1 Creating working folder which has the same root as sphinxtrain folder and pocketsphinx folder and name it the same as your language model.



mm	7/10/2554 11:11	File folder	
pocketsphinx	6/10/2554 0:29	File folder	
sphinxbase	6/10/2554 0:19	File folder	
SphinxTrain	6/10/2554 0:45	File folder	
cmusphinx-pocketsphinx.tar	6/10/2554 0:14	WinRAR archive	22,988 KB
cmusphinx-sphinxbase.tar	6/10/2554 0:11	WinRAR archive	2,812 KB
cmusphinx-SphinxTrain.tar	28/8/2554 18:42	WinRAR archive	8,577 KB

Figure 3.17 how to create working folder.

3.3.2.2 Go into working folder to create "etc folder" and "wav folder"

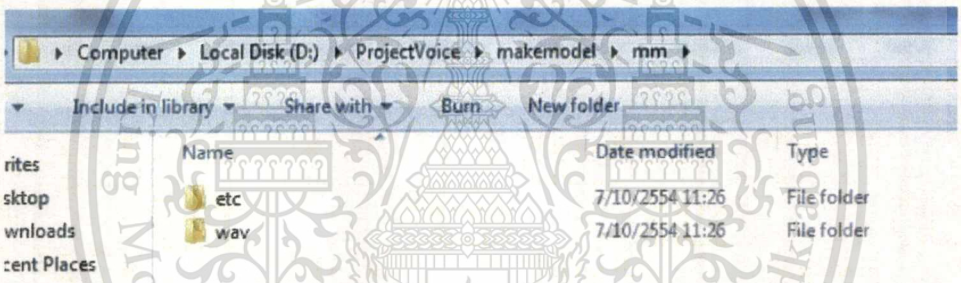


Figure 3.18 creating etc and wav folder working folder.

3.3.2.3 Prepare the database for training

- etc
  - your_db.dic - *Phonetic dictionary*
  - your_db.phone - *Phoneset file*
  - your_db.lm.DMP - *Language model*
  - your_db.fillers - *List of fillers*
  - your_db_train.fileids - *List of files for training*
  - your_db_train.transcription - *Transcription for training*
  - your_db_test.fileids - *List of files for testing*
  - your_db_test.transcription - *Transcription for testing*
- wav
  - speaker_1
    - file_1.wav - *Recording of speech utterance*
  - speaker_2
    - file_2.wav

Figure 3.19 file structure for the training the model.

- Phonetic Dictionary (your_db.dic) should have one line per word with word following the phonetic transcription

```
1  DOWN          D O W N
2  FORWARD       F O R W A R D
3  LEFT          L E F T
4  RIGHT         R I GH T
5  SIT           S I T
6  UP            U P
7  WALK          W A L K
8
```

Figure 3.20 Phonetic dictionary (mm.dic).

- Phoneset file (your_db.phone) should have one phone per line. The number of phones should match the phones used in the dictionary plus the special SIL phone for silence.

```
A
D
E
F
GH
I
K
L
N
O
P
R
S
SIL
T
U
W
```

Figure 3.21 Phoneset file (mm.phone).

- Language model file (your_db.lm.DMP) should be in DMP format that was generated from cmuclmtk tool.
- Filler dictionary (your_db.filler) contains filler phones (not-covered by language model non-linguistic sounds like breathe, hmm or laugh). It can contain just silences.

```
<s>             SIL
</s>            SIL
<sil>           SIL
```

Figure 3.22 Filler dictionary (mm.filler).

- Transcription file (your_db_train.transcription and your_db_test.transcription) is a text file listing the transcription for each audio files.

```
<s> LEFT </s> (0001)
<s> RIGHT </s> (0002)
<s> UP </s> (0003)
<s> DOWN </s> (0004)
<s> FORWARD </s> (0005)
<s> SIT </s> (0006)
<s> WALK </s> (0007)
<s> LEFT RIGHT </s> (0008)
<s> RIGHT UP </s> (0009)
<s> DOWN FORWARD </s> (0010)
<s> FORWARD SIT </s> (0011)
<s> SIT WALK </s> (0012)
<s> LEFT UP </s> (0013)
<s> LEFT DOWN </s> (0014)
<s> LEFT FORWARD </s> (0015)
<s> LEFT SIT </s> (0016)
<s> LEFT WALK </s> (0017)
<s> RIGHT DOWN </s> (0018)
<s> RIGHT FORWARD </s> (0019)
<s> RIGHT SIT </s> (0020)
<s> RIGHT WALK </s> (0021)
<s> UP FORWARD </s> (0022)
<s> UP SIT </s> (0023)
<s> UP WALK </s> (0024)
<s> DOWN SIT </s> (0025)
<s> DOWN WALK </s> (0026)
<s> FORWARD WALK </s> (0027)
<s> LEFT RIGHT UP </s> (0028)
<s> DOWN FORWARD SIT </s> (0029)
<s> WALK RIGHT UP </s> (0030)
<s> SIT WALK DOWN </s> (0031)
<s> LEFT FORWARD SIT </s> (0032)
<s> UP DOWN WALK </s> (0033)
```

```
LEFT (0001)
RIGHT (0002)
UP (0003)
DOWN (0004)
FORWARD (0005)
SIT (0006)
WALK (0007)
```

Figure 3.23 file mm_train.transcription (left) and mm_test.transcriptsion (right).

It's important that each lines starts with <s> and ends with </s> are followed by id in parentheses

- Creating folder train and test in wav file to contain separating sound file for training in train folder and for testing in test folder.

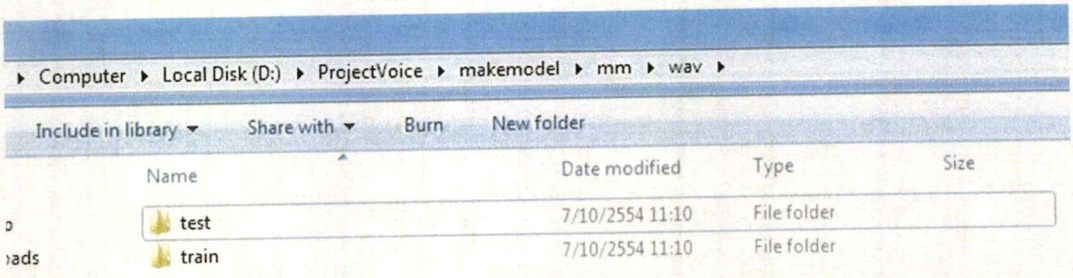


Figure 3.24 creating test and train folder in wav folder.

- Creating your recording file. Recoding file must be in MS WAV format with specific sample rate 8kHz, 16bit, mono for mobile applications. There are many ways to record the sound.

a) Manually segment audio recordings with existing transcription (podcasts, news, etc).

b) Record your friends', families', and colleagues'.

Figure 3.25 wav sounds record for train.

The word are spoken in wav sound must match the word are spoken in transcription file.

- Fileids (your_db_train.fileids and your_db_test.fileids) file is a text file listing the names of the records (utterance ids) one by line.

-

```
train/0001
train/0002
train/0003
train/0004
train/0005
train/0006
train/0007
train/0008
train/0009
train/0010
train/0011
train/0012
train/0013
train/0014
train/0015
train/0016
train/0017
train/0018
train/0019
train/0020
train/0021
train/0022
train/0023
train/0024
train/0025
train/0026
train/0027
train/0028
train/0029
train/0030
train/0031
train/0032
train/0033
```

```
test/0001
test/0002
test/0003
test/0004
test/0005
test/0006
test/0007
```

Figure 3.26 file mm_train.fileids (left) and mm_test.fileids (right).

- Now you are ready to create acoustic model.

mm	6/10/2554 1:22	Text Document	1 KB
mm.filler	6/10/2554 1:24	FILLER File	1 KB
mm.lm	7/10/2554 10:15	Crash Dump File	3 KB
mm.phone	6/10/2554 1:27	PHONE File	1 KB
mm_test.fileids	6/10/2554 1:41	FILEIDS File	1 KB
mm_test.transcription	6/10/2554 1:40	TRANSCRIPTION ...	1 KB
mm_train.fileids	6/10/2554 1:37	FILEIDS File	1 KB
mm_train.transcription	6/10/2554 1:36	TRANSCRIPTION ...	1 KB

Figure 3.27 complete prepared file for train.

3.3.2.4 Open Cygwin then go to your working folder.

For this project D > projectvoice > makemodel > mm

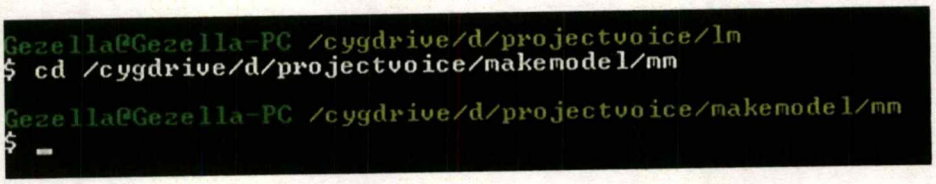


Figure 3.28 how to go to working folder.

3.3.2.5 Type "../sphinxtrain/scripts_pl/setup_sphinxtrain.pl -task your task name"

Figure 3.29 how to install sphinxtrain on working folder.

3.3.2.6 Type "../pocketsphinx/scripts/setup_sphinx.pl -task your task name"

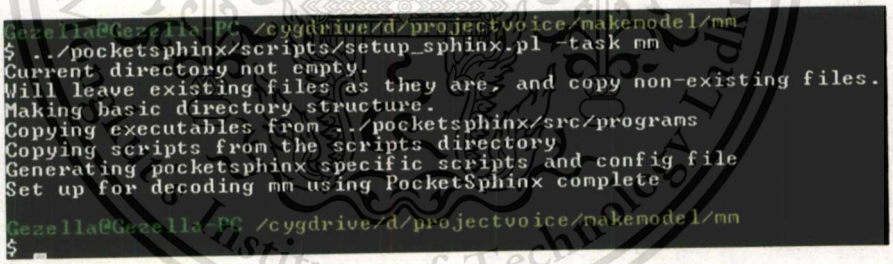
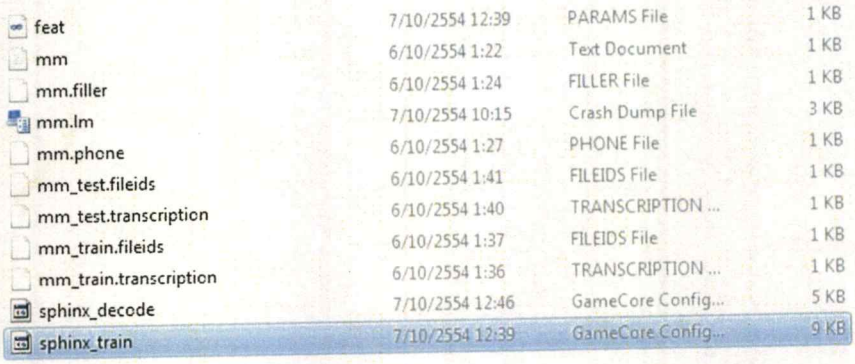


Figure 3.30 how to install pocketsphinx on working folder.

3.3.2.7 After that, the configuration files in etc folder need to be edited. Go to etc folder and open sphinx_train.cfg.



feat	7/10/2554 12:39	PARAMS File	1 KB
mm	6/10/2554 1:22	Text Document	1 KB
mm.filler	6/10/2554 1:24	FILLER File	1 KB
mm.lm	7/10/2554 10:15	Crash Dump File	3 KB
mm.phone	6/10/2554 1:27	PHONE File	1 KB
mm_test.fileids	6/10/2554 1:41	FILEIDS File	1 KB
mm_test.transcription	6/10/2554 1:40	TRANSCRIPTION ...	1 KB
mm_train.fileids	6/10/2554 1:37	FILEIDS File	1 KB
mm_train.transcription	6/10/2554 1:36	TRANSCRIPTION ...	1 KB
sphinx_decode	7/10/2554 12:46	GameCore Config...	5 KB
sphinx_train	7/10/2554 12:39	GameCore Config...	9 KB

Figure 3.31 file sphinx_train.cfg that generate from sphinxtrain tool.

3.3.2.8 Configuring the value in sphinx_train.cfg.

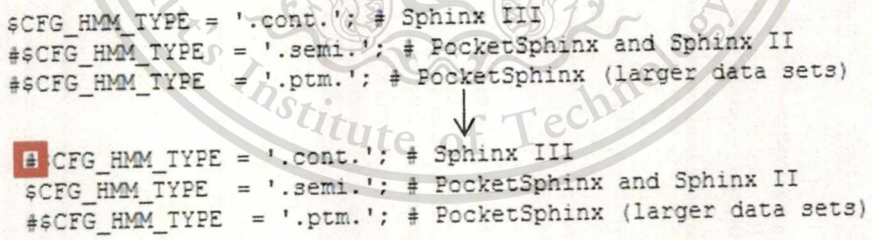
At line 27-30 change the value to be as in the figure 3.32

```
# Feature extraction parameters
$CFG_WAVFILE_SRATE = 16000.0;
$CFG_NUM_FILT = 40;
$CFG_LO_FILT = 133.3334;
$CFG_HI_FILT = 6855.4976;
```

```
# Feature extraction parameters
$CFG_WAVFILE_SRATE = 8000.0;
$CFG_NUM_FILT = 31;
$CFG_LO_FILT = 200;
$CFG_HI_FILT = 3500;
```

Figure 3.32 adjust values of sample rate and its match.

At line 106-108 change the model type used to train.



```
$CFG_HMM_TYPE = '.cont.'; # Sphinx III
#$CFG_HMM_TYPE = '.semi.'; # PocketSphinx and Sphinx II
#$CFG_HMM_TYPE = '.ptm.'; # PocketSphinx (larger data sets)
```

```
$CFG_HMM_TYPE = '.cont.'; # Sphinx III
$CFG_HMM_TYPE = '.semi.'; # PocketSphinx and Sphinx II
#$CFG_HMM_TYPE = '.ptm.'; # PocketSphinx (larger data sets)
```

Figure 3.33 how to change model for train.

At line 127-128 change NUM_DENSITIES to match your project following figure 3.34.

```
$CFG_INITIAL_NUM_DENSITIES = 256;
$CFG_FINAL_NUM_DENSITIES = 256;
```
→
```
$CFG_INITIAL_NUM_DENSITIES = 4;
$CFG_FINAL_NUM_DENSITIES = 4;
```

Figure 3.34 configure value densities.

Vocabulary	Hours in db	Senones	Densities	Example
20	5	200	8	Tidigits Digits Recognition
100	20	2000	8	RM1 Command and Control
5000	30	4000	16	WSJ1 5k Small Dictation
20000	80	4000	32	WSJ1 20k Big Dictation
60000	200	6000	16	HUB4 Broadcast News
60000	2000	12000	64	Fisher Rich Telephone Transcription

Figure 3.35 number of vocabulary to match configures.

At line 167

```
$CFG_N_TIED_STATES = 1000;
```
→
```
$CFG_N_TIED_STATES = 100;
```

Figure 3.36 configure values of senones to match the densities.

Now ready to create acoustic model.

3.3.2.9 Open Cygwin then type "./scripts_pl/runall.pl"

```
Gezella@Gezella-PC /cygdrive/d/projectvoice/makemodel/mm
$ ./scripts_pl/runall.pl
```

Figure 3.37 using "runall.pl" command.

```
        T 1
        T 2
        U 0
        U 1
        U 2
        W 0
        W 1
        W 2
MODULE: 45 Prune Trees
    Phase 1: Tree Pruning
    Phase 2: State Tying
MODULE: 50 Training Context dependent models
    Phase 1: Cleaning up directories:
        accumulator...logs...qmanager...
    Phase 2: Copy CI to CD initialize
    Phase 3: Forward-Backward
        Baum welch starting for 4 Gaussian(s), iteration: 1 (1 of 2)
        0% 50% 100%
WARNING: This step had 0 ERROR messages and 1 WARNING messages.  Please check th
e log file for details.
        Baum welch starting for 4 Gaussian(s), iteration: 1 (2 of 2)
        0% 50% 100%
WARNING: This step had 0 ERROR messages and 1 WARNING messages.  Please check th
e log file for details.
```

Figure 3.38 during process in creating acoustic model.

```
    Phase 1: Cleaning up directories: logs...
    Phase 2: Doing interpolation...
WARNING: This step had 0 ERROR messages and 11 WARNING messages.  Please check t
he log file for details.
    Phase 3: Dumping senones for PocketSphinx...

Gezella@Gezella-PC /cygdrive/d/projectvoice/makemodel/mm
$
```

Figure 3.39 finish process in creating acoustic model.

Now finish creating acoustic model.

3.3.2.10 Testing accuracy of the model by typing "./scripts_pl/decode/slave.pl"

In figure 3.40 SENTENCE ERROR show percentage of error rate for mm model

```
Gezella@Gezella-PC /cygdrive/d/projectvoice/makemodel/mm
$ ./scripts_pl/decode/slave.pl
MODULE: DECODE Decoding using models previously trained
        Decoding 7 segments starting at 0 (part 1 of 1)
        0%
        Aligning results to find error rate
        SENTENCE ERROR: 57.1% (4/7)   WORD ERROR RATE: 57.1% (3/7)

Gezella@Gezella-PC /cygdrive/d/projectvoice/makemodel/mm
$
```

Figure 3.40 process to test the accuracy of model.

3.3.3 Using the model

To use your own speech model on Android device; acoustic model files, language model, and dictionary need to be copied and placed on the device folder.

For acoustic model, you can find it in folder model_parameters/mm.cd_semi_100 copy all the files in this folder to your device.

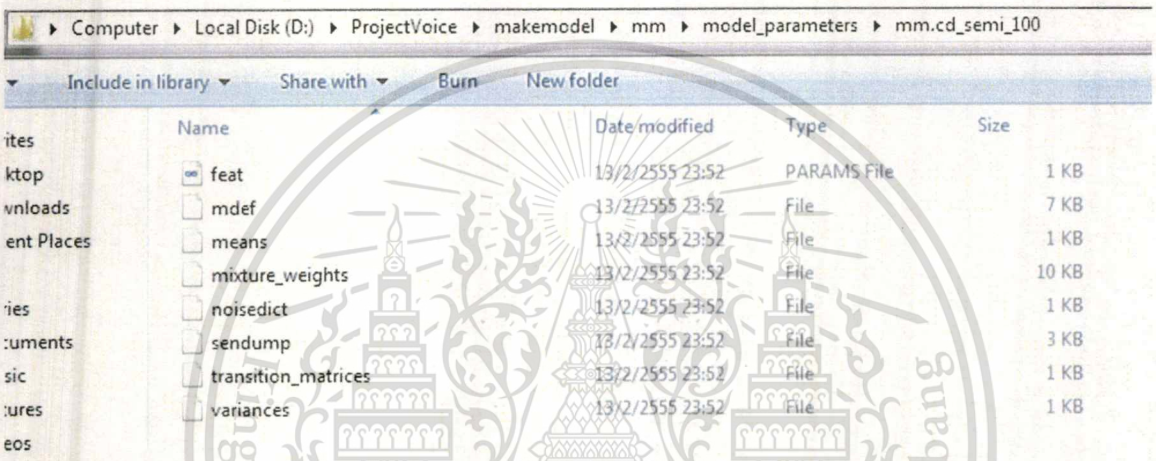


Figure 3.41 acoustic model files.

For language model and dictionary, you can find it in etc folder in the working folder.

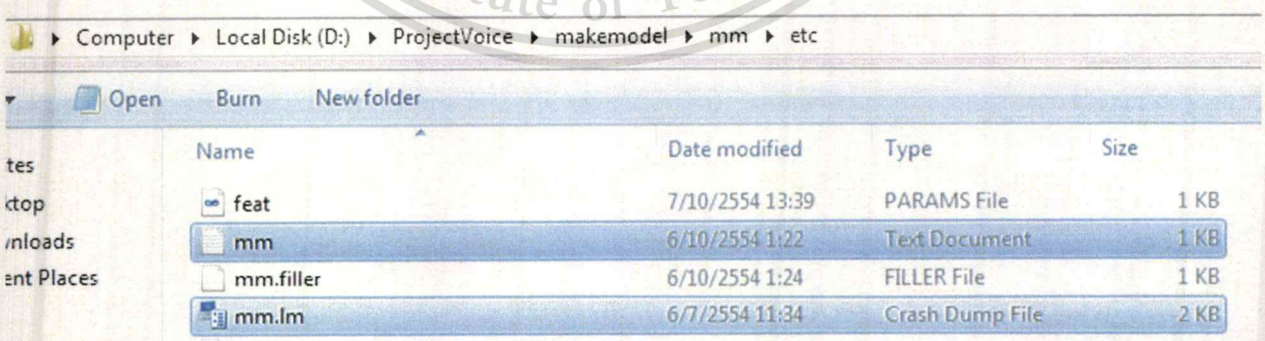


Figure 3.42 language model and dictionary files.

3.4 Programming

3.4.1 Setting up project for Pocketsphinx on Android project

In order to use PocketSphinx tool on Android, it needs to be built first and then port it from C files to java files.

3.4.1.1 Setting up environment

See Appendix C.

3.4.1.2 Building PocketSphinx Android Project

First open Cygwin and cd to your Android project in working folder.

Figure 3.43 cd into PocketSphinxAndroidDemo.

Then type /cygdrive/d/voiceproject/sphinx2/android-ndk-r7b/ndk-build

to create "libpocketsphinx_jni.so" file and "obj" folder for native library android uses.

```
Gezella@Gezella-PC ~
$ cd /cygdrive/d/voiceproject/sphinx2/pocketsphinxandroiddemo

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/pocketsphinxandroiddemo
$ /cygdrive/d/voiceproject/sphinx2/android-ndk-r7b/ndk-build
Gdbserver      : [arm-linux-androideabi-4.4.3] libs/armeabi/gdbserver
Gdbsetup       : libs/armeabi/gdb.setup
Cygwin         : Generating dependency file converter script
Compile thumb  : pocketsphinx_jni <= pocketsphinx_wrap.c
Compile thumb  : pocketsphinx <= acmod.c
Compile thumb  : pocketsphinx <= bin_mdef.c
```

```
Gdbserver      : [arm-linux-androideabi-4.4.3] libs/armeabi/gdbserver
Gdbsetup       : libs/armeabi/gdb.setup
Install        : libpocketsphinx_jni.so => libs/armeabi/libpocketsphinx_jni.so

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/pocketsphinxandroiddemo
$
```

Figure 3.44 building process.

Next cd into pocketsphinx/swig in working folder and type "./makefile" to build java library for android.

```
Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/pocketsphinxandroiddemo
$ cd /cygdrive/d/voiceproject/sphinx2/pocketsphinx/swig

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/pocketsphinx/swig
$ makefile
bash: makefile: command not found

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/pocketsphinx/swig
$ ./makefile
./makefile: line 1: CFLAGS: command not found
Package sphinxbase was not found in the pkg-config search path.
Perhaps you should add the directory containing `sphinxbase.pc'
to the PKG_CONFIG_PATH environment variable
```

Figure 3.45 create java library pocketsphinx for android.

Now look into working folder -> pocketsphinx -> swig -> edu -> cmu -> pocketsphinx path, you will see many java files that were created.

Config	31/3/2555 18:13	JAVA File	3 KB
ConfigTest	1/7/2553 8:56	JAVA File	2 KB
Decoder	31/3/2555 18:13	JAVA File	3 KB
DecoderTest	1/7/2553 13:22	JAVA File	2 KB
goforward	1/7/2553 8:56	Wave Sound	88 KB
Hypothesis	31/3/2555 18:13	JAVA File	2 KB
Lattice	31/3/2555 18:13	JAVA File	1 KB
LatticeTest	1/7/2553 11:36	JAVA File	1 KB
pocketsphinx	31/3/2555 18:13	JAVA File	1 KB
pocketsphinxJNI	31/3/2555 18:13	JAVA File	4 KB
SegmentIterator	31/3/2555 18:13	JAVA File	2 KB
test	1/7/2553 8:56	CFG File	1 KB

Figure 3.46 java library generate from swig.

You need to copy these following files to your working folder /PocketSphinxAndroidDemo/jni/edu/cmu/pocketsphinx

- Config.java
- Decoder.java
- Hypothesis.java
- Lattice.java
- pocketsphinx.java
- pocketsphinxJNI.java
- segmentIterator.java

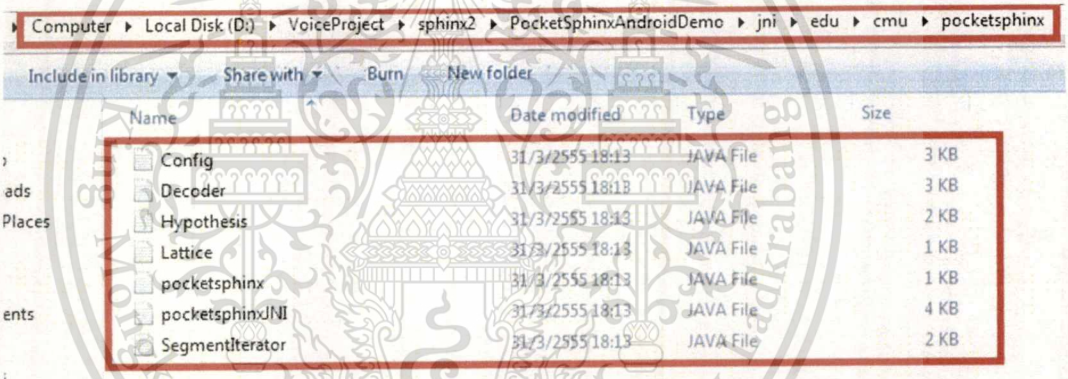


Figure 3.47 copy pocketsphinx java library into android project.

Next open eclipse and select File -> Import…

When import screen is popped up select General -> Existing Projects into Workspace then click next.

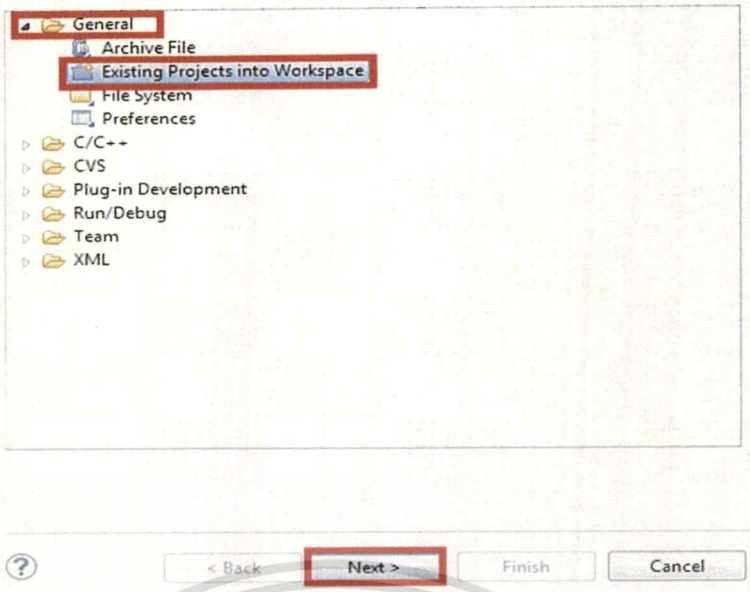


Figure 3.48 import android project into eclipse workspace (1).

Click Browse and select PocketSphinxAndroidDemo after that check "copy projects into workspace" then click finish.

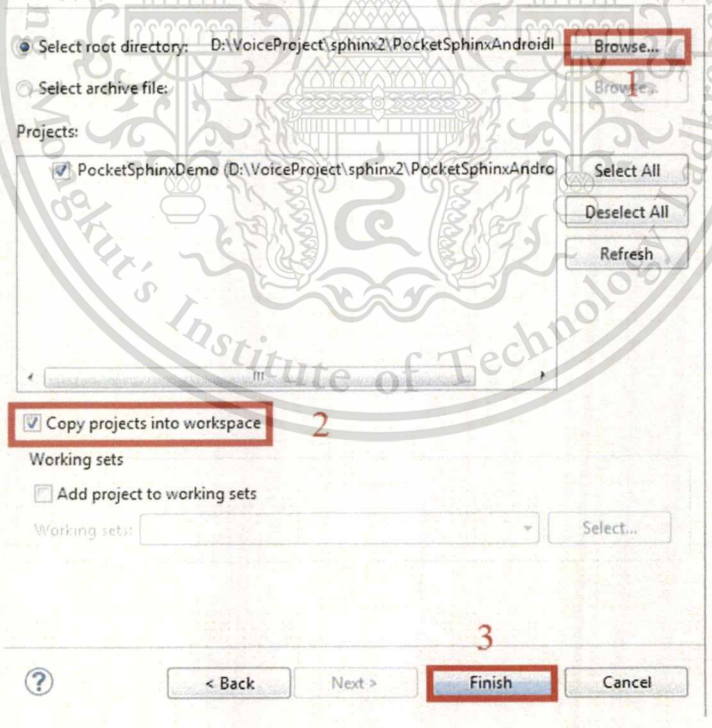


Figure 3.49 import android project into eclipse workspace (2).

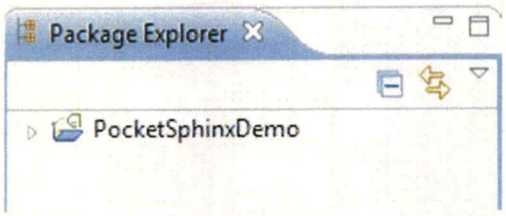


Figure 3.50 import PocketSphinxAndroidDemo.

Now you have finished import and build library pocketsphinx for Android.

3.4.2 Create package in Eclipse

Package is the way to group classes together to make it easier to control everything. To create package in Eclipse right click at src -> select new -> package specify the source folder and the name of package then click finish.

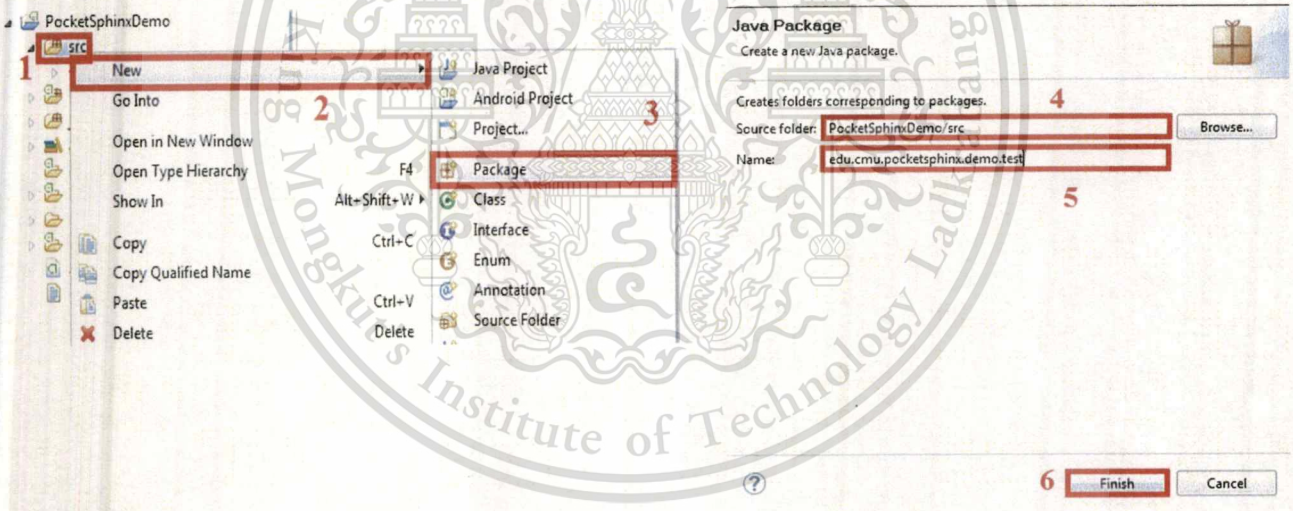


Figure 3.51 show how to create package in Eclipse.

3.4.3 Create your own Game Framework

3.4.3.1 Overview of game framework

Every game needs some basic framework such as how to add images, input, etc. so that if they are separated from the actual game, it will be easier to control everything. Usually every framework is split up into modules, as follows:

Window management: This module will deal with the sleep screen such as pause and resume the application.

Input: This module will deal with user's inputs such as touch events, keystrokes and accelerometers.

File I/O: This module will make us get the bytes of our resources that will store in assets folder.

Graphics: This module is the most complex module. It deals with loading pictures and drawing them on the screen.

Game Framework: This module will combine all modules together and provides an easy-to-use to write our games.

3.4.3.2 Game framework diagram

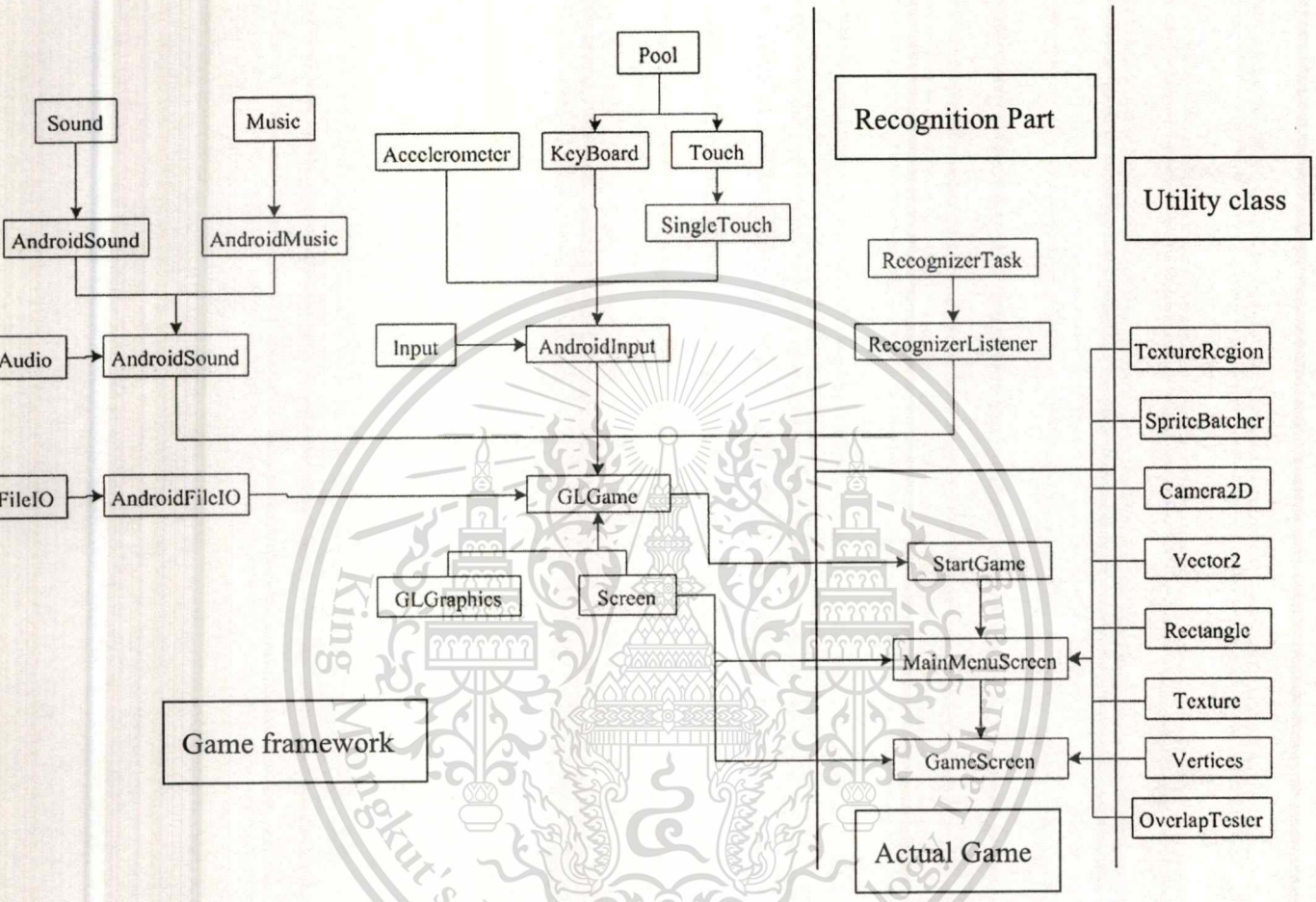


Figure 3.52 Game framework diagram.

3.4.4 Interface for each Module.

3.4.4.1 Input

```
package edu.cmu.pocketsphinx.framework;


import java.util.List;
```

```
public interface Input { // interface for input implement later...
   public static class KeyEvent { // defined constants that encode key's event
      public static final int KEY_DOWN = 0;  // key is down
      public static final int KEY_UP = 1; // key is up
      public int type; // key's type  (up or down)
      public int keyCode; //key's code in number
      public char keyChar; //character of that key e.g. a, b etc.
   }
   public static class TouchEvent { // defined constants that encode touch's event
      public static final int TOUCH_DOWN = 0; // touch down
      public static final int TOUCH_UP = 1; // touch up
      public static final int TOUCH_DRAGGED = 2; // touch drag
      public int type; // touch event's type (up, down or drag)
      public int x, y; // x and y coordinate on the UI where touch occur
      public int pointer; // for multitouch in this project never use assign 0 only
   }
   public boolean isKeyPressed(int keyCode); //check is specify key is pressed or not
   public boolean is TouchDown(int pointer); // check touch event
   public int getTouchX(int pointer); // get x's position where touch occur
   public int getTouchY(int pointer); // get y's position where touch occur
   public float getAccelX(); // get accelerometer's value for x-axis
   public float getAccelY(); // get  accelerometer's value for y-axis
   public float getAccelZ(); // get accelerometer's value for z-axis
   public List<KeyEvent> getKeyEvents();  // get instance of key event
   public List<TouchEvent> getTouchEvents(); // get instance of touch event
}
```

3.4.4.2 File I/O

```
package edu.cmu.pocketsphinx.framework;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

public interface FileIO { // interface for FileIO implement later...

   public InputStream readAsset(String fileName) throws IOException;

   // read file from asset folder by giving file's name

   public InputStream readFile(String fileName) throws IOException;

   // read file from storage by giving file's name

   public OutputStream writeFile(String fileName) throws IOException;

  // write file to storage by file's name

}
```

3.4.4.3 Audio

- Sound : sound effect such as "bang", "boom".

```
package edu.cmu.pocketsphinx.framework;

public interface Sound { // interface for sound implement later

   public void play(float volume); // play sound effect with volume (0-1)

   public void dispose(); // once don't need sound instance use dispose method

}
```

- Music: songs or background music.

```
package edu.cmu.pocketsphinx.framework;

public interface Music { // interface for music implement later

   public void play(); // play music

   public void stop(); // stop music

   public void pause(); // pause music

   public void setLooping(boolean looping); // set loop playback when reach the end

   public void setVolume(float volume); // set volumn in (0-1)

   public boolean isPlaying(); // check is music playing

   public boolean isStopped(); // check music is stop

   public boolean isLooping(); // check music is set to loop

   public void dispose(); // don't need this instance music anymore dispose it

}
```

- Audio interface is used for tying sounds and music together in audio which produces either new sound or music.

```
package edu.cmu.pocketsphinx.framework;

public interface Audio { // audio interface implement later

   public Music newMusic(String filename); // create new music instance

   public Sound newSound(String filename); // create new sound instance

}
```

3.4.4.4 Graphics: for OpenGL, the game uses GLSurfaceView which already has been provided by Android API.

3.4.4.5 Game framework ties all modules together.

```
package edu.cmu.pocketsphinx.framework;

public interface Game { // game interface implement later

   public Input getInput(); // get Input instance

   public FileIO getFileIO(); // get FileIO instance

   public Audio getAudio(); // get Audio instance

   public void setScreen(Screen screen); // set current screen

   public Screen getCurrentScreen(); // return current active screen

   public Screen getStartScreen(); // use to set up start game

}
```

Example of usage.

```
// to get input

        float accelY = game.getInput().getAccelY();

// to provide background music

        Music bgMusic = game.getAudio().newMusic("bgTheme.mp3");

        bgMusic.setLooping(true);

        bgMusic.setVolumn(0.5f);

        bgMusic.play();
```

3.4.4.6 Screen: In the game, every screen needs to be updated and presented so abstract classes need to be provided for less coding time.

```
package com.example.sample.framework;

public abstract class Screen{ // abstract class to use every screen in game

        protected final Game game; // game instance for every screen

        public Screen(Game game){  // constructure

                this.game = game; // set game instance

        }

        public abstract void update(float deltaTime); // update screen

                // deltatime = how long between each frame

        public abstract void present(float deltaTime); // present screen

        public abstract void pause(); // pause screen

        public abstract void resume(); // resume screen

        public abstract void dispose(); // don't need screen any more dispose it.
```

3.4.5 Implement Game Framework

In this part interfaces and abstract classes are implemented from the interfaces that were created earlier. Interfaces are located in the package edu.cmu.pocketsphinx.framework then implementation will put into the package edu.cmu.pocketsphin.framework.impl. For an import section in the code, only AndroidFileIO class will be shown because in Eclipse, it have function that help user to add an import file by clicking on it.

See Appendix D.

3.4.5.1 AndroidFileIO class

The FileIO interface contains three methods. First, it has an InputStream for an asset; asset is the folder in Android project where the resources are kept such as music, pictures etc. (In Android, it has two ways to put resources in folder /res and folder /assets, this project uses /assets instead of /res because it can put different file types in the same folder). Second it requires an InputStream for a file on the external storage. Third, it returns an OutputStream for a file on the external storage.

```
package edu.cmu.pocketsphinx.framework.impl;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;              //import needed classes

import java.io.OutputStream;

import android.content.res.AssetManager;

import edu.cmu.pocketsphinx.framework.FileIO;
```

```
public class AndroidFileIO implements FileIO {
    AssetManager assets; //declare for access to folder /assets
    String externalStoragePath; // path to external storage
    public AndroidFileIO(AssetManager assets) {
        this.assets = assets;
        this.externalStoragePath = "/sdcard/Android/data/edu.cmu.pocketsphinx/setting/";
    }
    @Override
    public InputStream readAsset(String fileName) throws IOException {
        return assets.open(fileName); // read file name from folder /assets
    }
    @Override
    public InputStream readFile(String fileName) throws IOException {
        return new FileInputStream(externalStoragePath + fileName);
    } // read file from external storage
    @Override
    public OutputStream writeFile(String fileName) throws IOException {
        return new FileOutputStream(externalStoragePath + fileName);
    } // write file to external storage
}
```

FileIO interface was implemented by storing an AssetManager together with the path to external storage. The game interface will holds the instance of this class and gives the AssetManager to make it work.

3.4.5.2 AndroidMusic, AndroidSound and AndroidAudio

Earlier three interfaces for audio have been created: Audio, Music and Sound. Audio responses to create Sound and Music instances from asset files, Sound responses to playback sound effects and Music responses to play bigger music files.

First AndroidAudio.

```
public class AndroidAudio implements Audio {
  AssetManager assets; // need to load sound effect from asset files
  SoundPool soundPool; // store sound effect
  public AndroidAudio(Activity activity) {
    activity.setVolumeControlStream(AudioManager.STREAM_MUSIC);
    this.assets = activity.getAssets(); // get and store AssetManger instance
    this.soundPool = new SoundPool(20, AudioManager.STREAM_MUSIC, 0);
  } // set soundPool configures to be able to play 20 sound effects in parallel
```

In the constructor, the class receives activity because it uses to setup volume in the media stream and gives AssetManager instance.

```
@Override
  public Music newMusic(String filename) {
    try {
      AssetFileDescriptor assetDescriptor = assets.openFd(filename);
      return new AndroidMusic(assetDescriptor);
    } catch (IOException e) {
      throw new RuntimeException("Couldn't load music '" + filename + "'");
    } // end catch  }// end newMusic
```

```
@Override
  public Sound newSound(String filename) {
    try {
      AssetFileDescriptor assetDescriptor = assets.openFd(filename);
      int soundId = soundPool.load(assetDescriptor, 0);
      return new AndroidSound(soundPool, soundId);
    } catch (IOException e) {
      throw new RuntimeException("Couldn't load sound '" + filename + "'");
    }
  }
```

For both Music and Sound, they will be loaded from folder /assets by method AssetManager.openFd(). In Music, AssetFileDescriptor will response to read data as well as offset and length of that entry's data for streaming. In Sound, the sound effect's data are small enough to store in RAM. Then provides SoundPool class to load and assign id to each sound effect.

Second AndroidSound. The methods should be self-explanation.

```
public class AndroidSound implements Sound {
  int soundId; // store sound effect's id
  SoundPool soundPool; // store soundPool instance
  public AndroidSound(SoundPool soundPool,int soundId) {
    this.soundId = soundId;
    this.soundPool = soundPool;
  } // this constructor call from AndroidAudio
```

```
@Override
    public void play(float volume) {
        soundPool.play(soundId, volume, volume, 0, 0, 1); // play sound effect
    }
    @Override
    public void dispose() {
        soundPool.unload(soundId); // release RAM that store sound effect
    }
}
```

Third AndroidMusic.

```
public class AndroidMusic implements Music, OnCompletionListener {
    MediaPlayer mediaPlayer;
    boolean isPrepared = false;
```

The AndroidMusic class stores a MediaPlayer instance along with a boolean member isPrepared. This boolean member will let us keep track of MediaPlayer's state. The AndroidMusic class not only implements the Music interface, but also implements OnCompletionListener interface to inform that when the music is stopping, it needs to be prepared again before other methods are invoked.

```
public AndroidMusic(AssetFileDescriptor assetDescriptor) {
    mediaPlayer = new MediaPlayer();
    try {
      mediaPlayer.setDataSource(assetDescriptor.getFileDescriptor(),
          assetDescriptor.getStartOffset(), // create and prepare
          assetDescriptor.getLength()); // the MediaPlayer from AssetFileDescriptor
      mediaPlayer.prepare();
      isPrepared = true; // set prepare flag
      mediaPlayer.setOnCompletionListener(this);
       // register AndroidMusic as an OnCompletionListener with the MediaPlayer
    } catch (Exception e) {
      throw new RuntimeException("Couldn't load music");
    } // end catch
} //end AndroidMusic
@Override
  public void dispose() {  // when no longer use this music just call this method
    if (mediaPlayer.isPlaying()) // check is music playing or not
      mediaPlayer.stop(); // if player then stop
    mediaPlayer.release(); // release music from memory
  }
@Override
  public boolean isLooping() { // check that music is set to looping or not
    return mediaPlayer.isLooping();
```

```
@Override
  public boolean isPlaying() { // check that music is playing or not
    return mediaPlayer.isPlaying();
  }
  @Override
  public boolean isStopped() { // check that music is stop or not
    return !isPrepared;  // stop mean isPrepared = false
  }
  @Override
  public void pause() { // want to pause music call this method
    if (mediaPlayer.isPlaying()) // check that music is playing or not
      mediaPlayer.pause(); // set music to pause
  }
@Override
  public void play() { // play music by call this method
    if (mediaPlayer.isPlaying())  // check is music is playing or not
      return; // if music is currently playing just return from method
    try {
      synchronized (this) {
        if (!isPrepared)  // check that music is prepare or not
          mediaPlayer.prepare(); // if not prepare the music
        mediaPlayer.start(); // start to play music
      }
    } catch (IllegalStateException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    } //end catch  } // end play()
```

Synchronize block is used for locking the resources from getting accessed by different thread at the same time because isPrepared flag needs to be used for getting "set" on a separate thread by onCompletion() method.

```
@Override
  public void setLooping(boolean isLooping) { // set music to playback when reach the end
    mediaPlayer.setLooping(isLooping); // true to let music playback, false to not let it.
  }
  @Override
  public void setVolume(float volume) { // set volumn of music
    mediaPlayer.setVolume(volume, volume);
  }
  @Override
  public void stop() { // use this method to stop music
    mediaPlayer.stop(); // stop music from playing
    synchronized (this) { // because isPrepared flag set from different thread use this to lock
      isPrepared = false; // set isPrepared flag to false
    }
  }
  @Override
  public void onCompletion(MediaPlayer arg0) {
    synchronized (this) {
      isPrepared = false;
    }
  }
}
```

3.4.5.3 AndroidInput, AccelerometerHandler, KeyboardHandler, TouchHandler

The input interface let developers access to the accelerometer, the touchscreen, and the keyboard, but putting all the code of these modules into a single file is a bit messy. So the input event is separated into handler classes. Then the AndroidInput class will tie all those handlers together.

AccelerometerHandler. This class is self-explanation.

```
public class AccelerometerHandler implements SensorEventListener {
  float accelX; // store value accelerometer of x-axis
  float accelY; // store value accelerometer of y-axis
  float accelZ; // store value accelerometer of z-axis
  public AccelerometerHandler(Context context) {
    SensorManager manager = (SensorManager) context
        .getSystemService(Context.SENSOR_SERVICE);
    if (manager.getSensorList(Sensor.TYPE_ACCELEROMETER).size() != 0) { // check, accel is available
      Sensor accelerometer = manager.getSensorList(
          Sensor.TYPE_ACCELEROMETER).get(0);
      manager.registerListener(this, accelerometer,
          SensorManager.SENSOR_DELAY_GAME); // specify how often sensor should be update
    }
  }
  @Override
  public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // nothing to do here
  }
```

```
@Override
    public void onSensorChanged(SensorEvent event) {
        accelX = event.values[0]; // set accelX to specify value accel of x-axis
        accelY = event.values[1]; // set accelY to specify value accel of y-axis
        accelZ = event.values[2]; // set accelZ to specify value accel of z-axis
    }
    public float getAccelX() {
        return accelX;
    }
    public float getAccelY() {
        return accelY;
    }
    public float getAccelZ() {
        return accelZ;
    }
}
```

Next is the Pool class which reuses touch event and keyboard event because the methods getTouchEvent() and getKeyEvent() which were created earlier will return list of that events. That means developers always create instances of those two classes and store them in lists in the handlers. The Android input system will automatically do a lot of those events, when the key is pressed or the screen is touched, which will consume the resources and slow down the device. So Pool class for reuse is required.

Pool class

```
public class Pool<T> { // generic class, it is like collection class such as ArrayList
   public interface PoolObjectFactory<T> {
      public T createObject(); // interface for return new create object depend on
   }                           // which class implement this Pool (touch or key)
   private final List<T> freeObjects; // ArrayList to store pool object
   private final PoolObjectFactory<T> factory;
   private final int maxSize; // max number pool can hold.
public Pool(PoolObjectFactory<T> factory, int maxSize) {
      this.factory = factory; // use to generate new instances of the type the class holds
      this.maxSize = maxSize;
      this.freeObjects = new ArrayList<T>(maxSize);
   }
public T newObject() {
      T object = null;
      if (freeObjects.size() == 0)
         object = factory.createObject();
      else
         object = freeObjects.remove(freeObjects.size() - 1);
      return object;
   }
```

The newObject() method is responsible for either creates a new instance of the type that the Pool holding, or returns the Pool instance in case it has free one in

the freeObjects ArrayList. This method is used to recycle objects as long as the Pool has some free object stored in the freeObjects list, otherwise the method will creates a new one.

```
public void free(T object) {
    if (freeObjects.size() < maxSize)
        freeObjects.add(object);
  }
}
```

free() method can insert objects that the application is no longer use them as long as the capacity of freeObjects list is not exceed the maxSize. Otherwise Android system will handle them itself.

KeyboardHandler class

```
public class KeyboardHandler implements OnKeyListener { // access to onKey
  boolean[] pressedKeys = new boolean[128];
  Pool<KeyEvent> keyEventPool;
  List<KeyEvent> keyEventsBuffer = new ArrayList<KeyEvent>();
  List<KeyEvent> keyEvents = new ArrayList<KeyEvent>();
```

pressedKeys is an array holding 128 booleans. Those are current states of each key. Next is the Pool that hold instance of KeyEvent class. The third member keyEventsBuffer stores the KeyEvents that have not yet been consumed by our game. The last member store KeyEvents that return when call to getKeyEvents().

```
public KeyboardHandler(View view) {

    PoolObjectFactory<KeyEvent> factory = new
PoolObjectFactory<KeyEvent>() {

      @Override

      public KeyEvent createObject() {

        return new KeyEvent(); // implement createObject() to return

      }                              // newly create KeyEvent instance

    };

    keyEventPool = new Pool<KeyEvent>(factory, 100);

       // create Pool instance with proper PoolObjectFactory for KeyEvent

    view.setOnKeyListener(this); // register onKeyListener with the view

    view.setFocusableInTouchMode(true);

    view.requestFocus(); // make sure that the view will receive key events

  }

@Override

  public boolean onKey(View v, int keyCode, android.view.KeyEvent event) {

    if (event.getAction() == android.view.KeyEvent.ACTION_MULTIPLE)

      return false;

    synchronized (this) {

      KeyEvent keyEvent = keyEventPool.newObject();

      keyEvent.keyCode = keyCode;

      keyEvent.keyChar = (char) event.getUnicodeChar();

      if (event.getAction() == android.view.KeyEvent.ACTION_DOWN) {

        keyEvent.type = KeyEvent.KEY_DOWN;

        if(keyCode > 0 && keyCode < 127)

          pressedKeys[keyCode] = true;

      }
```

```
        if (event.getAction() == android.view.KeyEvent.ACTION_UP) {
            keyEvent.type = KeyEvent.KEY_UP;
            if(keyCode > 0 && keyCode < 127)
                pressedKeys[keyCode] = false;
        }
        keyEventsBuffer.add(keyEvent);
    }
    return false;
}
```

onKey() method is responsible for handle key events. First the method ignores when multiple key is pressed. Next is synchronized block to make sure that none of the members are accessed in parallel because those events are received from the UI thread and read from main thread. In synchronized block the program first fetches a key event instance from Pool. This will let the application recycle instance or create a new one depends on the state of the Pool. Then decode the type of KeyEvent and set it along with the element pressedKey array accordingly. Finally, adding KeyEvent to the keyEvenyBuffer list.

```
public boolean isKeyPressed(int keyCode) {
    if (keyCode < 0 || keyCode > 127)
        return false;
    return pressedKeys[keyCode]; }
```

Using integer to specify the key code and return whether that key is pressed or not. In this method we deal with primitive type -> no synchronized block.

```
public List<KeyEvent> getKeyEvents() {
    synchronized (this) {
        int len = keyEvents.size();
        for (int i = 0; i < len; i++)
            keyEventPool.free(keyEvents.get(i));
        keyEvents.clear();
        keyEvents.addAll(keyEventsBuffer);
        keyEventsBuffer.clear();
        return keyEvents;
    }
}
```

The last method starts with synchronized block because this method will be called from other threads. Next, copying all of the keyEvents methods and store them into Pool, clear all elements in keyEvents and fill it with keyEventsBuffer. Finally the program clears the keyEventsBuffer and returns newly filled keyEvents list to the caller. This method need to be called frequently so that the keyEvents list doesn't full.

TouchHandler interface for single touch or multi touch.

```
public interface TouchHandler extends OnTouchListener {
    public boolean isTouchDown(int pointer);
    public int getTouchX(int pointer);
    public int getTouchY(int pointer);
    public List<TouchEvent> getTouchEvents();
}
```

Creating TouchHandler interface in order to handle both single touch and multi touch; however, only single touch needs to be implemented.

SingleTouchHandler class

```
public class SingleTouchHandler implements TouchHandler {
    boolean isTouched;
    int touchX;
    int touchY;
    Pool<TouchEvent> touchEventPool;
    List<TouchEvent> touchEvents = new ArrayList<TouchEvent>();
    List<TouchEvent> touchEventsBuffer = new ArrayList<TouchEvent>();
    float scaleX;
    float scaleY;
```

First three members are used for storing the current state of the touch screen for one finger, followed by a Pool and two lists holding TouchEvent that looks similar to KeyBoardHandler. The last two members are for scaling the different screen resolution. However, these two scale members are needed for this game that does not need to use OpenGL as graphic renderer.

```
public SingleTouchHandler(View view, float scaleX, float scaleY) {
    PoolObjectFactory<TouchEvent> factory = new
PoolObjectFactory<TouchEvent>() {
      @Override
      public TouchEvent createObject() {
        return new TouchEvent(); // same as keyEvent
      }                          // but instead of KeyEvent return TouchEvent
    };
    touchEventPool = new Pool<TouchEvent>(factory, 100);
    view.setOnTouchListener(this);
    this.scaleX = scaleX; // set pass scaleX value
    this.scaleY = scaleY; // set pass scaleY value  }
```

In the constructor the program register SingleTouchHandler as an OnTouchListener and set up the Pool to recycle TouchEvents and also store scaleX and scaleY parameters.

```
@Override
  public boolean onTouch(View v, MotionEvent event) {
    synchronized(this) {
      TouchEvent touchEvent = touchEventPool.newObject();
      switch (event.getAction()) {
      case MotionEvent.ACTION_DOWN:
        touchEvent.type = TouchEvent.TOUCH_DOWN;
        isTouched = true;
        break;
```

```
        case MotionEvent.ACTION_MOVE:
            touchEvent.type = TouchEvent.TOUCH_DRAGGED;
            isTouched = true;
            break;
        case MotionEvent.ACTION_CANCEL:
        case MotionEvent.ACTION_UP:
            touchEvent.type = TouchEvent.TOUCH_UP;
            isTouched = false;
            break;
        }
        touchEvent.x = touchX = (int)(event.getX() * scaleX);
        touchEvent.y = touchY = (int)(event.getY() * scaleY);
        touchEventsBuffer.add(touchEvent);
        return true;
    }
}
```

onTouch() method handle all touch events such as touch up, touch down or touch drag. In every case they set isTouched and touchEvent properly. Finally, the program scales touchEvent position x and y based on scale parameter and add touchEvent to the touchEventsBuffer list.

```
@Override
    public int getTouchX(int pointer) {
        synchronized(this) {
            return touchX;
        } }
```

```
@Override
    public int getTouchY(int pointer) {
        synchronized(this) {
            return touchY;
        }
    }
@Override
    public boolean isTouchDown(int pointer) {
        synchronized(this) {
            if(pointer == 0)
                return isTouched;
            else
                return false;
        }
    }
```

For method getTouchX(), getTouchY() and isTouchDown() allow developers to retrieve the touchscreen state. The pointer ID uses only zero because it will deal with only single touch. The last one is getTouchEvents() method, it works similarly to the KeyBoardHandler.getKeyEvents() methods.

```
@Override
    public List<TouchEvent> getTouchEvents() {
        synchronized(this) {
            int len = touchEvents.size();
            for( int i = 0; i < len; i++ )
                touchEventPool.free(touchEvents.get(i));
            touchEvents.clear();
            touchEvents.addAll(touchEventsBuffer);
            touchEventsBuffer.clear();
            return touchEvents;
        }
    }
```

AndroidInput class ties all handlers together.

```
public class AndroidInput implements Input {
    AccelerometerHandler accelHandler;
    KeyboardHandler keyHandler;
    TouchHandler touchHandler;
    public AndroidInput(Context context, View view, float scaleX, float scaleY) {
        accelHandler = new AccelerometerHandler(context);
        keyHandler = new KeyboardHandler(view);
        touchHandler = new SingleTouchHandler(view, scaleX, scaleY);
}
```

The class starts by letting its implements the Input interface and creates three members for accelerometer, keyboard and touch. Next, initializing the constructor, which takes a Context, View, scaleX and scaleY.

```
@Override
  public boolean isKeyPressed(int keyCode) {
    return keyHandler.isKeyPressed(keyCode);
  }
  @Override
  public boolean isTouchDown(int pointer) {
    return touchHandler.isTouchDown(pointer);
  }
  @Override
  public int getTouchX(int pointer) {
    return touchHandler.getTouchX(pointer);
  }
  @Override
  public int getTouchY(int pointer) {
    return touchHandler.getTouchY(pointer);
  }
  @Override
  public float getAccelX() {
    return accelHandler.getAccelX();
  }
```

```
@Override
    public float getAccelY() {
        return accelHandler.getAccelY();
    }
    @Override
    public float getAccelZ() {
        return accelHandler.getAccelZ();
    }
    @Override
    public List<TouchEvent> getTouchEvents() {
        return touchHandler.getTouchEvents();
    }
    @Override
    public List<KeyEvent> getKeyEvents() {
        return keyHandler.getKeyEvents();
    }
}
```

The rest of this class is self-explanation. Each method is call to the proper handler, which does all the works for you.

3.4.5.4 GLGraphic class for make use of OpenGL ES

```
public class GLGraphics {
        GLSurfaceView glView;
        GL10 gl;
        GLGraphics(GLSurfaceView glView){
                this.glView = glView;
        }
        public GL10 getGL(){
                return gl;  // return GL10 instance
        }
        void setGL(GL10 gl){
                this.gl = gl; // set GL10 instance
        }
        public int getWidth(){
                return glView.getWidth();  // get width of GLSurfaceView
        }
        public int getHeight(){
                return glView.getHeight(); // get height of GLSurfaceView
        }       } // end GLGraphics
```

The GLGraphic class will keep track of the GL10 instance that has gotten from the GLSurfaceView. GLSurfaceView is used instead of normal View because it allows the application to draw via OpenGL ES. The rest is a few getter and setter methods.

3.4.5.5 GLGame ties everything together. This class is a little bit more complex because it needs to synchronize between the rendering and UI threads.

```
public abstract class GLGame extends Activity implements Game, Renderer {
    enum GLGameState{ // state of GLGame
        Initialized,
        Running,
        Paused,
        Finished,
        Idle
    }
    GLSurfaceView glView;
    GLGraphics glGraphics;
    Audio audio;
    Input input;
    FileIO fileIO;
    Screen screen;
    GLGameState state = GLGameState.Initialized;
    Object stateChanged = new Object(); // create for lock thread
    long startTime = System.nanoTime(); // for calculate delta time
    WakeLock wakeLock;
```

The class extends the Activity class and implements the Game and GLSurfaceView.Renderer interface. It has an enum called GLGameState that keeps track of the state the GLGame instance is currently in. The members of this

class consist of a GLSurfaceView and GLGraphics instance. The class also has Audio, Input, FileIO and Screen instance, which is needed for writing the game. The state member keeps track of the state via one of the GLGameState enums. The stateChanged member is an object that will be used to synchronize the UI thread and the rendering thread. Finally the class has a member to keep track of the delta time between each frames and a WakeLock that use to keep the screen from dimming.

```
@Override
        public void onCreate(Bundle savedInstanceState){
                super.onCreate(savedInstanceState);
                requestWindowFeature(Window.FEATURE_NO_TITLE); // set for full screen

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                                WindowManager.LayoutParams.FLAG_FULLSCREEN);
                glView = new GLSurfaceView(this); // instantiate GLSurfaceView
                glView.setRenderer(this);
                setContentView(glView); // set GLSurfaceView as a content view
                glGraphics = new GLGraphics(glView);
                fileIO = new AndroidFileIO(getAssets());
                audio = new AndroidAudio(this);
                input = new AndroidInput(this, glView, 1, 1); // scale to 1
                PowerManager powerManager = (PowerManager)
getSystemService(Context.POWER_SERVICE); // get powerManager for dimming
                wakeLock = powerManager.newWakeLock(PowerManager.FULL_WAKE_LOCK,
"GLGame");      // set for no dimming
        }
```

The onCreate() method is performed all the setup. The work starts by making activity become full-screen and instantiate the GLSurfaceView, setting it as the content View. Then this method also instantiate all other game modules such as AndroidFileIO or AndroidInput classes. The scale touch coordinates values are both 1 because OpenGL ES will used to handle this. The last thing is creating WakeLock instance.

```
@Override
    public void onResume(){
        super.onResume();
        glView.onResume(); // let GLSurfaceView start rendering thread
        wakeLock.acquire(); // acquire the WakeLock
    }
    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config){
        glGraphics.setGL(gl);
        synchronized(stateChanged){ // lock with stateChanged
            if(state == GLGameState.Initialized)
                screen = getStartScreen();
            state = GLGameState.Running;
            screen.resume();
            startTime = System.nanoTime();
        }
    }
    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height){
    } // this method is a stub. Nothing to do here
```

The onSurfaceCreate() method is invoked on the rendering thread. If application is started for the first time, the getStartScreen() will be called to return the starting screen of the game. If not then it is means the game just resume from paused state. In any case the GLGameState is set to Running and then call the current Screen's resume() method. This method also keeps track of the current time so the delta time can be calculated. The synchronization is used since the members that have been manipulated within the synchronized block can be manipulated in the onPause() method on the UI thread. That is something which has to prevent, so an object as a lock is used.

```
@Override
public void onDrawFrame(GL10 gl){
        GLGameState state = null;
        synchronized(stateChanged){
                state = this.state;
        }
```

The onDrawFrame() method is called by the rendering thread as often as possible. Here the state game is checked which state is currently in and react to it accordingly. As the state can be set on the onPause() method on the UI thread, synchronize is used to prevent the access to it.

```
        if(state == GLGameState.Running){
                float deltaTime = (System.nanoTime() - startTime) / 1000000000.0f;
                startTime = System.nanoTime();
                screen.update(deltaTime); // call to screen update
                screen.present(deltaTime); // call to screen present
        }
        if(state == GLGameState.Paused){
                screen.pause();
                synchronized(stateChanged){
                        this.state = GLGameState.Idle;
                        stateChanged.notifyAll();
                }
        }
        if(state == GLGameState.Finished){
                screen.pause();
                screen.dispose();
                synchronized(stateChanged){
                        this.state = GLGameState.Idle;
                        stateChanged.notifyAll();
                }
        }
}
```

If the game is running the method will calculates and tells the current Screen to update and present of itself.

If the game is paused, the current Screen is told to pause itself as well. Then the state is changed to GLGameState.Idle, indicating that the pause request is received from the UI thread. Since the program wait for this to happen in the onPause() method in the UI thread, the UI thread will be notified that it can now really pause the application.

If the Activity is being closed, GLGameState.Finished is reacted. In this case the current Screen is told to pause and dispose of itself, and then send another notification to the UI thread, which waits for the rendering thread to properly shutting down.

```
@Override
public void onPause(){
        synchronized(stateChanged){
                if(isFinishing()) // call to know whether Activity going to be destroyed or not
                        state = GLGameState.Finished;
                else
                        state = GLGameState.Paused;
                while(true){
                        try{
                                stateChanged.wait(); // wait for notification
                                break;
                        }catch(InterruptedException e){ }
                }// end while } // end synchronized block
                wakeLock.release();
                glView.onPause();
                super.onPause();  } // end onPause()
```

The onPause() method is a usual Activity notification method that is called on the UI thread when the Activity is paused. Depending on whether the application is closed or paused, the program sets accordingly and waits for the rendering thread to process the new state. Finally the method releases the WakeLock and tells the GLSurfaceView and the Activity to pause themselves.

```
public GLGraphics getGLGraphics(){
        return glGraphics;
}
@Override
public Input getInput(){
        return input;
}
@Override
public FileIO getFileIO(){
        return fileIO;
}
@Override
public Audio getAudio(){
        return audio;
}
```

The getGLGraphics(), getFileIO(), getInput() and getAudio() methods will return the respective instances to the caller. The caller will always be one of our Screen implementations of our game.

```
@Override
public void setScreen(Screen screen){
        if(screen == null)
                throw new IllegalArgumentException("Screen must not be null");
        this.screen.pause(); // set current screen to pause
        this.screen.dispose(); // set current screen to dispose
        screen.resume(); // start new screen
        screen.update(0); // update new screen once
        this.screen = screen; // set current screen = new screen
}
public Screen getCurrentScreen(){
        return screen; // return current active screen
} // end getCurrentScreen()   } // end GLGame class
```

The setScreen() method start off with null-checking because a null Screen cannot be allowed here. Next the current Screen is told to pause and dispose of itself so it can make room for the new Screen. Then the new Screen is asked to resume and update itself once with a delta time of zero. Finally the method sets the Screen member to the new Screen.

The last method getCurrentScreen() is simply returns the currently active Screen of the game.

3.4.5.6 Example creating simple triangle

```
public class FirstTriangleTest extends GLGame {
   @Override
   public Screen getStartScreen() {
      return new FirstTriangleScreen(this);
   }
```

The FirstTrianagleTest class derives from GLGame and has to implement the Game.getStartScreen() method. In that method a new FirstTriangleScreen is created, which will then be frequently called to update and present itself by the GLGame. Thus, OpenGL ES methods can be used in the constructor of the FirstTriangleScreen class.

```
class FirstTriangleScreen extends Screen {
    GLGraphics glGraphics;
    FloatBuffer vertices;
    public FirstTriangleScreen(Game game) {
       super(game);
       glGraphics = ((GLGame)game).getGLGraphics(); // get GLGraphics instance
       ByteBuffer byteBuffer = ByteBuffer.allocateDirect(3 * 2 * 4);
       byteBuffer.order(ByteOrder.nativeOrder()); // make sure byte order is correct
       vertices = byteBuffer.asFloatBuffer(); // convert byte buffer to float buffer
       vertices.put( new float[] {   0.0f,   0.0f,
                                     319.0f,   0.0f,
                                     160.0f, 479.0f});
       vertices.flip();  // set pointer in vertices to zero }
```

The FirstTriangleScreen class holds two members: a GLGraphics instance, which it have to cast to a GLGame instance so the GLGame.getGLGraphics() method can be used and FloatBuffer, which stores the 2D position of the three vertices of our triangle. In constructor the GLGraphics instance is fetched from the GLGame. The program allocate byte based on (number of vertices) *(number of coordinate points)*(bytes use to store float) = 3 * 2 * 4. Then developers specify 3 points for triangle and call flip() method to move pointer back to starting position.

```
@Override
    public void present(float deltaTime) {
      GL10 gl = glGraphics.getGL();
      gl.glViewport(0, 0, glGraphics.getWidth(), glGraphics.getHeight());
      gl.glClear(GL10.GL_COLOR_BUFFER_BIT);
      gl.glMatrixMode(GL10.GL_PROJECTION);
      gl.glLoadIdentity();
      gl.glOrthof(0, 320, 0, 480, 1, -1); // set camera position
      gl.glColor4f(1, 0, 0, 1); // set color to red
      gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
      gl.glVertexPointer( 2, GL10.GL_FLOAT, 0, vertices); // give vertices to OpenGL
      gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
    }
```

In the present() method is where all graphics which will be shown place here. It start off by set the viewport, clear the screen set the projection matrix so that developers can work in our custom coordinate system, set default vertex color (red in this case), specify that our vertices will have positions, tell OpenGL ES where it can find those vertex positions and finally render the red triangle.

```
@Override
    public void update(float deltaTime) { // all logic of game put in this method
      game.getInput().getTouchEvents();
      game.getInput().getKeyEvents();
    }
    @Override
    public void pause() {
    }
    @Override
    public void resume() {
    }
    @Override
    public void dispose() {
    }
  } }
```

The rest of the class is just prototype code. In the update() method we make sure that our event buffers don't get filled up. The rest of the code does nothing.

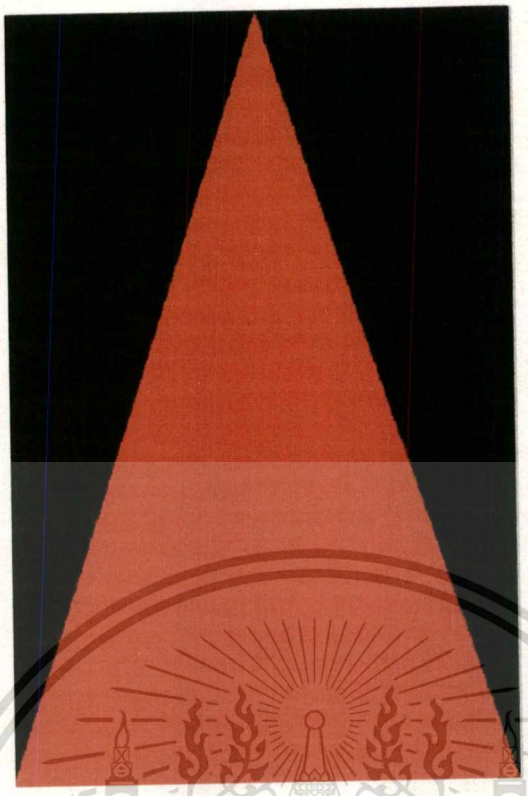


Figure 3.53 Show our red triangle.

As you can see only draw red triangle in Open GL ES needs a lot of things to create. If it is an actual game, it will be a huge mess. That makes next part a couple of class were created to help you deal with Open GL ES easier.

3.4.5.7 Utility class, to help us deal with Open GL ES easier.

3.4.5.7.1 Vector2 class, this class is used to represent positions, directions, distances and velocities in our 2D game.

```
public class Vector2 {
   public static float TO_RADIANS = (1 / 180.0f) * (float)Math.PI;
   public static float TO_DEGREES = (1 / (float)Math.PI) * 180;
   public float x, y;
```

The class starts off by defining two static constant, TO_RADIANS and TO_DEGREES for converting and the two members x and y are designed for storing the components of the vector.

```
public Vector2() {
  }
  public Vector2(float x, float y) {
    this.x = x;
    this.y = y;
  }
  public Vector2(Vector2 other) {
    this.x = other.x;
    this.y = other.y;
  }
```

Next a couple of constructors are created for different instantiate.

```
public Vector2 cpy() {
    return new Vector2(x, y); }
public Vector2 set(float x, float y) {
    this.x = x;
    this.y = y;
    return this; }
  public Vector2 set(Vector2 other) {
    this.x = other.x;
    this.y = other.y;
    return this; }
```

Next is cpy() method that will create a duplicate instance of the current vector and return it. The set() method allow the class to set the x and y components of a vector, based on either two float arguments or other vectors.

```
public Vector2 add(float x, float y) {
    this.x += x;
    this.y += y;
    return this;
}
public Vector2 add(Vector2 other) {
    this.x += other.x;
    this.y += other.y;
    return this;
}
public Vector2 sub(float x, float y) {
    this.x -= x;
    this.y -= y;
    return this;
}
public Vector2 sub(Vector2 other) {
    this.x -= other.x;
    this.y -= other.y;
    return this;
}
```

The add() and sub() methods also come in two options: in the first case they work with two floats arguments and in the other case they take another Vector2 instance.

```
public Vector2 mul(float scalar) {
    this.x *= scalar;
    this.y *= scalar;
    return this;
  }
public float len() {
    return FloatMath.sqrt(x*x + y*y);
  }
  public Vector2 nor() {
    float len = len();
    if(len!=0) {
      this.x /= len;
      this.y /= len;
    }
    return this;
  }
```

The mul() method just multiplies the x and y components. The len() method is to calculate the length of the vector and the nor() method normalizes the vector to unit length.

```
public float angle() {
    float angle = (float)Math.atan2(y, x) * TO_DEGREES;
    if(angle < 0)
       angle += 360;
    return angle;
}
public Vector2 rotate(float angle) {
    float rad = angle * TO_RADIANS;
    float cos = FloatMath.cos(rad);
    float sin = FloatMath.sin(rad);
    float newX = this.x * cos - this.y * sin;
    float newY = this.x * sin + this.y * cos;
    this.x = newX;
    this.y = newY;
    return this;
}
```

The angle() method calculates the angle between the vector and the x-axis using the atan2() method from Math class provided by JAVA. The rotate() method simply rotates the vector around the origin by the giving angle.

```
public float dist(Vector2 other) {
    float distX = this.x - other.x;
    float distY = this.y - other.y;
    return FloatMath.sqrt(distX*distX + distY*distY);
}
```

```
public float dist(float x, float y) {
    float distX = this.x - x;
    float distY = this.y - y;
    return FloatMath.sqrt(distX*distX + distY*distY);
  }
```

Finally the last two methods are used to calculate the distance between this and another vector.

3.4.5.7.2 Rectangle class, for creating bounding shape for each object in the game.

```
public class Rectangle {
  public final Vector2 lowerLeft;
  public float width, height;
  public Rectangle(float x, float y, float width, float height) {
    this.lowerLeft = new Vector2(x,y);
    this.width = width;
    this.height = height;
  }
}
```

The class stores the lower-left corner's position in a Vector2 class and the width and height in two floats.

3.4.5.7.3 OverlapTester class, for checking collision.

```
public class OverlapTester {

  public static boolean overlapRectangles(Rectangle r1, Rectangle r2) {

    if(r1.lowerLeft.x < r2.lowerLeft.x + r2.width &&

      r1.lowerLeft.x + r1.width > r2.lowerLeft.x &&

      r1.lowerLeft.y < r2.lowerLeft.y + r2.height &&

      r1.lowerLeft.y + r1.height > r2.lowerLeft.y)

       return true;

    else

       return false;

  }
```

The overlapRectangles() method will check for overlap between two rectangles. If all these conditions are met, then two rectangles overlap.

```
public static boolean pointInRectangle(Rectangle r, Vector2 p) {

    return r.lowerLeft.x <= p.x && r.lowerLeft.x + r.width >= p.x &&

        r.lowerLeft.y <= p.y && r.lowerLeft.y + r.height >= p.y;

  }

  public static boolean pointInRectangle(Rectangle r, float x, float y) {

    return r.lowerLeft.x <= x && r.lowerLeft.x + r.width >= x &&

        r.lowerLeft.y <= y && r.lowerLeft.y + r.height >= y;

  }

}
```

Finally the pointInRectangle() methods, these methods are called based on the given arguments either in positions or vector . They are used for checking a touch point with the game object.

3.4.5.7.4 Texture class, this method is responsible for load a bitmap from /assets and create a texture object from it.

```
public class Texture {
  GLGraphics glGraphics;
  FileIO fileIO;
  String fileName;
  int textureId;
  public Texture(GLGame glGame, String fileName) {
    this.glGraphics = glGame.getGLGraphics();
    this.fileIO = glGame.getFileIO();
    this.fileName = fileName;
    load();
  }
```

Texture class defines a set of attributes that store instances which will be used in this class. The textureId attribute is used for storing id of the texture because Open GL ES is C class. It cannot return java object so instead of returning java object, it uses the id to map.

In the contructor, it takes GLGame instance to make use of GLGraphics , FileIO, and fileName to know where texture is stored then call load() method.

```
private void load() {
    GL10 gl = glGraphics.getGL();
    int[] textureIds = new int[1];
    gl.glGenTextures(1, textureIds, 0);
    textureId = textureIds[0];
    InputStream in = null;
    try {
      in = fileIO.readAsset(fileName); // read byte from asset with name of file = fileName
      Bitmap bitmap = BitmapFactory.decodeStream(in); // decode byte in bitmap format
      gl.glBindTexture(GL10.GL_TEXTURE_2D, textureId); // bind texture with id
      GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmap, 0); // map image to texture
      gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_NEAREST);
      gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_NEAREST);
      gl.glBindTexture(GL10.GL_TEXTURE_2D, 0); // unbind texture
      bitmap.recycle(); // release memory
    } catch(IOException e) {
      throw new RuntimeException("Couldn't load texture '" + fileName + "'", e);
    } finally {
      if(in != null)
        try { in.close(); } catch (IOException e) { }
    }
  }
```

The gl.glGenTextures() is the command to generate id for texture. In try...catch block the program first loads the bitmap then sets id for texture, maps it with the image

and then sets filter to it in case the rendering is not pixel perfect. Next the texture is bind and unbind then call bitmap.recycle() to release the memory.

```
public void reload() {
    load();
    bind();
    gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_NEAREST);
    gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_NEAREST);
    glGraphics.getGL().glBindTexture(GL10.GL_TEXTURE_2D, 0);
  }
public void bind() {
    GL10 gl = glGraphics.getGL();
    gl.glBindTexture(GL10.GL_TEXTURE_2D, textureId);
  }
  public void dispose() {
    GL10 gl = glGraphics.getGL();
    gl.glBindTexture(GL10.GL_TEXTURE_2D, textureId);
    int[] textureIds = { textureId };
    gl.glDeleteTextures(1, textureIds, 0);
  }
}
```

The reload() method is called when game resumes again from minimize state. The bind() method is used for binding texture. The last dispose() method is called when the texture is no longer uses.

3.4.5.7.5 Vertices class, this class uses to hold a maximum number of vertices and indices to be used for rendering. It also takes care of enabling all the states needed to rendering such as coordinates, colors and textures.

```
public class Vertices {
    final GLGraphics glGraphics;
    final boolean hasColor;
    final boolean hasTexCoords;
    final int vertexSize;
    final FloatBuffer vertices;
    final ShortBuffer indices
```

The Vertices class has a reference to the GLGraphics instance, that hold GL10 instance. The class also stores whether the vertices have colors and texture coordinates and also stores a FloatBuffer that holds our vertices and a ShortBuffer that holds the optional indices.

```
public Vertices(GLGraphics glGraphics, int maxVertices, int maxIndices, boolean hasColor,
boolean hasTexCoords) {
    this.glGraphics = glGraphics;
    this.hasColor = hasColor;
    this.hasTexCoords = hasTexCoords;
    this.vertexSize = (2 + (hasColor?4:0) + (hasTexCoords?2:0)) * 4;
    ByteBuffer buffer = ByteBuffer.allocateDirect(maxVertices * vertexSize);
    buffer.order(ByteOrder.nativeOrder());
    vertices = buffer.asFloatBuffer();
```

```
    if(maxIndices > 0) {
        buffer = ByteBuffer.allocateDirect(maxIndices * Short.SIZE / 8);
        buffer.order(ByteOrder.nativeOrder());
        indices = buffer.asShortBuffer();
    } else {
        indices = null;
    }
}
```

In the constructor, the class specifies how many vertices and indices the Vertices instance can hold maximally, as well as whether the vertices have colors or texture coordinates. Next the members is set accordingly and the buffers is instantiated

```
public void setVertices(float[] vertices, int offset, int length) {
    this.vertices.clear();
    this.vertices.put(vertices, offset, length);
    this.vertices.flip();
}
public void setIndices(short[] indices, int offset, int length) {
    this.indices.clear();
    this.indices.put(indices, offset, length);
    this.indices.flip(); }
```

Next are the setVertices() and setIndices() methods. It is set accordingly.

```
public void draw(int primitiveType, int offset, int numVertices) {
    GL10 gl = glGraphics.getGL();
    gl.glEnableClientState(GL10.GL_VERTEX_ARRAY); // enable coordinate array
    vertices.position(0); // tell Open GL ES to start finding coordinate at position 0
    gl.glVertexPointer(2, GL10.GL_FLOAT, vertexSize, vertices); // tell OpenGL ES coordinates position has 2
    if(hasColor) { // if vertices enable color tell OpenGL ES to enable color vertex.
      gl.glEnableClientState(GL10.GL_COLOR_ARRAY); // enable color array
      vertices.position(2); // tell OpenGL ES to start finding color at position 2 (0,1) for coordinate
      gl.glColorPointer(4, GL10.GL_FLOAT, vertexSize, vertices); // tell OpenGL ES color position has 4
    }
    if(hasTexCoords) { // if vertices enable color tell OpenGL ES to enable texture vertex.
      gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY); // enable texture array
      vertices.position(hasColor?6:2); // if it has color start at position 6 otherwise start at position 2
      gl.glTexCoordPointer(2, GL10.GL_FLOAT, vertexSize, vertices); // tell OpenGL ES texture position has 2
    }
    if(indices!=null) {
      indices.position(offset); // if indices is specify draw by indices
      gl.glDrawElements(primitiveType, numVertices, GL10.GL_UNSIGNED_SHORT, indices);
    } else {
      gl.glDrawArrays(primitiveType, offset, numVertices); // draw by vertices
    }
    if(hasTexCoords)
      gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
    if(hasColor)
      gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
  } }
```

The final method of the Vertices class is draw() method. It takes the type of the primitive, the offset into the vertices buffer and the number of vertices to use for rendering. Depending on whether the vertices have colors and texture coordinates, the class will enable the relevant OpenGL ES states and tell OpenGL ES where to find the data. Depending on whether indices are used or not, the program either call glDrawElements() or glDrawArrays() with the parameters passed to the method.

3.4.5.7.6 Camera2D class, to set view port, projection matrix and translate touch coordinates to world coordinates.

```
public class Camera2D {
  public final Vector2 position;
  public float zoom;
  public final float frustumWidth; // the width of camera
  public final float frustumHeight; // the height of camera
  final GLGraphics glGraphics;
```

Here the camera's position, frustum width and height and zoom factor are stored as members. The class also needs to refer to GLGraphics so it can get the width and height of the screen in pixels for transforming touch coordinates to world coordinates.

```
public Camera2D(GLGraphics glGraphics, float frustumWidth, float frustumHeight) {
    this.glGraphics = glGraphics;
    this.frustumWidth = frustumWidth;
    this.frustumHeight = frustumHeight;
    this.position = new Vector2(frustumWidth / 2, frustumHeight / 2);
    this.zoom = 1.0f;
}
```

In the constructor, the class takes a GLGraphics instance and the frustum's width and height at the zoom factor 1 as parameters. It stores them and initializes the position of the camera to look at the center of the screen.

```
public void setViewportAndMatrices() {
    GL10 gl = glGraphics.getGL();
    gl.glViewport(0, 0, glGraphics.getWidth(), glGraphics.getHeight());
    gl.glMatrixMode(GL10.GL_PROJECTION);
    gl.glLoadIdentity();
    gl.glOrthof(position.x - frustumWidth * zoom / 2,
            position.x + frustumWidth * zoom/ 2,
            position.y - frustumHeight * zoom / 2,
            position.y + frustumHeight * zoom/ 2,
            1, -1); // we working in 2D so for z-axis just leave it there
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();
}
```

The setViewPortAndMatrices() method sets the viewport and to span the whole screen and sets the projection matrix in accordance with the camera's parameters.

```
public void touchToWorld(Vector2 touch) {
    touch.x = (touch.x / (float) glGraphics.getWidth()) * frustumWidth * zoom;
    touch.y = (1 - touch.y / (float) glGraphics.getHeight()) * frustumHeight * zoom;
    touch.add(position).sub(frustumWidth * zoom / 2, frustumHeight * zoom / 2);
  }
}
```

The touchToWorld() method takes a Vector2 instance containing touch coordinates and transforms that coordinates to world space.

3.4.5.7.7 TextureRegion class, cut down some parts of texture into sprites.

```
public class TextureRegion {
  public final float u1, v1;
  public final float u2, v2;
  public final Texture texture;
  public TextureRegion(Texture texture, float x, float y, float width, float height) {
    this.u1 = x / texture.width;
    this.v1 = y / texture.height;
    this.u2 = this.u1 + width / texture.width;
    this.v2 = this.v1 + height / texture.height;
    this.texture = texture;
  }
}
```

The TextureRegion class stores the texture coordinates of the top-left corner(u1,v1) and bottom-right corner(u2,v2) of the region in texture coordinates. The constructor takes a Texture and the top-left corner, as well as width and height of the region, in pixel coordinates.

3.4.5.7.8 SpriteBatcher class, to draw sprites that are created from TextureRegion class to the game.

```
public class SpriteBatcher {
        final float[] verticesBuffer;
        int bufferIndex;
        final Vertices vertices;
        int numSprites;
```

The member verticesBuffer is the temporary float array that is created to store the vertices of the sprites of the current batch in. The member bufferIndex indicates in the float array where it should start to write the next vertices. The member vertices is the Vertices instance which used to render the batch. The last member numSprites holds the number which is drawn so far in the current batch.

```
public SpriteBatcher(GLGraphics glGraphics, int maxSprites){
                this.verticesBuffer = new float[maxSprites * 4 * 4];
                this.vertices = new Vertices(glGraphics, maxSprites * 4, maxSprites * 6,
false, true); // choose to disable color vertices for not.
                this.bufferIndex = 0;
                this.numSprites = 0;
```

```
            short[] indices = new short[maxSprites * 6];
            int len = indices.length;
            short j = 0;
            for(int i = 0; i < len; i += 6, j += 4){
                indices[i + 0] = (short)(j + 0);
                indices[i + 1] = (short)(j + 1);
                indices[i + 2] = (short)(j + 2);
                indices[i + 3] = (short)(j + 2);
                indices[i + 4] = (short)(j + 3);
                indices[i + 5] = (short)(j + 0);
            }
            vertices.setIndices(indices, 0, indices.length);
        }
```

In the constructor, it contains two arguments: the GLGraphics instance which is needed for creating the vertices instance and the maximum number of sprites that the batcher can render in one batch. The first thing is creating the float array. It has to be four vertices per sprite and each vertices takes up four floats. Next the class creates the Vertices instance and specify the parameters. Then initialize the bufferIndex and numSprites members to zero. Next indices is created for our Vertices instance. The class needs to do this only once because the indices will never be changed. It precomputes those and stores them in the Vertices instance.

```
public void beginBatch(Texture texture){
        texture.bind();
        numSprites = 0;
        bufferIndex = 0;
}
```

Next up is the beginBatch() method. It binds the texture and resets the numSprites and bufferIndex members so the first sprite's vertices will get inserted at the front of the verticesBuffer float array.

```
public void endBatch(){
        vertices.setVertices(verticesBuffer, 0, bufferIndex);
        vertices.bind();
        vertices.draw(GL10.GL_TRIANGLES, 0, numSprites * 6);
        vertices.unbind();
}
```

The next method is endBatch(): it is called to finalize and draw the current batch. It transfers the vertices which are defined for this batch from the float array to the Vertices instance. The rest is binding the Vertices instance, drawing the pictures and unbinding the Vertices.

```
public void drawSprite(float x, float y, float width, float height, TextureRegion region){
        float halfWidth = width / 2;
        float halfHeight = height / 2;}
```

```
        float x1 = x - halfWidth;
        float y1 = y - halfHeight;
        float x2 = x + halfWidth;
        float y2 = y + halfHeight;
        verticesBuffer[bufferIndex++] = x1;                // start vertex 1
        verticesBuffer[bufferIndex++] = y1;
        verticesBuffer[bufferIndex++] = region.u1;
        verticesBuffer[bufferIndex++] = region.v2;
        verticesBuffer[bufferIndex++] = x2;                // start vertex 2
        verticesBuffer[bufferIndex++] = y1;
        verticesBuffer[bufferIndex++] = region.u2;
        verticesBuffer[bufferIndex++] = region.v2;
        verticesBuffer[bufferIndex++] = x2;                // start vertex 3
        verticesBuffer[bufferIndex++] = y2;
        verticesBuffer[bufferIndex++] = region.u2;
        verticesBuffer[bufferIndex++] = region.v1;
        verticesBuffer[bufferIndex++] = x1;                // start vertex 4
        verticesBuffer[bufferIndex++] = y2;
        verticesBuffer[bufferIndex++] = region.u1;
        verticesBuffer[bufferIndex++] = region.v1;
        numSprites++;
    }
}
```

The final is drawSprite() method. It takes the x- and y- coordinates of the center of the sprite, its width and height, and the TextureRegion it maps to.

The method's responsibility is to add four vertices to the float array starting at the current bufferIndex. These four vertices form a texture-mapped rectangle. The method calculates the position of the bottom-left corner (x1,y1) and the top-right corner (x2,y2) and use these four variables to construct the vertices, together with the texture coordinates from the TextureRegion. The vertices are counterclockwise order, starting from the bottom-left vertex. Once they are added to the float array, it increments the numSprites counter and waits for either another sprite to be added or for the batch to be finalized.

3.4.5.8 Creating Example MainMenu Screen.

3.4.5.8.1 Creating the assets.

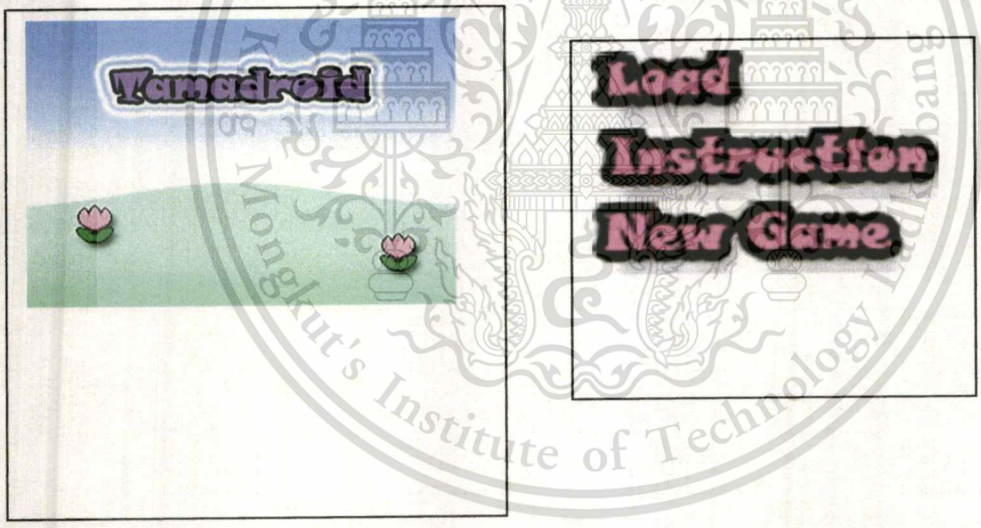


Figure 3.54 The background imange and the main menu entries.

The background image has the size of 480 * 320 but for OpenGL, it need to create at size 512*512 by adding white space to its.

The mainmenu has size of 128*128 with Load (63*27), New Game (117*25) and Instruction (132*25).

The music and sound, we provide only click sound.

Figure 3.55 Click sound file.

Then copy all these three files and store into folder /assets in Android project.

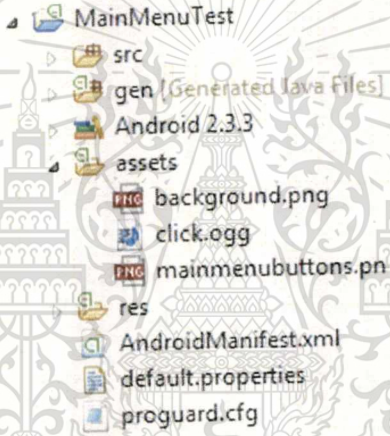


Figure 3.56 Assets folder.

3.4.5.8.2 Coding the program

It start off by creating Assets.java for loading all resources.

```
public class Assets{

        public static Texture bg;

        public static TextureRegion bgRegion;

        public static Texture buttons;

        public static TextureRegion newgame;

        public static TextureRegion load

        public static TextureRegion instruction;
```

```
public static Sound clickSound;
```

The class holds references to all the Texture, TextureRegion and Sound instances that are needed for the game.

```
public static void load(GLGame game){
        bg = new Texture(game, "background.png");
        bgRegion = new TextureRegion(bg, 0, 0, 480, 320);
        buttons = new Texture(game, "mainmenubuttons.png");
        newgame = new TextureRegion(buttons, 0, 27+1+25+1, 117, 25);
        load = new TextureRegion(buttons, 0, 0, 63, 27);
        instruction = new TextureRegion(buttons, 0, 27+1, 132, 25);
        clickSound = game.getAudio().newSound("click.ogg");
}
```

The load() method, which will be called once at the start of the game, is responsible for loading all assets of our game. It loads the background image and creates a corresponding TextureRegion for it. Next it loads the texture atlas and creates all necessary TextureRegions for all buttons. The last thing is we load click sound from the assets.

```
public static void playSound(Sound sound){
        sound.play(1);
}
}
```

The final method is a helper method that let we use it to play back audio.

StartGame.java

Next Main Activity class's name StartGame.java is defined, this class is responsible for an entry point of our game.

```
public class StartGame extends GLGame {
        @Override
        public Screen getStartScreen() {
                return new MainMenuScreen(this);
        }
        @Override
        public void onSurfaceCreated(GL10 gl, EGLConfig config){
                super.onSurfaceCreated(gl, config);
                Assets.load(this);
        }
}
```

The class derive from GLGame and implement the getStartScreen() method, which returns a MainMenuScreen instance and it needs to override onSurfaceCreated(), which is called each time the OpenGL ES context is created and it loads all of the assets here.

GLScreen.java class for fixing problem for casting all the reference to GLGraphics.

```
public abstract class GLScreen extends Screen {
    protected final GLGraphics glGraphics;
    protected final GLGame glGame;
    public GLScreen(Game game) {
        super(game);
        glGame = (GLGame)game;
        glGraphics = ((GLGame)game).getGLGraphics();
    }
}
```

The class stores the GLGraphics and GLGame instances that do the casting for you.

MainMenuScreen.java

Next is MainMenuScreen.java, this is the screen which is returned by calling getStartScreen() method from StartGame.java, so it is the first screen the player will see.

```
public class MainMenuScreen extends GLScreen{
        Camera2D guiCam;
        SpriteBatcher batcher;
        Rectangle newgameBounds;
        Rectangle loadBounds;
        Rectangle instructionBounds;
        Vector2 touchPoint;
```

The class derive from GLScreen so GLGraphics instance can be accessed more easily. The class creates a couple member Camare2D, SpriteBatcher to render background and UI elements. The class also creates Rectangles to determine if the user touched a UI element and Vector2 touchPoint to store and transform the touch coordinates to world coordinates.

```
public MainMenuScreen(Game game) {
        super(game);
        guiCam = new Camera2D(glGraphics, 480, 320);
        batcher = new SpriteBatcher(glGraphics, 100);
        newgameBounds = new Rectangle(240 - 58, 130 - 13, 117, 25);
        loadBounds = new Rectangle(240 - 32, 90 - 14, 63, 27);
        instructionBounds = new Rectangle(240 - 66, 50 - 13, 132, 25);
        touchPoint = new Vector2();
}
```

In the constructor, it simply sets up all the members. The Camera2D instance will allow the game to work in the target resolution of 480 * 320 pixels. The batcher will draw at most 100 pictures in one batch. The Rectangles which set up for each UI elements are given in pixel coordinates.

```
@Override
    public void update(float deltaTime) {
        List<TouchEvent> touchEvents = game.getInput().getTouchEvents();
        game.getInput().getKeyEvents();
```

```
int len = touchEvents.size();
        for(int i = 0; i < len; i++){
                TouchEvent event = touchEvents.get(i);
                if(event.type == TouchEvent.TOUCH_UP){
                        touchPoint.set(event.x, event.y);
                        guiCam.touchToWorld(touchPoint);
                        if(OverlapTester.pointInRectangle(newgameBounds, touchPoint)){
                                Assets.playSound(Assets.clickSound);
                                return;
                        }
                        if(OverlapTester.pointInRectangle(loadBounds, touchPoint)){
                                Assets.playSound(Assets.clickSound);
                                return;
                        }
                        if(OverlapTester.pointInRectangle(instructionBounds, touchPoint)){
                                Assets.playSound(Assets.clickSound);
                                return;
                        }       }// end if  } // end for  } // end update()
```

Next is the update() method. It loops through the TouchEvents that returned by the Input instance and check for touch-up events. In case it has such an event, it first translates the touch coordinates to world coordinates by using Camera2D. Once it has our world touching point, it can check that point against the rectangles of the UI elements. If NewGame, Load or Instruction was hit, the click sound will be played via Assets.playSound() method.

```
@Override
public void present(float deltaTime) {
        GL10 gl = glGraphics.getGL();
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT);
        guiCam.setViewportAndMatrices();
        gl.glEnable(GL10.GL_TEXTURE_2D);
        batcher.beginBatch(Assets.bg);
        batcher.drawSprite(240, 160, 480, 320, Assets.bgRegion);
        batcher.endBatch();
        gl.glEnable(GL10.GL_BLEND);
        gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
        batcher.beginBatch(Assets.buttons);
        batcher.drawSprite(240, 130, 117, 25, Assets.newgame);
        batcher.drawSprite(240, 90, 63, 27, Assets.load);
        batcher.drawSprite(240, 50, 132, 25, Assets.instruction);
        batcher.endBatch();
        gl.glDisable(GL10.GL_BLEND);
}
```

In the present() method it first clears the screen, sets up the projection matrices via the camera and renders the background and UI elements. Since the UI elements which are provided have transparent backgrounds, blending is enabled temporarily to render them.

```
    @Override
    public void pause() {
    }
    @Override
    public void resume() {
    }
    @Override
    public void dispose() {
    }
}
```

The rest of the method is just a stub, nothing to do with it.

3.4.5.8.3 Show the result. Click on each button will produce click sound.

Figure 3.57 Main menu screen.

3.4.6 Speech Recognition Code

Speech recognition part is used to hold-speak method which will give the user to hold the button while holding it, the device will record the sound and recognize each partial results. When the user releases the button, the device will stop and final result will be represented at this time.

Starting off by defining the manifest file to gain permission to record audio and write file to external storage.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.cmu.pocketsphinx.demo"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name" android:debuggable="true'
      <activity android:name=".PocketSphinxDemo"
                android:label="@string/app_name">
          <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
      </activity>

  </application>
  <uses-sdk android:minSdkVersion="8" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
<uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
</manifest>
```

Figure 3.58 Configure Android manifest.

Next, creating the simple layout and button.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:background="@color/white">

<EditText android:id="@+id/EditText01" android:layout_width="fill_parent"
    android:text="Text goes here..." android:contentDescription="Recognition results"
    android:layout_height="wrap_content" android:layout_weight="1"></EditText>

<Button android:id="@+id/Button01" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:layout_weight="0"
    android:layout_gravity="center_horizontal" android:text="Hold and Speak">]
    </Button>

</LinearLayout>
```

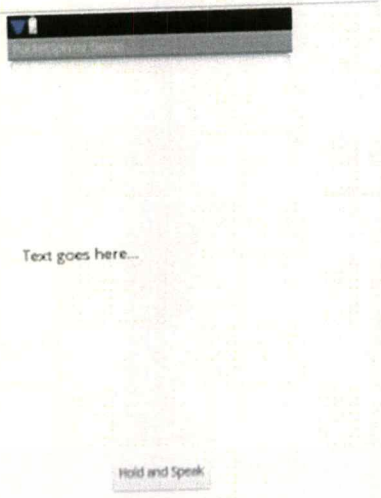


Figure 3.59 Creating layout

3.4.6.1 Create RecognitionListener interface, this class is used for speech recognition callbacks; this is a cut down version of android.speech.RecognitionListener.

```
public interface RecognitionListener {
        abstract void onPartialResults(Bundle b); // call when have partial result
        abstract void onResults(Bundle b); // call when have final result
        abstract void onError(int err);
}
```

3.4.6.2 PocketSphinxDemo class, this class will response to touch events on the speak button by allowing the speak button to function as a "push and hold" button and it will trigger the start of recognition when it is first pushed and the end of recognition when it is released.

```
public class PocketSphinxDemo extends Activity implements OnTouchListener,
RecognitionListener {
        static {
                System.loadLibrary("pocketsphinx_jni");
        }
```

```
RecognizerTask rec; // recognition task which run in worker thread
Thread rec_thread; // thread which recognition task run
boolean listening; // state of listening
ProgressDialog rec_dialog; // progress dialog for final result
EditText edit_text; // editable text view
```

The PocketSphinxDemo class implements OnTouchListener and RecognitionListener to register that this class will derive methods from these two interfaces. Next the native library is loaded in order to use pocketsphinx. Then the class creates a couple of members to use.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    this.rec = new RecognizerTask();
    this.rec_thread = new Thread(this.rec);
    this.listening = false;
    Button b = (Button) findViewById(R.id.Button01);
    b.setOnTouchListener(this);
    this.edit_text = (EditText) findViewById(R.id.EditText01);
    this.rec.setRecognitionListener(this);
    this.rec_thread.start();
}
```

In the onCreate() method, the program set up the content view to main.xml. Then instantiate RecognizerTask and tell the RecognizerTask to run in a worker thread.

Next the program registers the OnTouchListener and RecognitionListener then start the recognition thread.

```
public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                        this.listening = true;
                        this.rec.start();
                        break;
                case MotionEvent.ACTION_UP:
                        if (this.listening) {
                                this.rec_dialog =
ProgressDialog.show(PocketSphinxDemo.this, "", "Recognizing speech...", true);
                                this.rec_dialog.setCancelable(false);
                                this.listening = false;
                        }
                        this.rec.stop();
                        break;
                default:
                        ;
                }
                return false;
        }
```

The onTouch() method is called when the button is clicked. In case the button is down, setting the listening to true and call RecognizerTask.start() to start recording and recognizing the speech. If the button is up then the application will check that it is listening or not? If it is listening, that means it was released from holding the button state. Then set things up for final result and call RecognizerTask.stop() to stop recording and recognizing the speech.

```
public void onPartialResults(Bundle b) {
        final PocketSphinxDemo that = this;
        final String hyp = b.getString("hyp");
        that.edit_text.post(new Runnable() {
            public void run() {
                that.edit_text.setText(hyp);
            }
        });
}
public void onResults(Bundle b) {
        final String hyp = b.getString("hyp");
        final PocketSphinxDemo that = this;
        this.edit_text.post(new Runnable() {
            public void run() {
                that.edit_text.setText(hyp);
                that.rec_dialog.dismiss();
            }
        });
}
```

```
public void onError(int err) {
        }
}
```

The onPartialResult() and onResult() methods react accordingly to the recognition thread when partial result or final result are generated. Either case untying the message string type name's hyp from the bundle and let edit text to present it via post() method. It cannot be set directly because these results are generated from the other thread. It is something that java prevent.

3.4.6.3 RecognizerTask class, this class is separated into two classes: one is the AudioTask class which takes responsibility to record the speech and puts it into the buffer. Another class is RecognizerTask itself which will do the work to write the recorded speech file from AudioTask to external storage and recognize that speech for you.

```
public class RecognizerTask implements Runnable {
        class AudioTask implements Runnable {
                LinkedBlockingQueue<short[]> q; // warehouse to place the sound
                AudioRecord rec; // AudioRecord instance for record audio
                int block_size; // size of block that will be add to queue
                boolean done; // state of AudioTask
```

This class is designed to form as a consumer-producer pattern. The AudioTask class is the producer by keeping produces the record sound and put them into the warehouse in this case the warehouse is "q" member. The RecognizerTask will then retrieve the data from this queue and fetch them to recognition process and report the

result back to the main thread.

The AudioTask class is created inside the RecognizerTask because those two classes will run together and track can be kept from both of them at the same time.

In AudioTask class is implementing Runnable because this class will work as a producing thread. Next the program defines some members that will be used in this class.

```
        AudioTask(LinkedBlockingQueue<short[]> q, int block_size) {
                        this.done = false;
                        this.q = q;
                        this.block_size = block_size;
                        this.rec = new
AudioRecord(MediaRecorder.AudioSource.DEFAULT, 8000,
                                        AudioFormat.CHANNEL_IN_MONO,
                                        AudioFormat.ENCODING_PCM_16BIT,
8192);
        }
```

In constructor, we initialize the queue from RecognizerTask to get the same location of the queue. Then set the audio record to record with the 8000 sample rates.

```
        public void stop() {
                this.done = true;
        }
```

The stop() method will set the done attribute to true to stop this thread.

```
        public void run() {
                this.rec.startRecording(); // call to start recording
                while (!this.done) {
                        int nshorts = this.readBlock();
                        if (nshorts <= 0) // mean something error
                                break;
                }
                this.rec.stop(); // call to stop recording
                this.rec.release(); // release the resource AudioRecord holds
        }
        int readBlock() {
                short[] buf = new short[this.block_size];
                int nshorts = this.rec.read(buf, 0, buf.length);
                if (nshorts > 0) { // if nshorts has some data put it into queue
                        this.q.add(buf);
                }
                return nshorts;
        }
}
```

The run() method is where the actual work of AudioTask begins. It will loop through the readBlock() method to add sound buffer to the queue until the boolean "done" set to true.

Next is RecognizerTask class, this thread is initializing by the main thread.

```
Decoder ps; // pocketsphinx decoderobject

AudioTask audio; // AudioTask instance

Thread audio_thread; // Thread for AudioTask

LinkedBlockingQueue<short[]> audioq; // Queue for Audio buffers

RecognitionListener rl; // Listener for recognition results

enum State {  // state of main loop thread

        IDLE, LISTENING

};

enum Event { // event of main loop thread

        NONE, START, STOP

};

Event mailbox; // current event
```

This class starts with creating members. It also has enum State and Event for keeping track of what currently RecognizerTask is.

```
public void setRecognitionListener(RecognitionListener rl) {

        this.rl = rl;

}
```

This method is for registering the RecognitionListener to the main thread.

```
public RecognizerTask() {
        pocketsphinx

    .setLogfile("/sdcard/Android/data/edu.cmu.pocketsphinx/pocketsphinx.log");
        Config c = new Config();
        c.setString("-hmm",  // tell decorder where to find acoustic model
    "/sdcard/Android/data/edu.cmu.pocketsphinx/hmm/en_US/tamadroid");
        c.setString("-dict", // tell decorder where to find dictionary
    "/sdcard/Android/data/edu.cmu.pocketsphinx/lm/en_US/menu.dic");
        c.setString("-lm", // tell decorder where to find language model
    "/sdcard/Android/data/edu.cmu.pocketsphinx/lm/en_US/tamadroid.lm.DMP");
        c.setString("-rawlogdir", "/sdcard/Android/data/edu.cmu.pocketsphinx");
            // specify where to save output speech record file and type
        c.setFloat("-samprate", 8000.0); // set sample rate to 8000
        c.setInt("-maxhmmpf", 2000);
        c.setInt("-maxwpf", 10);
        c.setInt("-pl_window", 2);
        c.setBoolean("-backtrace", true);
        c.setBoolean("-bestpath", false);
        this.ps = new Decoder(c); // set decoder to specify configure
        this.audio = null;
        this.audioq = new LinkedBlockingQueue<short[]>(); // create the queue
        this.mailbox = Event.NONE; }
```

In the constructor, configure is set to the pocketsphinx decoder and initialize Event of the main loop to NONE.

```
public void run() {
        boolean done = false; // for keep looping
        State state = State.IDLE; // initialize state to IDLE
        String partial_hyp = null; // initialize partial_hyp result = null
```

In run() method, these three members are initialized.

```
while (!done) {
        Event todo = Event.NONE;
        synchronized (this.mailbox) {
            todo =  this.mailbox;
            if (state == State.IDLE && todo == Event.NONE) {
                try {
                    this.mailbox.wait(); // wait for something to happen
                    todo = this.mailbox;
                } catch (InterruptedException e) {
                }// end catch
            } // end if
            this.mailbox = Event.NONE;  } // end synchronized
```

While loop is a statement in which all the works are placed. First, define todo to keep what is going to do next. If the todo is NONE and state is IDLE then wait for main thread to call start() method. These are done in the synchronized block because it will use mailbox to lock thread and let main thread change the mailbox directly.

```
switch (todo) {
        case NONE:
                break; // nothing to do here
        case START:
                if (state == State.IDLE) {
                        this.audio = new AudioTask(this.audioq, 1024);
                        this.audio_thread = new Thread(this.audio);
                        this.ps.startUtt(); // call to create record file
                        this.audio_thread.start(); // start AudioTask thread
                        state = State.LISTENING;
                }
                break;
```

Then if the main thread is called to do something, it will switch todo to the case it is actually to do. If todo is NONE, then do nothing. If todo is START and state is IDLE, it will know that player just press the button to start recording and recognizing the speech from the main thread. Then initialize AudioTask, set thread and run it. Next, change the state to LISTENING which means right now it can recognize speech.

```
        case STOP:
                if (state == State.LISTENING) {
                        this.audio.stop(); // call AudioTask to stop
```

```
try {
        this.audio_thread.join(); // wait until die
}
catch (InterruptedException e) {
        done = true;
}
```

If todo is STOP and state is LISTENING, it will know that player just released the speak button. Then, call the AudioTask thread to stop and wait until it dies.

```
                                short[] buf;
// retrieve audio queue           while ((buf = this.audioq.poll()) != null) {
// call to recognize process        this.ps.processRaw(buf, buf.length, false, false);
                                }
```

Retrieving the entire buffer in queue and fetch it to the recognition process.

```
                                this.ps.endUtt(); // call to end recognize process
                                this.audio = null; // set AudioTask to initialize value
                                this.audio_thread = null; // initialize audio_thread
                                Hypothesis hyp = this.ps.getHyp(); // hypothesis
                                if (this.rl != null) {
                                        if (hyp == null) {
                                                this.rl.onError(-1);
                                        }
                                        else {
// create bundle to send the message to main thread        Bundle b = new Bundle();
// put result string to the bundle with name "hyp"         b.putString("hyp", hyp.getHypstr());
                // call to present final result             this.rl.onResults(b);
                                        } // end else
```

```
                    } // end if
                state = State.IDLE;
                } // end if
                break;
            } // end switch
```

Next, close all the attributes that was created and set in case todo = START. Then get the hypothesis result. If the result is not null, then sends it to the main thread as final result. Finally, set the state to IDLE to wait for next speech.

```
if (state == State.LISTENING) {
    try {
        short[] buf = this.audioq.take(); // take first buffer
        this.ps.processRaw(buf, buf.length, false, false);
        Hypothesis hyp = this.ps.getHyp(); // hypothesis result
        if (hyp != null) { // if it has hypothesis
            String hypstr = hyp.getHypstr(); // get result in string
            if (hypstr != partial_hyp) {
                if (this.rl != null && hyp != null) {
                    Bundle b = new Bundle();
                    b.putString("hyp", hyp.getHypstr());
                    this.rl.onPartialResults(b);
                }
            }
            partial_hyp = hypstr;
        }
    } catch (InterruptedException e) {}
} // end if
} // end while  } // end run() method
```

The last part in run() method can be accessed only during holding the speak button. It will wait the AudioTask to produce buffer and take the first buffer of the queue to hypothesis. If it has the result hypothesis, it will then check that the result is the same as previous result or not. If it is not, it will create bundle to send this partial result to the main thread by invoking the method onPartialResults() and set previous partial_hyp to the current one.

```
        public void start() {
                synchronized (this.mailbox) {
                        this.mailbox.notifyAll();
                        this.mailbox = Event.START;
                }
        }
        public void stop() {
                synchronized (this.mailbox) {
                        this.mailbox.notifyAll();
                        this.mailbox = Event.STOP;
                }
        }
} // end RecognizerTask
```

The last two methods are for the main thread to call. They will notify the recognizer thread and set todo method accordingly to the mailbox says to do.

3.4.6.4 Show the result

Holding button and speak "exercise food sleep clean healing" then release it.

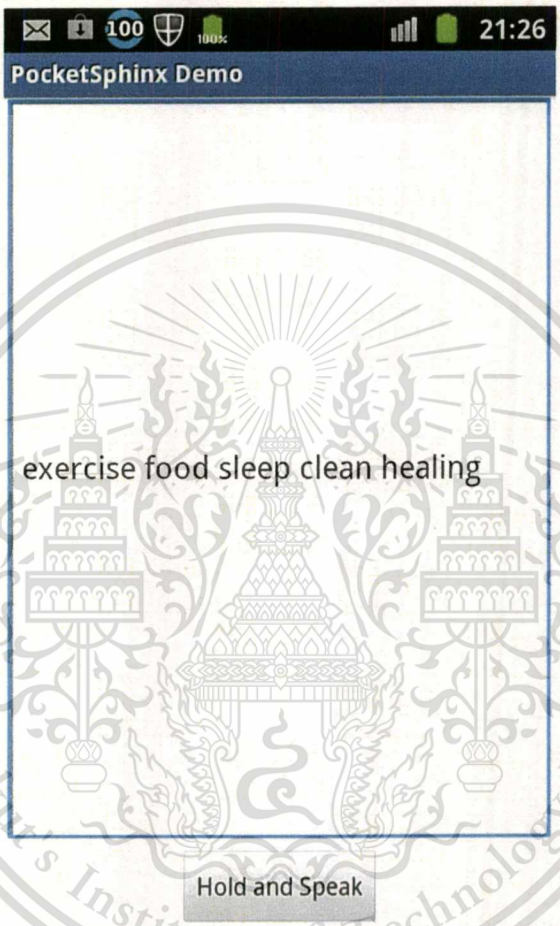


Figure 3.60 PocketSphinxDemo application.

Chapter 4

Implementation

This project is about game on android with speech recognition. This chapter will explain about how to install the game on the device and how to play this game.

4.1 Install the game

Install process can be divided into two parts

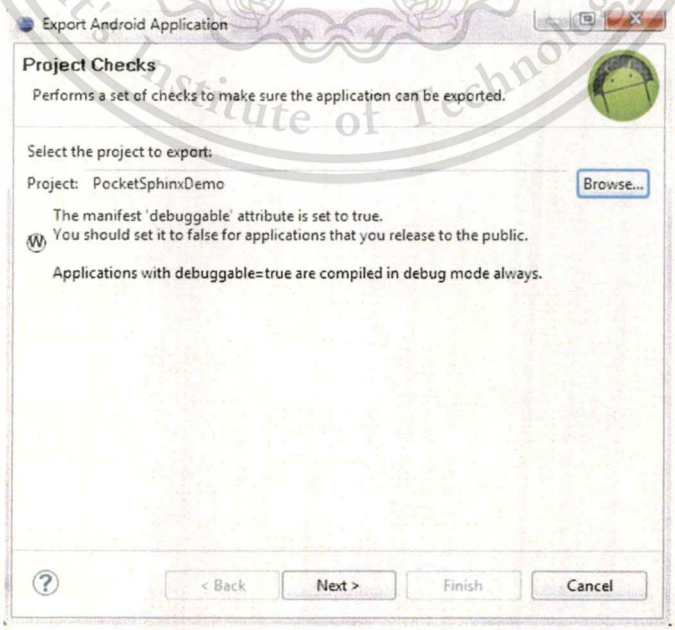
1. Export .apk file.

2. Copy .apk file that was exported from first step to the device and install it

4.1.1 Export file

4.1.1.1. Open Eclipse.

4.1.1.2. right-click your project in the package explorer view and select Android Tools -> Export Signed Application Package. You will see the dialog shown in figure 4-1.



4.1.1.3. Click the next button to see the dialog in figure 4-2.

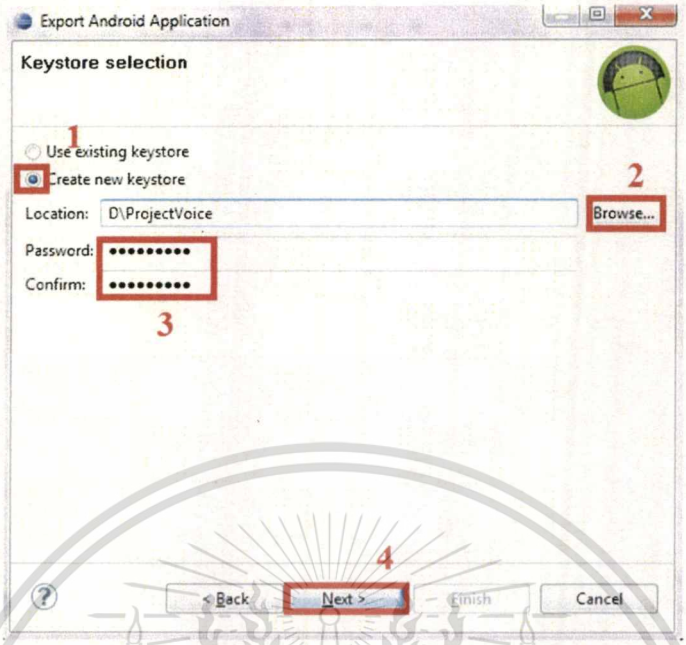


Figure 4.2 Creating keystore.

4.1.1.4 Check create new keystore then click browse... to select location to store keystore. Next type the password and confirm it then click next.

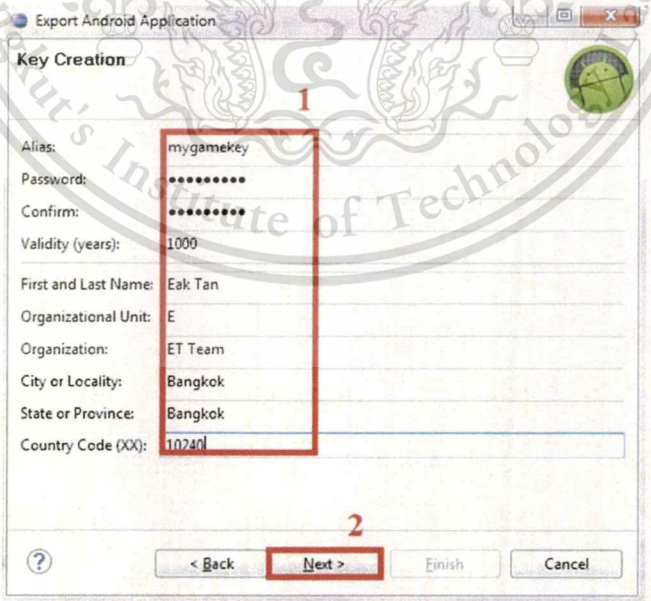


Figure 4.3 Creating the key for signing the APK.

4.1.1.5 Fill up the form then click next you will meet the dialog in figure 4.4.

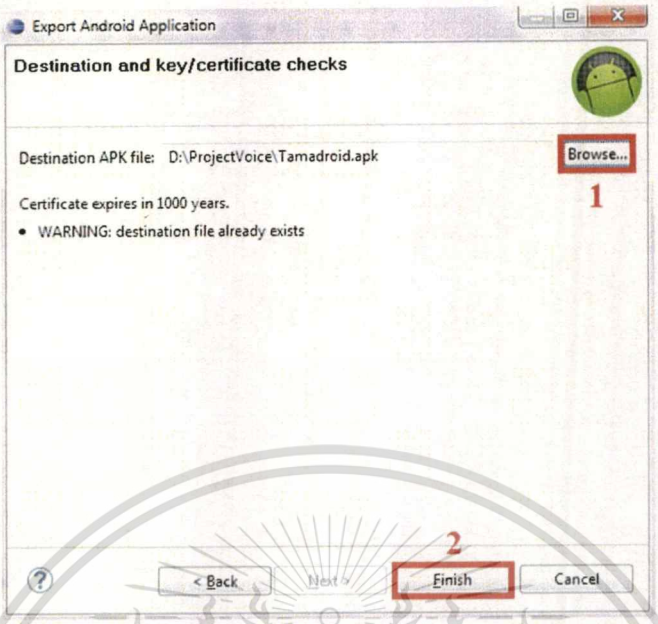


Figure 4.4 Specifying the destination file.

4.1.1.6 Click browse to specify where the exported APK file would be store. Then click finish.

4.1.2 Install game on device

4.1.2.1 Connect the device with computer by USB

4.1.2.2 Copy APK file from path specified in 4.1.1.6 to the device

4.1.2.3 Open My Files application in the device and go to the folder where file APK is placed.

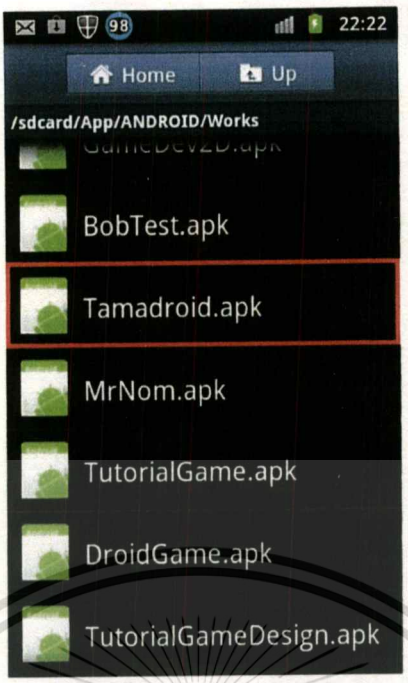


Figure 4.5 Placing file on the device.

4.1.2.4 Select that APK file and then select install.

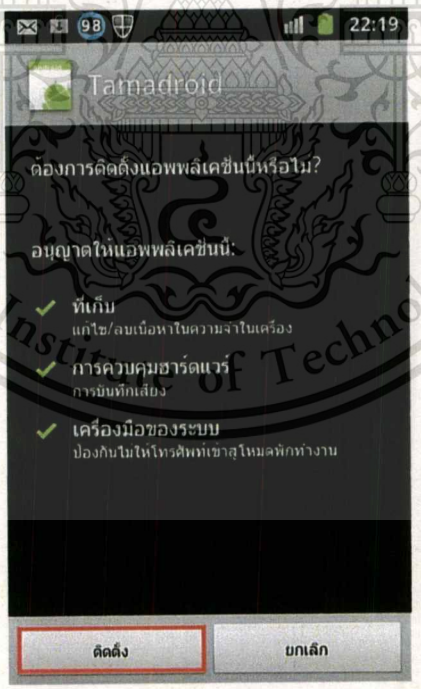


Figure 4.6 Install file on the device.

Then you are done to install game on the device.

4.2 Overview the game

This game is called "Tamadroid". Player must hatch an egg at the beginning and it will become a monster. It needs food, shower, sleep, and medicine. Player can play with it or ask it to do some exercises.

4.3 The explanation of the user interface

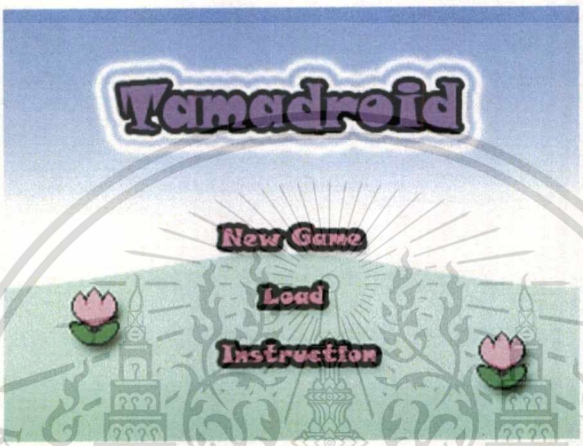


Figure 4.7 Main menu.

4.3.1 New Game button

For the first time playing, player must press on new game button to start a new game.

4.3.2 Load button

This button will load the latest gameplay which player had played before quit the game.

4.2.3 Instruction button

This button will show how to play Tamadroid.

4.4 How to play

After player pressed new game, the game will be started in figure 4.8. Player will press cold or heat buttons on the left hand side to adjust the temperature in Temp bar, which moves randomly. The egg will hatch when Time gauge is full.

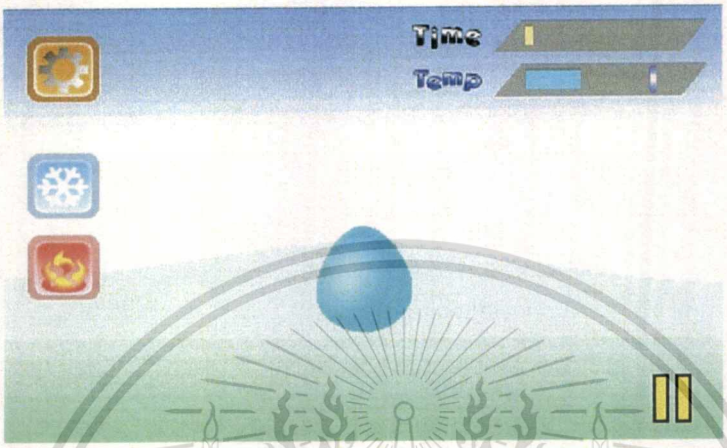


Figure 4.8 Egg screen.

Figure 4.9 Heater button.

Figure 4.10 Cooler button.

In figure 4.11 after the egg is hatched, the monster comes out. Now there are more options to play with your pet such as exercise, mini-game, etc. Hg bar is hunger point if it reduce, this means the monster become more hunger. Player has to feed it some foods. St bar is stamina point it will decrease when the monster do nothing. This point can be increased by letting the monster do some exercise.

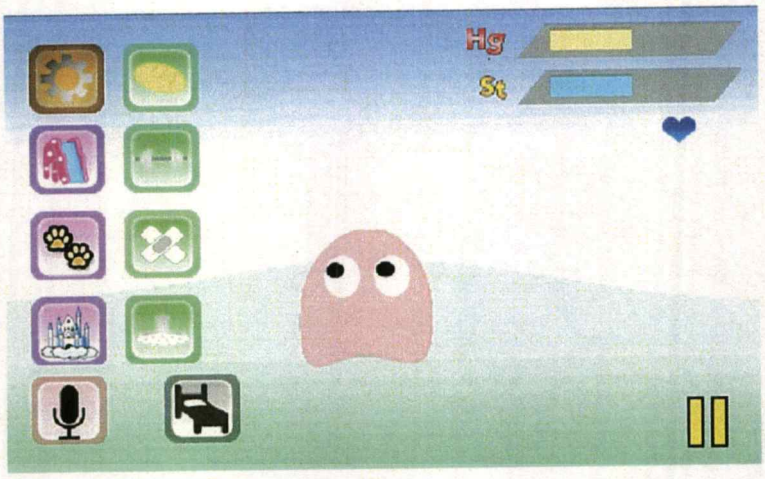


Figure 4.11 Monster screen.

When food button is pressed or speaking the word "food" while pressing microphone button there will be a food drop from the sky. Player needs to drag it to his pet. Beware of rotten food if there are more than 3 pieces in the screen, the monster will be sick.

Figure 4.12 Food button.

Figure 4.13 Food.

Figure 4.14 Rotten food.

Figure 4.15 Microphone button.

If the monster is sick, player must treat it by pressing plaster button to use medicine with the monster or speak "healing" while the microphone button is being pressed. A syringe will appear after plaster button is pressed. Player just drags the syringe to the monster to use.

Figure 4.16 Plaster button.

Figure 4.17 Syringe.

Player can ask the monster to do an exercise by pressing exercise button or speak "exercise" after microphone button is being pressed. Player can speak "left" or "right" while monster is exercising.

Figure 4.18 Exercise button.

When the monster is dirty, player should clean it. Shower button is for cleaning a monster or speak "clean" while pressing microphone button. After pressing shower button, the shower will appear then drag and hold it over the monster for a while to make it clean.

Figure 4.19 Shower button.

Figure 4.20 Dirty.

When the sky is getting dark, the monster needs to sleep so player should let it sleep by pressing bed button or say "sleep" to prepare to sleep and then say "good night" to sleep. Although you wake it up at night, it will not do anything but sleep.

Figure 4.21 Sleep button.

Figure 4.22 Sleep.

There are 3 mini-games. First, Fruit-Eating Monster game, player controls the direction of the monster by pressing left arrow or right arrow. Once the monster eats a fruit there will be a box behind the monster. Let the monster eats as much as possible. If the monster hit its boxes, that means game over.

Figure 4.23 Fruit-Eating Monster game.

Figure 4.24 Fruit-Eating Monster screen.

Second, Let's Run Together game, player tilts his device to control the direction and presses jump button to allow player command "jump" to dodge the obstacles. The monster will run until it reaches the goal.

Figure 4.25 Let's Run Together game.

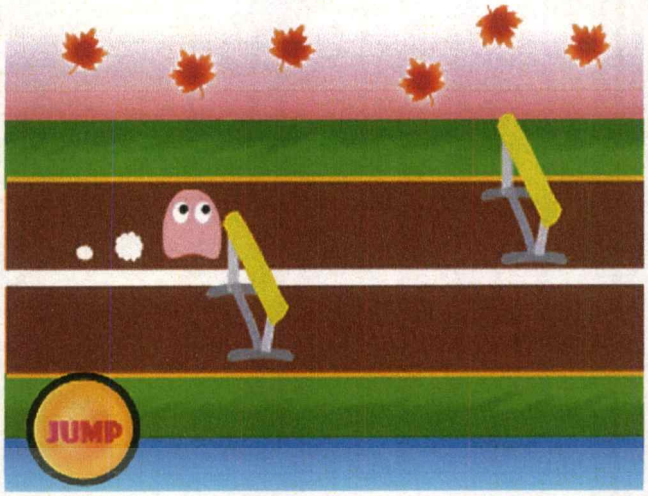


Figure 4.26 Let's Run Together screen.

The last mini-game is Jump to Heaven. Player tilts his device to control direction of monster jumping and collect the coins. There are jumpers help the monster jump higher. The floating stone will disappear, if the monster repeatedly jumps on the same stone. If the monster falls, game over. There are rippers who are going to stop the monster to get to the haven.

Figure 4.27 Jump to Heaven game.

Figure 4.28 Jumper.

Figure 4.29 Ripper.

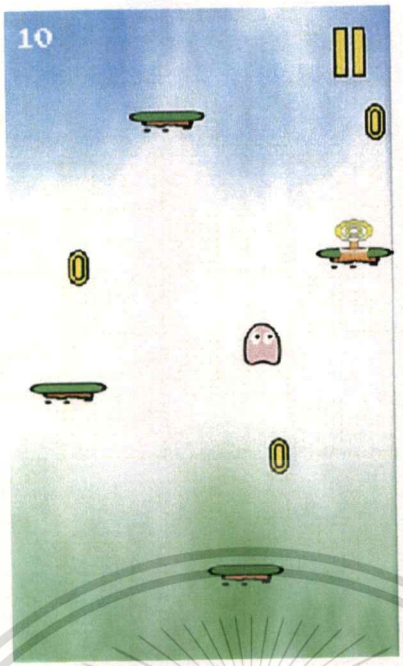


Figure 4.30 Jump to Haven screen.

Each mini-game can increase happiness level, which represented by heart color. From the least happiness to the most happiness: black, blue, yellow, and red.

Figure 4.31 Heart.

Chapter 5

Conclusion

5.1 Conclusion

In this special project, we create a pet game that embedded the speech recognition system to provide multimodal interaction. The purpose of this game is to allow the players to have a new experience in interacting with the game. Players can order or command their pet either by using touch or using speech. In speech recognition part, game speech model is created which include acoustic model, language model, and dictionary to map the commands in the game. CMU Sphinx Toolkit is used in this stage to consume game speech model and convert them to which CMU Sphinx Toolkit decoder can understand. In the game part, game framework is created to help writing the game source code much easier. Finally for graphic part, OpenGL ES is used even though that means more heavy-mass code, OpenGL ES is needed. OpenGL ES API comes with the functions that can control everything about graphics such as rotation and transparency that only Android API has limitation.

This game's title is "Tamadroid". The game task is to take care of the pet. If players do not take care of their pet well, it will end up with the death. The meters come in help in this stage to keep track of what currently status the pet is. Players can use these meters as a guideline to take care of the pet either by using touch or speech to command.

5.2 Problems

While we were researching and developing this game, the problems occurred in this special project are as follows:

- All CMU Sphinx tools are writing in C class, in order to use them on Android we need to port them to Java class.

- All CMU Sphinx tools need UNIX environment to do everything such as install. We need to install Cygwin to emulate UNIX environment on Window.

- Android emulator cannot use for testing speech recognition because it does not have microphone. So we have to test it on the device that is more time-consuming.

- Android emulator does not have enough resources to test when the application is more code complexity, which will make the emulator run very slow.

- The recognition process is very high resources usage, for those who have high-performance devices, there is no problem but for low-performance devices this process will take time around 10 seconds to be done. During that time the game will be hanging.

5.3 Suggestion

- In order to have high precision speech recognition, we need to record various and a lot of sound to achieve that.

- Players should stay in quite enough environments to use speech command; otherwise the speech recognition system will not work.

Appendix A

Install Cygwin

First you need to download Cygwin program from the site below

http://www.cygwin.com/

Because sphinx toolkit only need Linux environment to install and run.

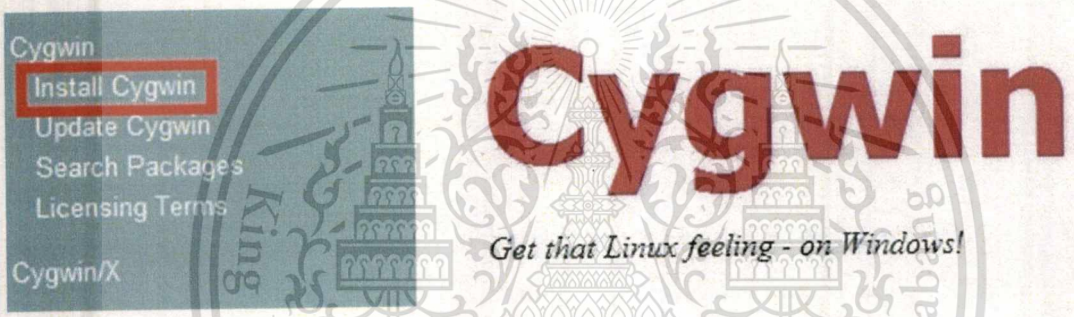


Figure A.1 Download Cygwin page.

There's a small grey box on the right side of the page that says "Install Cygwin". Click it, and Cygwin's setup.exe will be downloaded and run:

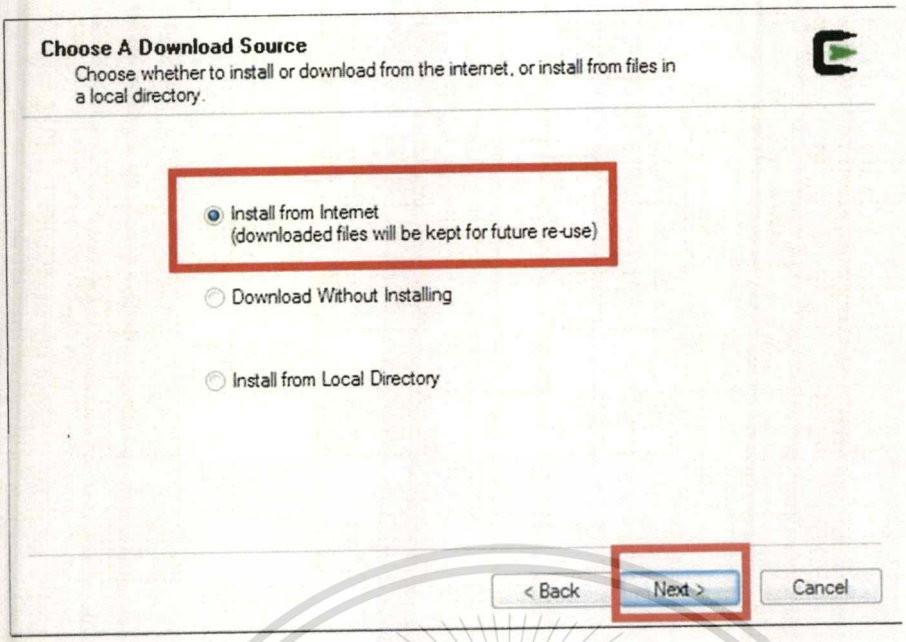


Figure A.2 Choose the way you need to install Cygwin.

Choose Install from Internet, then click Next, then choose the install directory (be sure to choose a directory path that contains no spaces in it)

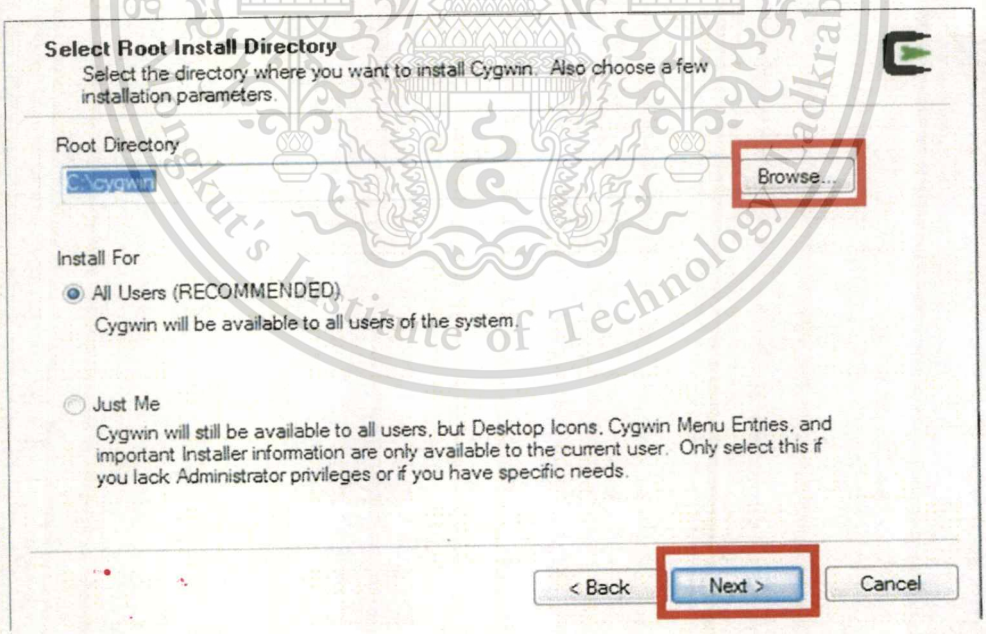


Figure A.3 Select directory.

At this point Cygwin will connect to its central site and download the list of mirror sites. Choosing a mirror site that ends up with .jp for better download speed.

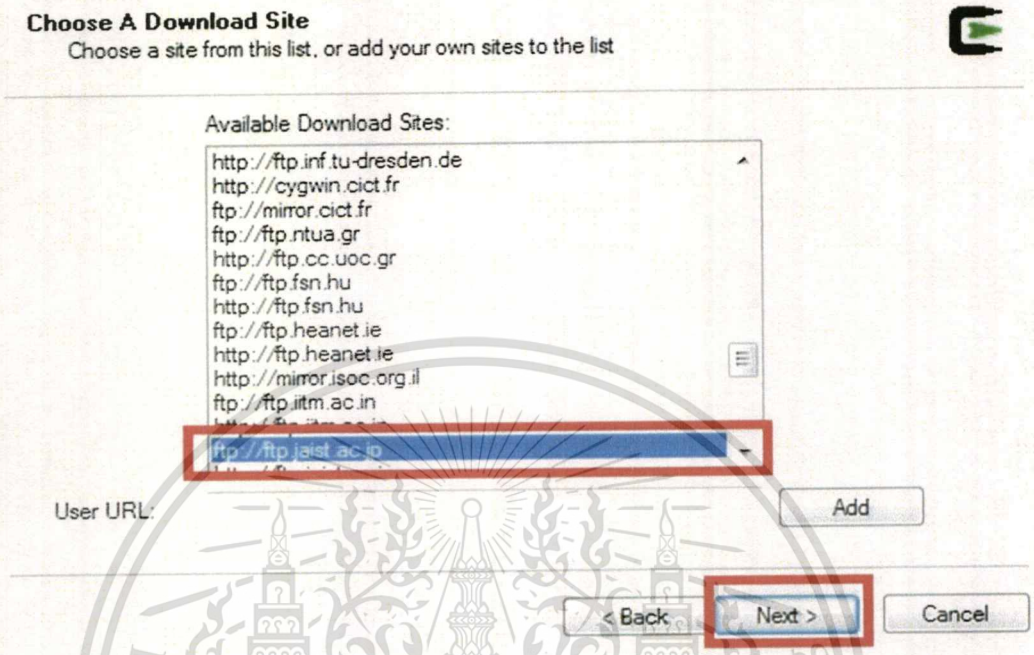


Figure A.4 Select mirror site to download Cygwin.

After you chose the mirror site and clicked Next, Cygwin will download and show you the lists of available packages:

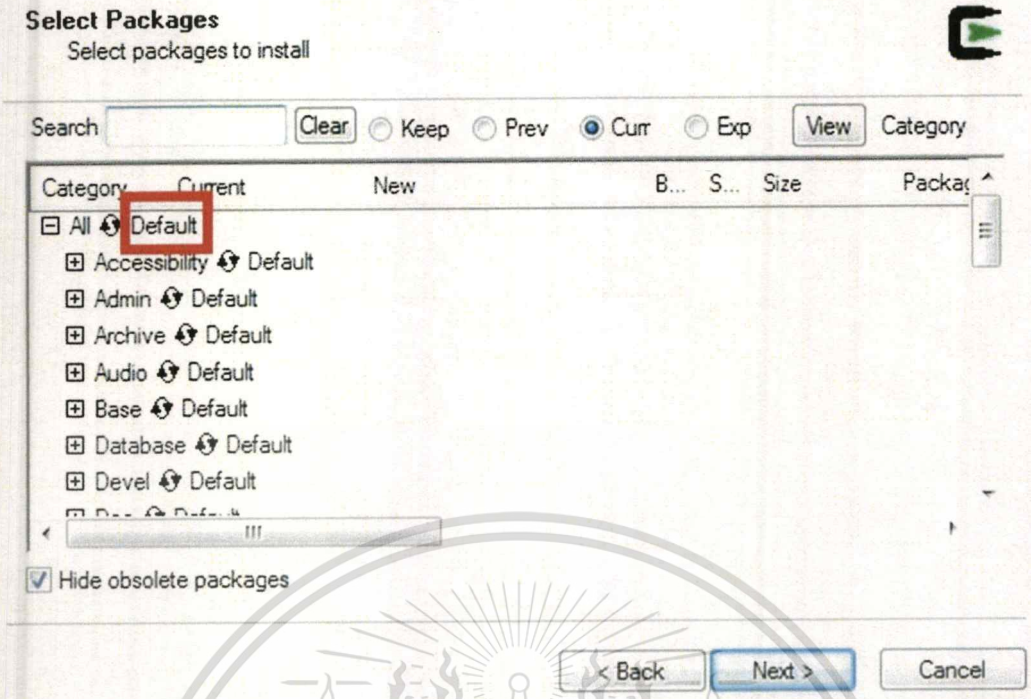


Figure A.5 Package to be select and install.

By default, only the basic packages are installed. We need to download all in order not to miss some functions. Click (once) on the word "Default" and the root of All nodes and wait for a few seconds for finishing the setup. Then, you will see "Default" are changed to "Install" for the All nodes.

Now click next and let Cygwin downloads the packages and installs the environments.

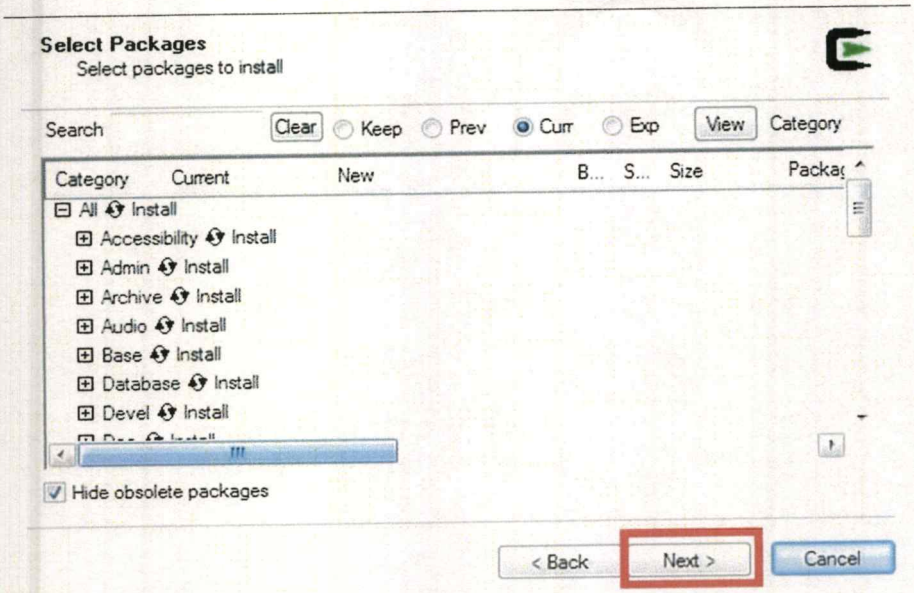


Figure A.6 Select package to be install.

This might take a huge time. Wait until it finishes. When it finishes. You will see the final setup screen

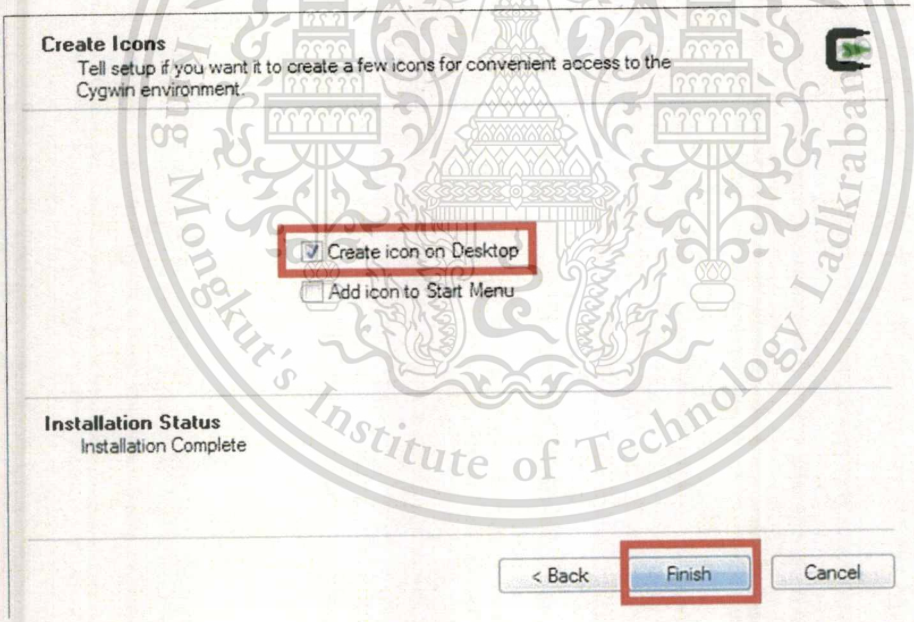


Figure A.7 show finish setup.

Allow it to create an icon on the desktop. Here's what you will see on your desktop after you click Finish – an icon that launches the Cygwin console.

Appendix B

Install Sphinx Toolkit

First of all, download the released packages, go to

http://cmusphinx.sourceforge.net/wiki/download/

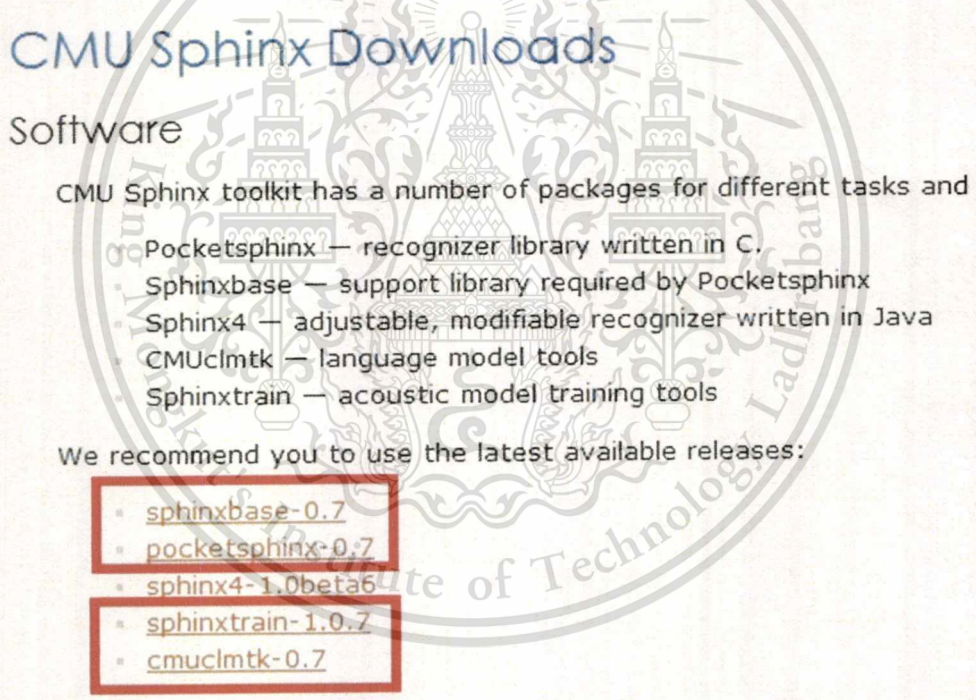


Figure B.1 Download sphinx toolkit page.

Click them to download and save to whatever folder you work with and extract all of them.

cmuclmtk	26/8/2554 16:45	File folder	
pocketsphinx	29/8/2554 22:32	File folder	
sphinxbase	29/8/2554 22:31	File folder	
sphinxtrain	30/8/2554 13:53	File folder	
cmuclmtk-0.7.tar	26/8/2554 16:43	WinRAR archive	8,911 KB
cmusphinx-pocketsphinx.tar	21/6/2554 18:53	WinRAR archive	22,988 KB
cmusphinx-sphinxbase.tar	27/6/2554 14:10	WinRAR archive	2,810 KB
cmusphinx-SphinxTrain.tar	28/8/2554 10:54	WinRAR archive	8,577 KB

Figure B.2 Sphinx tool download and extract.

Then open Cygwin. Double click on Cygwin icon on the desktop.

You will see the command line look like the figure below.

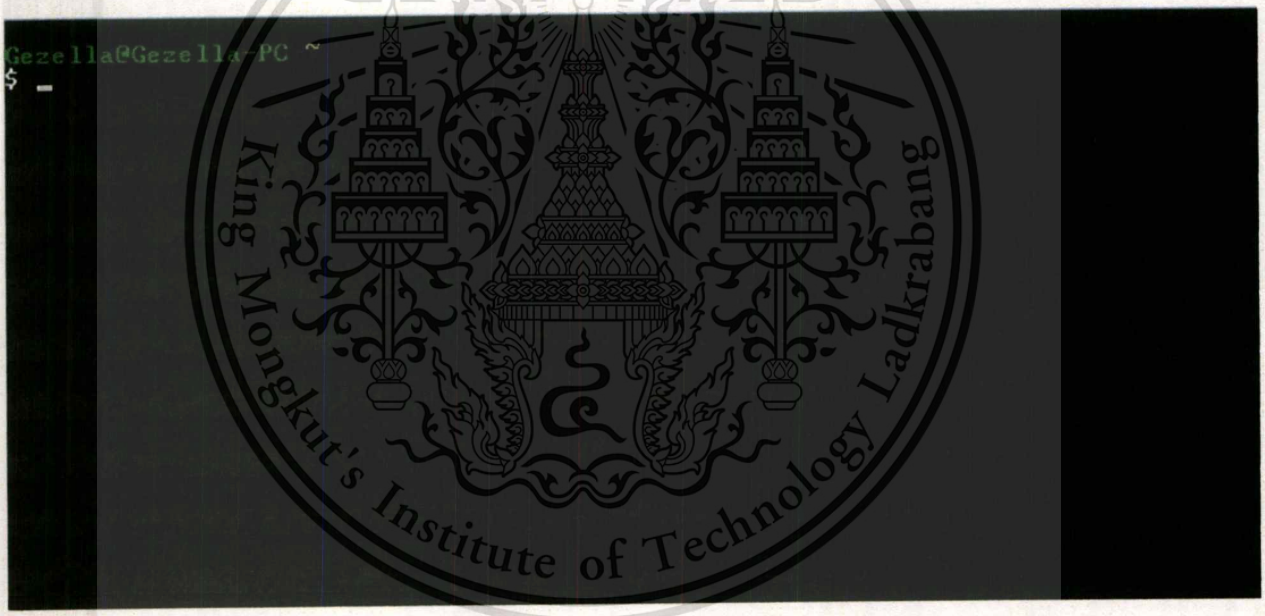


Figure B.3 Interface of Cygwin.

Then type.

"cd /cygdrive/where your sphinx tool located/sphinxbase"

For example my tool is located at D:\VoiceProject\sphinx2

So I type

“cd /cygdrive/d/voiceproject/sphinx2/sphinxbase”

It will look like this

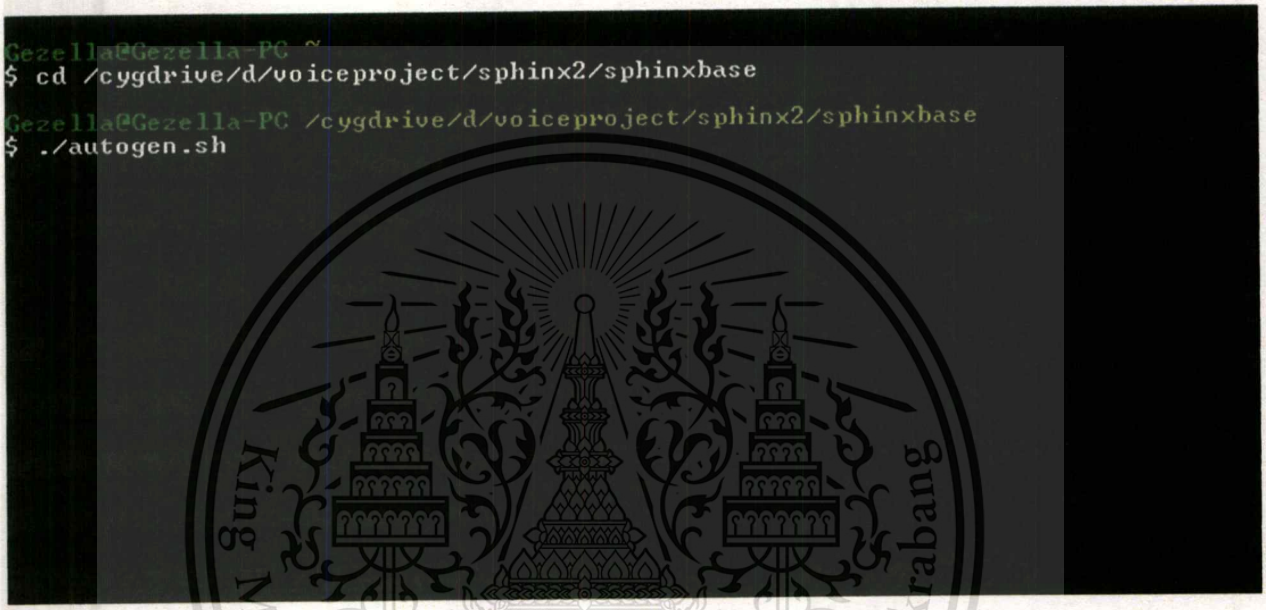


Figure B.4 How to change directory in Cygwin.

Next type “./autogen.sh” and wait until you can type again. Then type “make” and wait for processing to complete.

```
config.status: creating test/unit/test_ad/Makefile
config.status: creating test/unit/test_alloc/Makefile
config.status: creating test/unit/test_bitvec/Makefile
config.status: creating test/unit/test_case/Makefile
config.status: creating test/unit/test_string/Makefile
config.status: creating test/unit/test_cmdln/Makefile
config.status: creating test/unit/test_hash/Makefile
config.status: creating test/unit/test_matrix/Makefile
config.status: creating test/unit/test_feat/Makefile
config.status: creating test/unit/test_fe/Makefile
config.status: creating test/unit/test_logmath/Makefile
config.status: creating test/unit/test_ngram/Makefile
config.status: creating test/unit/test_fsg/Makefile
config.status: creating test/unit/test_thread/Makefile
config.status: creating test/unit/test_util/Makefile
config.status: creating test/regression/testfuncs.sh
config.status: creating test/regression/Makefile
config.status: creating include/config.h
config.status: creating include/sphinx_config.h
config.status: executing depfiles commands
config.status: executing libtool commands
Now type `make' to compile the package.

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/sphinxbase
$ make_
```

Figure B.5 Process for command "./autogen.sh".

```
sphinxbase.c: In function æ__pyx_f_10sphinxbase_9NGramIter_set_iterÆ:
sphinxbase.c:3440:16: warning: assignment discards qualifiers from pointer targe
t type
sphinxbase.c: In function æ__pyx_pf_10sphinxbase_8HuffCode_5decodeÆ:
sphinxbase.c:5390:52: warning: passing argument 2 of æhuff_code_decode_strÆ from
 incompatible pointer type
../include/sphinxbase/huff_code.h:138:13: note: expected æconst char **Æ but arg
ument is of type æchar **Æ
sphinxbase.c:5390:20: warning: assignment discards qualifiers from pointer targe
t type
sphinxbase.c: In function æ__pyx_pf_10sphinxbase_8HuffCode_9decode_from_fileÆ:
sphinxbase.c:5856:15: warning: assignment discards qualifiers from pointer targe
t type
creating build/lib.cygwin-1.7.9-i686-2.6
gcc -shared -Wl,--enable-auto-image-base build/temp.cygwin-1.7.9-i686-2.6/sphinx
base.o -L../src/libsphinxbase/.libs -L/usr/lib/python2.6/config -lsphinxbase -li
conv -lpython2.6 -o build/lib.cygwin-1.7.9-i686-2.6/sphinxbase.dll
touch pymod-build-stamp
make[1]: Leaving directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase/python'
make[1]: Entering directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase'
make[1]: Nothing to be done for `all-am'.
make[1]: Leaving directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase'

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/sphinxbase
$ make install_
```

Figure B.6 Process for command "make".

And then type "make install". When the process finished, it will show on the screen look like this then installation for sphinxbase is completed.

```
running install
running build
running build_ext
running install_lib
copying build/lib.cygwin-1.7.9-i686-2.6/sphinxbase.dll -> /usr/local/lib/python2
.6/site-packages
running install_egg_info
Removing /usr/local/lib/python2.6/site-packages/SphinxBase-0.7-py2.6.egg-info
Writing /usr/local/lib/python2.6/site-packages/SphinxBase-0.7-py2.6.egg-info
test -z "/usr/local/include/sphinxbase" || /usr/bin/mkdir -p "/usr/local/include
/sphinxbase"
 /usr/bin/install -c -m 644 sphinxbase.pxd '/usr/local/include/sphinxbase'
make[2]: Leaving directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase/python'
make[1]: Leaving directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase/python'
make[1]: Entering directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase'
make[2]: Entering directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase'
make[2]: Nothing to be done for `install-exec-am'.
test -z "/usr/local/lib/pkgconfig" || /usr/bin/mkdir -p "/usr/local/lib/pkgconfi
g"
 /usr/bin/install -c -m 644 sphinxbase.pc '/usr/local/lib/pkgconfig'
make[2]: Leaving directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase'
make[1]: Leaving directory `/cygdrive/d/voiceproject/sphinx2/sphinxbase'

Gezella@Gezella-PC /cygdrive/d/voiceproject/sphinx2/sphinxbase
$
```

Figure B.7 Process for command “make install”.

Then go into pocketsphinx folder by typing

“cd /cygdrive//where your sphinx tool located /pocketsphinx” and do the same thing as in sphinxbase

“./autogen.sh”

“make”

“make install”

For SphinxTrain you have to adjust some command by changing

“./autogen.sh” to “./configure”

“make” so we finish sphinxtrain no need to do ”make install”

For cmuclmtk do

"./configure"

"make"

"make install"

To test the tool is ready for you to use or not, typing

"pocketsphinx_continuous"

If it works correctly, it will look like this.

```
INFO: ngram_model_dmp.c(196): ngrams 1=5001, 2=436879, 3=418286
INFO: ngram_model_dmp.c(242):      5001 = LM.unigrams(+trailer) read
INFO: ngram_model_dmp.c(291):    436879 = LM.bigrams(+trailer) read
INFO: ngram_model_dmp.c(317):    418286 = LM.trigrams read
INFO: ngram_model_dmp.c(342):     37293 = LM.prob2 entries read
INFO: ngram_model_dmp.c(362):     14370 = LM.bo_wt2 entries read
INFO: ngram_model_dmp.c(382):     36094 = LM.prob3 entries read
INFO: ngram_model_dmp.c(410):       854 = LM.tseg_base entries read
INFO: ngram_model_dmp.c(466):      5001 = ascii word strings read
INFO: ngram_search_fwdtree.c(99): 788 unique initial diphones
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 60 single-phone
words
INFO: ngram_search_fwdtree.c(186): Creating search tree
INFO: ngram_search_fwdtree.c(191): before: 0 root, 0 non-root channels, 60 singl
e-phone words
INFO: ngram_search_fwdtree.c(326): after: max nonroot chan increased to 13428
INFO: ngram_search_fwdtree.c(338): after: 457 root, 13300 non-root channels, 26
single-phone words
INFO: ngram_search_fwdflat.c(156): fwdflat: min_ef_width = 4, max_sf_win = 25
INFO: continuous.c(371): pocketsphinx_continuous COMPILED ON: Oct  6 2011, AT: 0
0:30:40

Allocating 32 buffers of 2500 samples each
READY....
```

Figure B.8 Program pocketsphinx_continuous.

Then you are ready to create your own model.

Appendix C

Setting up project for PocketSphinx on Android Project

In order to build pocketsphinx on android, first you need to have "sphinxbase" and "pocketsphinx" installed in your computer and they would be in the same parent folder.

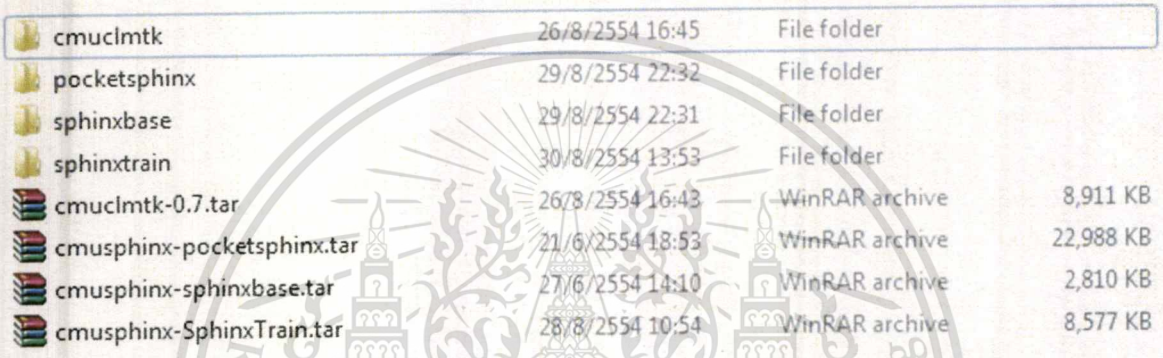


Figure C.1 Show working folder.

Then go to

http://cmusphinx.svn.sourceforge.net/viewvc/cmusphinx/trunk/PocketSphinxAndroidDemo/

And click "Download GNU tarball" save to your working folder and extract it.

gen/	10391	17 months	dhdfu	add r.java too
jni/	11117	8 months	nshmyrev	Wrapper for nbest
libs/	10391	17 months	dhdfu	add r.java too
res/	10219	20 months	dhdfu	Add a TextView to put performance in
src/	10375	17 months	dhdfu	remove bogus @overrides
.bzrignore	10220	20 months	dhdfu	try to run SWIG automatically (require
.classpath	10216	20 months	dhdfu	remove pocketsphinx sources and swig as...
.project	10223	20 months	dhdfu	totally rewrite the recognizer task to us
AndroidManifest.xml	10376	17 months	dhdfu	switch to froyo
default.properties	10376	17 months	dhdfu	switch to froyo

Download GNU tarball

SourceForge Help ViewVC Help Powered by ViewVC 1.1.6

Figure C.2 Download pocketsphinx android page.

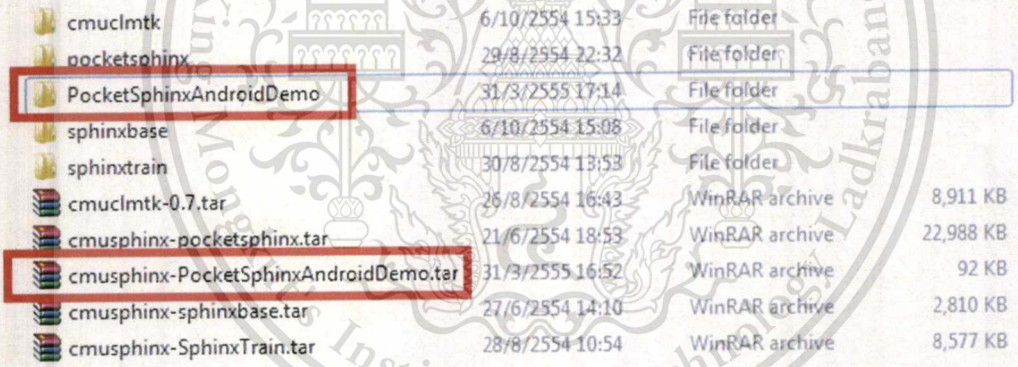


Figure C.3 Show the rar file and extract folder.

Next open file Android.mk in folder PocketSphinxAndroidDemo/jni

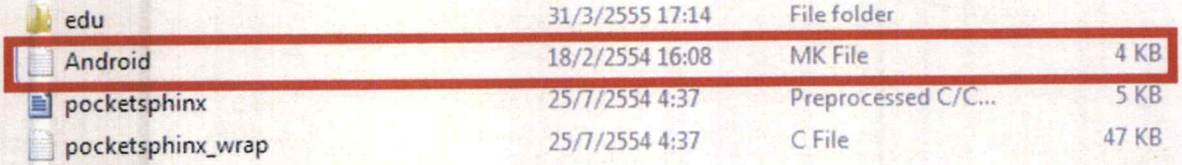


Figure C-4 show Android.mk file

Change SPHINX_PATH variable in line 5 to your working folder.

```
# You MUST change this to the absolute path of the directory containing
# sphinxbase and pocketsphinx source code.
SPHINX_PATH := $(HOME)/Projects/Sphinx/trunk
```

↓

```
SPHINX_PATH := /cygdrive/d/voiceproject/sphinx2
```

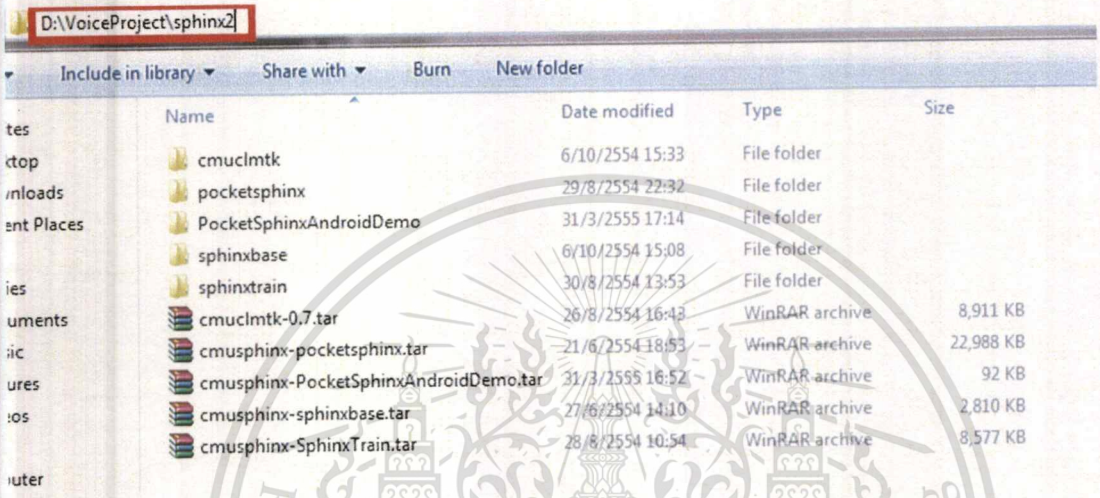


Figure C.5 Show specifying path for PocketSphinxAndroidDemo.

Next download Android NDK from http://developer.android.com/sdk/ndk/index.html choose platform for window.

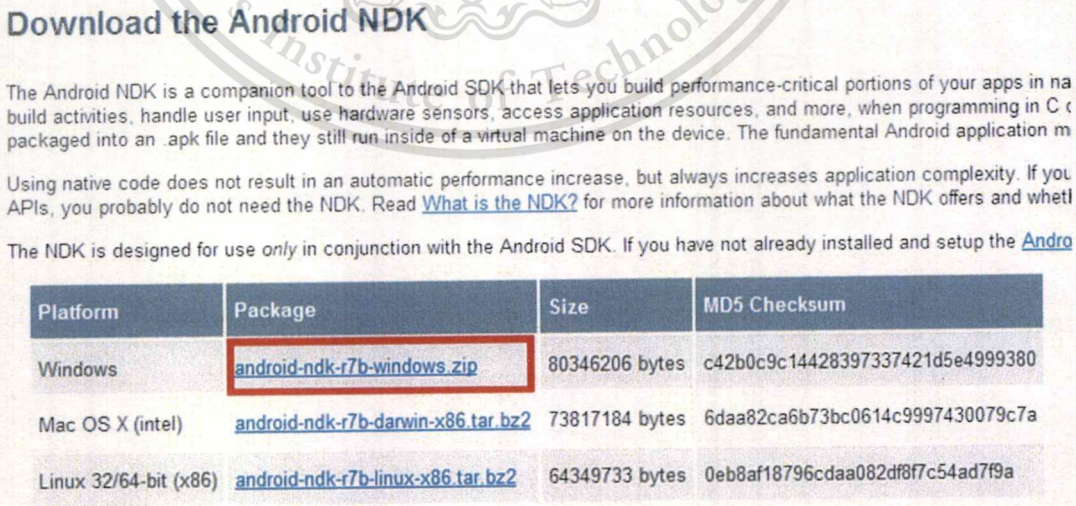
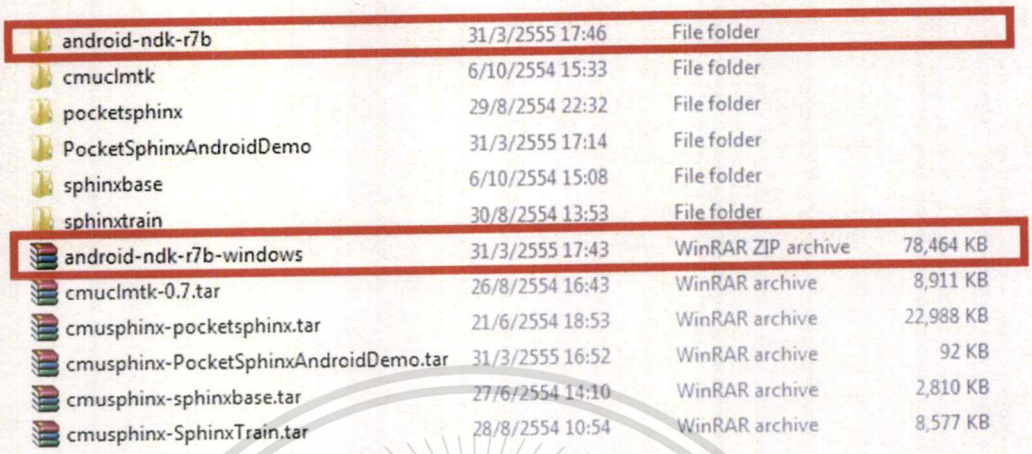


Figure C.6 Download Android NDK.

Extract Android NDK to working folder. Now you finish setting up an environment.



android-ndk-r7b	31/3/2555 17:46	File folder	
cmuclmtk	6/10/2554 15:33	File folder	
pocketsphinx	29/8/2554 22:32	File folder	
PocketSphinxAndroidDemo	31/3/2555 17:14	File folder	
sphinxbase	6/10/2554 15:08	File folder	
sphinxtrain	30/8/2554 13:53	File folder	
android-ndk-r7b-windows	31/3/2555 17:43	WinRAR ZIP archive	78,464 KB
cmuclmtk-0.7.tar	26/8/2554 16:43	WinRAR archive	8,911 KB
cmusphinx-pocketsphinx.tar	21/6/2554 18:53	WinRAR archive	22,988 KB
cmusphinx-PocketSphinxAndroidDemo.tar	31/3/2555 16:52	WinRAR archive	92 KB
cmusphinx-sphinxbase.tar	27/6/2554 14:10	WinRAR archive	2,810 KB
cmusphinx-SphinxTrain.tar	28/8/2554 10:54	WinRAR archive	8,577 KB

Figure C.7 Save and Extract Android NDK.

References

[1] Murphy, M.L. (2009). *Beginning Android*. New York, NY: Apress.

[2] Zechner, M. (2011). *Beginning Android Games*. New York, NY: Apress.

[3] So-In, C., and Janyoi, P. (2011). *Basic Android App Development*. Nonthaburi: IDC Premier.

[4] [Online]http://cmusphinx.sourceforge.net/, Retrieved at July 2011.

[5] [Online]http://project.uet.itgo.com/speech.htm, Retrieved at August 2011.

[6] [Online]http://badlogicgames.com, Retrieved at January 2012.

[7] [Online]http://www.speech.cs.cmu.edu/sphinx/tutorial.html, Retrieved at July 2011.

[8] [Online]http://www.droidnova.com/, Retrieved at November 2011.

[9] [Online]http://developer.android.com, Retrieved at September 2011.

[10] [Online]http://obviam.net/, Retrieved at December 2011.

[11] [Online]http://blog.jayway.com/category/android/, Retrieved at October 2011.

[12] [Online]http://stackoverflow.com/, Retrieved at November 2011.