# Convey Box: into a speaking world

Catarina Yamamoto and Koichiro Ochimizu
June 23, 2003
IS-RR-2003-004

School of Information Science
Japan Advanced Institute of Science and Technology, Hokuriku
Asahidai 1-1, Tatsunokuchi
Nomi, Ishikawa, 923-12, JAPAN
a_flama@hotmail.com, ochimizu@jaist.ac.jp

# Convey Box: into a speaking world

Catarina YAMAMOTO and Koichiro OCHIMIZU
Japan Advanced Institute of Science and Technology, Hokuriku
Asahidai 1-1, Tatsunokuchi, Nomi, Ishikawa, 923-12, JAPAN
Phone: +81-761- 51-1260
e-mail: a_flama@hotmail.com, ochimizu@jaist.ac.jp

## 1. Introduction

"Language is the ability to communicate with the people around us. Is perhaps the defining quality of human beings and it's so important to our survival and our joy" is a perfect definition said by Oral Deaf Association.

There are other ways to communicate with the world at large like sign and cued but it would be limited in some how. So the proposal software has the intuition to do a mix of the different kinds of communication used today to improve handicap children ability of listening skills and speech.

Auditory behaviors are pre-requisite to speech and language development. That's why poored and problematical speech and language disorders are usually related to auditory system that requires for example attention to focus the sounds, discrimination of it and association to the physical reality.

Children with hearing loss most of the time have fear to speak. They believe that anyone will complain about their speech, for example, too loud. But now a day, computers have been a great way of entertainment for kids. Using this strategy and a good visually software it will stimulate the young users to language activities, and be reward for it.

This software seems primary to hearing handicap children between four (4) and five (5) years old with hearing loss degree of 40-65 dB (speech/language retardation, learning disability, hears little or no speech at normal conversation levels) or up with assistance of the hearing aid device (this aid amplifies the sounds but not clearer it) to able then to talk with anyone. But it can be also used for anyone who intent to learn to speak or improve speaking skills.

The combination of technology: hearing aid and software - for effective practice (therapy), and support of family and friends will certainly give a chance to get these children to be into this huge hearing world.

## 2. Real Scenery

Children with hearing loss can hear and talk if they want, there is hope, but for that is necessary a lot of dedication, patience and hard work of the parents and especially of the kids.

All hearing loss people have some residual hearing and are based on that, that aid devices and implants work.

The usual treatment consists in individual and group classes of learning sounds. Each sound is repeated thousand of times to create a pattern (something that is ignored by us). Usually when a teacher is teaching the sound to the children hands is used to help the understanding at the beginning. Later a series of tests are made to test their abilities. For example, telling the child to identify a specific word (sound) that is said which the person who is saying has the mouth covered.

This learning should be started as soon as possible. Even a baby with hearing problem should have some language learning. All language is learning. To this small children treatment works with human contact and hearing - like songs, since the children are too young. As awards cheering and smiles are giving when words are saying by handicapped children.

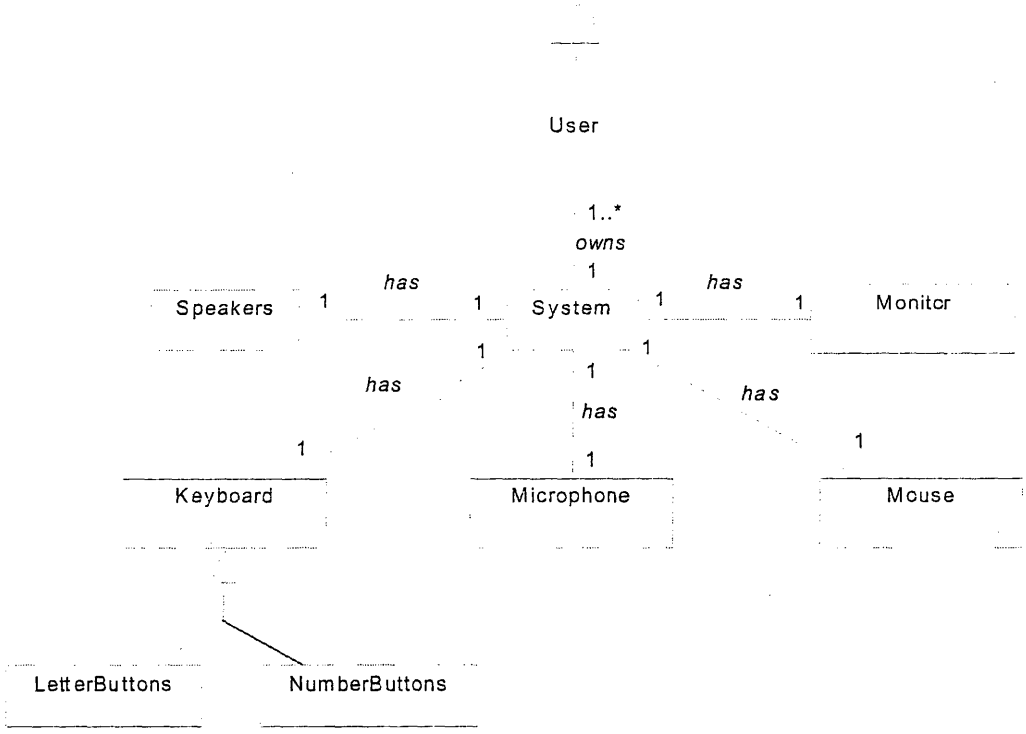
Imitate sounds like "baba", "dada", "dudu" is very important to development of hearing and talk. But this is a very slow process and takes long time. Some very simple words for us can be really powerful when said by handicap child.

Hard of hearing children has very strong mouth articulation and sounds that differ only by place of articulation having similar acoustic characteristics. For example, voiceless plosives (/p/, /t/ and /d/) as well as voiceless fricatives (/s/, /f/ and /th/). But in general you can understand what is being saying and with a lot of practice and work sound can be said almost, or even perfect.

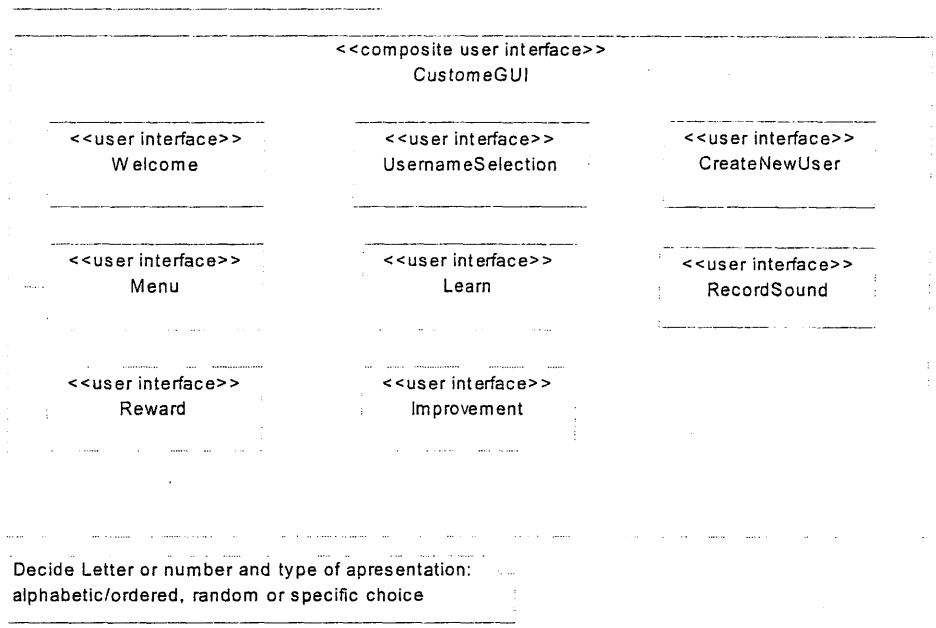
We can make an association of the different languages in world said by non-natives. For example a Japanese person with basic skills on English language can have some difficult saying some words that are not familiar in Japanese language as "think", "hard", etc. But after a lot of practice, this phonetics will be part of their understanding making them being able to say the words perfectly.

# 3. System Design

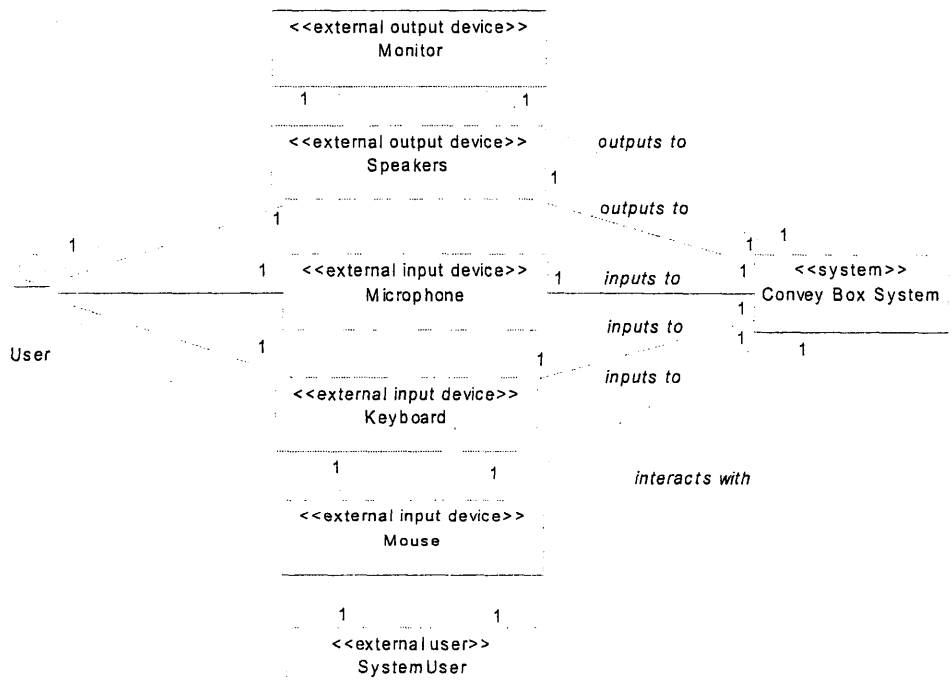
## *3.1 Conceptual Static Model*



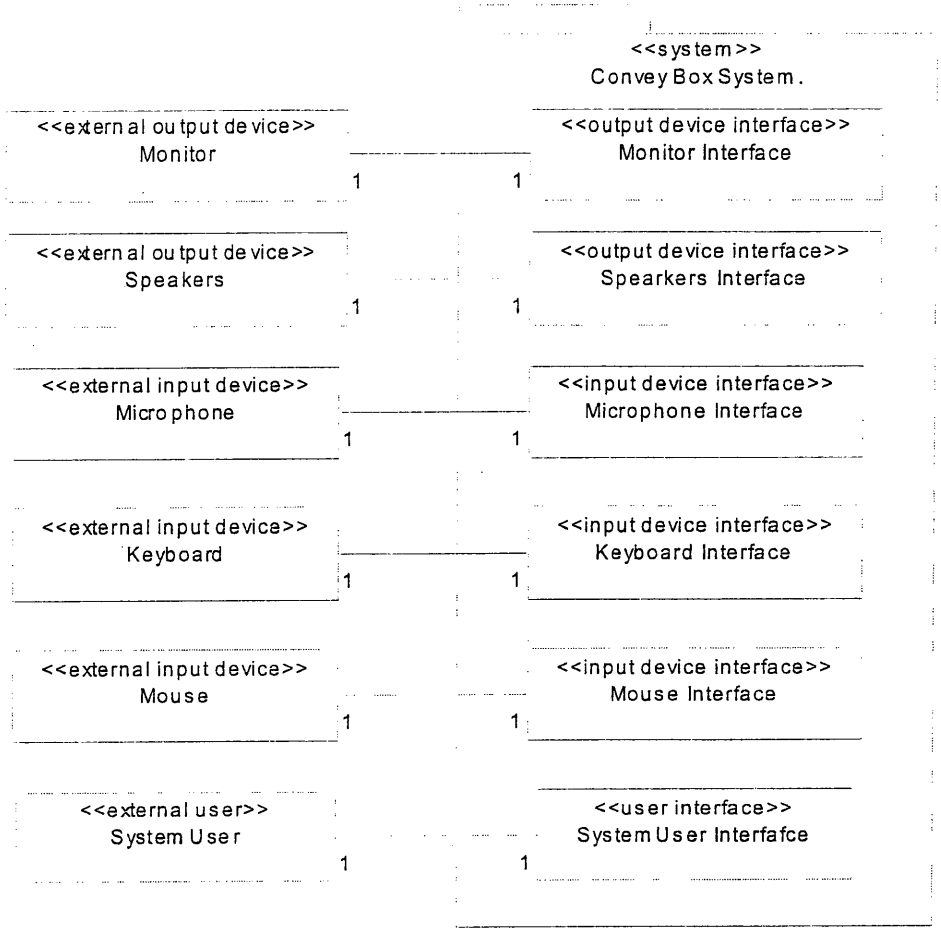
## 3.2 User Interface Classes




## 3.3 Context Class Diagram



## 3.4 System External Classes and Interface Classes



## *3.5 Use Case Model*

### 3.5.1 Create User Abstract Use Case

Use Case Name: Create User

Summary: System includes new user

Actor: End User

Precondition: CreateNewUser screen is displaying.

Description:

1. User fills in the form with name and username.
2. If the username is not registered (valid) yet, menu screen is shown.

Alternatives:

1. If there is a blank field, return to CreateNewUser screen.
2. If a username is already registered, try new username message is shown and CreateNewUser screen is shown again.

Postcondition: User is registered and logged on the system.

### 3.5.2 Validate User Abstract Use Case

Use Case Name: Validate User

Summary: System recognizes registered user

Actor: End User

Dependency: include *Create User* abstract use case

Precondition: System is started, displaying welcome screen.

Description:

1. User selects username on the list.
2. System displays menu screen.

Alternatives:

1. If user doesn't have a username, must register one to use the system.
2. If an incorrect username is choosen, user must click on exit button at menu screen and try again choosing the correct one.
3. If exit button is clicked, system shuts down.
4. If new user button is clicked, include *Create User*.

Postcondition: User is logged on the system.

### 3.5.3 Learn Sound Concrete Use Case

Use Case Name: Learn Sound

Summary: Sound and image is showing for association.

Actor: End User

Dependency: include *Validate User*

Precondition: System is displaying activities menu screen.

Description:

1. Include *Validate User* abstract use case.
2. User selects what kind of sounds want to learn (letters or numbers) then if it should be alphabetic/ordered, random or specific one.
3. Images and animations are shown with the sound.
4. If the next button is clicked, the next Learn sound is show.

Alternatives:

1. If exit button is clicked, menu screen is shown.
2. If listen again button is clicked, sound is played again.
3. If record button is clicked, record sound screen is shown.

Postcondition: User had the knowledge of the sound to record it.

### 3.5.4 Record Sound Concrete Use Case

Use Case Name: Record Sound

Summary: User will record imitation sound that was learned.

Actor: End User

Precondition: All input devices plugged in, record sound screen is showing.

Description:

1. Record button is clicked.
2. User reproduces sound on the microphone.
3. Stop button is clicked.
4. If listen me button is clicked, recorded sound will be reproduced.

Alternatives:

1. If user wants to record again the sound, description 1 (one) to 3 (three) is fallowed.
2. If there is no sound recorded, compare button will be disabled.

Postcondition: Compare button will be able.

### 3.5.5 Compare Sounds Concrete Use Case

Use Case Name: Compare Sounds

Summary: Sounds are compared to check improvements of user.

Actor: End User

Precondition: Compare button is able.

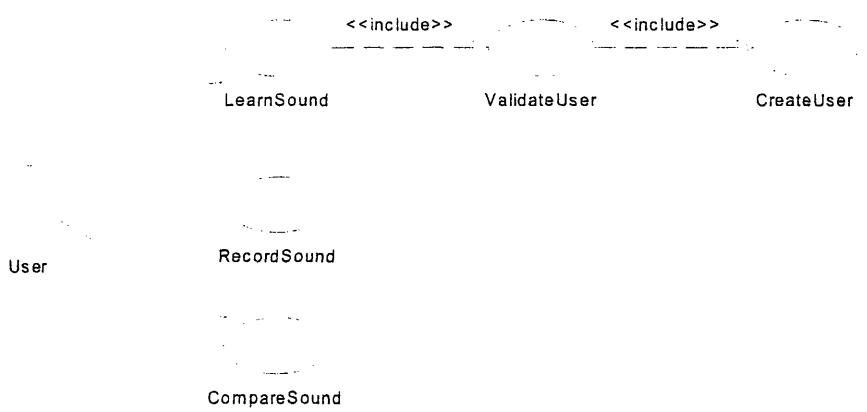
Description:

1. Recorded sound is compared with stored sound.
2. If it matches, reward[1] screen is shown.
3. If listen me button is clicked, recorded sound is played.
4. If listen sound button is clicked, stored sound is played.
5. If next button is clicked, the next sound/image is shown.

Alternatives:

1. If recorded sound does not match, improvement screen (tip) is shown.

Postcondition: Reward screen is displaying.



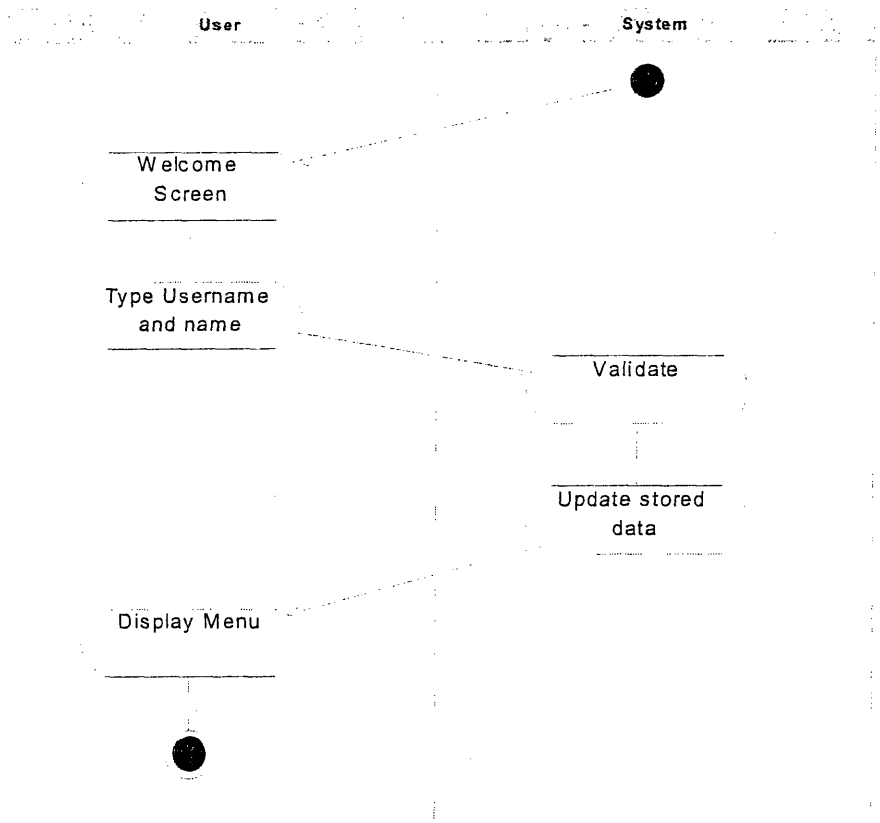

*Use Case Model with Abstract Use Cases*

[1] Reward is a given name for animations/images/sounds to congratulate successful operation

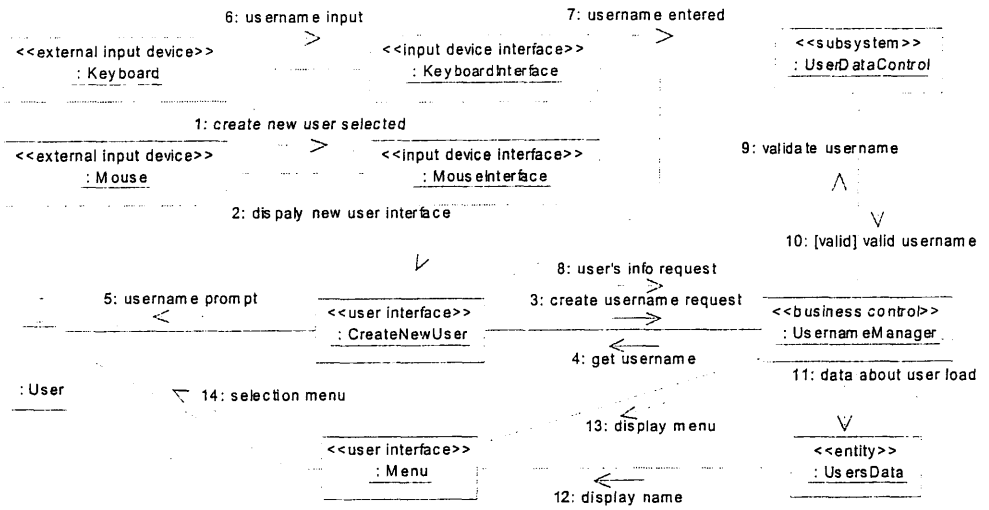
## *3.6 Diagrams*

### 3.6.1 Create User Use Case Diagrams

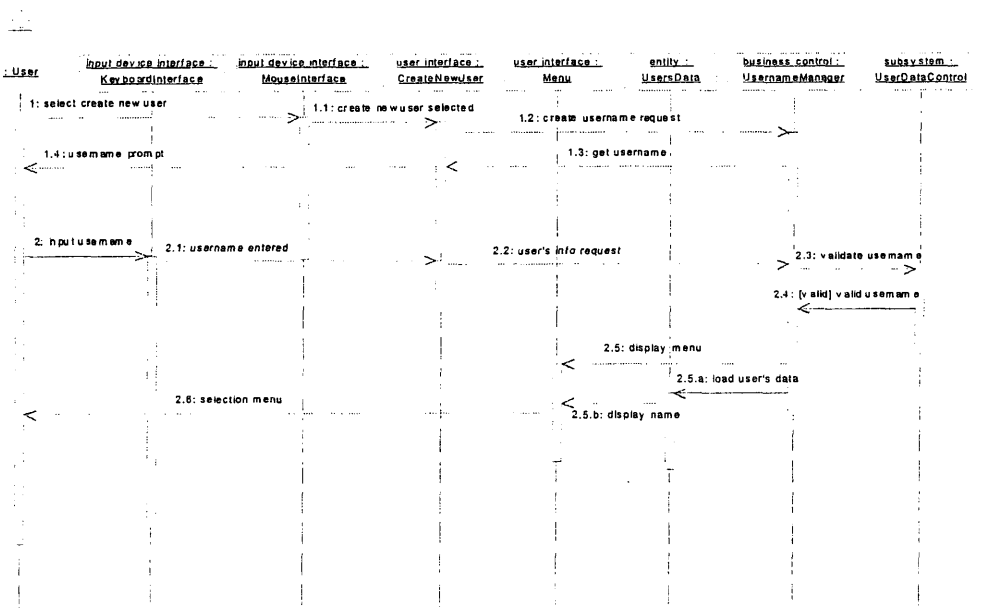
#### 3.6.1.1 Activity Diagram



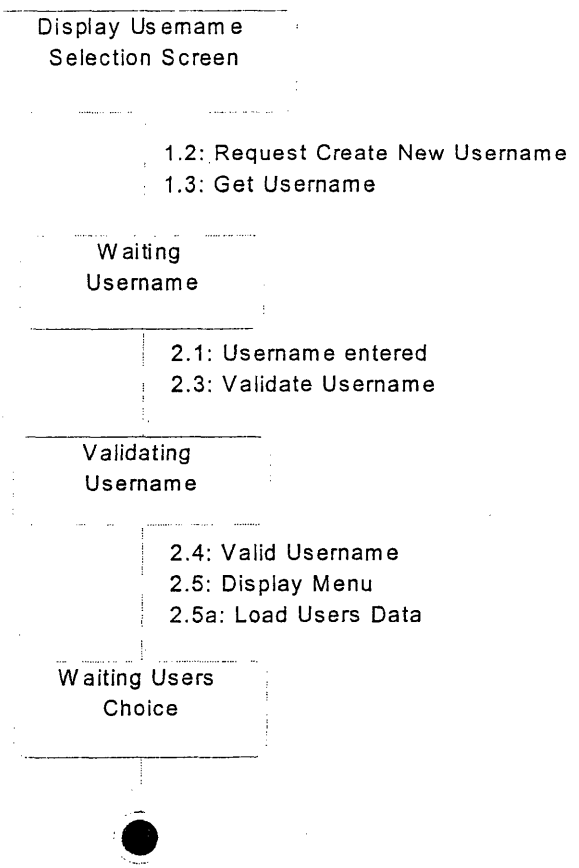
## 3.6.1.2 Collaboration Diagram




## 3.6.1.3 Sequence Diagram



### 3.6.1.4 Statechart Diagram



### 3.6.1.5 Sequence Description

**1:** The *user* actor select create a new user by clicking the button. The *mouseInterface* object will select information.

**1.1:** The *mouseInterface* object calls event *createNewUser* screen.

**1.2:** The *createNewUser* sends request to create username to *UsernameManager*. As result, the statechart transits from *Idle* state (the inicial state) to *waiting username* state. The output event associated with this transition is *Get username.*

**1.3:** *Username manager* sends the *Get username* event to *createNewUser* screen..

**1.4:** *CreateNewUser* screen displays the username prompt to the *User* actor.

**2:** *User* inputs the username and name to the *createNewUser* screen.

**2.1:** The input data is made thru the *keyboardInterface* object

**2.2:** *CreateNewUser* screen sends the *user's information*, containing *username* and *name* to the *UsernameManager* control.

**2.3:** The *username manager* sends a *valid username* request to *UserDataControl*. This causes a transition from *waiting username* to *validating username* state. The output event associated with this transition is *validate username*.

**2.4:** *UserDataControl* validates the username and sends a *valid username* response to the *Username manager*. As result of this event transition changes to *waiting User's Choice* state. The output events associated for this transition are *display menu* and *load user's data*.

**2.5:** *Username Manager* sends the *display menu* event to the *menu* screen.

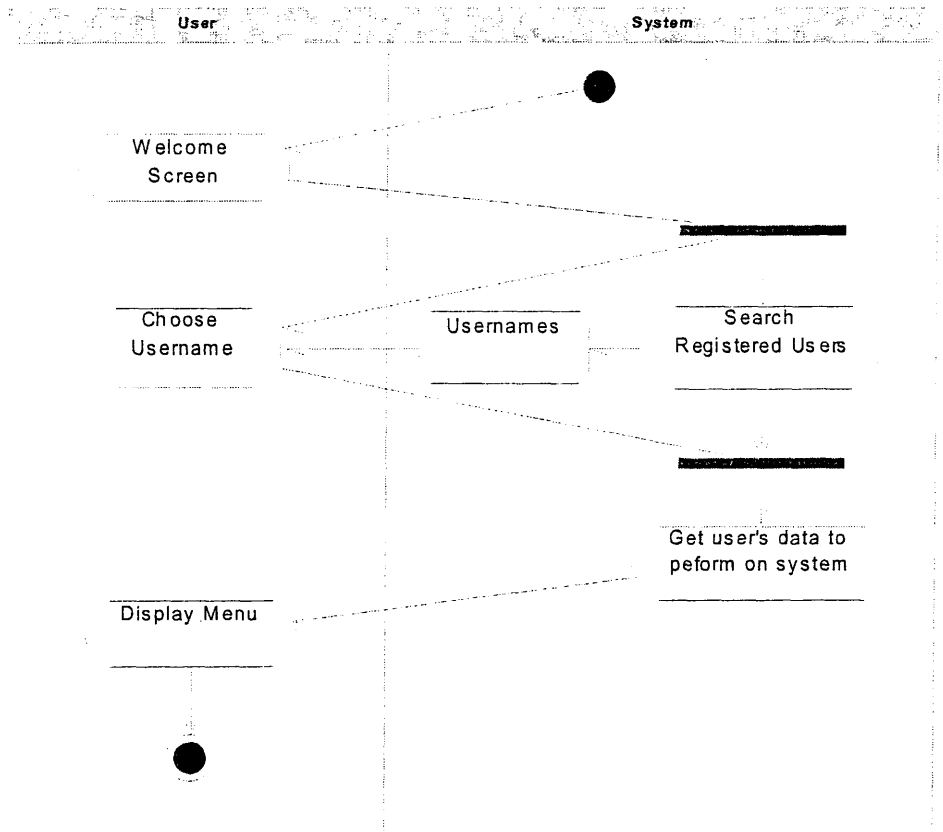
**2.5a:** *Username Manager* sends a *load user's data* message to the *UsersData* entity.

**2.5b:** *UsersData* entity *display name* related to username at *Menu* screen.

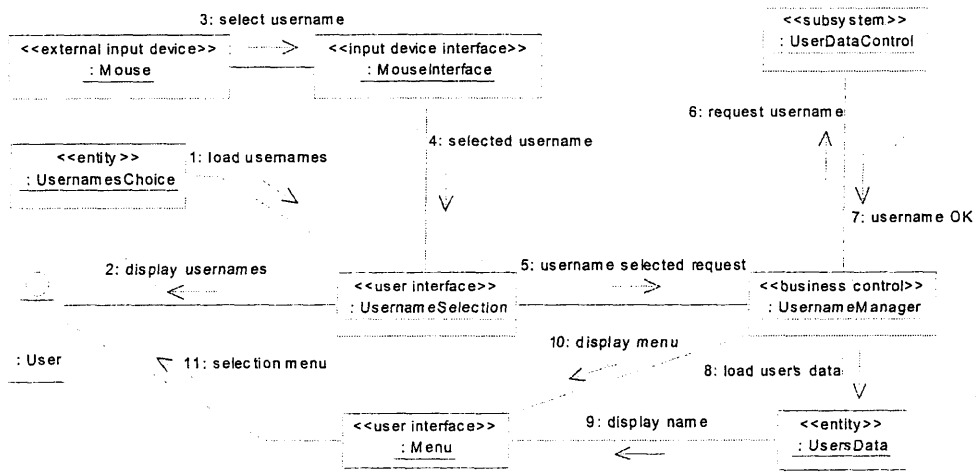
**2.6:** *Menu* screen displays user's name and a menu showing learn sound options to *User* actor.

## 3.6.2 Validate User Use Case Diagrams

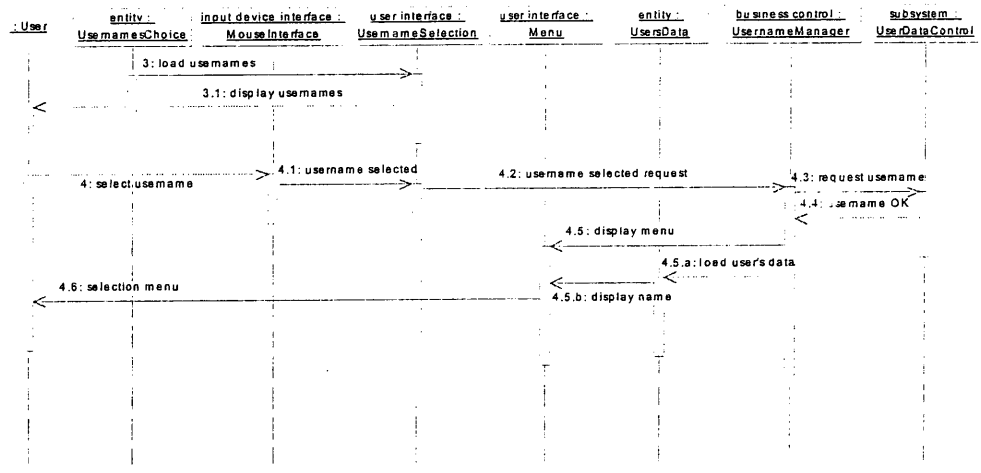
### 3.6.2.1 Activity Diagram



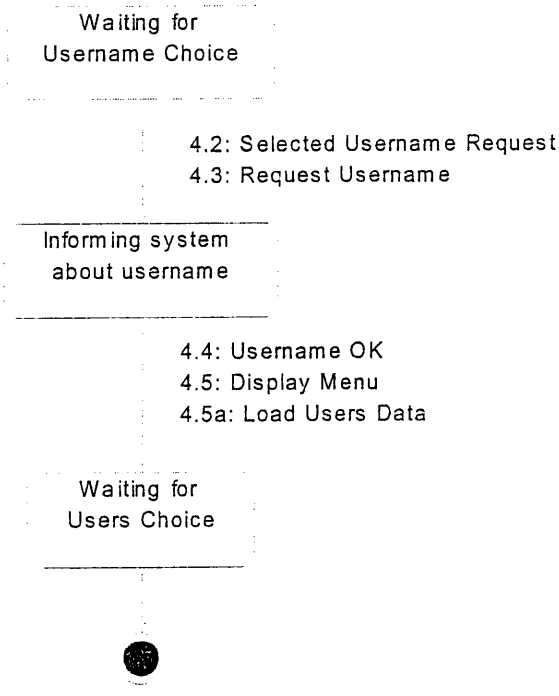

### 3.6.2.2 Collaboration Diagram



### 3.6.2.3 Sequence Diagram




### 3.6.2.4 Statechart Diagram



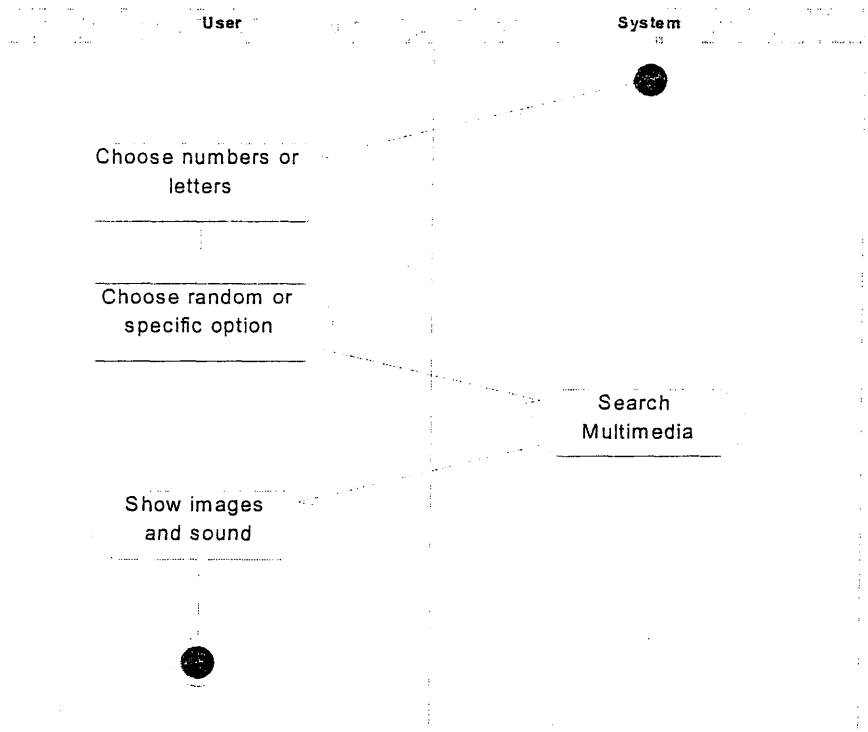
### 3.6.2.5 Sequence Description

**3:** The entity *Usernames Choice* load usernames into *Username* screen.
**3.1:** The *username* screen displays all usernames to the *User* actor.

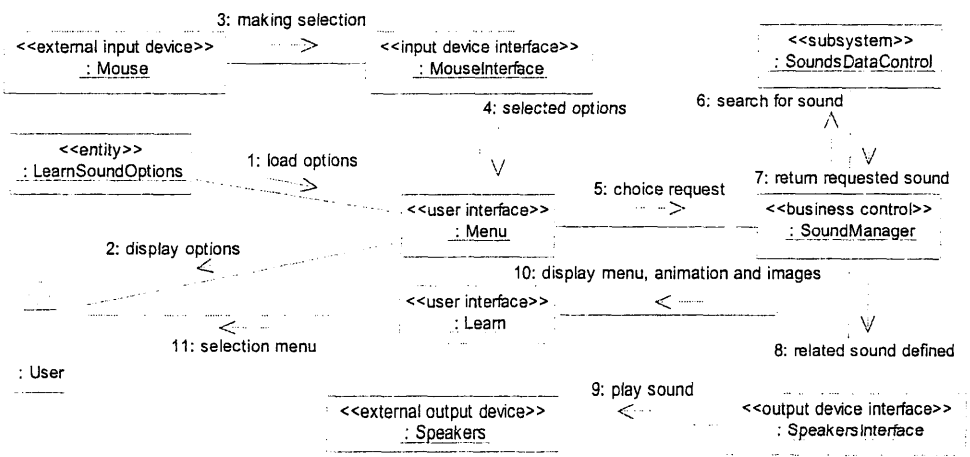
**4:** The *User* chooses one of the username displaying.
**4.1:** The *MouseInterface* object calls event *username selected request.*
**4.2:** *UsernameSelection* screen sends the *username selected request* to *Username Manager* Control.
**4.3:** The *Username Manager* sends the *request of a username* to *UserDataControl*. As result the statechart transits from *Idle* state (the initial state) to *informing system about username* state. The output event associated with this transition is *Request Username*.
**4.4:** *UserDataControl* validates the username and sends an *OK response* to *Username Manager* control. As result for the event, *Informing System about username* transition to *waiting for user's choice* state. The outputs for this event are *display menu* and *load user's data*.
**4.5:** *Username Manager* sends the *display menu* event to *menu* screen.
**4.5a:** *Username Manager* sends a *load user's data* message to the *usersData* entity.
**4.5b:** *UsersData* entity *display name* related to username at *Menu* screen.
**4.6:** *Menu* screen displays user's name and a menu showing learn sound options to *User* actor.

## 3.6.3 Learn Sound Use Case Diagrams

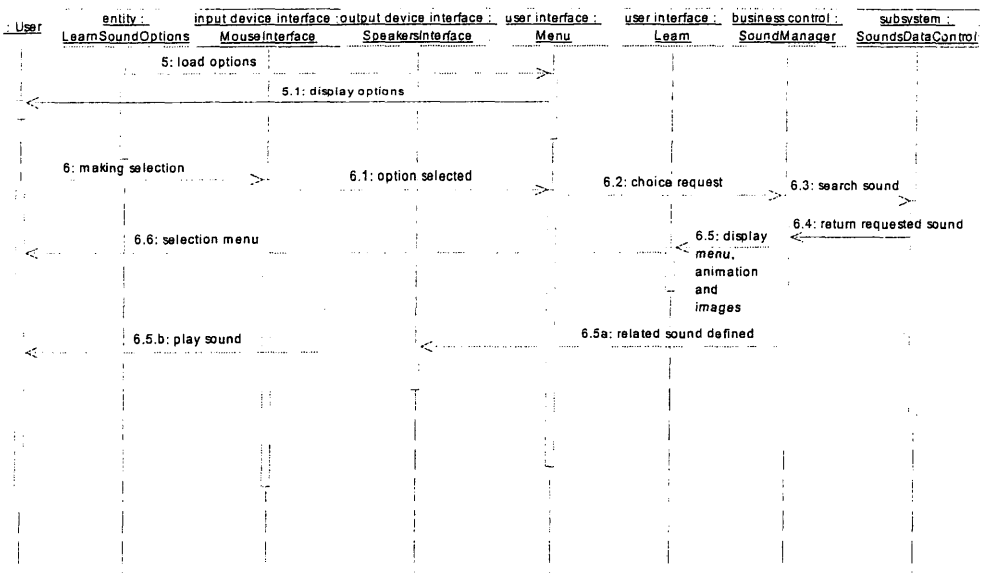
### 3.6.3.1 Activity Diagram



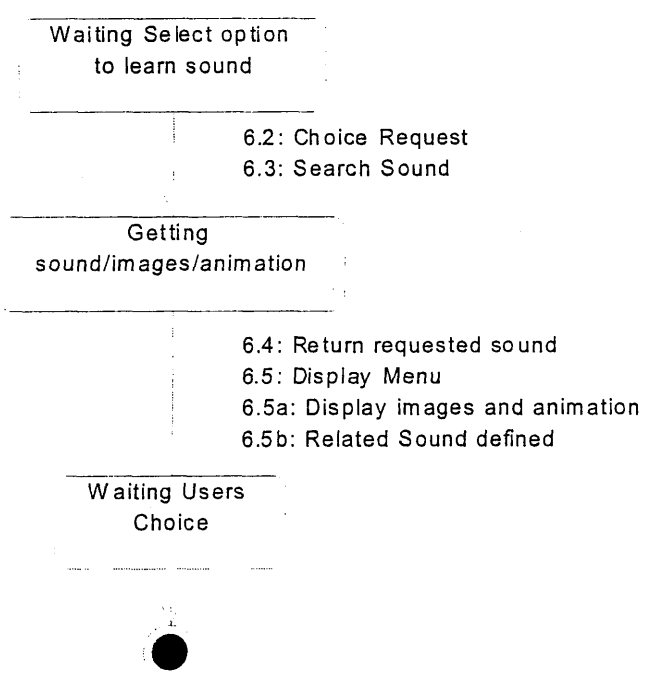

### 3.6.3.2 Collaboration Diagram



### 3.6.3.3 Sequence Diagram




### 3.6.3.4 Statechart Diagram



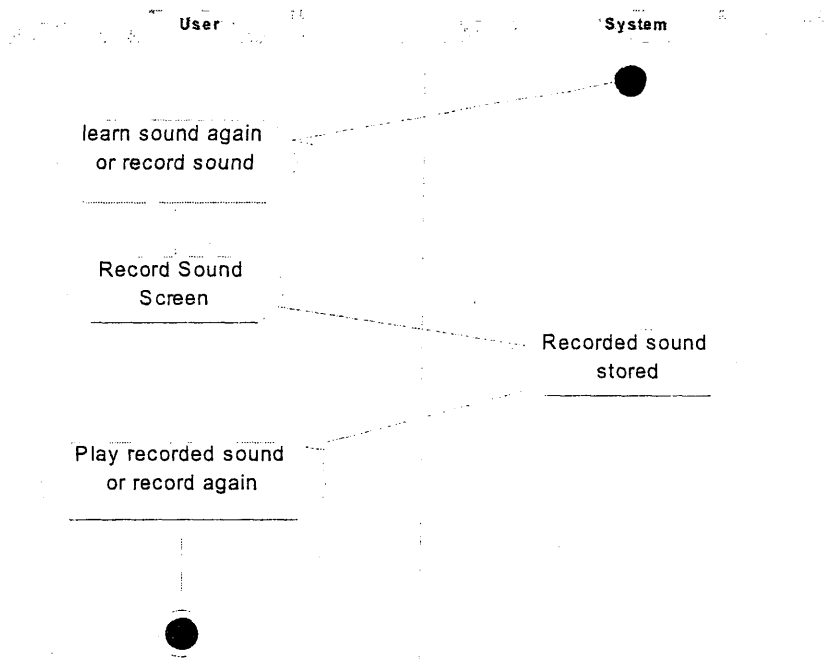
### 3.6.3.5 Sequence Description

**5:** The entity *LearnSound Options* load usernames into *Menu* screen.
**5.1:** The *menu* screen displays possible options to learn sound to the *User* actor.

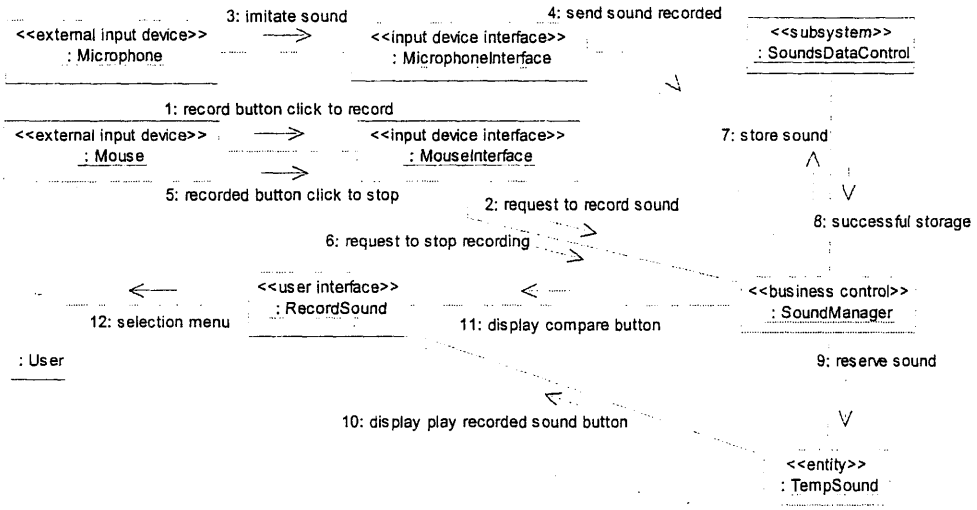
**6:** The *User* makes the selections of letter or numbers, alphabetic/ordered or specific one.
**6.1:** The *MouseInterface* object calls event *option selected.*
**6.2:** *Menu* screen sends the *choice request* to *Sound Manager* Control.
**6.3:** The *Sound Manager* sends the request to *search sound* to *SoundDataControl.* As result statechart transits from *Idle* state (the initial state) to *getting sound/images/animation* state. The output event associated with this transition is *Search Sound.*
**6.4:** *SoundDataControl* returns the *requested sound* to *Sound Manager* control. As result for the event, *Getting sound/images/animation* transits to *waiting for user's choice* state. The outputs for this event are *display menu, display image and animation* and *Related sound defined.*
**6.5:** *Sound Manager* sends the *display menu, animation and images* event to *learn* screen.
**6.5a:** *Sound Manager* sends the *related sound* message to *SpeakersInterface.*
**6.5b:** The *SpearkersInterface* object *plays sound* to *User.*
**6.6:** *Learn* screen displays information about the letter/number choosen and record option to *User* actor.

## 3.6.4 Record Sound Use Case Diagrams

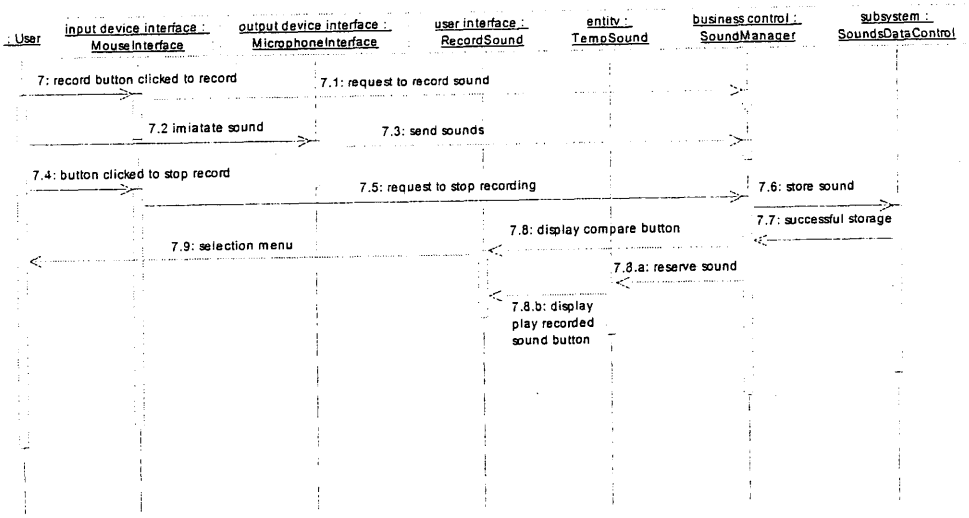
### 3.6.4.1 Activity Diagram



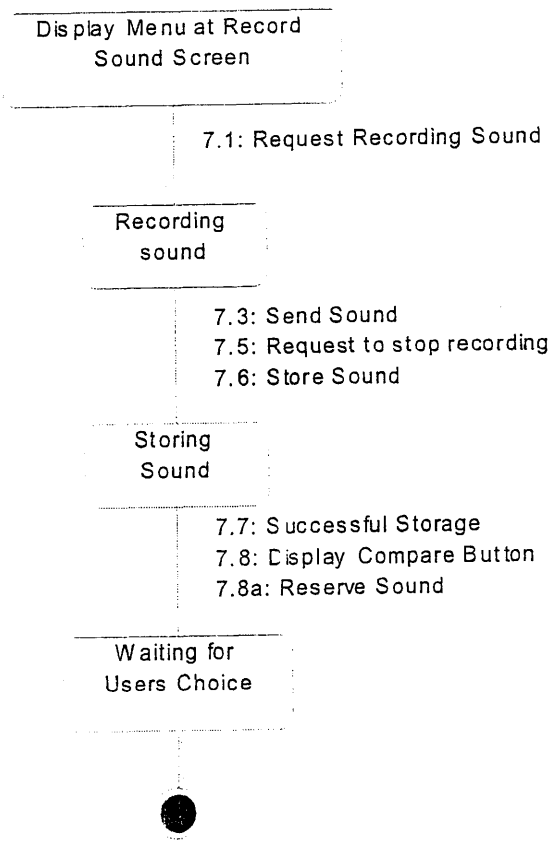

### 3.6.4.2 Collaboration Diagram



### 3.6.4.3 Sequence Diagram




### 3.6.4.4 Statechart Diagram



### 3.6.4.5 Sequence Description

**7:** The *User* actor clicks the button to record sound.

**7.1:** The *MouseInterface* object calls event *request recorded sound*. The statechart state changes from *Idle* state (the initial state) to *Recording Sound* state. The output for this event is *Request Recording Sound*.

**7.2:** The *User* starts to *imitate sound* on the microphone device.

**7.3:** The *MicrophoneInterface* object *send sounds* to *Sound Manager* Control.

**7.4:** The *User* clicks again button to stop record.

**7.5:** The *MouseInterface* object *request to stop recording* to *Record Sound Manager* Control.

**7.6:** The *Sound Manager* sends the request to *store sound* to *SoundDataControl*. As result statechart have transition from *Recording Sound* state to *storing sound* state. The output event associated with this transition is *Store Sound*.

**7.7:** *SoundDataControl* store the sound and send a message of *successful storage* to *Sound Manager* control. As result for the event, *Storing Sound* transits to *waiting for user's choice* state. The outputs for this event are *display compare button, reserve sound* and *Successful Storage*.

**7.8:** *Sound Manager* sends the *display compare button* event to *RecordSound* screen.

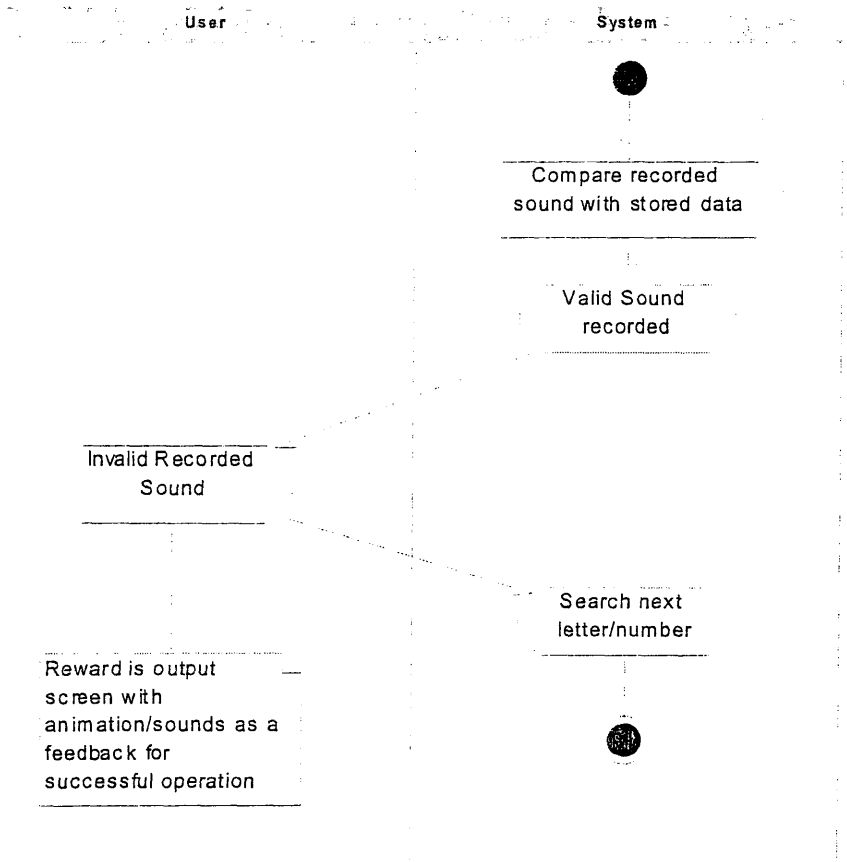
**7.8a:** *Sound Manager* sends to *TempSound* entity the request to *reserve sound* .

**7.8b:** *TempSound* sends the information to *display play recorded button* to *Record Sound* screen.

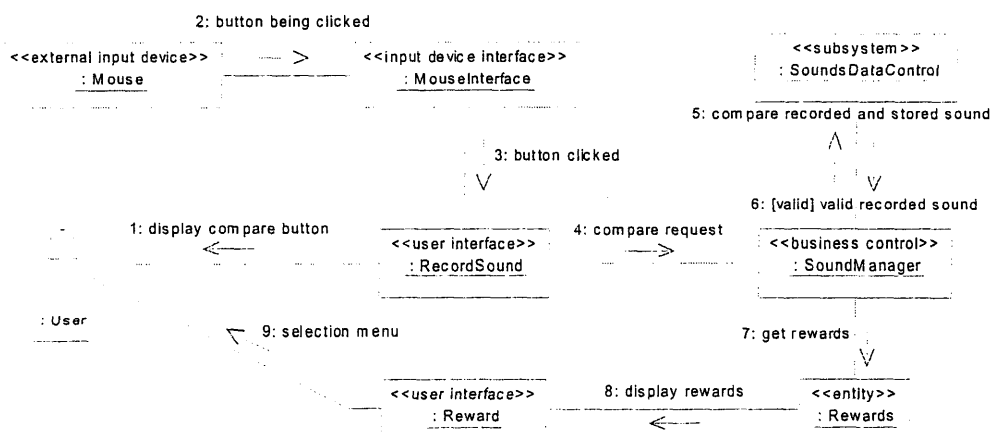
**7.9:** *RecordSound* screen displays record option as play sound again; see animation and buttons available after recording sound to *User* actor.

## 3.6.5 Compare Sound Use Case [valid] Diagrams

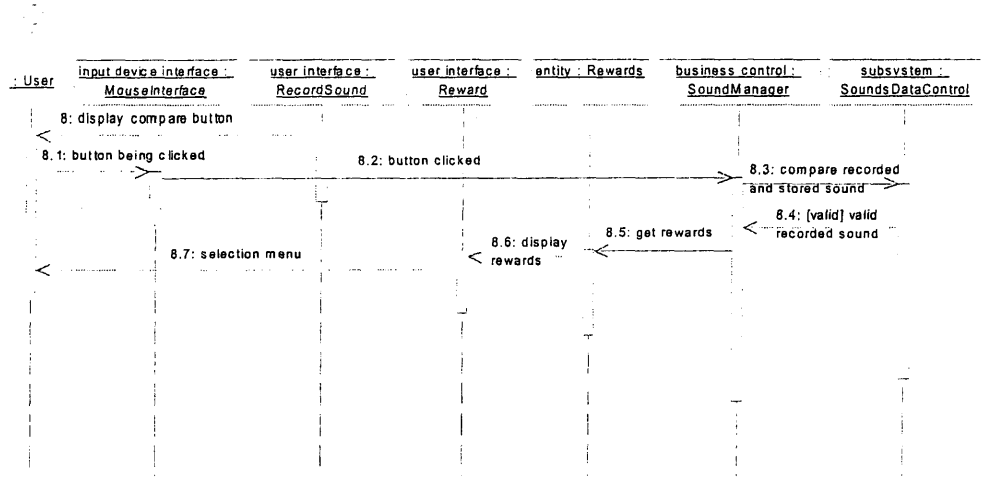
### 3.6.5.1 Activity Diagram



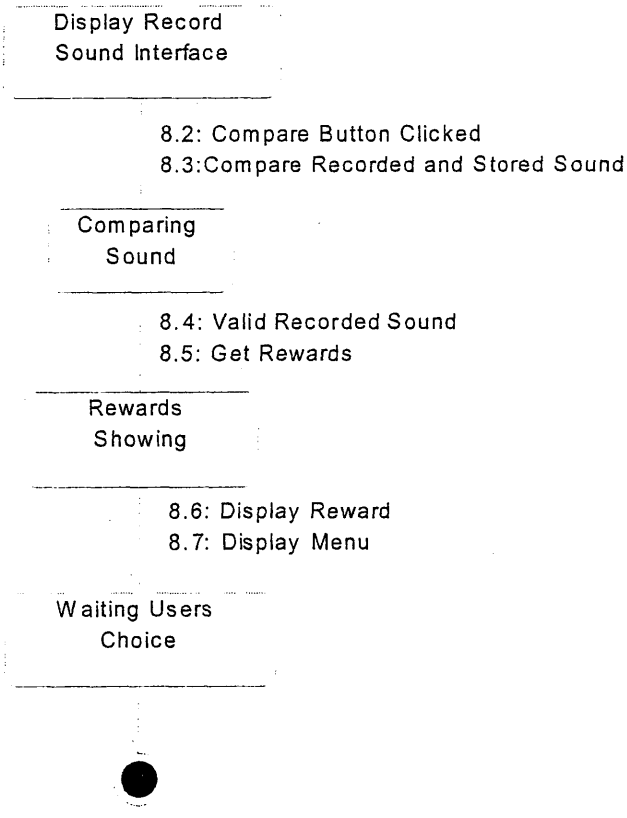

### 3.6.5.2 Collaboration Diagram



### 3.6.5.3 Sequence Diagram




### 3.6.5.4 Statechart Diagram



### 3.6.5.5 Sequence Description

**8:** The *RecordSound* screen has able the compare button.

**8.1:** The *User* actor clicks the button to compare sounds.

**8.2:** The *MouseInterface* object sends event *button clicked* to *Sound Manager* Control.

**8.3:** The *sound Manager* Control requests the *SoundDataControl* to *compare recorded and stored sound.* The statechart state changes from *Idle* state (the initial state) to *Comparing Sound* state. The output for this event is *compare recorded and stored sound.*

**8.4:** *SoundDataControl* validates the recorded sound and send a message of *valid recorded sound* to *Sound Manager* control. As result for the event, *Comparing Sound* transits to *Rewards Showing* state. The outputs for this event are *valid recorded sound* and *Get Rewards.*

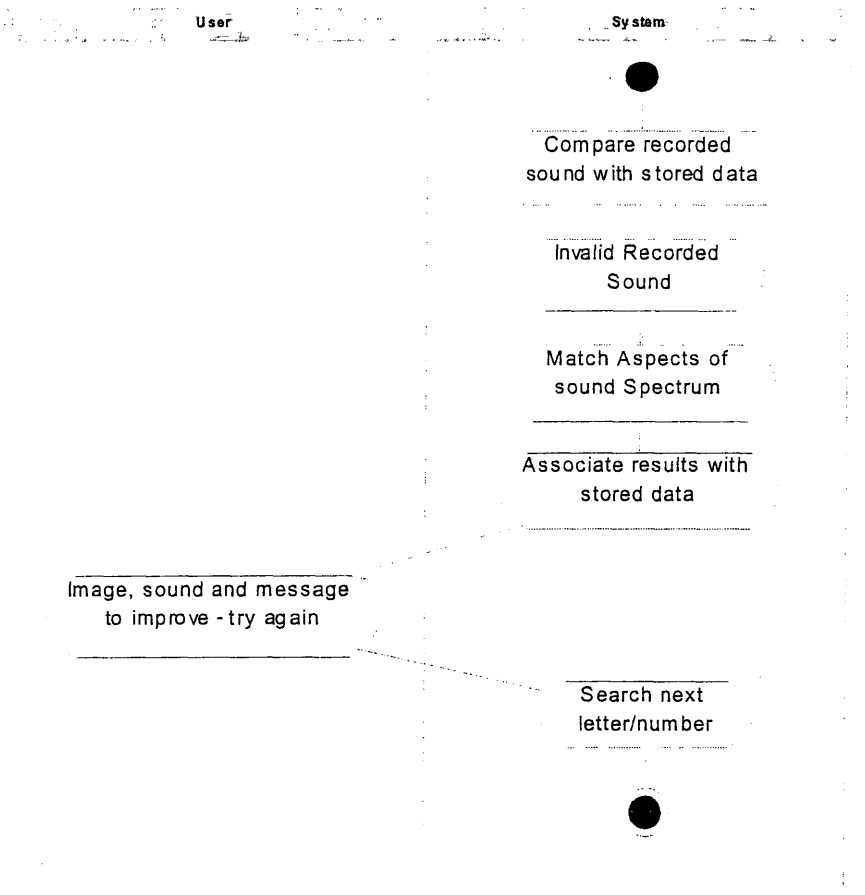
**8.5:** *Sound Manager* sends information to *get rewards* to *Rewards* entity. This event results in transition from *Rewards showing* to *Waiting User's Choice* state. The outputs for this event are *display Rewards* and *display menu.*

**8.6:** *Rewards* entity sends to *Reward* screen the information about prides.

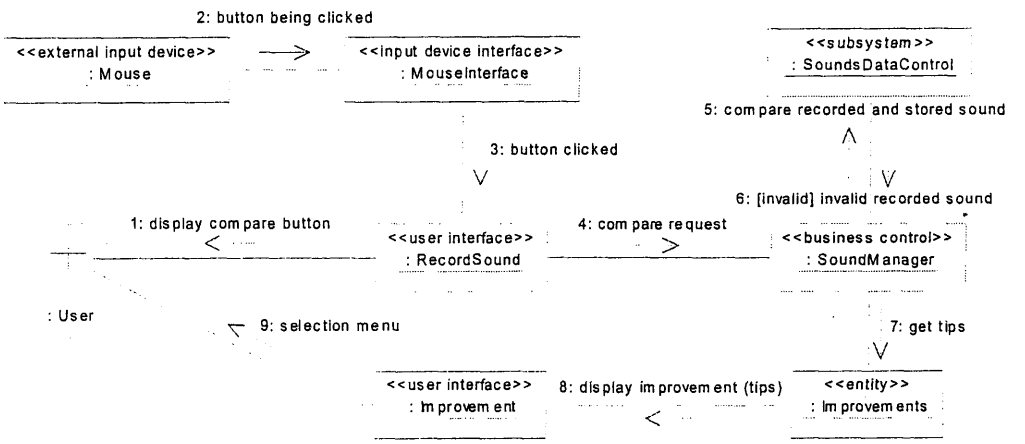
**8.7:** *Reward* screen displays a menu showing options like hear again this sound, go to next one and exit to *User* actor.

## 3.6.6 Compare Sound Use Case [invalid] Diagrams

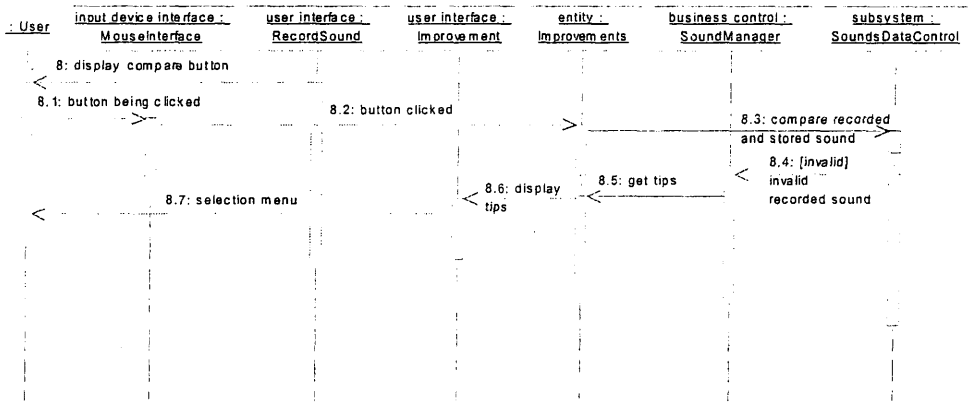
### 3.6.6.1 Activity Diagram



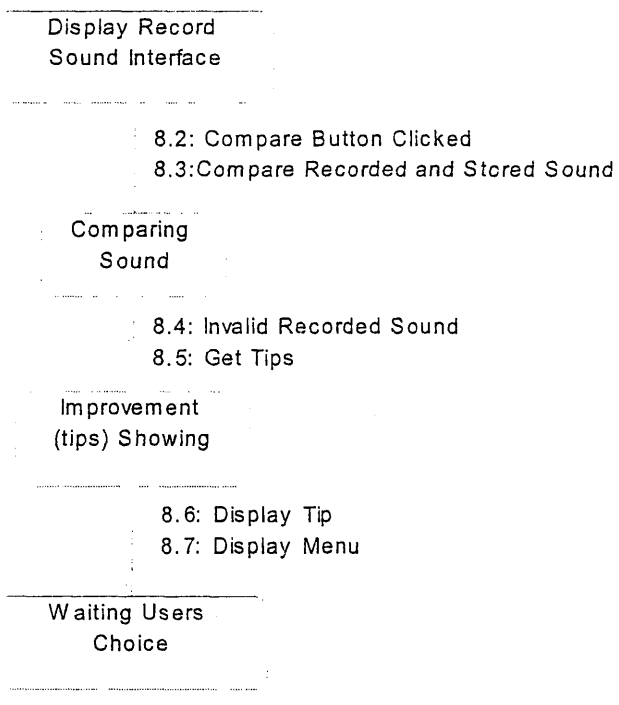

### 3.6.6.2 Collaboration Diagram



### 3.6.6.3 Sequence Diagram




### 3.6.6.4 Statechart Diagram



### 3.6.6.5 Sequence Description

**9:** The *RecordSound* screen has able the compare button.

**9.1:** The *User* actor clicks the button to compare sounds.

**9.2:** The *MouseInterface* object sends event *button clicked* to *Sound Manager* Control.

**9.3:** The *sound Manager* Control requests the *SoundDataControl* to *compare recorded and stored sound*. The statechart state changes from *Idle* state (the initial state) to *Comparing Sound* state. The output for this event is *compare recorded and stored sound.*

**9.4:** *SoundDataControl* invalidates the recorded sound and send a message of in*valid recorded sound* to *Sound Manager* control. As result for the event, *Comparing Sound* transits to *Improvement (tips) Showing* state. The outputs for this event are *valid recorded sound* and *Get Tips*.

**9.5:** *Sound Manager* sends information to *get tips* to *Improvements* entity. This event results in transition from *Improvement (tips) showing* to *Waiting User's Choice* state. The outputs for this event are *display Tips* and *display menu.*

**9.6:** *Improvements* entity sends to *Improvement* screen the information about how to improve the imitation of sound. For example, more loud, more slow, etc.

**9.7:** *Improvement* screen displays a menu showing options try again, learn again sound, go to next one and exit to *User* actor.

# 4. Software Development

## 4.1 Classes

ConveyBox.java
JSplash.java
ControlContext.java
CapturePlayback.java
PlayerPanel.java

## 4.2 Source Code

```
/**
 * Convey Box System
 *
 *       Software for hearing handcap children with hearing loss
 *
 * @version @(#)ConveyBox.java     1.0      03/04/23
 * @author Catarina Yamamoto
 */

package Box;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class ConveyBox extends JPanel implements ActionListener, KeyListener{

  // Declaring variables
  static private CardLayout cardManager;
  static private JPanel cardPanel;
  static private JLabel iconSound;
  static private String nowLearning;
  private JPanel topPanel, bottomPanel, mouthLetterLearnPanel;
  private JLabel variavel, signLanguage;
  private JRadioButton abc, qwe;
  private ButtonGroup radioGroup;
  private JButton letters[] = new JButton[26];
  private JButton numbers[] = new JButton[10];
  private Color colors[] ={Color.green, Color.black, Color.magenta, Color.red, Color.blue};
  private String alphabet[] =
{"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","
X","Y","Z"};
  private String keyboard[] =
{"Q","W","E","R","T","Y","U","I","O","P","A","S","D","F","G","H","J","K","L","Z","X","C","V","
B","N","M"};
```

```
static private String path = "C:/Program Files/JCreator LE/MyProjects/convey box/";
private int width = 800, height = 580;
private CapturePlayback capturePlayback = new CapturePlayback();
private PlayerPanel mouthMove = new PlayerPanel(true, false);

public ConveyBox() {

        setLayout(new BorderLayout());

 // Create and configure the main panel
 topPanel = new JPanel () ;
 topPanel.setLayout(new BorderLayout ()) ;
 topPanel.setBorder(BorderFactory.createEmptyBorder(15,15,5,15));
                                                          //(top, left, bottom, right)

 // Create the cardlayout panel
 cardPanel = new JPanel () ;
 cardManager = new CardLayout();
 cardPanel.setLayout( cardManager );

 /*
  * set up Card1
  *
  */
 JLabel logo = new JLabel(new ImageIcon(path + "images/logo.gif"),
SwingConstants.CENTER);

 JLabel labelQuestion = new JLabel( "Hello, which sound would you like to learn
today?", SwingConstants.CENTER );
 JLabel labelBlank = new JLabel( " ", SwingConstants.CENTER );

 JPanel card1 = new JPanel(new BorderLayout ());

 JPanel top = new JPanel(new BorderLayout ());
 top.add(logo, BorderLayout.NORTH);
 top.add(labelQuestion, BorderLayout.CENTER);
 top.add(labelBlank, BorderLayout.SOUTH);

 TitledBorder lettersTitled = new TitledBorder("Letters");
 TitledBorder numbersTitled = new TitledBorder("Numbers");
 JPanel lettersPanel = new JPanel();
 lettersPanel.setLayout(new GridLayout(4,1));
 lettersPanel.setBorder(lettersTitled);

 //panel lines for keyboard letters
 JPanel rowOnePanel = new JPanel();
 JPanel rowTwoPanel = new JPanel();
 JPanel rowThreePanel = new JPanel();
 JPanel rowFourPanel = new JPanel();

 JPanel numbersPanel = new JPanel();
```

```
    numbersPanel.setBorder(numbersTitled);

    //letters buttons
    for (int i = 0; i < (letters.length); i++ ){
                letters[i] = new JButton(new ImageIcon(path + "images/letters_numbers/"
+ alphabet[i].toLowerCase() +"2.gif"));
                letters[i].setActionCommand(alphabet[i]);
                letters[i].setBackground(Color.black);
                letters[i].addActionListener(this);
                letters[i].addKeyListener(this);
                if (i<10)
                        rowOnePanel.add(letters[i]);
                else if ( i > 9 & i < 19)
                                rowTwoPanel.add(letters[i]);
                        else
                                rowThreePanel.add(letters[i]);
        }

        lettersPanel.add(rowOnePanel);
        lettersPanel.add(rowTwoPanel);
        lettersPanel.add(rowThreePanel);

        //numbers buttons
        for (int i = 0; i < (numbers.length); i++ ){
                numbers[i] = new JButton(new ImageIcon(path +
"images/letters_numbers/" + i +"2.gif"));
                numbers[i].setActionCommand(i+"");
                numbers[i].setBackground(Color.white);
                numbers[i].addActionListener(this);
                numbers[i].addKeyListener(this);
                numbersPanel.add(numbers[i]);
        }


        //RadioButtons
        //alphabet
        abc = new JRadioButton("ABC", true);
        abc.setActionCommand("order");
        //keyboard
        qwe = new JRadioButton("QWE", false);
        qwe.setActionCommand("noorder");

        //logic relation between buttons
        radioGroup = new ButtonGroup();
        radioGroup.add(abc);
        radioGroup.add(qwe);

        //register events of Radio Buttons
        ActionListener handler = new RadioButtonHandler();
        abc.addActionListener(handler);
        qwe.addActionListener(handler);
```

```
        //Add radioButton to lettersPanel
        rowFourPanel.add(abc);
        rowFourPanel.add(qwe);
        lettersPanel.add(rowFourPanel);

    //adding panels to card
    JPanel center = new JPanel(new BorderLayout ());
    center.add( lettersPanel, BorderLayout.CENTER );
    center.add( numbersPanel, BorderLayout.SOUTH );

    //create random button
        JButton randomButton = new JButton("RANDOM");
        randomButton.setActionCommand("random");
        randomButton.setMnemonic(KeyEvent.VK_R);
        randomButton.setFont( new Font("Arial", Font.BOLD, 15 ) );
        randomButton.addActionListener(this);
        randomButton.addKeyListener(this);
        randomButton.setBackground( Color.white );
        //randomButton.setForeground( Color.black );

        JPanel randomPanel = new JPanel(new BorderLayout ());
        randomPanel.setBorder(BorderFactory.createEmptyBorder(8,1,5,1));
        randomPanel.add(randomButton);

    card1.add( top, BorderLayout.NORTH );
    card1.add( center, BorderLayout.CENTER );
    card1.add( randomPanel, BorderLayout.SOUTH );

    cardPanel.add( card1, "card1");


    /*
     * set up Card2
     *
     */
    JLabel labelLearnRecord = new JLabel( "LET'S LEARN THE SOUNDS",
  SwingConstants.LEFT );
    labelLearnRecord.setFont( new Font("Verdana", Font.PLAIN, 18 ) );
        labelLearnRecord.setForeground( Color.black );

        //Control Menu for card 2
        JButton playSound2 = new JButton("LISTEN AGAIN", new ImageIcon(path +
 "images/speaker.gif"));
        playSound2.setActionCommand("listenAgain");
        playSound2.setBackground( Color.yellow );
        playSound2.addActionListener( this );

    JButton voltar2 = new JButton("GO BACK", new ImageIcon(path +
  "images/voltar.gif"));
    voltar2.setActionCommand("goBack");
```

```
    voltar2.setBackground( Color.yellow );
    voltar2.addActionListener( this );

    JButton next2 = new JButton("NEXT", new ImageIcon(path + "images/next.gif"));
    next2.setActionCommand("next");
    next2.setBackground( Color.yellow );
    next2.addActionListener( this );

    JButton random2 = new JButton("RANDOM", new ImageIcon(path +
"images/rando.gif"));
    random2.setActionCommand("random");
    random2.setBackground( Color.yellow );
    random2.addActionListener( this );

    JPanel card2 = new JPanel(new BorderLayout() );
    JPanel tituloLearn = new JPanel( );
    JPanel imagem = new JPanel( new BorderLayout() );
    JPanel control = new JPanel();
    control.setLayout(new GridLayout(4,1));
    JPanel central = new JPanel(new BorderLayout());
    JPanel recordBar = new JPanel();
    recordBar.setLayout(new GridLayout(1,1));

    //Adding sign language icon
    signLanguage = new JLabel( "", SwingConstants.CENTER );
    imagem.setBackground( Color.yellow );
    imagem.add( signLanguage , BorderLayout.CENTER );

    //left Panel contains empty border and mouth/letter panel
    JPanel leftLearnPanel = new JPanel();
    leftLearnPanel.setLayout(new BorderLayout());

    JPanel leftEmptyLearnPanel = new JPanel();
    leftEmptyLearnPanel.setBorder(new EmptyBorder(5,5,5,5));
    leftEmptyLearnPanel.setBackground( Color.yellow );

    //Label for actual learning sound (letter/number)
    variavel = new JLabel( "", SwingConstants.CENTER );
    variavel.setVerticalTextPosition(JLabel.BOTTOM);
    variavel.setFont( new Font("Verdana", Font.BOLD , 180 ) );

        //Mouth Moviment Panel
    mouthLetterLearnPanel = new JPanel(new BorderLayout());

    mouthLetterLearnPanel.setBackground( Color.yellow );
    mouthLetterLearnPanel.add(mouthMove, BorderLayout.CENTER);
    mouthLetterLearnPanel.add(variavel, BorderLayout.SOUTH);

    JLabel a= new JLabel(new ImageIcon(path + "images/line.gif"));
    mouthLetterLearnPanel.add(a);
```

```
    leftLearnPanel.add(leftEmptyLearnPanel, BorderLayout.WEST);
    leftLearnPanel.add(mouthLetterLearnPanel, BorderLayout.EAST);

    control.add(playSound2);
    control.add(voltar2);
    control.add(next2);
    control.add(random2);

    tituloLearn.setBackground( Color.yellow );
    tituloLearn.add( labelLearnRecord );

    recordBar.add(capturePlayback);

    central.add(leftLearnPanel, BorderLayout.WEST );
    central.add(imagem, BorderLayout.CENTER );
    central.add(control, BorderLayout.EAST );

    card2.add( central, BorderLayout.CENTER );
    card2.add( tituloLearn , BorderLayout.NORTH);
    card2.add( recordBar , BorderLayout.SOUTH);

    cardPanel.add( card2, "card2");

    /*
     * set up Card3
     *
     */
    JLabel labelComparation = new JLabel( "COMPARATION RESULT: ",
SwingConstants.CENTER );
    JLabel labelResult = new JLabel( "GOOD JOB / TRY AGAIN",
SwingConstants.CENTER );
    JPanel central3 = new JPanel();
    JPanel card3 = new JPanel();

    //Control Menu for card 3
    JButton playSound3 = new JButton("LISTEN AGAIN", new ImageIcon(path +
"images/speaker.gif"));
        playSound3.setActionCommand("listenAgain");
        playSound3.setBackground( Color.yellow );
        playSound3.addActionListener( this );

        JButton playSoundMe3 = new JButton("LISTEN ME", new ImageIcon(path +
"images/speaker.gif"));
        playSoundMe3.setActionCommand("listenMe");
        playSoundMe3.setBackground( Color.yellow );
        playSoundMe3.addActionListener( this );

    JButton voltar3 = new JButton("GO BACK", new ImageIcon(path +
"images/voltar.gif"));
    voltar3.setActionCommand("goBack");
    voltar3.setBackground( Color.yellow );
```

```
    voltar3.addActionListener( this );

    JButton next3 = new JButton("NEXT", new ImageIcon(path + "images/next.gif"));
    next3.setActionCommand("next");
    next3.setBackground( Color.yellow );
    next3.addActionListener( this );

    JButton random3 = new JButton("RANDOM", new ImageIcon(path +
"images/rando.gif"));
    random3.setActionCommand("random");
    random3.setBackground( Color.yellow );
    random3.addActionListener( this );

    iconSound = new JLabel("");

    JPanel control3 = new JPanel();
    control3.setLayout(new GridLayout(5,1));
    control3.add(playSound3);
    control3.add(playSoundMe3);
    control3.add(voltar3);
    control3.add(next3);
    control3.add(random3);

    central3.add(labelComparation);
    central3.add(iconSound);
    central3.add(labelResult);

    card3.setLayout( new BorderLayout() );
    card3.add( central3, BorderLayout.CENTER );
    card3.add( control3, BorderLayout.EAST );
    cardPanel.add( card3, "card3" );


    // Create exit Panel
    JButton exit = new JButton("EXIT", new ImageIcon(path + "images/door.gif"));
    exit.setVerticalTextPosition(SwingConstants.CENTER);
    exit.setHorizontalTextPosition(SwingConstants.RIGHT);

    exit.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
                exitProgram();
          }
    });

    JPanel exitPanel = new JPanel () ;
    FlowLayout exitLayout = new FlowLayout ();
    exitLayout.setAlignment(FlowLayout.RIGHT);
          exitPanel.setLayout(exitLayout) ;
          exitPanel.add(exit);

    JButton about = new JButton("?");
```

```
   about.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                String textAbout = "Convey Box: Into a Speaking World \n Version 1.0 -
2003"+
                                                "\n \n Catarina Yamamoto\n
a_flama@hotmail.com \n"+
                                                "JICA - Japan International Cooporation
Agency \n \n" +
                                                "Koichiro Ochimizu \n
ochimizu@jaist.ac.jp \n" +
                                                "JAIST - Japan Advaced Institute of
Science and Technology \n"+
                                                "";
                JOptionPane.showMessageDialog(null, textAbout, "ABOUT",
JOptionPane.INFORMATION_MESSAGE);
        }
   });
   //about.setForeground(Color.blue);

   JPanel aboutPanel = new JPanel () ;
   FlowLayout aboutLayout = new FlowLayout ();
   aboutLayout.setAlignment(FlowLayout.LEFT);
        aboutPanel.setLayout(aboutLayout) ;
        aboutPanel.add(about);

   bottomPanel = new JPanel () ;
   bottomPanel.setLayout( new BorderLayout() );

   bottomPanel.add(aboutPanel, BorderLayout.WEST);
   bottomPanel.add(exitPanel, BorderLayout.EAST);

   //Adding cardPanel and exitPanel to topPanel
   topPanel.add(cardPanel, BorderLayout.CENTER) ;
   topPanel.add(bottomPanel, BorderLayout.SOUTH) ;

        add(topPanel, BorderLayout.CENTER);

 }


   public void keyPressed(KeyEvent evt){}

        public void keyReleased(KeyEvent evt){}

        public void keyTyped(KeyEvent evt){
                char keyChar = Character.toUpperCase(evt.getKeyChar());
                nowLearning = keyChar + "";

                if (keyChar >= 'A' && keyChar <= 'Z')
                        eventButton(keyChar + "");
                else if (keyChar >= '0' && keyChar <= '9')
```

```
                    eventButton(keyChar + "");

                //make ALT + R keyfunction work

        }


    public void actionPerformed( ActionEvent e ){

                eventButton(e.getActionCommand());
        }


  void eventButton(String command) {
        //If it's a letter
        if
(command.equals("A")||command.equals("B")||command.equals("C")||command.equals("
D")||command.equals("E")

||command.equals("F")||command.equals("G")||command.equals("H")||command.equals(
"I")||command.equals("J")

||command.equals("K")||command.equals("L")||command.equals("M")||command.equals(
"N")||command.equals("O")

||command.equals("P")||command.equals("Q")||command.equals("R")||command.equals(
"S")||command.equals("T")

||command.equals("U")||command.equals("V")||command.equals("W")||command.equals
("X")||command.equals("Y")
            ||command.equals("Z")) learnSound(command);
        //If it's a number
        if
(command.equals("0")||command.equals("1")||command.equals("2")||command.equals("
3")||command.equals("4")

||command.equals("5")||command.equals("6")||command.equals("7")||command.equals("
8")||command.equals("9")
            ) learnSound(command);
        //If random is choosen
        if (command.equals("random")) variavelRandom();
//JOptionPane.showMessageDialog(null, "Random");
        //If want to return to main screen
        if (command.equals("goBack")) {
                capturePlayback.resetButtons();
                cardManager.show(cardPanel, "card1"); // show first card on click =
cardManager.first( cardPanel );
        }
        if (command.equals("listenMe")) capturePlayback.playback.start();
        if (command.equals("listenAgain")) {
                                videoMouth(nowLearning);
```

```
                    }
        if (command.equals("next")){
                int i = 0;
                capturePlayback.resetButtons();

                if (Character.isDigit(nowLearning.charAt(0))) {

                        // start for numbers
                        for (i = 0; i < 10; i++ )
                                if (nowLearning.charAt(0) == 57){
                                        learnSound("A");
                                        break;
                                }else{
                                        learnSound(((nowLearning.charAt(0)-48)+1) +"");

                                        break;
                                }
                        //end for

                }else{
                        // start for letters
                        for (i = 0; i < (letters.length); i++ )
                                if (nowLearning.equals(alphabet[i]))
                                        if (alphabet[i].toUpperCase() =="Z"){
                                                learnSound("0");
                                                break;
                                        }
                                else{
                                                learnSound(alphabet[i+1]);
                                                break;
                                        }
                        // end for
                }

                }
 }

 void learnSound(String tecla){

        nowLearning = tecla;
        int x=(int)(Math.random()*5);
        variavel.setText( tecla );
        variavel.setForeground( colors[x] );
        signLanguage.setIcon(new ImageIcon(path + "images/letters_numbers/" +
tecla.toLowerCase() +".gif"));
        cardManager.show(cardPanel, "card2");
        videoMouth(tecla);

 }

 void videoMouth(String tecla){
```

```
        mouthMove.loadFile(path + "video/"+ tecla +".AVI");
 }


 void variavelRandom(){

        capturePlayback.resetButtons();

        int x=(int)(Math.random()*35);          //between 0 and 35
                                                                    //if start with 1 ->
Math.random()*4+1
                                                                    //ex between 100 and
500 -> Math.random()*400+100
        if (x > 9)
                learnSound(alphabet[(x-10)]);
        else
                learnSound(x+"");

 }


 static void turnPanel(){

        iconSound.setIcon(new ImageIcon(path + "images/letters_numbers/" +
nowLearning.toLowerCase() +"2.gif"));
        cardManager.show(cardPanel, "card3");

 }


 class RadioButtonHandler implements ActionListener{
        public void actionPerformed ( ActionEvent ex){
                if (radioGroup.getSelection().getActionCommand() == "order")
                        for (int i = 0; i < (letters.length); i++ ){

                                letters[i].setIcon(new ImageIcon(path +
"images/letters_numbers/" + alphabet[i].toLowerCase() +"2.gif"));
                                letters[i].setActionCommand(alphabet[i]);
                                //letters[i].setText("" + alphabet[i]);
                                //letters[i].setBackground(Color.black);

                        }
                else if (radioGroup.getSelection().getActionCommand() == "noorder")
                        for (int i = 0; i < (letters.length); i++ ){
                                letters[i].setIcon(new ImageIcon(path +
"images/letters_numbers/" + keyboard[i].toLowerCase() +"2.gif"));
                                letters[i].setActionCommand(keyboard[i]);
                                //letters[i].setText("" + keyboard[i]);
                                //letters[i].setBackground(Color.black);
                        }
```

```
        }
    }

    static void exitProgram(){
                if (JOptionPane.showConfirmDialog(null,
"Are you sure you want to leave?", "Convey Box", JOptionPane.YES_NO_OPTION) ==
0)
                System.exit(0);
    }


        public static void main(final String[] args) {

        final ConveyBox mainWindow = new ConveyBox() ;

        // Configure the top-level JFrame
        JFrame mainFrame = new JFrame("Convey Box");
        mainFrame.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                        System.exit(0);
                }
        });
        mainFrame.getContentPane().add("Center", mainWindow);
        mainFrame.pack();

        /*
         *SPLASH
         */
        JSplash splash = new JSplash(mainFrame, path + "images/splash.jpg", 4000);
                splash.show();
                splash.requestFocus();
                // This causes the whole application to block until the splash is gone
                splash.block();
                // Once splash is done, show main window
                mainFrame.show();
        /*
                */

        //center window
        Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
        mainFrame.setLocation(d.width/2 - mainWindow.width/2, d.height/2 -
mainWindow.height/2);
                mainFrame.setSize(new Dimension(mainWindow.width,
mainWindow.height));
                mainFrame.setVisible(true);


        }
```

```
}

package Box;

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

/**
 *      Implements a basic splash screen for the Swing Application Framework.
 *      The splash screen will show for a fixed amount of time unles the user
 *      clicks the mouse on the window.
 *
 *      This code is based on code published by the Java Developer's Journal
 * <http://www.sys-con.com/java/index2.html<
 *
 *      @version        1.0
 */

public class JSplash extends JWindow
        implements KeyListener, MouseListener, ActionListener {

        /*
         * JSplash constructs a splash screen (JWindow).
         *              parent is the parent frame for the window
         *              filename is the JPEG/GIF file to show as the splash
         *              timeout is time in milliseconds to display the splash
         */
        public JSplash(JFrame parent, String filename, int timeout) {
                super(parent);

                // Note, this code does no error checking
                ImageIcon image = new ImageIcon(filename);
                // The splash will be centered on the screen
                int w = image.getIconWidth() + 5;
                int h = image.getIconHeight() + 5;
                Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
                int x = (screen.width - w) / 2;
                int y = (screen.height - h) / 2;
                setBounds(x, y, w, h);

                getContentPane().setLayout(new BorderLayout());
                JLabel picture = new JLabel(image);
                getContentPane().add("Center", picture);
                picture.setBorder(new BevelBorder(BevelBorder.RAISED));

                // Listen for key strokes
                addKeyListener(this);
                // Listen for mouse events from here and parent
                addMouseListener(this);
```

```
            parent.addMouseListener(this);
            // Timeout after a while
            Timer timer = new Timer(0, this);
            timer.setRepeats(false);
            timer.setInitialDelay(timeout);
            timer.start();
        }

        // This method is called in order to block the application until
        // the splash screen times out or is dismissed.
        // This is a bit of a kludge, actually, and the actual affect
        // varies based on platform.
        public void block() {
            while(isVisible()) {}
        }

        // Dismiss the window on a key press, ignore rest.
        public void keyTyped(KeyEvent event) {}
        public void keyReleased(KeyEvent event) {}
        public void keyPressed(KeyEvent event) {
            setVisible(false);
            dispose();
        }

        // Dismiss the window on a mouse click, ignore rest.
        public void mousePressed(MouseEvent event) {}
        public void mouseReleased(MouseEvent event) {}
        public void mouseEntered(MouseEvent event) {}
        public void mouseExited(MouseEvent event) {}
        public void mouseClicked(MouseEvent event) {
            setVisible(false);
            dispose();
        }

        // Dismiss the window on a timeout
        public void actionPerformed(ActionEvent event) {
            setVisible(false);
            dispose();
        }
}

/*
 * @(#)ControlContext.java    1.5      99/11/01
 *
 * Copyright (c) 1998, 1999 by Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license to use,
 * modify and redistribute this software in source and binary code form,
 * provided that i) this copyright notice and license appear on all copies of
 * the software; and ii) Licensee does not utilize the software in a manner
 * which is disparaging to Sun.
```

```
 *
 * This software is provided "AS IS," without a warranty of any kind. ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY
 * IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE OR
 * NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS
SHALL NOT BE
 * LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING,
MODIFYING
 * OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL
SUN OR ITS
 * LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR
DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES,
HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF
THE USE OF
 * OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line control of
 * aircraft, air traffic, aircraft navigation or aircraft communications; or in
 * the design, construction, operation or maintenance of any nuclear
 * facility. Licensee represents and warrants that it will not use or
 * redistribute the Software for such purposes.
 */


/**
 * The interface for the JavaSound tabs to open and close audio resources.
 */

package Box;

public interface ControlContext {
      public void open();
      public void close();
}

/*
 * @(#)CapturePlayback.java 1.11     99/12/03
 *
 * Copyright (c) 1999 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license to use,
 * modify and redistribute this software in source and binary code form,
 * provided that i) this copyright notice and license appear on all copies of
 * the software; and ii) Licensee does not utilize the software in a manner
 * which is disparaging to Sun.
 *
```

```
 * This software is provided "AS IS," without a warranty of any kind. ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY
 * IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE OR
 * NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS
SHALL NOT BE
 * LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING,
MODIFYING
 * OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL
SUN OR ITS
 * LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR
DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES,
HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF
THE USE OF
 * OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line control of
 * aircraft, air traffic, aircraft navigation or aircraft communications; or in
 * the design, construction, operation or maintenance of any nuclear
 * facility. Licensee represents and warrants that it will not use or
 * redistribute the Software for such purposes.
 */

package Box;

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Line2D;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import java.util.Vector;
import java.util.Enumeration;
import java.io.*;
import javax.sound.sampled.*;
import java.awt.font.*;
import java.text.*;



/**
 * Capture/Playback sample.  Record audio in different formats
 * and then playback the recorded audio.  The captured audio can
 * be saved either as a WAVE, AU or AIFF.  Or load an audio file
 * for streaming playback.
 *
```

```
 * @version @(#)CapturePlayback.java      1.11    99/12/03
 * @author Brian Lichtenwalter
 */
public class CapturePlayback extends JPanel implements ActionListener,
ControlContext{

   final int bufSize = 16384;

   FormatControls formatControls = new FormatControls();
   Capture capture = new Capture();
   Playback playback = new Playback();

   AudioInputStream audioInputStream;
   SamplingGraph samplingGraph;

   JButton playB, captB, compB;
   JButton auB, aiffB, waveB, teste;
   JTextField textField;

   String fileName = "untitled";
   String errStr;
   double duration, seconds;
   File file;
   Vector lines = new Vector();


   public CapturePlayback() {

      setLayout(new BorderLayout());
      SoftBevelBorder sbb = new SoftBevelBorder(SoftBevelBorder.LOWERED);
      setBorder(new EmptyBorder(5,5,5,5));
      JPanel p2 = new JPanel();
      p2.setBorder(sbb);
      p2.setLayout(new BoxLayout(p2, BoxLayout.X_AXIS));

      JPanel buttonsPanel = new JPanel();
      buttonsPanel.setBorder(new EmptyBorder(10,0,5,0));
      playB = addButton("Play", buttonsPanel, false);
      captB = addButton("Record", buttonsPanel, true);
      compB = addButton("Compare", buttonsPanel, false);
      p2.add(buttonsPanel);

      JPanel samplingPanel = new JPanel(new BorderLayout());
      samplingPanel.add(samplingGraph = new SamplingGraph());
      p2.add(samplingPanel);

      add(p2);
   }
```

```
public void open() { }


public void close() {
    if (playback.thread != null) {
        playB.doClick(0);
    }
    if (capture.thread != null) {
        captB.doClick(0);
    }
}

public void resetButtons(){

        playB.setEnabled(false);
    captB.setEnabled(true);
    compB.setEnabled(false);

    lines.removeAllElements();
    samplingGraph.repaint();

}


private JButton addButton(String name, JPanel p, boolean state) {
    JButton b = new JButton(name);
    b.addActionListener(this);
    b.setEnabled(state);
    p.add(b);
    return b;
}


public void actionPerformed(ActionEvent e) {
    Object obj = e.getSource();
    if (obj.equals(playB)) {
        if (playB.getText().startsWith("Play")) {
            playback.start();
            samplingGraph.start();
            captB.setEnabled(false);
            compB.setEnabled(false);
            playB.setText("Stop");
        } else {
            playback.stop();
            samplingGraph.stop();
            captB.setEnabled(true);
            compB.setEnabled(true);
            playB.setText("Play");
        }
    } else if (obj.equals(captB)) {
        if (captB.getText().startsWith("Record")) {
```

```
            file = null;
            capture.start();
            fileName = "untitled";
            samplingGraph.start();
            playB.setEnabled(false);
            compB.setEnabled(false);
            captB.setText("Stop");
         } else {
            lines.removeAllElements();
            capture.stop();
            samplingGraph.stop();
            playB.setEnabled(true);
            compB.setEnabled(true);
            captB.setText("Record");
         }
      } else if (obj.equals(compB)) {
         ConveyBox.turnPanel();
      }
   }

   public void createAudioInputStream(File file, boolean updateComponents) {
      if (file != null && file.isFile()) {
         try {
            this.file = file;
            errStr = null;
            audioInputStream = AudioSystem.getAudioInputStream(file);
            playB.setEnabled(true);
            fileName = file.getName();
            long milliseconds = (long)((audioInputStream.getFrameLength() * 1000) /
audioInputStream.getFormat().getFrameRate());
            duration = milliseconds / 1000.0;
            if (updateComponents) {
               formatControls.setFormat(audioInputStream.getFormat());
               samplingGraph.createWaveForm(null);
            }
         } catch (Exception ex) {
            reportStatus(ex.toString());
         }
      } else {
         reportStatus("Audio file required.");
      }
   }

   public void saveToFile(String name, AudioFileFormat.Type fileType) {

      if (audioInputStream == null) {
         reportStatus("No loaded audio to save");
         return;
      } else if (file != null) {
```

```
            createAudioInputStream(file, false);
        }

        // reset to the beginnning of the captured data
        try {
            audioInputStream.reset();
        } catch (Exception e) {
            reportStatus("Unable to reset stream " + e);
            return;
        }

        File file = new File(fileName = name);
        try {
            if (AudioSystem.write(audioInputStream, fileType, file) == -1) {
                throw new IOException("Problems writing to file");
            }
        } catch (Exception ex) { reportStatus(ex.toString()); }
        samplingGraph.repaint();
    }


    private void reportStatus(String msg) {
        if ((errStr = msg) != null) {
            System.out.println(errStr);
            samplingGraph.repaint();
        }
    }


    /**
     * Write data to the OutputChannel.
     */
    public class Playback implements Runnable {

        SourceDataLine line;
        Thread thread;

        public void start() {
            errStr = null;
            thread = new Thread(this);
            thread.setName("Playback");
            thread.start();
        }

        public void stop() {
            thread = null;
        }

        private void shutDown(String message) {
            if ((errStr = message) != null) {
                System.err.println(errStr);
```

```
            samplingGraph.repaint();
        }
        if (thread != null) {
            thread = null;
            samplingGraph.stop();
            captB.setEnabled(true);
            compB.setEnabled(true);
            playB.setText("Play");
        }
    }

    public void run() {

        // reload the file if loaded by file
        if (file != null) {
            createAudioInputStream(file, false);
        }

        // make sure we have something to play
        if (audioInputStream == null) {
            shutDown("No loaded audio to play back");
            return;
        }
        // reset to the beginnning of the stream
        try {
            audioInputStream.reset();
        } catch (Exception e) {
            shutDown("Unable to reset the stream\n" + e);
            return;
        }

        // get an AudioInputStream of the desired format for playback
        AudioFormat format = formatControls.getFormat();
        AudioInputStream playbackInputStream =
AudioSystem.getAudioInputStream(format, audioInputStream);

        if (playbackInputStream == null) {
            shutDown("Unable to convert stream of format " + audioInputStream + " to
format " + format);
            return;
        }

        // define the required attributes for our line,
        // and make sure a compatible line is supported.

        DataLine.Info info = new DataLine.Info(SourceDataLine.class,
            format);
        if (!AudioSystem.isLineSupported(info)) {
            shutDown("Line matching " + info + " not supported.");
            return;
        }
```

```
        // get and open the source data line for playback.

        try {
            line = (SourceDataLine) AudioSystem.getLine(info);
            line.open(format, bufSize);
        } catch (LineUnavailableException ex) {
            shutDown("Unable to open the line: " + ex);
            return;
        }

        // play back the captured audio data

        int frameSizeInBytes = format.getFrameSize();
        int bufferLengthInFrames = line.getBufferSize() / 8;
        int bufferLengthInBytes = bufferLengthInFrames * frameSizeInBytes;
        byte[] data = new byte[bufferLengthInBytes];
        int numBytesRead = 0;

        // start the source data line
        line.start();

        while (thread != null) {
            try {
                if ((numBytesRead = playbackInputStream.read(data)) == -1) {
                    break;
                }
                int numBytesRemaining = numBytesRead;
                while (numBytesRemaining > 0 ) {
                    numBytesRemaining -= line.write(data, 0, numBytesRemaining);
                }
            } catch (Exception e) {
                shutDown("Error during playback: " + e);
                break;
            }
        }
        // we reached the end of the stream.  let the data play out, then
        // stop and close the line.
        if (thread != null) {
            line.drain();
        }
        line.stop();
        line.close();
        line = null;
        shutDown(null);
    }
} // End class Playback


/**
 * Reads data from the input channel and writes to the output stream
```

```
*/
class Capture implements Runnable {

  TargetDataLine line;
  Thread thread;

  public void start() {
    errStr = null;
    thread = new Thread(this);
    thread.setName("Capture");
    thread.start();
  }

  public void stop() {
    thread = null;
  }

  private void shutDown(String message) {
    if ((errStr = message) != null && thread != null) {
      thread = null;
      samplingGraph.stop();
      playB.setEnabled(true);
      compB.setEnabled(true);
      captB.setText("Record");
      System.err.println(errStr);
      samplingGraph.repaint();
    }
  }

  public void run() {

    duration = 0;
    audioInputStream = null;

    // define the required attributes for our line,
    // and make sure a compatible line is supported.

    AudioFormat format = formatControls.getFormat();
    DataLine.Info info = new DataLine.Info(TargetDataLine.class,
      format);

    if (!AudioSystem.isLineSupported(info)) {
      shutDown("Line matching " + info + " not supported.");
      return;
    }

    // get and open the target data line for capture.

    try {
      line = (TargetDataLine) AudioSystem.getLine(info);
      line.open(format, line.getBufferSize());
```

```
        } catch (LineUnavailableException ex) {
            shutDown("Unable to open the line: " + ex);
            return;
        } catch (SecurityException ex) {
            shutDown(ex.toString());
            return;
        } catch (Exception ex) {
            shutDown(ex.toString());
            return;
        }

        // play back the captured audio data
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int frameSizeInBytes = format.getFrameSize();
        int bufferLengthInFrames = line.getBufferSize() / 8;
        int bufferLengthInBytes = bufferLengthInFrames * frameSizeInBytes;
        byte[] data = new byte[bufferLengthInBytes];
        int numBytesRead;

        line.start();

        while (thread != null) {
            if((numBytesRead = line.read(data, 0, bufferLengthInBytes)) == -1) {
                break;
            }
            out.write(data, 0, numBytesRead);
        }

        // we reached the end of the stream.  stop and close the line.
        line.stop();
        line.close();
        line = null;

        // stop and close the output stream
        try {
            out.flush();
            out.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        // load bytes into the audio input stream for playback

        byte audioBytes[] = out.toByteArray();
        ByteArrayInputStream bais = new ByteArrayInputStream(audioBytes);
        audioInputStream = new AudioInputStream(bais, format, audioBytes.length /
frameSizeInBytes);

        long milliseconds = (long)((audioInputStream.getFrameLength() * 1000) /
format.getFrameRate());
        duration = milliseconds / 1000.0;
```

```
        try {
            audioInputStream.reset();
        } catch (Exception ex) {
            ex.printStackTrace();
            return;
        }

                        //audioBytes is the vector of input stream.
        samplingGraph.createWaveForm(audioBytes);

    }
} // End class Capture


/**
 * Controls for the AudioFormat.
 */
class FormatControls extends JPanel {

    Vector groups = new Vector();
    JToggleButton linrB, ulawB, alawB, rate8B, rate11B, rate16B, rate22B, rate44B;
    JToggleButton size8B, size16B, signB, unsignB, litB, bigB, monoB,sterB;

    public FormatControls() {
        setLayout(new GridLayout(0,1));
        EmptyBorder eb = new EmptyBorder(0,0,0,5);
        BevelBorder bb = new BevelBorder(BevelBorder.LOWERED);
        CompoundBorder cb = new CompoundBorder(eb, bb);
        setBorder(new CompoundBorder(cb, new EmptyBorder(8,5,5,5)));
        JPanel p1 = new JPanel();
        ButtonGroup encodingGroup = new ButtonGroup();
        linrB = addToggleButton(p1, encodingGroup, "linear", true);
        ulawB = addToggleButton(p1, encodingGroup, "ulaw", false);
        alawB = addToggleButton(p1, encodingGroup, "alaw", false);
        add(p1);
        groups.addElement(encodingGroup);

        JPanel p2 = new JPanel();
        JPanel p2b = new JPanel();
        ButtonGroup sampleRateGroup = new ButtonGroup();
        rate8B = addToggleButton(p2, sampleRateGroup, "8000", false);
        rate11B = addToggleButton(p2, sampleRateGroup, "11025", false);
        rate16B = addToggleButton(p2b, sampleRateGroup, "16000", false);
        rate22B = addToggleButton(p2b, sampleRateGroup, "22050", false);
        rate44B = addToggleButton(p2b, sampleRateGroup, "44100", true);
        add(p2);
          add(p2b);
        groups.addElement(sampleRateGroup);

        JPanel p3 = new JPanel();
```

```
    ButtonGroup sampleSizeInBitsGroup = new ButtonGroup();
    size8B = addToggleButton(p3, sampleSizeInBitsGroup, "8", false);
    size16B = addToggleButton(p3, sampleSizeInBitsGroup, "16", true);
    add(p3);
    groups.addElement(sampleSizeInBitsGroup);

    JPanel p4 = new JPanel();
    ButtonGroup signGroup = new ButtonGroup();
    signB = addToggleButton(p4, signGroup, "signed", true);
    unsignB = addToggleButton(p4, signGroup, "unsigned", false);
    add(p4);
    groups.addElement(signGroup);

    JPanel p5 = new JPanel();
    ButtonGroup endianGroup = new ButtonGroup();
    litB = addToggleButton(p5, endianGroup, "little endian", false);
    bigB = addToggleButton(p5, endianGroup, "big endian", true);
    add(p5);
    groups.addElement(endianGroup);

    JPanel p6 = new JPanel();
    ButtonGroup channelsGroup = new ButtonGroup();
    monoB = addToggleButton(p6, channelsGroup, "mono", false);
    sterB = addToggleButton(p6, channelsGroup, "stereo", true);
    add(p6);
    groups.addElement(channelsGroup);
}

private JToggleButton addToggleButton(JPanel p, ButtonGroup g,
                        String name, boolean state) {
    JToggleButton b = new JToggleButton(name, state);
    p.add(b);
    g.add(b);
    return b;
}

public AudioFormat getFormat() {

    Vector v = new Vector(groups.size());
    for (int i = 0; i < groups.size(); i++) {
      ButtonGroup g = (ButtonGroup) groups.get(i);
      for (Enumeration e = g.getElements();e.hasMoreElements();) {
        AbstractButton b = (AbstractButton) e.nextElement();
        if (b.isSelected()) {
          v.add(b.getText());
          break;
        }
      }
    }

    AudioFormat.Encoding encoding = AudioFormat.Encoding.ULAW;
```

```
        String encString = (String) v.get(0);
        float rate = Float.valueOf((String) v.get(1)).floatValue();
        int sampleSize = Integer.valueOf((String) v.get(2)).intValue();
        String signedString = (String) v.get(3);
        boolean bigEndian = ((String) v.get(4)).startsWith("big");
        int channels = ((String) v.get(5)).equals("mono") ? 1 : 2;

        if (encString.equals("linear")) {
            if (signedString.equals("signed")) {
                encoding = AudioFormat.Encoding.PCM_SIGNED;
            } else {
                encoding = AudioFormat.Encoding.PCM_UNSIGNED;
            }
        } else if (encString.equals("alaw")) {
            encoding = AudioFormat.Encoding.ALAW;
        }
        return new AudioFormat(encoding, rate, sampleSize,
                channels, (sampleSize/8)*channels, rate, bigEndian);
    }


    public void setFormat(AudioFormat format) {
        AudioFormat.Encoding type = format.getEncoding();
        if (type == AudioFormat.Encoding.ULAW) {
            ulawB.doClick();
        } else if (type == AudioFormat.Encoding.ALAW) {
            alawB.doClick();
        } else if (type == AudioFormat.Encoding.PCM_SIGNED) {
            linrB.doClick(); signB.doClick();
        } else if (type == AudioFormat.Encoding.PCM_UNSIGNED) {
            linrB.doClick(); unsignB.doClick();
        }
        float rate = format.getFrameRate();
        if (rate == 8000) {
            rate8B.doClick();
        } else if (rate == 11025) {
            rate11B.doClick();
        } else if (rate == 16000) {
            rate16B.doClick();
        } else if (rate == 22050) {
            rate22B.doClick();
        } else if (rate == 44100) {
            rate44B.doClick();
        }
        switch (format.getSampleSizeInBits()) {
            case 8  : size8B.doClick(); break;
            case 16 : size16B.doClick(); break;
        }
        if (format.isBigEndian()) {
            bigB.doClick();
        } else {
```

```
            litB.doClick();
        }
        if (format.getChannels() == 1) {
            monoB.doClick();
        } else {
            sterB.doClick();
        }
    }
} // End class FormatControls


/**
 * Render a WaveForm.
 */
class SamplingGraph extends JPanel implements Runnable {

    private Thread thread;
    private Font font10 = new Font("serif", Font.PLAIN, 10);
    private Font font12 = new Font("serif", Font.PLAIN, 12);
    Color jfcBlue = new Color(204, 204, 255);
    Color pink = new Color(255, 175, 175);


    public SamplingGraph() {
        setBackground(new Color(20, 20, 20));
    }


    public void createWaveForm(byte[] audioBytes) {

        lines.removeAllElements(); // clear the old vector

        AudioFormat format = audioInputStream.getFormat();
        if (audioBytes == null) {
            try {
                audioBytes = new byte[
                    (int) (audioInputStream.getFrameLength()
                    * format.getFrameSize())];
                audioInputStream.read(audioBytes);
            } catch (Exception ex) {
                reportStatus(ex.toString());
                return;
            }
        }

        Dimension d = getSize();
        int w = d.width;
        int h = d.height-15;
        int[] audioData = null;
        if (format.getSampleSizeInBits() == 16) {
            int nlengthInSamples = audioBytes.length / 2;
```

```
            audioData = new int[nlengthInSamples];
            if (format.isBigEndian()) {
              for (int i = 0; i < nlengthInSamples; i++) {
                 /* First byte is MSB (high order) */
                 int MSB = (int) audioBytes[2*i];
                 /* Second byte is LSB (low order) */
                 int LSB = (int) audioBytes[2*i+1];
                 audioData[i] = MSB << 8 | (255 & LSB);
              }
            } else {
               for (int i = 0; i < nlengthInSamples; i++) {
                 /* First byte is LSB (low order) */
                 int LSB = (int) audioBytes[2*i];
                 /* Second byte is MSB (high order) */
                 int MSB = (int) audioBytes[2*i+1];
                 audioData[i] = MSB << 8 | (255 & LSB);
               }
            }
         } else if (format.getSampleSizeInBits() == 8) {
            int nlengthInSamples = audioBytes.length;
            audioData = new int[nlengthInSamples];
            if (format.getEncoding().toString().startsWith("PCM_SIGN")) {
               for (int i = 0; i < audioBytes.length; i++) {
                  audioData[i] = audioBytes[i];
               }
            } else {
               for (int i = 0; i < audioBytes.length; i++) {
                  audioData[i] = audioBytes[i] - 128;
               }
            }
         }

         int frames_per_pixel = audioBytes.length / format.getFrameSize()/w;
         byte my_byte = 0;
         double y_last = 0;
         int numChannels = format.getChannels();
         for (double x = 0; x < w && audioData != null; x++) {
            int idx = (int) (frames_per_pixel * numChannels * x);
            if (format.getSampleSizeInBits() == 8) {
               my_byte = (byte) audioData[idx];
            } else {
               my_byte = (byte) (128 * audioData[idx] / 32768 );
            }
            double y_new = (double) (h * (128 - my_byte) / 256);
            lines.add(new Line2D.Double(x, y_last, x, y_new));
            y_last = y_new;
         }

         repaint();
      }
```

```
    public void paint(Graphics g) {

        Dimension d = getSize();
        int w = d.width;
        int h = d.height;
        int INFOPAD = 15;

        Graphics2D g2 = (Graphics2D) g;
        g2.setBackground(getBackground());
        g2.clearRect(0, 0, w, h);
        g2.setColor(Color.white);
        g2.fillRect(0, h-INFOPAD, w, INFOPAD);

        if (errStr != null) {
            g2.setColor(jfcBlue);
            g2.setFont(new Font("serif", Font.BOLD, 18));
            g2.drawString("ERROR", 5, 20);
            AttributedString as = new AttributedString(errStr);
            as.addAttribute(TextAttribute.FONT, font12, 0, errStr.length());
            AttributedCharacterIterator aci = as.getIterator();
            FontRenderContext frc = g2.getFontRenderContext();
            LineBreakMeasurer lbm = new LineBreakMeasurer(aci, frc);
            float x = 5, y = 25;
            lbm.setPosition(0);
            while (lbm.getPosition() < errStr.length()) {
                TextLayout tl = lbm.nextLayout(w-x-5);
                if (!tl.isLeftToRight()) {
                    x = w - tl.getAdvance();
                }
                tl.draw(g2, x, y += tl.getAscent());
                y += tl.getDescent() + tl.getLeading();
            }
        } else if (capture.thread != null) {
            g2.setColor(Color.black);
            g2.setFont(font12);
            g2.drawString("Length: " + String.valueOf(seconds), 3, h-4);
        } else {
            g2.setColor(Color.black);
            g2.setFont(font12);
            g2.drawString("File: " + fileName + "  Length: " + String.valueOf(duration) + "
Position: " + String.valueOf(seconds), 3, h-4);

            if (audioInputStream != null) {
                // .. render sampling graph ..
                g2.setColor(jfcBlue);
                for (int i = 1; i < lines.size(); i++) {
                    g2.draw((Line2D) lines.get(i));
                }

                // .. draw current position ..
```

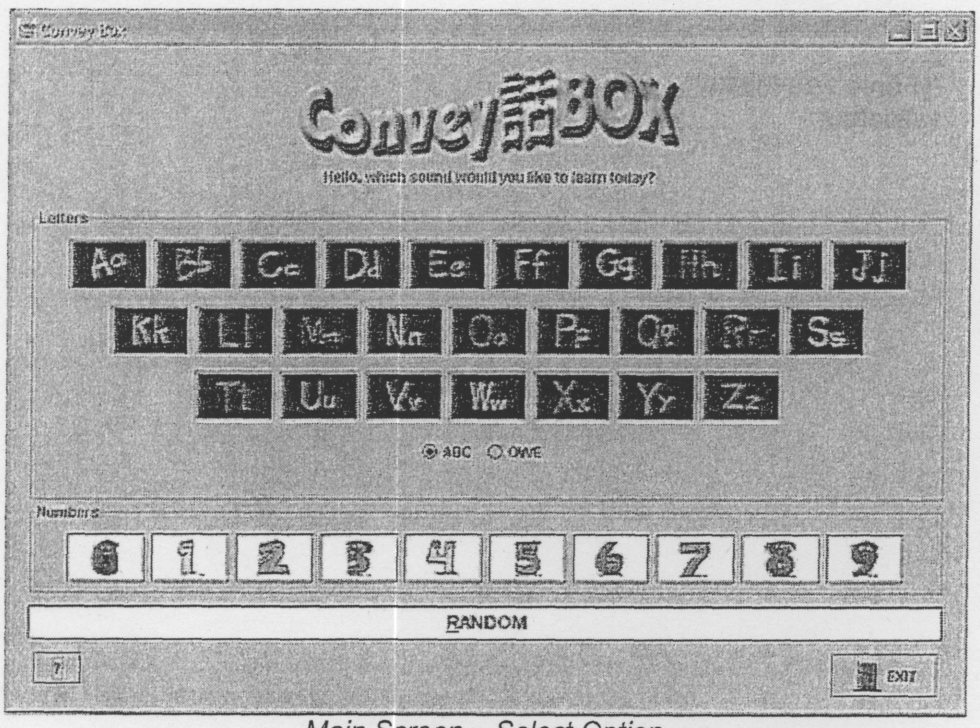
```
            if (seconds != 0) {
               double loc = seconds/duration*w;
               g2.setColor(pink);
               g2.setStroke(new BasicStroke(3));
               g2.draw(new Line2D.Double(loc, 0, loc, h-INFOPAD-2));
            }
         }
      }
   }

   public void start() {
      thread = new Thread(this);
      thread.setName("SamplingGraph");
      thread.start();
      seconds = 0;
   }

   public void stop() {
      if (thread != null) {
         thread.interrupt();
      }
      thread = null;
   }

   public void run() {
      seconds = 0;
      while (thread != null) {
         if ((playback.line != null) && (playback.line.isOpen()) ) {

            long milliseconds = (long)(playback.line.getMicrosecondPosition() / 1000);
            seconds =  milliseconds / 1000.0;
         } else if ( (capture.line != null) && (capture.line.isActive()) ) {

            long milliseconds = (long)(capture.line.getMicrosecondPosition() / 1000);
            seconds =  milliseconds / 1000.0;
         }

         try { thread.sleep(100); } catch (Exception e) { break; }

         repaint();

         while ((capture.line != null && !capture.line.isActive()) ||
               (playback.line != null && !playback.line.isOpen()))
         {
            try { thread.sleep(10); } catch (Exception e) { break; }
         }
      }
      seconds = 0;
      repaint();
   }
} // End class SamplingGraph
```

```
}

package Box;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.media.*;
import javax.swing.*;

public class PlayerPanel extends    JPanel
                  implements ControllerListener
{
   protected boolean    includeVisual, includeControls;
   protected Player     player;
   File file;

   public PlayerPanel(boolean includeVisual,
                      boolean includeControls)
   {
      player = null;

      this.includeVisual   = includeVisual;
      this.includeControls = includeControls;

      setLayout(new BorderLayout());
      setSize(200, 150);
   }


   public void loadFile(String name)
   {

      file = new File(name);
      Component c;

      /**
       * Verify if exists player and remove it
       */
      removePreviousPlayer();

      try {

         /**
          * Inicialize player
          */

         player = Manager.createPlayer(file.toURL());
```

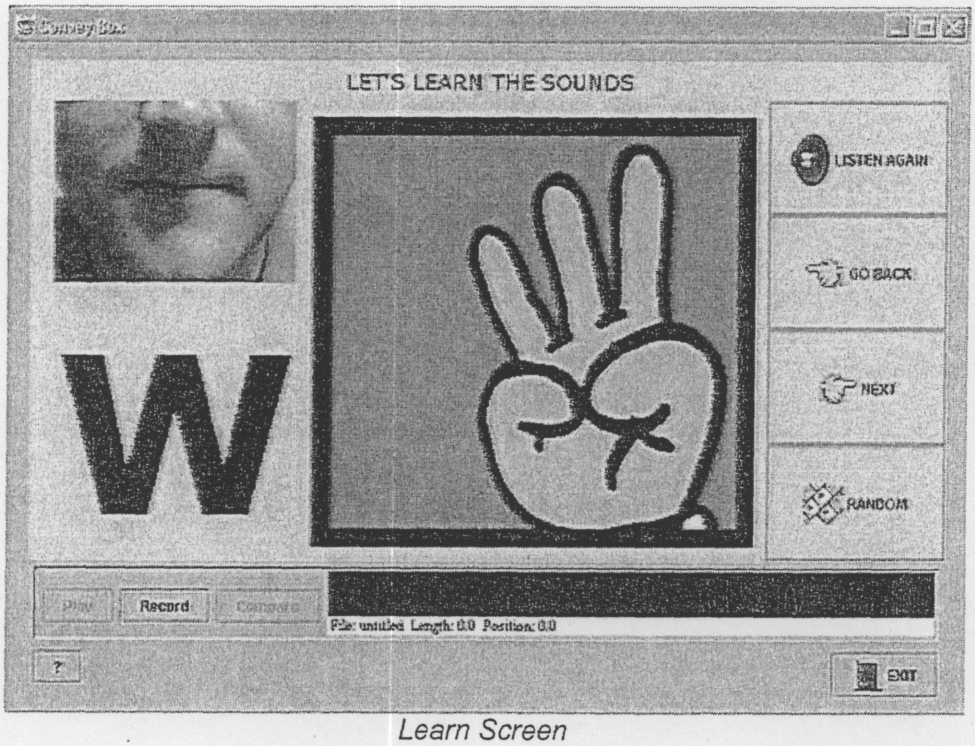
```
            player.addControllerListener(this);
            player.start();

        } catch (MalformedURLException mue) {
            JOptionPane.showMessageDialog(this, "Unable to open file!");
        } catch (NoPlayerException npe) {
            JOptionPane.showMessageDialog(this, "Unable to create a player!");
        } catch (IOException ioe) {
            JOptionPane.showMessageDialog(this, "Unable to connect to source!");
        }

    }


    /**
     * Remove previous player function
     */
     private void removePreviousPlayer()
    {
      if ( player == null )
        return;

      player.close();

      Component visual = player.getVisualComponent();
      Component control = player.getControlPanelComponent();

      if ((visual != null) && (includeVisual))
        remove( visual );

      if ((control != null) && (includeControls))
        remove( control );

    }


    /**
    * The controllerUpdate method takes care of events from the Player object
    * @param ControllerEvent The controller event from a Player object
    */
     public synchronized void controllerUpdate(ControllerEvent evt)
     {
        Component c;

        if (evt instanceof RealizeCompleteEvent) {

            c = player.getVisualComponent();
            if ((c != null) && (includeVisual)) {
                add(c, BorderLayout.CENTER);
            }
```

```
            c = player.getControlPanelComponent();
            if ((c != null) && (includeControls)) {
                add(c, BorderLayout.SOUTH);
            }

            // Force a re-layout
            validate();
        }

    }

}
```

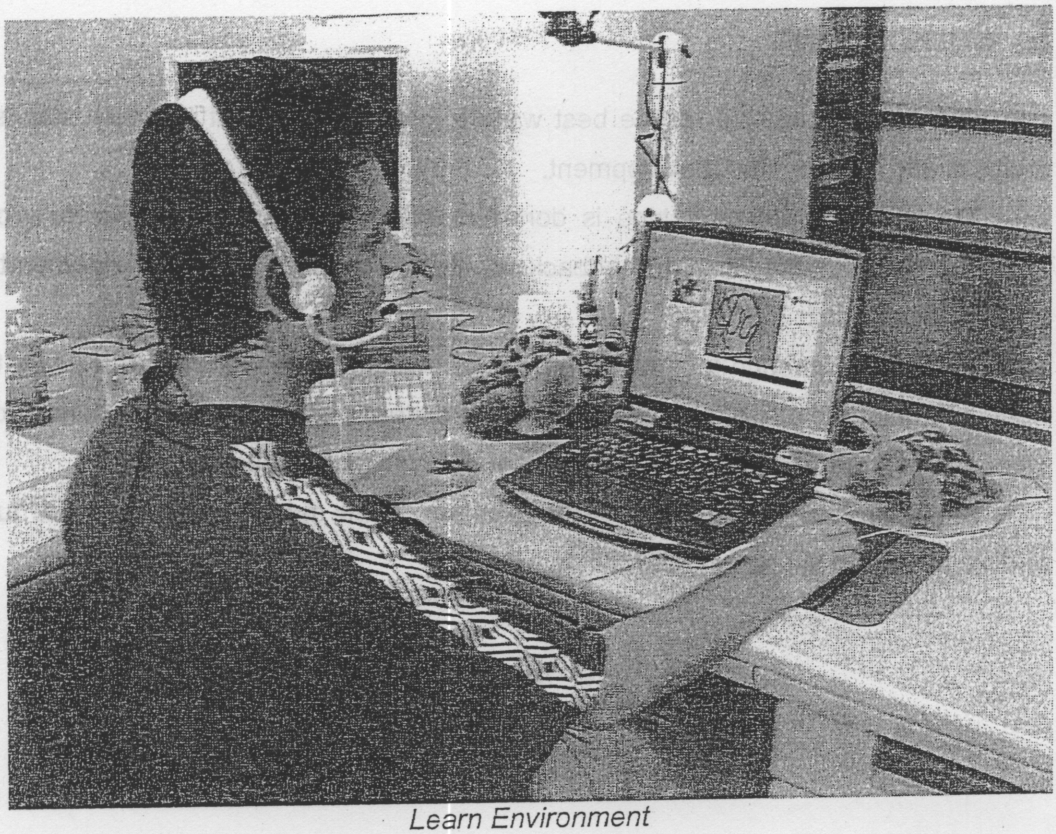
## 4.3 User's Interface




*Main Screen – Select Option*




*Learn Screen*



*Learn Environment*

# 5. Conclusion

Our research looked for the best ways to get to the goal at first then after some analyses and design – the development.

The software first prototype is done but still in need of some tests to prove it efficiency. The evaluation and feedback will be fundamental to future implementation and improvements.

A year was enough to create the basis for this software but it has a lot work to be made and researched. As for future work we can already summarize some ideas: songs, score games, new vocabulary and implement different languages.

The wish to have a product to help others is the motivation to continue further studies in ways to improve quality of life.

# 6. Bibliography

**Deitel**, Java How to Program – Java2 featuring Swing
**Gomaa**, Designing Concurrent, Distributed and Real-Time Applications with UML
**Eckstein**, Java Swing, O'Reilly
**Zukawski**, Java AWT Reference, O'Reilly

***Internet Sources***

http://java.sun.com/
http://www.mrsalphabet.com/
http://www.latticeworksw.com/roxabc.htm
http://abc.pa.net/
http://www.kidgen.com/
http://ourworld.compuserve.com/homepages/RayLec/lettersa.htm
http://msnhomepages.talkcity.com/DeckDr/marsberg/
http://www.lawrencegoetz.com/programs/keys/
http://schoolexpress.com/
http://kidsfreeware.com/school/play-plus.html
http://teacher.scholastic.com/products/interactiveabc.htm
http://learn-math-playing-games.com/index2.html
http://homeschoolinformation.com/Resources/free_software.htm
http://www.learningplanet.com/act/fl/aact/index.asp
http://www.gnu.org/graphics/agnuhead.html
http://www.schoolzone.co.uk/

***Convey Box Homepage***

http://conveybox.visit.ws