Supporting Multimodal Collaboration with Digital Ink and Audio

(*Spline Title*: Multimodal Collaboration with Digital Ink and Audio)
(*Thesis Format*: Monograph)

By

Amit Regmi

Graduate Program in Computer Science

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

January 30, 2009

THE UNIVERSITY OF WESTERN ONTARIO

SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES

CERTIFICATE OF EXAMINATION

Chief Adviser:

_____

Dr. Stephen M. Watt

Advisory Committee:

_____

_____

Examiners:

_____

Dr. Hanan Lutfiyya

_____

Dr. Mahmoud El-Sakka

_____

Dr. Gregory J. Reid

The thesis by

Amit Regmi

entitled

Supporting Multimodal Collaboration with Digital Ink and Audio

is accepted in partial fulfillment of the
requirements for the degree of

Master of Computer Science

Date: January 30, 2009

Dr. Sylvia Osborn
Chair of Examining Board

Abstract

The increased use of handheld and tablet devices enables people to share a wide variety of media and collaborate. It becomes interesting to conduct and to archive communication sessions that involve audio and digital ink on a shared canvas. Collaborative whiteboards do exist today and most of them use complex protocols for communication. As a rule, the existing whiteboards are not interoperable across multiple platforms and do not support archival of collaborative sessions for later reference. In this thesis, we examine the question of what is required to support these capabilities. We examine how various data formats may be used to capture, to transmit, to record and to synchronize ink and audio channels. To test our ideas we have developed a concrete software implementation.

We present InkAndAudio Chat, a whiteboard collaborative application, along with InkAMP, an Ink and Audio Media Player that plays the archived ink and audio sessions. We also present PyInkAndAudio Chat, a similar chat client for Linux and Mac OS X, to stretch our multimodal application framework further and demonstrate its cross-platform support. All our chat clients make use of InkML to save the digital ink which is later used by the InkAMP player in a platform independent manner.

Keywords: Multimodal collaboration, InkML, Skype, Whiteboard, Pen-based computing

Acknowledgments

I am grateful to my supervisor, Prof. Stephen M. Watt, for his constant support, patient guidance and the encouragement. Special thanks to him for his generous flexibility without which this work wouldn't have been complete.

Many thanks to my past and present roommates who have been great source of motivation and strength during my research.

Thanks also to those who answered my queries in various technical forums, mailing lists and IRC channels.

Finally, thanks to my colleagues at the ORCCA (Ontario Research Centre for Computer Algebra) Laboratory, for their help and support throughout the year.

Contents

List of Figures

Chapter 1

Introduction

In the last fifty years, the computing hardware has gone through a drastic change. With the availability of a myriad of computing devices, the computing environment is going through a technical evolution. Such devices today are equipped with multiple modes of input. In order to give the users a good computing experience, applications need to incorporate those input modes. This makes it possible for a user to compute in different set of environments. Therefore, multimodal user interactions has become an established subject of research.

Laptop computers are becoming smaller and mobile devices are coming up with more computation power. Mobile computation is becoming a reality. More users wish to migrate their desktop computations to mobile devices. The fact that mobile tasks are different from workstation tasks is easy to see. Mobile devices deal primarily into note taking, form filling, email reading, database browsing *etc.* while workstations are used more for document preparation, application development, email reading and composition, graphics design *etc.* Due to the widespread use and acceptance of mobile devices, multimodal applications have a larger role to play in the future of digital communication.

Oviatt [41] expresses this position as follows :

> "The flexibility of a multimodal interface can accommodate a wide range of users, tasks, and environments for which any given single mode may not suffice."

Multimodal applications not only provide combined features of uni-modal inputs; they also bring about additional benefits. Such applications help users establish effective communication, enhance group collaboration and help make the workforce more productive. In the work of Philip Cohen *et al.* [9] it has been shown that multimodal interactions have higher efficiency compared to simple GUI-based interactions.

The ongoing research in human-computer interaction has also touched the multimodal aspect of computations. Much work has been done in the field of multimodal interface design. Applications have been developed, usable by diverse population including the physically challenged. Homer [14], a self voicing Web browser designed for visually impaired people, is such an example.

1.1 Multimodal Interaction Working Group

The World Wide Web Consortium (W3C) is an international standards organization for the World Wide Web, founded by Tim Berners-Lee in 1994, at the Massachusetts Institute of Technology (MIT). W3C aims at providing open standards for the evolution of the World Wide Web. The W3C Multimodal Interaction Working Group [19] was launched in 2002. It aims to extend the Web to allow users to select dynamically the most appropriate interaction mode for their needs, and different abilities. It also enables developers to provide an effective user interface for the modes a user selects [19]. The Multimodal Interaction Working Group has played an active role in the formulation of new standards such as InkML [6, 55], VoiceXML *etc.* for the data representation involving multimodal interactions.

1.2 Cross-Platform Multimodal Computing

Cross-platform computing has been another active field of research. Commercialized hardware manufacture gives us scores of different software platforms on computing devices. Various researchers are looking into possibilities of making the multimodal applications run in different existing platforms which eventually would benefit the users. InkML [6, 55], a W3C open standard for digital ink representation, is one such technology that could support cross-platform application development by providing an open and flexible framework for data representation.

InkML, an XML-based language for digital ink traces, is a work in progress from the Multimodal Interaction Working Group. It defines an XML data exchange format for ink, entered with stylus or an electronic pen in a multimodal system. InkML provides the capture and processing support for the digital ink involved in drawings, handwriting, gestures, mathematical notations, music notes *etc.*

The use of proprietary data representation standards such as Microsoft's ISF (Ink Serialized Format) limits the availability of a multimodal application to a specific platform. Therefore, open standards such as InkML, VoiceXML *etc.* may be helpful for data representation in cross-platform multimodal computing.

1.3 Objectives

The primary objectives of this thesis were to analyze:

- The issues involved in the development of a cross-platform multimodal application with ink and audio and
- The issues involved in using various data formats to provide archival support for the collaborative sessions involving the multimodal data.

We have developed InkAndAudio Chat, a multimodal chat application that uses digital ink with audio for collaborative discussions. PyInkAndAudio, our chat client for Linux and Mac OS X, supports our multimodal application further and justifies on its being a truly cross-platform tool.

A number of collaborative whiteboards have been developed. Most of them make use of complex protocols for data exchange. These do not yet have the technology that can archive collaborative sessions involving digital ink and audio and be able to play them back later. One of our objectives was to be able to simplify the data exchange mechanisms involved in the whiteboard application and be able to archive such sessions for future reference.

Also, our aim was to be able to test the suitability of the definition of InkML for interactive collaboration and as a multimodal component. We have used InkML as the means of storing the digital ink strokes generated when participants use InkAndAudio Chat application to collaborate.

1.4 Related Work

Pen input has several applications. Digital whiteboards have been an interesting area for some time with their usefulness in distance education, classroom teaching, business meetings, collaborative document annotation *etc.* As more and more collaborative platforms and devices emerge, the implementation becomes more challenging.

Whiteboard sharing is one of the most widely used applications in pen computing. In the earliest forms of whiteboards, portability and efficiency of data exchange were not the prime concern. Tivoli [43] was one such whiteboard designed for work-group meetings and with essentially no cross-platform support. Xerox LiveBoard [15] and the Tivoli system were whiteboards designed for small group meetings. Work on large-scale classroom presentations have also been discussed in some recent systems [3, 17, 26]. InkBoard [38] was another collaborative sketching application where designers could exchange ink strokes over the network and store the drawings in a central InkBoard server.

Multimodal systems with pen and voice interfaces were in use in the 1990s and earlier. QuickSet [10] was a multimodal framework used by the US Navy and US Marine Corps to set up training scenarios and to control the virtual environment.

Among other systems, Speech Pen [29] uses handwriting recognition techniques to predict the written words.

With the growing importance of multimodal applications, several application development frameworks came into existence. RiverInk [37], an ink application development framework focuses mainly on interoperability and uses a subset of InkML to represent the ink messages. To achieve lossless ink storage, it proposes to use the native-ink format. To support interoperability, the framework sends ink data as a collection of the native ink format, InkML and PNG images. This makes the data exchange bulky enough for the mobile networks. Similarly, IBM Multimodal Toolkit [22] is another framework based on XHTML and VoiceXML that provides a development environment for interactive multimodal web applications.

Recent work [1, 30] at the HP Labs, describe a heterogeneous collaborative environment for digital ink exchange. It makes use of the XMPP protocol (Extensible Messaging and Presence Protocol) for the information exchange over the network. The client server architecture is rather complex with an InkML plug-in necessary as a server component and an XMPP client manager component necessary at the client application to communicate InkML messages with the server.

Multimodal interactions has been an active research area for a long time. The gaining popularity of handhelds and mobile devices has made it even more popular and much more research work can be expected in the future.

We have included a detailed discussion of the most relevant systems in Section 2.1.

1.5 Thesis Organization

InkAndAudio Chat and the InkAMP player are our implementations to demonstrate the ideas of the thesis. We have chosen to use the Skype peer-to-peer network as the backbone for our implementation, to allow a broad test userbase.

In Chapter 2, we discuss the necessary background behind the theory and our implementation. We also cover some discussion on existing multimodal applications for collaboration, their usefulness and shortcomings. We look into different data representation standards for a variety of input modes. We discuss various InkML concepts and features that have been used in our implementation. Finally, we will discuss the Skype peer-to-peer network which has been the backbone of our implementation.

In Chapter 3, we cover the design issues that arise while developing cross-platform multimodal applications.

Chapter 4 presents various design and implementation issues we faced while building the Skype-based InkAndAudio multimodal chat system.

Chapter 5 covers the system architecture. It describes the two major components of our implementation, InkAndAudio Chat client and the InkAMP player that plays the recorded chat sessions. We also cover the design of PyInkAndAudio Chat, a chat client for Linux and Mac OS X, based on Skype4Py wrapper.

We conclude with Chapter 6, and discuss some future extension possibilities to our implementation.

Chapter 7 presents a user manual that helps in the installation of InkAndAudio Chat client as an add-on for Skype.

Chapter 2

Background

Since its inception, people have been using the Internet for collaboration and building up community with other professionals. Various whiteboard applications that help multiple users to work together are being sold commercially. Groove Networks [39] has been developing collaborative multimodal applications for a variety of end users. After its acquisition by Microsoft, their products have been marketed as Office Groove, which is a collaboration software program that helps teams work together dynamically and effectively, even if members work for different organizations, work remotely, or work offline.

Group interactions with a computing application is difficult using the traditional mouse and keyboard methods *e.g.* in collaboration. More time is spent on handling the application rather than on the discussion of the subject matter. While a computer mouse is a pointing device, a computer pen is used for writing or drawing. Pens are better suited for gestural techniques and so offer the best medium for effective interactions.

Use of digital ink for capture of handwritten strokes has become an important input mode for multimodal applications. Work by Anderson *et al.* [3] provides an insight on how digital ink can be used in practical context. It attempts to establish digital ink capture as a mechanism that could overcome the lost flexibility of real-time interactions with a group while using only presentation slides.

Other possible handwriting usage include:

- Prevention of online fraud
- Convenient mode of communication at crowded places with high noise level
- Mode for identification and security clearance
- When language fonts are not available in a system

Audio input is convenient when a user is incapable of using hands for writing or gestures *e.g.* while driving. Hence, depending on the circumstance, a user will prefer to use either handwriting or audio or both if such modes are available.

Most of the research work on collaborative computation are centered on whiteboards which have been used extensively in group meetings and classrooms. The early whiteboards [15] did not use audio as an input mode. With more progress in the research, audio/video modality came into use, making the applications truly multimodal.

In the next section, we will discuss a number of such multimodal applications that use ink and/or audio as input modes.

2.1 Multimodal Systems of the Past

2.1.1 QuickSet

QuickSet [10] was a multimodal framework developed in 1997 and used by the US Army to set up training scenarios. It supported digital ink and audio as input modes. QuickSet prototypes were successfully tested to train the US Army in various environments. One such test was conducted in a noisy environment with explosions and low-flying jet aircrafts. As expected, the users gestured successfully when spoken interaction was not feasible. This was interpreted as showing that the freedom of choice for the input mode enables a better and efficient communication.

2.1.2 InkBoard

InkBoard was a network-sharing Tablet PC sketching application developed at the MIT Intelligent Engineering Systems Lab (IESL) and published by Hai Ning *et al.* [38], in 2004. It was an application for collaboration and helped drawings to be shared among designers. Its use was limited to the Windows environment as it was integrated with Microsoft's ConferenceXP research platform for audio/video conferencing and used Microsoft's proprietary Ink Serialized Format (ISF) for stroke streaming.

2.1.3 LiveBoard

The LiveBoard [15] system was developed in 1990 for group collaboration at Xerox-Palo Alto Research Center. It had a huge interactive, stylus-based display system that could be used in slide presentation and group collaboration. Its major drawbacks include the hardware installation cost, use of proprietary software and portability. Though it could run several other applications, the whiteboard application was the most widely used. LiveBoard whiteboard used an X11 based bitmap paint application. It could remember new pages drawn and could recover those pages for printing when necessary, but could only print the final version of the annotated document. It had a primitive network support and was used mostly in a classroom environment. However, it was not sufficiently mature to be used for distance education.

The whiteboard application in LiveBoard had limitations on the way the digital ink was handled due to the immature pen-based computation of that time. The application was simply a multiple pen-based image editor.

2.1.4 Tivoli

Tivoli [43] came into existence in part to overcome the limitations of LiveBoard. It made use of a proprietary Software Development Kit (SDK) to handle the digital ink elegantly. The major

changes seen in Tivoli are in the drawing representation which had *stroke objects* instead of pixel map images as in LiveBoard. Strokes were seen as atomic units and were transformable in nature. Tivoli was made more collaborative than its ancestor, LiveBoard, by providing multiple cordless pens for multiple users.

2.1.5 Tivoli with Audio

While the Tivoli system had no audio capability, pairing ink with audio had been anticipated by other authors. Moran *et al.* [36] did a case study on the *"salvation"* of the captured multimedia recordings in meeting discussions. The term *"salvation"* meant the act of sorting out useful pieces from the recordings, indexing them, relating them, organizing them and creating new materials with them.

The tool made use of Tivoli system and the audio input from a microphone. In this tool, Tivoli timestamps every action involved such as drawing a stroke with the LiveBoard pen, changing a page *etc.* These timestamps were then used as access points for audio records.

2.1.6 Classroom 2000

ClassRoom 2000 [2] was one of the first collaborative tools aimed at capturing the presentation material along with audio in a classroom setting. It uses LiveBoard for presentation and PDAs or Tablets for note taking. Classroom 2000 is basically an educational application for the automated capture of discussion material in a ubiquitous computing environment. The stored material is then accessed using HTML based web browsers. Real-time collaboration amongst geographically separate participants was not possible, as the presentation material had to be uploaded to a server before participants could access it.

2.1.7 Electronic Chalkboard

Pen computation has generally been limited to small screens. Electronic Chalkboard [17] was developed at Freie Universität Berlin and published in 2004. It takes pen computing to a large screen, suitable for blackboard sized display with real-time interactions. It has been extended to communicate with external web services and computer algebra servers. Electronic Chalkboard obtains the computation results from a server, and the results are then displayed on a whiteboard. Multimedia data can be stored along with strokes. Participants can connect to the system using Tablet PCs. Software agents running in the background connect with the external resources. The system has been integrated and tested successfully with the Maple computer algebra system and the Mathematica system (from Wolfram Research) [47]. The authors of the Electronic Chalkboard multimodal system claim that their system has taken the traditional classroom teaching to a whole new level, making way for an incredibly effective teaching and learning experience.

2.1.8 Audio Notebook and Audio Indexing

Audio indexing becomes necessary when one has to play multimodal data archives where the audio needs to be synchronized with other forms of data. Audio indexing along with ink strokes on a sketch application seems to be first reported on NoTime [31].

Audio Notebook [52] was one of the initial works on audio indexing. It helped in note taking with traditional paper notebooks and the audio recording was indexed based on explicit user directed actions such as a button push to mark important parts. Audio Notebook was aimed more towards augmenting the note taking rather than making it a tool for collaboration.

Audio Notebook was at a primitive stage in its development and suffered several limitations. Too much attention had to be given while generating the indexing information, *e.g.* frequent pressing of buttons. Information search was time consuming as everything was recorded, which often included long stretch of audio with little information. In contrast, Dynomite [57] records

just the important portions of the audio permanently resulting in quicker information search.

2.1.9 Dynomite

Dynomite was another such application similar to Audio Notebook and was used for note taking. It recorded audio continuously and generated timestamps synchronously with the ink stroke input, in a manner similar to FiloChat [56]. Dynomite not only timestamps the strokes but also saves the ink properties. It targets mobile devices and due to their known storage limitation, it saves only the important parts of the audio. Dynomite gives a paper-like interface to users and tries to reduce distractions while recording.

2.1.10 Other systems

There are several other applications that use digital ink and/or audio as input modes. Some of them include applications for note taking and sharing [13] , meetings [43, 51], classroom presentation and capture [2] *etc.*

LiveNotes [26], FiloChat [56] and Notepals [13] helped in note taking in a classroom environment. A study with FiloChat [56] has shown that the indexing of audio while note taking enhances the note taking experience.

2.2 Data Representations

2.2.1 Digital Ink Standards

In an electronic whiteboard environment, digital ink is by far the most favored and the most natural form of input. Pen-based computation in a multimodal setting needs a high degree of flexibility in the way data is represented. The digital ink format should support features such

as annotation, modifications, sharing, archival and streaming of the ink data. A common standard for the digital ink representation becomes important as it helps in the development of a robust and platform independent digital ink-enabled application and provides an easy medium for information interchange.

Jot specification [12]

Jot came into existence in 1992 as a result of a joint work by Slate, Lotus, GO, Microsoft, Apple, General Magic and others. Before Jot, there were no other agreed specifications available to represent digital-ink. Jot defines a format for the electronic ink storage and interchange across applications. As an initial specification, Jot had several limitations. Jot was not designed to support lots of in-memory manipulations of the digital ink such as changing stroke thickness, modifying stroke color, deleting strokes *etc.* Jot also lacked the flexibility to support real-time sharing and streaming of digital ink.

UNIPEN [21]

Another digital-ink representation format, UNIPEN, was formulated in 1994 at the AT&T Bell Laboratories. It was focused on representing training data for handwriting recognition. It failed to provide enough flexibility for other applications.

XPEN [32]

XPEN is another ink format that came into existence in 2003, based on UNIPEN. It is XML based and mainly used for distributed online handwriting recognition.

ISF (Ink Serialized Format) [11]

ISF, Microsoft's proprietary format, is a closed binary digital ink format whose usage is limited to the Windows environment. It is not a suitable data representation standard for a cross-platform application.

SVG (Scalable Vector Graphics) [16]

SVG 1.1 specification for digital ink representation was released by W3C in 2003. The design of SVG lacks scalability as it supports only X and Y channels and therefore, not a good choice for multimodal application development where one has to store the information about a number of channels.

InkML [6]

InkML is the latest standard for the representation of digital ink, whose first working draft was published by W3C in 2003. At present, it is the most flexible and a feature rich digital ink representation standard. We will have a detailed discussion about InkML specification in Section 2.3 below.

2.2.2 Audio Standards

Electronic whiteboards, when integrated with audio, can help in the recording of information-rich multimedia data which is beneficial for future retrieval. Dealing with audio needs a careful consideration, so that efficient storage and fast retrieval is possible. Special care should be taken to set the level of acceptable loss in audio storage, so as to ensure reliability and clarity when retrieved.

Three major groups of audio formats exist:

1. Uncompressed audio format
   e.g. WAV, AIFF, AU *etc.*
2. Formats with lossless compression

   e.g. FLAC, Monkey's Audio, WavPack, Tom's lossless Audio Kompressor (TAK), TTA, ATRAC, Apple lossless, lossless Windows Media Audio (WMA) *etc.*
3. Formats with lossy compression

   e.g. MP3, Vorbis, MusePack, ATRAC, Lossy WMA, AAC *etc.*

As we use Skype in our trials, we pay particular attention to the format it uses. Skype handles audio in the PCM representation, which is an uncompressed audio data format. The Skype audio is 16000 Hz 16 bit mono and streamed in the WAV format in Windows or in the AIFF format on a Mac OS X. Considering the limited storage capability of mobile devices, our application does further compression of the WAV format to produce an MP3 file.

The details of audio handling are explained further in Section 5.1.3.

2.2.3 Multimodal Data Representation Standards

SMIL, Synchronized Multimedia Integration Language, is a multimodal data representation standard proposed by W3C. It is a text-based multimodal data representation standard. It can handle audio, text, video, images and animations suitably, but does not have a strong support for digital ink representation. Its limitations are presented in Section 4.1.3, where we also talk about why we chose InkTracer application over SMIL for our implementation.

VoiceXML is another W3C standard for voice-based web applications. Processing of VoiceXML documents is done by a voice browser. Applications with interactive voice dialogs can be created with VoiceXML, similar to visual applications developed with HTML. Figure 2.1 shows a simple VoiceXML document with a voice prompt.

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
    <form>
       <block>
           <prompt> Voice XML </prompt>
       </block>
    </form>
</vxml>
```

Figure 2.1: An example of a simple VoiceXML document.

2.3 InkML

InkML [6] is the Ink-Markup Language that provides a data format for the representation of the digital ink generated by a stylus or an electronic pen. Being XML-based, it gives a neat and flexible way of representing the digital ink. Applications such as handwriting recognizers, paint programs *etc.* can process the digital ink with InkML as the common format. InkML also has an extensive support for a wide range of hardware devices. It is a platform-independent standard that can be used by applications on different operating systems. Hardware and software vendors had previously been using proprietary formats for ink representation. With the emergence of InkML as an open standard, proposed by the W3C Multimedia Interaction Activity Group [19], a wide variety of applications have been using it for data representation.

InkML is the medium for digital ink representation in our InkAndAudio multimodal chat application. Although other digital ink formats such as UNIPEN [21], Jot [12] *etc.* do exist, our choice was InkML based on its being an open and an up-to-date standard.

2.3.1 InkML Components

The *<ink>* Element

All InkML documents are well-formed XML documents. Based on the InkML specification syntax, every InkML document has an *<ink>* element as the root. All the allowed sub-elements of the *<ink>* element are enclosed within the *<ink>* and *</ink>* pair. The allowed sub-

elements include <*definitions*>, <*annotation*>, <*traceGroup*>, <*traceView*>, <*context*>, <*trace*> and <*annotationXML*>, which can occur any number of times under the root.

The *<definitions>* Element

The purpose of a *<definitions>* element is to define reusable contents in an InkML file. Definitions within a *<definitions>* element are then referenced by other elements in the InkML file using *id* attribute of the definition. Contextual information such as brush width and color, canvas properties, device specific ink source, different trace formats *etc.* can also be stored within *<definitions>*. These contextual information stored are then referenced by elements using attributes *brushRef*, *canvasRef*, *inkSourceRef*, *traceFormatRef* respectively for <*brush*>, <*canvas*>, <*inkSource*> and <*traceFormat*> elements. Figure 2.2 illustrates usage of the *<definitions>* element.

```
<ink>
   <definitions>
     <brush xml:id="redPen"/>
     <brush xml:id="bluePen"/>
     <traceFormat xml:id="normal"/>
     <traceFormat xml:id="noForce"/>
     <context xml:id="context1" brushRef="#redPen"
                                traceFormatRef="#normal"/>
     <context xml:id="context2" contextRef="#context1"
                                brushRef="#bluePen"/>
   </definitions>
   <context contextRef="#context2" traceFormatRef="#noForce"/>
   <context xml:id="context3"/>
</ink>
```

Figure 2.2: An example of <*definitions*> element in an InkML file, from [6].

The *<inkSource>* Element

The format and quality of ink from a digitizer is device specific. For applications to handle captured strokes coming from various sources it becomes necessary to understand the reported ink format. The *<inkSource>* element comes handy to store source related information such as the device model, ink channels captured, trace format, sampling rate, channel properties *etc.* Sometimes devices have certain systematic errors in their measurements and these must be taken into account for applications such as forensic handwriting analysis. The *<inkSource>* element is useful for this purpose. Figure 2.3 illustrates usage of the *<inkSource>* element.

```
<inkSource xml:id = "mytablet"
    manufacturer = "Example.com"
    model = "ExampleTab 2000 USB"
    specificationRef="http://www.example.com/docs/2000usb.html">

    <traceFormat href="#XYF"/>
    <sampleRate uniform="True" value="200"/>
    <activeArea size="A6" height="100" width="130" units="mm"/>
    <srcProperty name="weight" value="100" units="g"/>
    <channelProperties>
         <resolution channel="X" value="5000" units="1/in"/>
         <resolution channel="Y" value="5000" units="1/in"/>
         <channelProperty channel="Y" name="peakRate" value="50"
                                      units="cm/s">
         <resolution channel="F" value="1024" units="dev"/>
    </channelProperties>
</inkSource>
```

Figure 2.3: An example of <*inkSource*> element in an InkML file, from [6].

Trace and Trace Groups

The *<trace>* element represents individual ink strokes and stroke properties such as *X*, *Y* coordinates, pen tip force, brush reference, time offset, duration of stroke *etc.* The data stored by a *<trace>* elements have to be in accordance to the format of the trace as defined by a *<traceFormat>* element.

The *<traceGroup>* element is used to represent a collection of traces that have common attributes *e.g.* a group of traces coming from the same source.

As shown in Figure 2.4, the *<traceFormat>* element defines a sequence of channel values that occur within a *<trace>* element. The order of channel values within a *<trace>* element has to be in accordance to the order as defined by a *<traceFormat>* element.

```
<definitions>
   <traceFormat id="format1">
       <channel name="X" type="integer"/>
       <channel name="Y" type="integer"/>
   </traceFormat>
</definitions>
<traceGroup>
    <trace brushRef = "bb0b9659-96a4-4263-ae82-861cf146bdc4"
           timeOffset = "4449"
           duration= "674">
      68 41,
      69 42,
      69 44,
    </trace>

    <trace brushRef = "bb0b9659-96a4-4263-ae82-861cf146bdc4"
           timeOffset = "19398"
           duration= "100">
      102 95,
      101 94,
    </trace>
</traceGroup>
```

Figure 2.4: An example of <*trace*> and <*traceGroup*> in an InkML file.

Channel and Channel Properties

In an InkML file, a *<traceFormat>* has one or more <*intermittentChannel*> and *<channel>* elements as its children, the order of which determines the order of channel values inside a *<trace>* element. A *<channel>* element sets the properties of a channel using attributes such as *name*, *type*, *min* value, *max* value *etc.*

Each channel has a required *name* attribute which has to be one of the reserved channel names provided by the InkML specification. Table 2.1 shows a list of reserved channel names and their meanings.

The *<channelproperties>* element is used to store the properties of a channel for a *<traceFormat>* defined within a *<definitions>* element.

Channel Name	Dimension	Interpretation
X	length	X coordinate. This is the horizontal pen position on the writing surface, increasing to the right for +ve orientation.
Y	length	Y coordinate. This is the vertical position on the writing surface, increasing downward for +ve orientation.
Z	length	Z coordinate. This is the height of pen above the writing surface, increasing upward for +ve orientation.
F	force	pen tip force
S		tip switch state (touching/not touching the writing surface)
B1...Bn		side button states
OTx	angle	tilt along the x-axis
OTy	angle	tilt along the y-axis
OA	angle	azimuth angle of the pen (yaw)
OE	angle	elevation angle of the pen (pitch)
OR	angle	rotation (rotation about pen axis)
C		color value (device-specific encoding)
CR,CG,CB		color values (Red/Green/Blue)
CC,CM,CY,CK		color values (Cyan/Magenta/Yellow/Black)
W	length	stroke width (orthogonal to stroke)
T	time	time (of the sample point)

Table 2.1: Channel names and their meanings, from [6].

Annotations

The *<annotation>* element can be used to save some textual information in an InkML file. Traces can be tagged using the *<annotation>* element. For example, the GUID (Global Unique ID) of an application can be stored using the *<annotation>* element and associated

with a *<trace>* element. Figure 2.5 illustrates usage of the *<annotation>* element.

```
<ink xmlns:dc="http://dublincore.org/documents/dcmi-namespace/">
    <annotation type="description">A Sample theorem</annotation>
    <annotation type="writer">Albert Einstein</annotation>
    <annotation type="contentCategory">Text/en</annotation>
    <annotation type="language" encoding="ISO639">en</annotation>
    <annotation dc:language="en"/>
    <trace id="trace1"> ... </trace>

    <traceGroup id="tg1">
        <annotation type="truth">Hello World</annotation>
        <traceGroup>
            <annotation type="truth">Hello</annotation>
            <trace> ... </trace>
            ...
        </traceGroup>

        <traceGroup>
            <annotation type="truth">World</annotation>
            <trace> ... </trace>
            ...
        </traceGroup>
    </traceGroup>
    <traceView href="#tg1">
        <annotation type="style">Cursive</annotation>
    </traceview>
</ink>
```

Figure 2.5: Using <*annotation*> to save extra information in an InkML file, from [6].

2.3.2 Handling Digital Ink with InkML

When a digital pen or stylus writes on a surface sensitive to pen motion, the device can record those pen strokes in the form of X, Y coordinates, pen tip force, pen tilt and many other useful information. This stroke information can be saved for future reference or can be streamed to some other application by means of a transfer protocol. This is where InkML comes into picture. The input strokes are stored in an InkML file using the available constructs as explained in the previous Section 2.3.1.

Figure 2.6 shows an InkML document consisting of the digital ink information involved in a 2-party chat session. Such InkML documents can be rendered by our InkAMP player, covered in Section 5.4.

```
<ink>
  <definitions>
    <traceFormat id="fmt1">
       <channel name="X" type="integer"/>
       <channel name="Y" type="integer"/>
    </traceFormat>
  </definitions>
  <inkSource id="myTablet" manufacturer="HP.COM" model="HP-IpaQ">
     <sampleRate uniform="True" value="200"/>
     <channelProperties>
         <channelProperty channel="X" name="quantization"
                                      value="500" units="pixels"/>
         <channelProperty channel="Y" name="quantization"
                                      value="500" units="pixels"/>
     </channelProperties>
   </inkSource>
   <traceGroup brushRef="bb0b9659-96a4-4263-ae82-861cf146bdc4">
      <annotation type="Color">126247</annotation>
      <trace  timeOffset = "4449"
              duration= "674">
          68      41,
          69      42,
          69      44,
      </trace>
      <trace   timeOffset = "19398"
               duration= "100">
          102      95,
          101      94,
          101      94,
      </trace>
    </traceGroup>
</ink>
```

Figure 2.6: Sample InkML file for a 2-participant chat session.

2.3.3 Ink Capture and InkML Rendering

Several SDKs (Software Development Kits) such as IBM's CrossPad and Microsoft's Tablet PC SDK provide APIs that can be used to capture the ink information generated by a stylus. The introduction of <*InkCanvas*> control in Microsoft's Windows Presentation Foundation (WPF), included in .NET Framework 3.0 gives us a simple and easy to use API for the development of digital ink applications. As compared to the other SDKs, WPF's <*InkCanvas*> defines the handling of ink strokes more neatly.

A simple InkML file looks like the example in Figure 2.7.

```
<ink>
<trace>
    10 0, 9 14, 8 28, 7 42, 6 56, 6 70, 8 84, 8 98, 8 112,
    9 126, 10 140, 13 154, 14 168, 17 182, 18 188, 23 174,
    30 160, 38 147, 49 135, 58 124, 72 121, 77 135, 80 149,
    82 163, 84 177, 87 191, 93 205
</trace>
<trace>
    130 155, 144 159, 158 160, 170 154, 179 143, 179 129,
    166 125, 152 128, 140 136, 131 149, 126 163, 124 177,
    128 190, 137 200, 150 208, 163 210, 178 208, 192 201,
    205 192, 214 180
</trace>
<trace>
    227 50, 226 64, 225 78, 227 92, 228 106, 228 120, 229 134,
    230 148, 234 162, 235 176, 238 190, 241 204
</trace>
<trace>
    282 45, 281 59, 284 73, 285 87, 287 101, 288 115, 290 129,
    291 143, 294 157, 294 171, 294 185, 296 199, 300 213
</trace>
<trace>
    366 130, 359 143, 354 157, 349 171, 352 185, 359 197,
    371 204, 385 205, 398 202, 408 191, 413 177, 413 163,
    405 150, 392 143, 378 141, 365 150
</trace>
</ink>
```

Figure 2.7: Ink traces for *hello* strokes, from [6].

When rendered, the strokes in Figure 2.7 gives the handwritten impression of *hello* as shown in Figure 2.8.

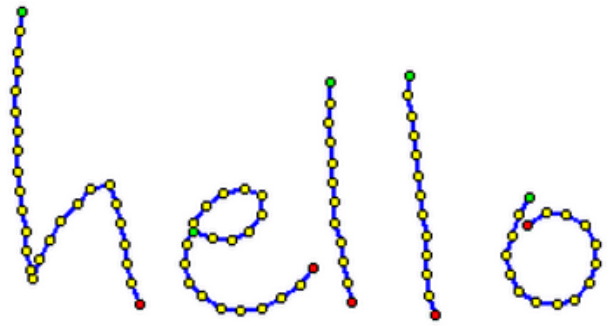


Figure 2.8: InkTracer when used for *hello* stroke rendering, from [6].

2.3.4 Archival *vs.* Streaming of Digital Ink

InkML, being XML-based, is flexible and can be extended to suit our implementation requirements. Two forms of InkML, archival InkML and streaming InkML have been discussed in the work by Keshari *et. al.* [28]. The former representation is used by applications that stream digital ink and the latter by applications that store digital ink for future reference. A conversion technique has also been discussed that helps inter-application communication involving either forms [28].

Our InkAndAudio Chat application saves the digital ink strokes in the archival form along with audio. The InkAMP player is capable of reading such archives. Our InkAMP player renders the digital ink into a drawing canvas and plays the audio with synchronization.

2.4 InkTracer

For our demonstration implementation of a combined ink and audio media player, we required an ink animation component. InkTracer is a Java application, developed by colleagues at Ontario Research Center for Computer Algebra (ORCCA). It first reads a valid InkML file to capture all

the stored stroke information. It then paints the strokes on the application-canvas. The canvas in InkTracer is a JPanel component that can be drawn and painted.

InkTracer was a suitable digital ink data renderer for our InkAMP media player, as it was developed in Java and it could be run on Windows, Linux, Mac OS X, Solaris and many other operating systems. The design of InkTracer is shown in Figure 2.9.

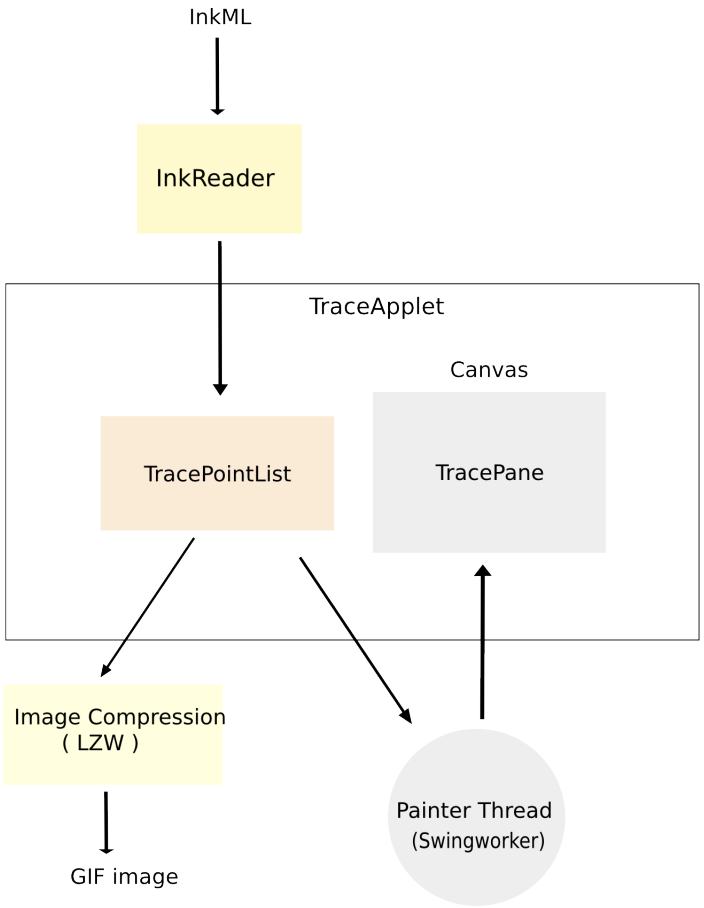


Figure 2.9: The InkTracer design.

2.5 The jlGui Music Player

For our demonstration implementation of a combined ink and audio player, we also require an audio player component. The jlGui [24] is an open source graphical music player written in Java. Its front-end is compliant with Winamp 2.0 skins and it has often been regarded as the Java clone for Winamp media player. The jlGui player was our choice as an audio player in InkAMP as it was a platform independent player and could easily be integrated with the InkTracer application. The jlGui player supports WAV, AU, AIFF, SPEEX, MP3, OGG, Vorbis, FLAC, Monkey's Audio and other audio formats.

The jlGui player relies on the JavaSound Service Provider Interface (SPI) architecture, to load audio codecs on JVM startup. Use of the SPI plug-in interface makes the design more elegant as it can further be used to extend JavaSound to support new file formats. A simple addition of a JAR file in the CLASSPATH is all that is required to add support for a new format. This design approach also makes the BasicPlayer source code independent of any support enhancement for new audio file format.

The design of jlGui player has two main layers as shown in Figure 2.10. BasicPlayer layer is based on JavaSound API and is the player API of jlGui. It provides high level API to stream audio or to play, pause, stop, resume and seek the audio. BasicPlayer layer can be used in any application.

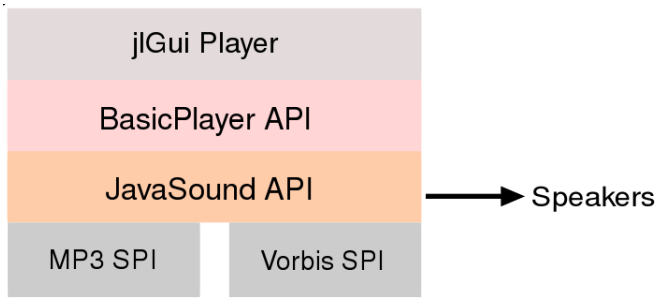


Figure 2.10: The jlGui player design, from [24].

The jlGui Player layer gives a simple to use front-end. The classes presented by this layer interact with classes in BasicPlayer layer to handle an audio playback.

2.6 Peer-to-Peer Networks and Skype

We have chosen to use Skype as a backbone for ink and audio collaboration at a distance. This section outlines the aspects of peer-to-peer networks, and Skype in particular, on which we rely.

2.6.1 Peer-to-Peer Networks

The history of peer-to-peer network dates back to July 1999 with the publication of Freenet protocol [7] and escalates to fame in September 1999 with the creation of Napster. Every new technology is sceptically viewed in being a successful model unless there is at least one so called killer application as a proof-of-concept. Napster was the pioneer, a music file-sharing application and the killer application that drew attention of many researchers by attracting 26.4 million users within a short span (Sep 1999 - Feb 2001). Since then, many successful applications have been developed over peer-to-peer and the applications have gone beyond computer science and touched bioinformatics, education, military, business, telecommunication and many more.

2.6.2 Skype VoIP

Skype is at present the worldwide leader in VoIP and has an ever-increasing user-base with over 300 million registered users [44]. Skype looks like the famous MSN or Yahoo instant messaging applications and these have many features in common, *e.g.* audio conferencing, instant messaging, voice calls, buddy lists *etc.* The underlying network protocols and implementation techniques in Skype, however, are quite different from that of those instant messengers [4].

Skype has been a front-runner in the VoIP business because of its better voice quality compared to the famous Yahoo or the MSN Instant Messaging services.

KaZaA [27], the predecessor of Skype, was the most popular file sharing application of its time. Similar to its predecessor KaZaA, Skype is an overlay peer-to-peer network. It is composed of normal-nodes (NN) and super-nodes (SN) which are organized into two levels as shown in Figure 2.11. P2P overlay networks are decentralized distributed systems where peers form a self-organizing network which is more robust and scalable than the traditional client-server architecture. As compared to other P2P networks, Skype uses super-nodes to improve the scalability.

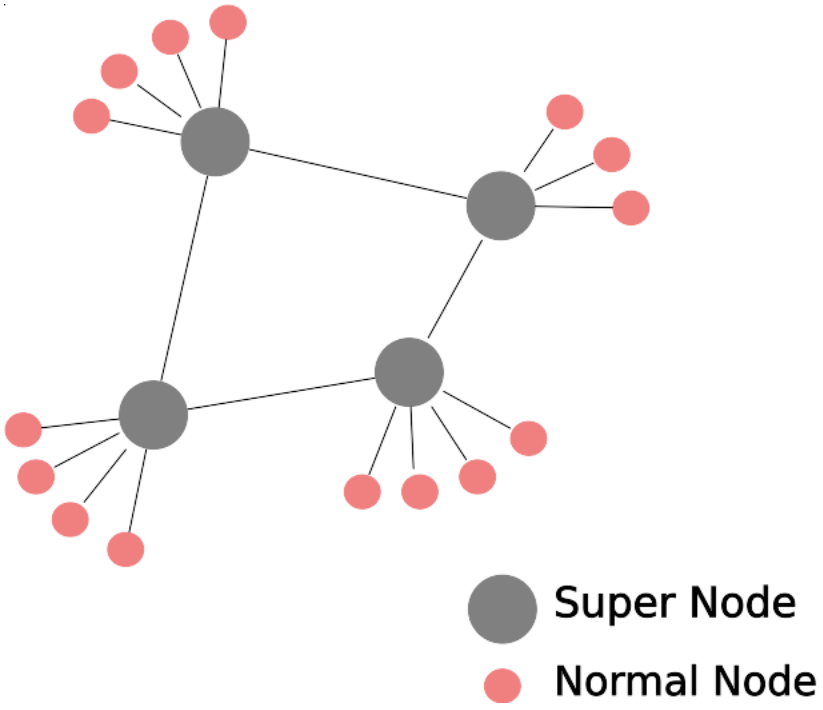


Figure 2.11: Skype P2P network layout.

Skype super-nodes are special nodes with spare bandwidth and strong hardware configuration. They are interconnected special nodes that help relay the queries amongst each other and act as a proxy connection to nodes that are behind firewalls. Super-nodes are publicly reachable, unlike the nodes that are behind Network Address Translators (NATs) or firewalls. Normal-nodes

keep a *"host cache"* of publicly available nodes. Each node in the network gets connected to one such super-node which helps in relaying its queries.

2.6.3 Skype API

Skype API has two layers, the Communication layer and the Command Protocol layer. The Communication layer aims at providing a communication channel between external applications and Skype. First thing an application does is to get attached to Skype using this layer. Once attached, the application makes use of the Command Protocol layer. The Protocol layer is a language of commands which are in UTF-8 format, that Skype knows how to respond to [34].

Text-based cross-platform API provided by Skype works based on Operating System messaging with Windows Messaging on Windows; Cocoa, Carbon or AppleScript in Mac OSX and X11 or D-BUS messaging in Linux environment.

The Skype API can be used by Skype applications for the following forms of communication:

- **Call :** Start, Control, Conference, Transfer, Record.
- **Chat :** Send/Receive.
- **SMS :** Send/Receive from any mobile in the world.
- **App2App :** Send/Receive messages invisible to the users.

Skype public API have been exported in various forms. The Skype client API is composed of a simple protocol with commands, responses and notifications. Skype4COM, the COM wrapper for Skype API represents Skype API as objects with properties, commands, events and notifications. The main objective of this COM wrapper is to reduce the complexity in Skype-based application development for Windows. Skype4COM was our choice for the implementation as it could be used in any ActiveX environment, such as Visual Studio or Delphi, and use familiar scripting languages, such as Visual Basic, PHP, JavaScript *etc.* Skype4COM integrates well with the digital ink handling features available in Windows Presentation Foundation (WPF), included

in the .NET Framework 3.0. Figure 2.12 illustrates an example C# code that uses Skype4COM.

```
SKYPE4COMLib.Skype objSkype;
SKYPE4COMLib.Call objCallOne, objCallTwo;
objSkype = new SKYPE4COMLib.Skype();
objSkype.Attach(7,true);
objCallOne = objSkype.PlaceCall("test","","","");
while(objCallOne.Status != SKYPE4COMLib.TCallStatus.clsInProgress)
    { }
objCallOne.Hold();

objCallTwo = objSkype.PlaceCall("echo123", "", "", "");
while(objCallTwo.Status != SKYPE4COMLib.TCallStatus.clsInProgress)
    { }
objCallTwo.Join(objCallOne.Id);
```

Figure 2.12: Using Skype4COM in C# to demonstrate a conference call.

Skype API is also available for web applications. Skype presence buttons can be embedded into web pages and Skype4COM can be programmed using JavaScript. Figure 2.13 and Figure 2.14 illustrate the use of Skype4COM in web applications.

```
<html>
<body>
   <object id=Skype name=Skype
              classid=clsid:830690FC-BF2F-47A6-AC2D-330BCB402664
              codebase="http://www.yoursite.com/Skype4COM.dll" >
        <span style="color: red">Failed to load control.</span>
   </object>
</body>
</html>
```

Figure 2.13: Using Skype4COM in an HTML file.

```
function playAudio(oCall)
{
    var cmdid = 1001;
    var callid = oCall(1).Id;
    var filename = "C:\\test.wav";
    var command = 'ALTER CALL ' + callid + ' SET_INPUT FILE="'
                  + filename + '"';
    Skype.SendCommand(Skype.Command(cmdid, command, 'INPUT'));
    alert(command);
}
<input type="button" onclick="playAudio(Skype.ActiveCalls)"
       value="Play Audio Message" />
```

Figure 2.14: JavaScript code making use of Skype4COM wrapper functions.

Since Skype 1.4, (*"Skype: links"*) URIs can be used on any browser or any application that has hyperlink support and it does not need authorization [25].

Pre-built libraries such as Skype4COM, Skype4Py and Skype4Java make it easier to handle the native Skype API calls, written in C++. Skype4COM implements Skype API through ActiveX object on Windows. Skype4Java is a Java wrapper and Skype4Py is a Python wrapper which works on Windows, Linux and Mac OS X. More work is being done to make the API feature rich and robust.

2.6.4 Skype Advantages

According to a security evaluation done by Tom Berson [5], Skype has used cryptography widely and well, which made it gain a solid foundation of trust, authenticity and confidentiality for the peer-to-peer services it provides. Content transmitted across Skype network cannot be easily disclosed by parties other than the peers. Compared to other VoIP systems, the typical bandwidth load on a Skype super-node is relatively low, even while it is relaying the VoIP traffic [20].

Some of the services provided by Skype include VoIP, IM, file transfer *etc.* SkypeIn and SkypeOut are paid services that let a user initiate and receive calls via regular telephone numbers through VoIP-PSTN gateways. User registration and username/password authentication is

handled by the Skype login server which is located in Europe (apparently in Amsterdam [18]).

The major advantages of Skype can be listed as follows:

1. **Multipurpose application**

   Skype client can be used for IM, VoIP, file transfer *etc.* Unlike KaZaA, the free downloadable Skype client comes without spyware.

2. **Secure and confidential communication**

   Unlike ISDN phone calls which generally are not encrypted, Skype calls are encrypted with 128-bit or better cryptography ciphers, making it difficult for someone to decipher any intercepted Skype conversation [18].

3. **Cross-platform availability**

   Skype client works on Windows, Linux, Mac OS X, Pocket PC and many other environments. Also, Skype API is open to public and its availability in many languages such as C#, Java, Python, JavaScript, Delphi *etc.* makes Skype and Skype-based applications more acceptable in a number of known platforms.

4. **Cost effective**

   Along with free IM, VoIP and file transfer facilities, even the SkypeIn and SkypeOut paid services are cost effective as compared to other VoIP and ISDN call charges.

5. **Reliable service**

   Besides being extremely easy to install and use, Skype offers reliable services, as it works seamlessly behind firewalls and NAT systems. Skype uses STUN [50] and TURN [49] protocols to determine the type of NATs and firewalls and then bypass them when necessary.

Skype has an established reputation for audio and text chat. The Skype API is available to the public and is evolving rapidly. Skype4Java and Skype4Py, the Java and Python extensions have also been developed to support cross-platform application development, based on the Skype API [34].

Because of all these benefits, it would be hard to imagine any better back-end support for the audio chat in our multimodal application and thus, Skype peer-to-peer network was our choice for the implementation.

Chapter 3

Multimodal Application Design Issues

Pervasive computing is gaining more popularity in everyday life. More and more multimodal applications are developed that make use of the ubiquitous computing available through countless new miniature devices. Use of multimodal applications is spreading to a larger pool of the world's population. Handheld devices have limited computing capacity which has to be addressed during the design phase, so that the limited resources can be used more efficiently.

When it comes to multimodal applications, the system design becomes more complex because of the availability of multiple input modes. Front-end design should take care of the user behavior and needs a careful understanding of the human-computer interaction. Back-end design should be focused on making the application more secure and robust.

To make the system design simple, several application development frameworks have been created. To produce computer-generated speech, some applications make use of a TTS (Text-To-Speech) engine. This helps an application to generate voice-based user interfaces. Several news channels already have an automated news reader for the website articles. Browsers such as Opera are sometimes speech enabled and can run such multimodal applications with voice-interfaces. The IBM Multimodal Toolkit [22] is an effort to assist such types of application development. It makes use of XHTML and VoiceXML to provide suitable interfaces to the web applications. With such applications, users need not use a mouse, keyboard, stylus or even the monitor if voice input can be parsed in the web forms [48].

The fact that the IBM Multimodal Toolkit needs Windows 2000 or later and the Opera browser for rendering, restricts applications from reaching a larger pool of computing devices. Our prime focus here is to come up with a platform-independent multimodal chat tool.

3.1 Input Modes

Modern technology has brought about numerous input and output modes that include digital ink, touch screen, voice, video *etc*. Multimodal interface design therefore has become an active field of research. Work by Oviatt *et al.* [40] discusses various interface design aspects of multimodal application development and remark on their higher potential to bring computation to more challenging applications and to be used by a broader spectrum of the world's population.

3.1.1 Voice as Input

Voice is the most favored mode of input. Work by Oviatt *et al.* [42] shows that in a pen-voice multimodal interaction, voice is the preferred mode for descriptive information exchange whereas pen is preferred for graphics and symbols.

The advantages of voice as an input mode are:

- Speed of communication
- Easy to use
- Requires less attention *e.g.* while driving

3.1.2 Digital Ink as Input

With the wide acceptance of tablet PCs for our day-to-day work and with the rapid development of pen computation enabled handhelds, digital ink has become a major input mode.

The advantages of digital ink as an input mode are:

- Portability; PDAs and even mobile phones offer pen input.
- Convey symbols, signs and graphics effectively
- Point and select
- Privacy; gives a private and a socially acceptable form of input in social settings.
- Useful in extreme noise

3.1.3 Voice and Pen Multimodal Input

Multimodality with voice and pen as input modes not only brings the advantages of the individual input modes but also adds more benefits.

The advantages of voice-pen multimodal systems are:

- Adaptive to changing environment
- Freedom of use
- More expressive communication
- Easy collaboration
- Complementary capabilities of the input modes
- Graceful error handling with error avoidance and easy error resolution capabilities
- Broad range of users based on age, skill level, language, physical disability *etc*.

3.2 Design Issues

Cross-platform multimodal application design and development often face the following considerations [33].

3.2.1 Resource Management

A resource is a physical component of a computing system with limited availability. One aspect of having multimodal applications is to be able to use them in various mobile devices, avoiding the use of keyboard and mouse. Such devices have resource limitations such as less storage capacity, lower system memory, lower processing speed *etc*. Therefore, it becomes necessary to manage the resources well, failure of which results in poor performance of the system which might lead to an application breakdown. Unreliable applications are less preferred and a user will look for better alternatives.

Hard disk storage, RAM and virtual memory, processor time, electric power, network throughput are such system resources that have to be managed properly. The design of multimodal applications should take into account the storage limitation of a device to make an application run seamlessly. Proper compression techniques need to be applied while storing the data.

3.2.2 Interoperability

New devices and technologies are coming up in various sectors that include medical establishments, military, education *etc*. Such innovations have an immense potential to improve the communication if integrated effectively with *"plug-and-play"* interoperability. When an application needs to be run in heterogeneous devices and environments, the design should address its interoperability. This ensures higher accessibility and attracts more users. Similar features need to be made available across platforms. The foundation of the information exchange needs a com-

mon set of data formats and needs to use the same protocol. Our use of open standards for data representation is one step closer to gaining higher interoperability. Our implementation targets the most common operating systems in use, such as Windows, Linux and Mac OS X. Once the goal of providing interoperability across these platforms has been met, further extensions can be developed for other mobile platforms such as Pocket PCs, PDAs, smart phones *etc.*

3.2.3 Dialogs

Dialogs or prompts play a vital role in the design of a multimodal user interface. Users should not be in a dilemma regarding how to answer a prompt. Well co-ordinated dialogs make the communication more effective. Special care should be taken while designing real-time multimodal systems. Un-synchronized dialogs can lead to mis-interpretation which might lead to a disastrous situation while handling sensitive operations.

3.2.4 Synchronization

Synchronization is the most important issue to be dealt while developing a multimodal application. Visual interface must be in synchronization with the audio interface, failing to do so brings confusion and unexpected dialog transitions. Keeping the content short and to the point might be beneficial for maintaining synchronization.

Synchronization is also important for a multimedia player that plays the recorded multimodal interactions. Saved data must be carefully timestamped using features provided by the data format specification and rendering should make use of those timestamps to maintain proper data synchronization while playback.

3.2.5 Voice *vs.* Visual Modes

A multimodal interaction has voice and visual components. This is helpful as some information is better represented in voice, whereas, others are more suitable in visual form. A multimodal application will have better impact when the input modes are used suitably *e.g.* introductory information will grab more attention when presented with voice, visual information will leave better impact with color, graphics or diagrams.

Multimodal applications have a large pool of users. Visual information will not be useful for visually impaired users. Such cases have to be addressed while designing the user interface. Proper selection of input and output modality is therefore very important while designing multimodal applications.

3.2.6 Static *vs.* Dynamic

Depending on the purpose of an application, the information display might have to change, based on time or user input. Online ticket purchase is one such example where information display keeps changing based on the user input. Such application's interface must be dynamic. When there is no such need for visual information to change, static application interfaces will suffice.

A cross-platform multimodal application for collaboration should include dynamic interfaces that can hold information about the participants activities. Such information helps resolve ambiguities, especially when one can not see what the other participant is doing.

3.2.7 Secure Communication

When multimodal applications are used for collaboration over public networks, it becomes necessary to ensure secure and confidential communication. The design has to address the use of encryption techniques for data transfer across the network, making it difficult for someone to decipher the intercepted data. Nowadays, when people do business with their handhelds and pre-

fer to migrate most of the work from their office workstations to mobile platforms, secure data transfer is crucial as it might lead to important business information leakage to the outside world.

Chapter 4

Skype-based Multimodal Chat Design Issues

We have seen examples of collaborative multimodal applications with voice and digital ink, but most of them have complex data exchange mechanisms [1, 38]. A considerable amount of complexity arises while developing a multimodal application based on a client server model [1]. Most of them have platform dependence because of their use of proprietary standards for data representation [15, 43].

In contrast, we base our design on Skype. Skype is a peer-to-peer network-based VoIP service that claims to be providing a secure and reliable services to more than 300 million registered users [44]. Skype peer-to-peer network comes to our rescue with robustness in its design and feature-rich API available to the public. Inclusion of Skype peer-to-peer network in the implementation solves most of the design issues we discussed in Chapter 3.

Resource Management

Most of the CPU intensive workload that comes with data transfer across the network is handled by the Skype client which is very reliable. Moreover, the client is available for Pocket PCs and many smart phones. This shows that the client efficiently manages the resources even for mobile devices and handhelds where system resources are limited.

Interoperability

The Skype client has been deployed across platforms such as Windows, Linux, Mac OS X, Pocket PC, Smart Phones, PDAs *etc*. Skype uses operating system messaging to handle the communication and the Skype API has been made available to many known platforms.

Conditions for a digital ink and audio application to be truly interoperable are:

- Digital ink should be understood in all the platforms
- Difference in the range of the channel-values captured should be understood (for example, the scale of X and Y)
- Difference in the resolution and the sampling rate of various ink sources should be understood
- Need for an efficient protocol for ink exchange across platforms

Skype peer-to-peer gives us the efficient protocol for stroke exchange. InkML further helps us achieve interoperability as InkML is extensible, gives a platform independent representation for digital ink, gives methods to group and transform digital ink when necessary and supports archival and streaming of ink [28].

Secure Communication

Skype makes use of Advanced Encryption Standard (AES) to encrypt data before transfer over the network. Skype uses 256-bit encryption that has a total of about 1.1×10^{77} possible keys, in order to encrypt data in each Skype interaction and uses 1536 to 2048 bit RSA to negotiate symmetric AES keys [4]. It is almost impossible for someone to decode the data transferred through Skype. Therefore, the InkAndAudio application can be used for business meetings and conferences which demand a secure mode of communication.

4.1 Issues Faced in Application Development

The InkAndAudio multimodal chat system is a collection of integrated applications. To meet the required cross-platform support, PyInkAndAudio client was developed for Linux and Mac OS X. It was necessary to understand various platforms and their application development environment in order to develop the chat clients. At the time of writing this thesis, Skype API wrappers were undergoing rapid development. Just a limited amount of online documentation was available. This made the application development even more challenging.

The development of a multimedia player for archived multimodal chat sessions was another challenge. First, we had to find independent players for ink data and audio. Then, we had to integrate those players to be able to play audio and ink together, with synchronization.

In the following subsections, we discuss the various issues faced during our application's development.

4.1.1 Choice of Data Representation

In Section 2.1, we discussed several multimodal applications that use digital ink. Most of them use proprietary data representation standards. This limits interoperability and the application's use is confined to fewer platforms. Section 2.2 explains the existing digital ink and audio standards and their features.

Our aim was to make the multimodal chat application available across multiple platforms. InkML, being an open, XML-based language, gave us the flexibility we were looking for. InkML became our choice for ink data representation along with MP3 file format for audio data, which again is an acceptable audio format across platforms.

Section 2.3 further demonstrates how InkML can be used for digital ink data representation.

4.1.2 Choice of Programming Environment

Several application development frameworks exist such as RiverInk [37], IBM CrossPad, Microsoft's Tablet PC SDK *etc.* It becomes necessary to choose a suitable application development environment. Failing to do so would result in slower development, certain inflexibilities in design, platform dependence and other problems.

Microsoft's Tablet PC SDK

Microsoft's Tablet PC Platform SDK had features to support digital ink in the programming environment and help in the Tablet PC application development. Since its first release in 2001, it has been improved significantly.

The API functions fall into 3 groups:

1. **Tablet input API**

   This API handles pen input from tablet digitizers. The InkCollector class and The InkOverlay class facilitate tablet input in the application code.

2. **Ink data management API**

   This API handles digital ink data management and basically deals with ink stroke objects. Ink class, being the entry point in the Ink Data API, defines a scope where multiple stroke objects can exist. Stroke objects represent a digital ink stroke and are the fundamental building blocks of an ink document.

3. **Ink recognition API**

   This API handles recognition of the input ink strokes.

Microsoft's Tablet PC SDK makes it possible for .NET developers to add ink support to their applications easily.

The Windows Presentation Foundation

The Windows Presentation Foundation (WPF) is a graphical subsystem in the Microsoft .NET framework 3.0. It uses the XAML markup language for rich user interface components such as UI elements, events, data bindings *etc.* Digital ink has been integrated directly into this presentation subsystem. Handwritten strokes can be parsed elegantly with the new ink analysis methods. With XAML, user interface design is simple and easy.

Digital ink collection and rendering, functions traditionally present only on the Tablet PC platform have been incorporated into WPF as first class member of the framework [53]. The InkCollector, InkPicture and InkOverlay controls found in the classic Tablet PC platform API are put together in the *<InkCanvas>* element with all of their functionalities, which was the main advantage of using WPF to build our Windows chat client. Figure 4.1 illustrates the use of *<InkCanvas>* element.

```
<InkCanvas xmlns="http://schemas.microsoft.com/avalon/2005">
    <InkCanvas.Background>
        <ImageBrush ImageSource="myBackground.jpg"/>
    </InkCanvas.Background>
</InkCanvas>
```

Figure 4.1: Using *<InkCanvas>* to setup a background image.

Tkinter with Python

Tkinter [46] is the most commonly used GUI programming toolkit for Python. Tkinter is portable between Mac OS X, Linux and Windows. Skype applications can be developed quickly with Skype4Py wrapper and Tkinter, for operating systems like Linux or Mac OS X, where Skype4COM wrapper and the .NET application development environment are not available.

Rapid Python Development with Tkinter, abbreviated as Rapyd-Tk [54], is a graphical user interface development tool that simplifies the GUI development with Tkinter, Python and Pmw modules on various platforms.

The challenges faced on Linux and Mac OS X chat client development will be covered in Section 4.1.6.

Cocoa and PyObjC for Mac OS X

Cocoa [23] is one of the native application programming environments provided by Apple Inc. for the Mac OS X operating system.

Applications developed in Cocoa comply with Apple's human interface guidelines. Usual Cocoa applications are written using Objective-C language. Now, bridging languages such as PyObjC, RubyCocoa *etc.* allow Python and Ruby scripts to use the existing Objective-C class libraries. This makes it possible for one to develop Skype applications for Mac OS X, using Cocoa, Skype4Py and PyObjC.

4.1.3 SMIL *vs.* InkTracer for Digital Ink Rendering

Synchronized Multimedia Integration Language, or SMIL was developed to bring presentation level interactive multimedia to the web. It is a text based open standard from the World Wide Web Consortium (W3C). SMIL has capabilities to combine multimedia such as text, audio, video, images, animations *etc*. into a single presentation and control the rendering with suitable time synchronization. SMIL markup has been considered as the multimedia version of HTML, but being written in XML, it has more flexibility. SMIL defines markup for layout, timing and visual transitions of the multimedia content. Figure 4.2 illustrates a simple SMIL document.

```
<smil>
   <body>
     <seq repeatCount="indefinite">
         <img src="house.jpg" dur="3s" />
         <img src="garage.jpg" dur="5s" />
         <img src="garden.jpg" dur="5s" />
     </seq>
   </body>
</smil>
```

Figure 4.2: A simple SMIL file for image slideshow.

SMIL presentations are played using SMIL players. Internet Explorer 5.5+ offers rendering capabilities for SMIL content embedded in an HTML code. Other available players include:

- RealOne platform by RealNetworks
- GRiNS for SMIL-2.0 by Oratrix
- SMIL player by InterObject

Though SMIL is a W3C recommended tool for multimedia presentation, it does not have features that could support easy rendering of the digital ink. As shown in the code listing of Figure 4.3, we can use the <*brush*> element of the *brushMedia* module in SMIL, to paint some section of the window. This looks like an answer to our problem of ink rendering. In reality, SMIL is not a feasible solution when our motive is to render a large number of strokes and maintain synchronization with the audio simultaneously.

```
<smil>
    ...
    <region regionName="window1" ... />
    ...
    <brush color="blue" region="window1" .../>
    ...
</smil>
```

Figure 4.3: Using <*brush*> element in SMIL.

4.1.4 Real-Time Data Transfer

For an interactive whiteboard application to be effective, data transfer among the participants should take place in real-time. Delay in the transmission results in an un-synchronized relay of information. To ensure real-time data transmission, care should be taken to set a proper size for the data packets and the frequency of data transfer. Sending small chunks of data more frequently gives better results than sending large chunks of data in wide intervals.

To best utilize Skype peer-to-peer network, it is more advisable to encourage participants to speak one at a time. This prevents the higher bandwidth requirement that can arise when more participants speak simultaneously. It is necessary, however, to support multiple simultaneous speakers.

Though digital strokes do not occupy as much volume as audio data, real-time stroke transmission needs to be ensured. As shown in Figure 4.4, long traces drawn are to be broken and transmitted in parts while sharing the whiteboard. Such long strokes can be stored as short *"continuation"* strokes in an InkML file and grouped together into a <*traceGroup*> element.

```
BEGIN:
    while(stroke capture continues)
    {
      if (stroke length exceeds the threshold)
            send the collected stroke-part as a stroke
    }

    if( stroke was broken and sent in parts)
        send the remaining part as a continuation
    else
        send the full stroke collected
END:
```

Figure 4.4: Algorithm to break lengthy strokes.

4.1.5 Synchronization

One of our objectives was to be able to play the audio and ink contained in the archived multimodal data. Proper synchronization during playback was crucial, without which the multimedia player would lose significance.

Our idea was to timestamp all the ink strokes involved in the interaction. The flexibility of InkML allows us to save the stroke timestamps in an InkML file. Every stroke stores the ink with a global time offset, which is the time elapsed since the start of the recording. The audio player also gives us a global time offset, which is the time elapsed corresponding to the player's slider position.

The InkAMP player, which will be described in Section 5.4, is a multi-threaded Java application with multiple painter threads, for multiple participants. The synchronization between these threads and the main audio playback thread is based on the global time offset saved with each stroke. The Java threads make use of "*Java barrier*" for synchronization as shown in Figure 4.5.

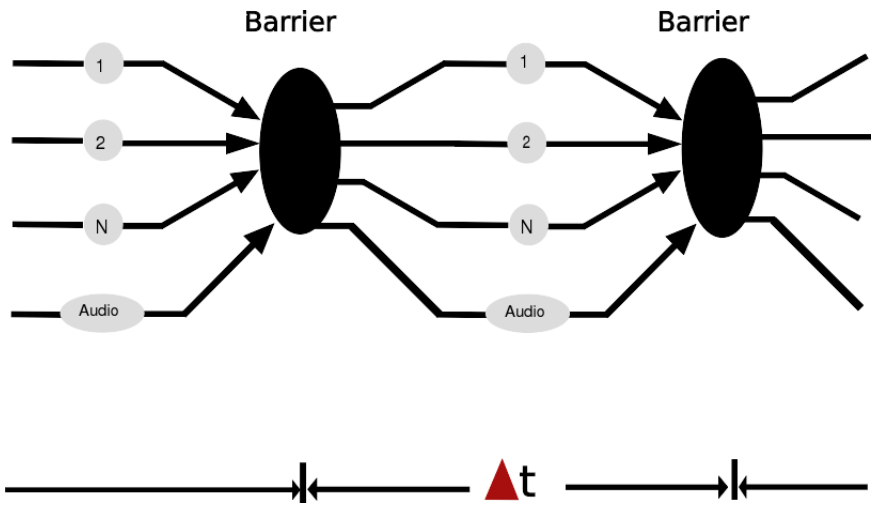


Figure 4.5: Java threads synchronization with "*Java barrier*".

The basic idea, as shown in Figure 4.6, was to make the audio thread wait for the painter threads when they lag behind. After each barrier interval (Δt), the audio thread blocks and waits for the painter threads, until they reach the barrier mark. The process repeats until the audio

thread completes the audio playback.

```
THREAD BEGIN
    set timer = 0
    while( painting of strokes in progress )
    {
         if( timer == Δt )
         {
              if( last thread to be blocked )
                   releaseAll()
              else
                   block()
              reset timer = 0
         }
         stroke = nextStroke
    }
THREAD END
```

Figure 4.6: Thread synchronization technique.

4.1.6 Challenges in Linux and Mac OS X Chat Client Development

As discussed in Section 4.1.2, the client application development for Linux and Mac OS X is made simple by the Skype4Py wrapper. Though our framework back-end can be easily integrated with Skype4Py, the application development has the following challenges.

Handling the Audio Data

Skype gives audio output as WAV format data. One might have to write WAV mixing algorithms to be able to combine the microphone input and the incoming Skype audio. As our InkAMP player plays *.REC* files that save audio as an MP3, there is a need for an audio encoder to convert WAV data into MP3. This was done by LAME encoder [45] in InkAndAudio Chat for Windows. Similar encoder needs to be used for a chat client in Linux or Mac OS X.

Handling the Digital Ink

With the introduction of the <*InkCanvas*> element in WPF, digital ink support has become robust for a Tablet PC environment in Windows. The advantages of using <*InkCanvas*> have already been discussed in Section 4.1.2. Linux and Mac OS X, however, have minimal or no such Tablet PC support. This brings a major challenge in the chat client development using our framework in such environments. Even trivial WPF-tasks such as stroke generation and stroke serialization before transfer might have to be simulated based on the X, Y cursor position, generated by the mouse movement. The simulation of numerous stroke events provided by <*InkCanvas*> in WPF will be difficult to achieve.

The Canvas widget provided by Tkinter can be helpful, as it is used to display and edit graphs and other drawings.

Handling the Data Exchange

A major challenge is to provide interoperability across different clients. One has to map the events generated in one application environment to those in the other. Skype API wrappers support exchange of textual data. This makes it necessary to serialize the strokes at the sending end and de-serialize data at the receiving end.

The client application design for Linux and Mac OS X will therefore need a proper understanding of the native application development environment. One should have a sound knowledge about Windows WPF, so as to be able to provide interoperability to InkAndAudio Chat client for Windows and the others.

Chapter 5

System Architecture

Most of the available whiteboard applications used for collaboration are built on their own complex protocols for data exchange [1, 38]. Instead of re-inventing a new protocol, our system design takes the Skype peer-to-peer protocol as the data exchange back-end after a careful analysis on its advantages. The Skype client can run on different mobile devices and on various platforms. The availability of public API to interact with the Skype network makes it a good choice for the implementation. The rest of this chapter focuses on the system architecture of the InkAndAudio Chat client, the InkAMP multimedia player for playing saved chat sessions, the data formats involved and the process of synchronization while playing back a saved session.

5.1 The InkAndAudio Chat Client for Windows

A simple design is the key to a successful implementation. The InkAndAudio Chat client uses the Skype API for data exchange among participants and thus the underlying communication protocol is hidden. Skype has been known for its efficient and reliable data exchange even at a sluggish internet speed. User-application interactions is made simple with graphical interface with features that support stroke erasing, conference setup, background image sharing *etc.* Figure 5.1 shows how two participants communicate with InkAndAudio Chat using our framework. For our implementation, InkAndAudio Chat handles the data exchange using the App2App com-

munication provided by the Skype4COM wrapper. This will be discussed further in Section 5.5 below.

Similar clients can be developed using Skype4Py or Skype4Java for the Linux and Mac OS X environments. Using available wrappers on top of the native C++ Skype API, the strokes exchanged among participants can be handled in multiple environments. This solves the main design issue of interoperability.

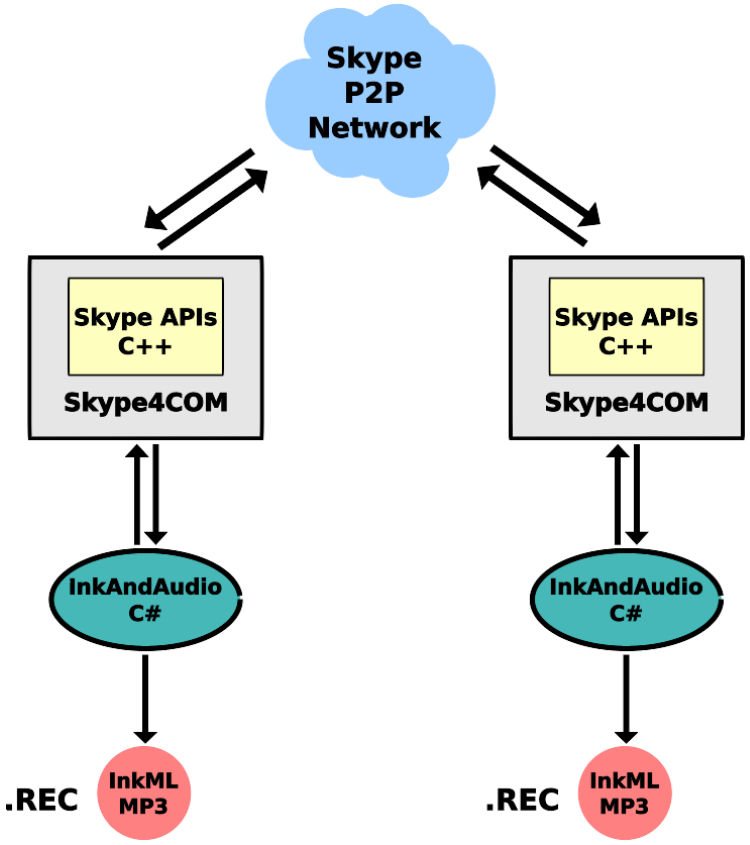


Figure 5.1: The InkAndAudio Chat application.

InkAndAudio Chat is an integration of several applications and external libraries. The major components of InkAndAudio Chat are listed in Sections 5.1.1 to 5.1.3.

5.1.1 The WPF's *<InkCanvas>* Element

In our implementation, an *InkCanvas* element serves as a whiteboard that is shared among chat participants. The WPF's *<InkCanvas>* element has consolidated the digital ink manipulation features available in Microsoft's Tablet PC SDK. The advantages of using *InkCanvas* have been discussed in Section 4.1.2. However, *InkCanvas* is not available for Linux and Mac OS X. This gives rise to a number of challenges in providing digital ink support for Linux and Mac OS X clients. These have been addressed in Section 4.1.6.

5.1.2 The clsWaveProcessor Class

Storage and processing of audio involved in the chat session is handled by the clsWaveProcessor class [8]. A clsWaveProcessor class can handle different WAV audio formats. As shown in Figure 5.2, clsWaveProcessor has been modified to handle 16000 Hz 16 bit mono audio, the standard Skype audio format. As shown in Figure 5.3, the clsWaveProcessor combines the Skype out audio stream coming from the other participants, with the microphone input and produces a single uncompressed WAV file.

```
// Customizing for Skype Stored WAV
private const short CHNL = 1;    // mono = 1, stereo = 2
private const int SMPL_RATE = 16000;// 8000,  16000, 44100 etc
private const int BIT_PER_SMPL = 16;// 8 bits = 8, 16 bits = 16
private const short FILTER_FREQ_LOW = -10000;
private const short FILTER_FREQ_HIGH = 10000;
```

Figure 5.2: Modification to handle 16000 Hz 16 bit mono Skype audio.

Integration of the clsWaveProcessor in InkAndAudio Chat was simplified by the fact that both were C# .NET applications. Similar WAV file processing methods need to be implemented for Linux or Mac OS X-like environment.

5.1.3 LAME (LAME Ain't an Mp3 Encoder)

Several handheld devices and smart phones have the capability to run a Skype client. For such devices with limited storage, data compression becomes crucial. The LAME encoder [45] is used for compression of the uncompressed WAV file produced by the clsWaveProcessor. As shown in Figure 5.3, the result of this compression is the MP3 audio format which is widely supported across platforms. LAME can also be used in other operating systems such as Linux, Mac OS X, Solaris, HP-UX, QNX *etc.*

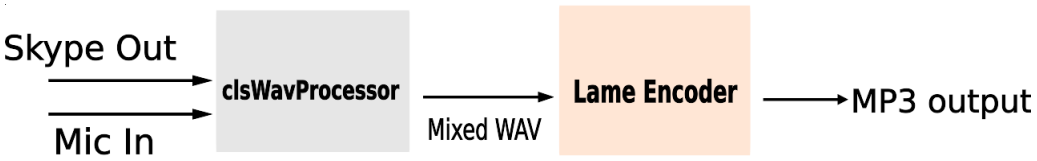
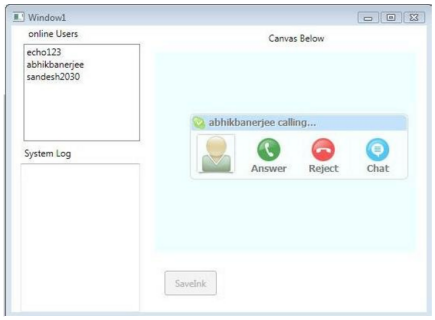


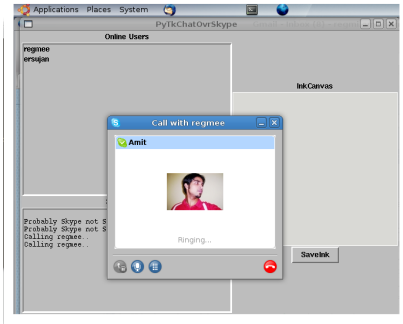
Figure 5.3: Audio compression with LAME.

5.2 PyInkAndAudio Chat Client for Linux and Mac OS X

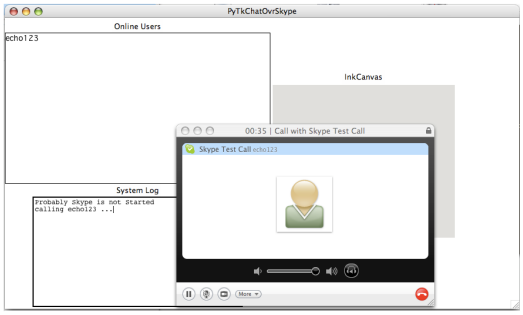
PyInkAndAudio is our chat client for Linux and Mac OS X. It is based on the Skype4Py multi-platform Skype API wrapper for Python [35]. At the time of writing this thesis, PyInkAndAudio, as shown in Figure 5.4 was at a primitive stage in its development. It could support only a few of the Skype interaction capabilities available in InkAndAudio Chat client for Windows, but nevertheless was a suitable proof-of-concept.



InkAndAudio Chat Client (Windows)



PyInkAndAudio Chat Client (Linux)



PyInkAndAudio Chat Client (MacOS)

Figure 5.4: Towards platform independence.

The Skype4Py wrapper works equally well on Windows, Linux and Mac OS X. Therefore, PyInkAndAudio Chat client can run both on Linux and Mac OS X. The Code listing of Figure 5.5 shows how an application makes use of Skype4Py to establish a connection and interact with Skype.

To make the client more robust and more native to Mac OS X environment, the Cocoa programming environment [23] can be used for the application development. Cocoa applications are generally written in the Objective-C language, but with the availability of PyObjC, higher level features of Python such as garbage collection and regular expressions can also be used in native Mac OS X applications.

```
import Skype4Py
# Create Skype instance
skype = Skype4Py.Skype()
# Connect Skype object to Skype client
skype.Attach()
print 'Your full name:', skype.CurrentUser.FullName
print 'Your contacts:'
for user in skype.Friends:
    print ' ', user.FullName
```

Figure 5.5: Python code using Skype4Py to connect to the Skype client.

PyObjC provides a Python/Objective-C bridge and enables interoperability between the two environments. It makes the GUI application development straightforward for developers. The code listing in Figure 5.6 shows how PyObjC can bridge Python and Objective-C together to allow an application interact with Skype.

```
# python
>>> import objc
>>> objc.loadBundle("SkypeAPI", globals(), bundle_path=
         objc.pathForFramework("/Applications/Skype.app/Contents
                                /Frameworks/Skype.framework"))
NSBundle </Users/amit/Library/Frameworks/Skype.framework>
(loaded)
>>> print SkypeAPI
<objective-c class SkypeAPI at 0xd0004140>
>>> SkypeAPI.isSkypeRunning()
1
```

Figure 5.6: Using PyObjC bridge to interact with Skype API on a Mac OS X.

5.3 Chat Session Archival

Recording digital media for future reference has always been an interesting topic for research. Whiteboards that use audio and digital ink have been used in education and for collaborative discussions, but no common technology for archival of such sessions exists. To make such archives available for applications in different platforms, we need to have a platform independent data format.

InkML gives us a platform-independent representation for digital ink archival. To store audio and ink together, we use a composite data format. As shown in the Figure 5.7, *.REC* file for a chat session contains the corresponding InkML file and an MP3 audio. The two files are zipped into a single *.REC* archive that may be used for playback by our InkAMP player.

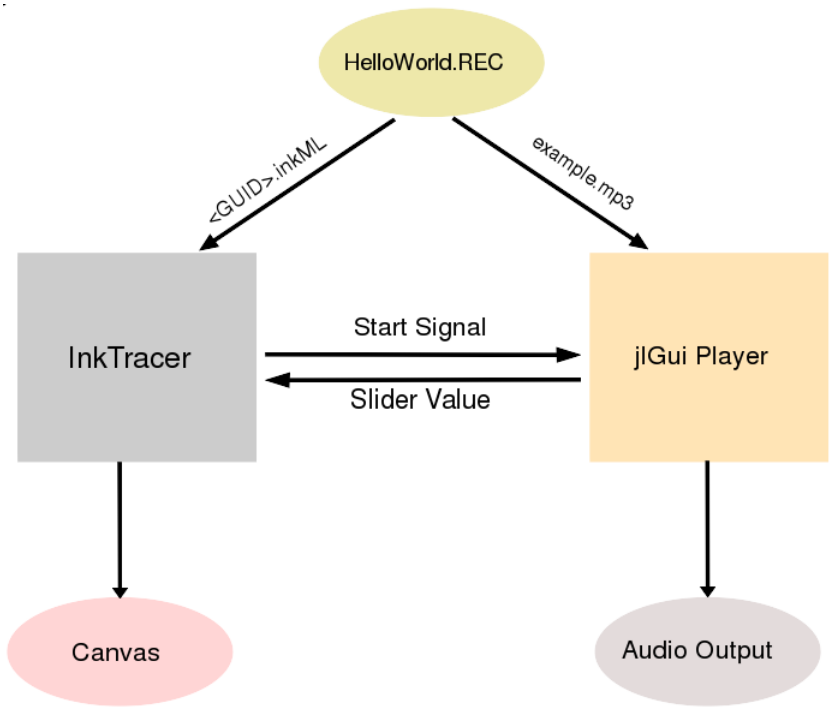


Figure 5.7: InkAMP data.

5.4 InkAMP : The Ink and Audio Media Player

InkAMP is a multi-threaded Java application whose thread model is depicted by Figure 5.8. It is an integration of two Java applications, InkTracer and the jlGui music player. We have included a brief discussion of these InkAMP components in Section 2.4 and Section 2.5. The modified version of InkTracer handles InkML files, based on the new InkML specification from W3C.

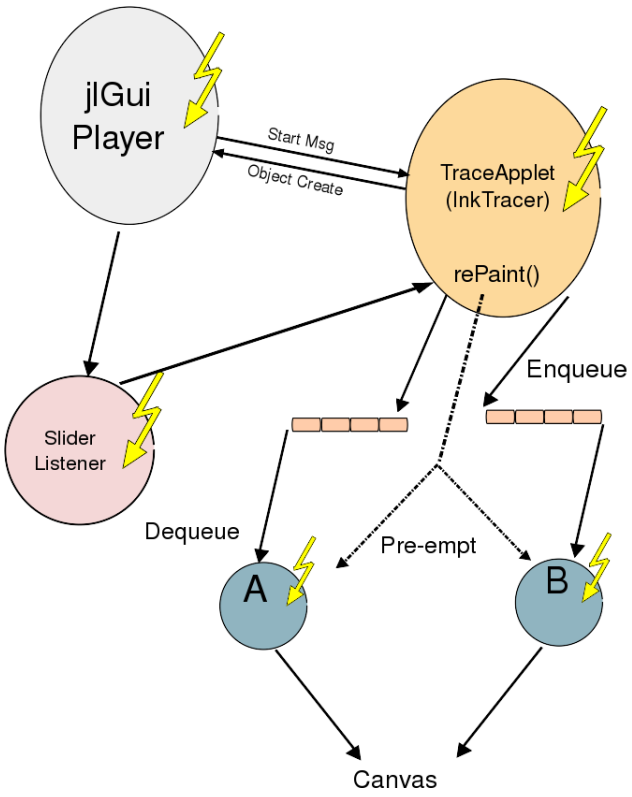


Figure 5.8: The multi-threaded model of InkAMP.

InkAMP can run in any platform that supports Java and operates on the following logic:

- It determines the number of participants *N*, depending on the number of different GUIDs stored in the InkML file.
- Creates *N* painter threads that paint on a common canvas.
- Each <*traceGroup*> in the InkML file is associated with one participant.
- Long traces would already have been broken as *"continuation"* traces by the chat application and stored accordingly.
- One thread reads the InkML and then populates the *N* queues with the respective traces coming from *N* participants.
- *N* painter threads read traces from respective queues and start painting on a common canvas.

5.5 Communication using the Skype Peer-to-Peer Protocol

5.5.1 The Need for Data Serialization

The primary advantage of serialization of objects in the .NET programming environment is to convert objects into a format that can be transmitted over the network. Objects can be serialized into two formats :

1. XML format
2. Binary format

Binary serialization gives a greater performance, whereas, XML serialization gives human readable data with greater flexibility. We have chosen binary serialization over XML serialization, since our aim was to make the stroke transmission faster and efficient.

Stroke collection and transmission across participating nodes in multiple platforms is a challenging task. Replicating the WPF *InkCanvas* behavior is a major hurdle in the development of an interoperable chat client for Linux or Mac OS X. Serialization of the collected strokes has to be handled differently for each environment.

5.5.2 Establishing Connection

We need to establish a communication channel before any two participants can start a chat session with InkAndAudio Chat. This process involves a sequence of message exchanges between the selected candidates. Whiteboard sharing can begin through the established communication streams once all the participants are ready.

Figure 5.10 shows how a communication channel is established when 2 participants interact, using InkAndAudio Chat. When a user selects an online user from the list, InkAndAudio Chat client initiates the communication with CHAT_INVITE message. It establishes a secure communication channel once it gets an acknowledgment from the other party *i.e.* the CHAT_ACK message. Once the stream for reading and writing has been set up, `stream.Write()` API is

used for stroke transfer with a CHAT_TRANSFER message. Figure 5.9 illustrates the message types used during a connection setup.

```
public class ApplicationMessage
{
    public enum MessageType
    {
       UNKNOWN,
       CHAT_INVITE,
       CHAT_ACK,
       CHAT_DECLINE,
       CHAT_TRANSFER,
       CHAT_TERMINATE,
       FILE_INVITE,
       FILE_ACK,
       FILE_DECLINE,
       FILE_TRANSFER,
       FILE_TERMINATE,
    }
}
```

Figure 5.9: Chat and File transfer messages.

Establishing connections in a conference call with InkAndAudio Chat is handled in a different way. A conference call is initiated by a host which maintains a one-to-one connection with other participants. The data exchange initiated by an ordinary participant is routed through the host. A conference call with InkAndAudio Chat is covered in Section 5.5.4 below.

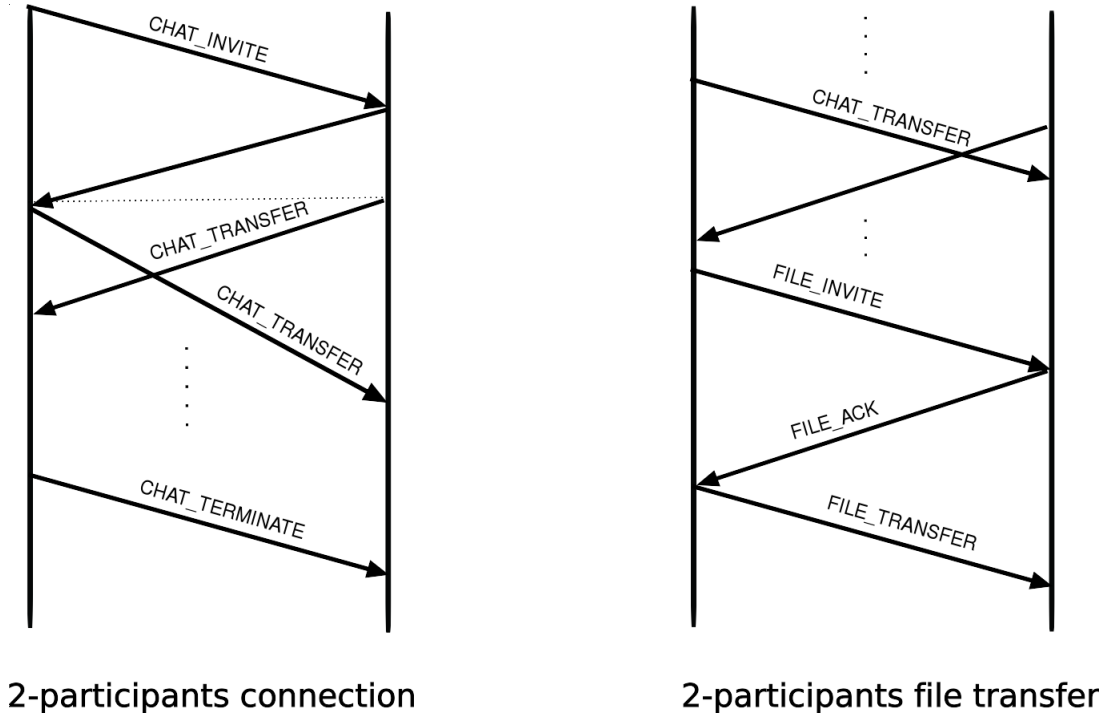


Figure 5.10: An interaction involving 2 participants.

5.5.3 Sharing Canvas Background

To give the users more ways to collaborate, background images can be shared and discussions can be followed with colored annotations. A canvas screenshot can be saved as an image file such as BMP, PNG, GIF *etc.* along with an InkML file. Discussion on a shared map is a typical scenario that can be handled using InkAndAudio Chat. Figure 5.10 shows the sequence of message exchange for a file transfer between two participants.

5.5.4 Conferencing with InkAndAudio

Multiple users can share the whiteboard using InkAndAudio conference. The conference is initiated by the host which has a connection with every other participant. Figure 5.11 shows the algorithm used by the host to setup a conference call. Once our host establishes the necessary communication streams, a collaboration can begin in the form of stroke exchange, stroke dele-

tion, stroke highlighting, canvas background sharing, voice chat *etc.*

```
BEGIN:
   if (participants do not have inProgress calls)
   {
       Call one participants from the list.
       Join all the remaining participants to the call.
   }
   else if (participants have inProgress calls)
   {
       Put existing calls OnHold.
       Wait until they all actually are on hold.
       Make a new call.
       Wait until that call actually starts.
       Search for OnHold calls and join them to the new call.
   }
END:
```

Figure 5.11: Algorithm to start a conference call, from [34].

Figure 5.12 shows a 3-user conference using InkAndAudio Chat. Node *A* is the host, while *B* and *C* are ordinary participants. Nodes *B* and *C* directly communicate with the host node *A*. However, a data exchange initiated by *B* or *C* has to go through the host node *A*.

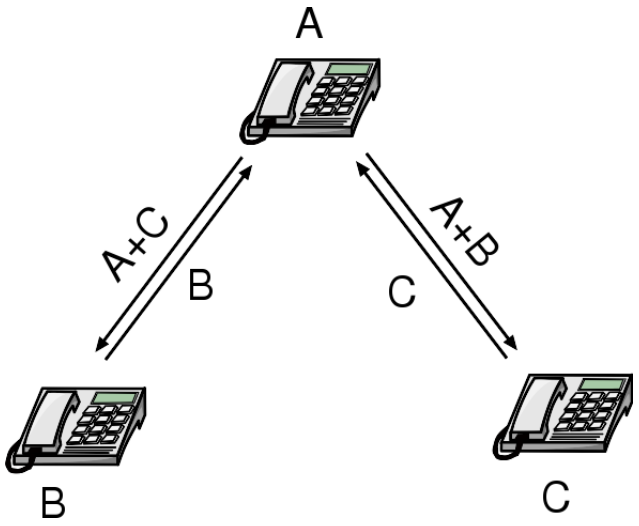


Figure 5.12: Skype 3 user conference, from [4].

5.5.5 Data Exchange in a Conference

Figure 5.13 shows the data exchange model for a conference call. Relay of data takes place through the host node. Audio routing to participating nodes is handled by the Skype client internally, and is written to the corresponding WAV files using the Skype API.

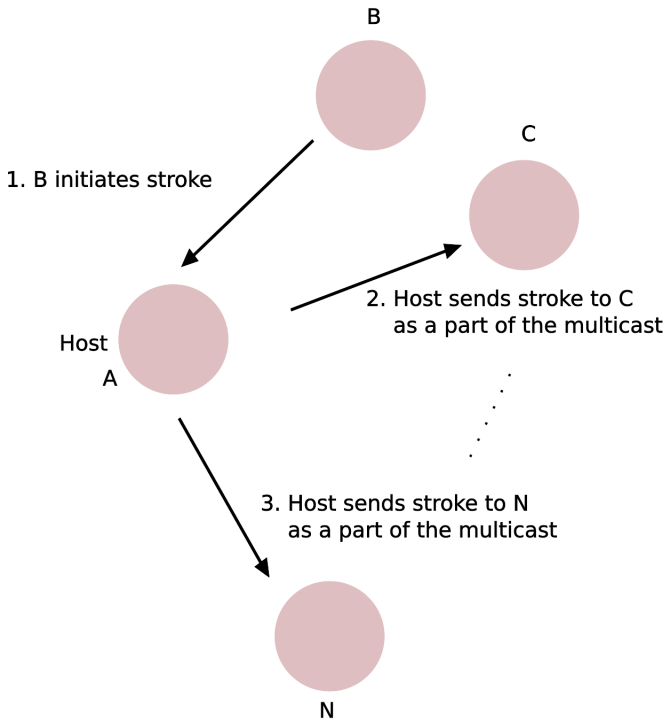


Figure 5.13: Skype multi-user conference and data multicast.

Our implementation follows the same data exchange model for the following activities:

- Stroke exchange
- Stroke deletion
- *InkCanvas* background sharing

Chapter 6

Conclusion and Future Directions

Multimodal applications reach a broad spectrum of the population and can be useful in various challenging conditions. Since multimodal systems support more natural interactions without special training, they have the potential to make computation available to researchers, students, business people, temporarily or permanently disabled, computer-illiterate and many others. The fact that multimodal applications reach out to a wider audience makes them a potentially good tool for collaboration. In today's diverse working environment, multimodal applications that integrate audio with other modes may become increasingly useful.

In this thesis, we have discussed a few examples of multimodal applications used for collaboration. We have discussed the issues that arise while designing applications involving ink and audio collaboration. We have studied the design and implementation of InkAndAudio Chat, which is used for collaboration and supports speech and ink input modes. InkAndAudio is not just a practically helpful multimodal tool, but also an example that shows how InkML is well suited as a means to represent digital ink for interactive collaboration. Also, PyInkAndAudio was introduced as a chat client for Linux and Mac OS X as a proof-of-concept to show cross-platform support provided by our multimodal application framework.

There are a number of interesting possible future directions for this work. An increasing number of modes of input gives a user the flexibility to choose the most convenient input mode for communication. Based on the availability of video capture API from Skype, it would be in-

teresting to include video input mode in the higher versions of the application. Then the question would arise naturally, as to how to organize ink annotation of video and still image capture.

The PyInkAndAudio Chat client can be developed further to make the multimodal application reach a larger user-base. It still lacks the WPF-*InkCanvas* functionality found in the InkAndAudio Chat application for Windows. Such functionality would have to be simulated for a better ink exchange support.

Our system can also be extended and integrated with external computation servers in a manner similar to what Electronic Chalkboard does with the Maple computer algebra system and the Mathematica system as explained in Section 2.1.7. Such capabilities would give users a good learning experience with spontaneous examples for clarification.

Chapter 7

User Manual

This chapter provides a user manual for the chat applications.

7.1 Project Homepage

A detailed description of the project is available at:

`http://www.orcca.on.ca/software/ink+audio`

7.2 The InkAndAudio Chat

Installation of the Skype add-on is straightforward. We need to download `InkAndAudio.spark` file from the project's homepage and install it after starting the Skype application.

7.2.1 System Requirements

- Windows (`32-bit`) with Skype installation.
- registered `Skype4COM.dll`

  Skype4Com.dll is automatically included together with the Skype Extras Manager during Skype installation. If a user has unchecked Extras Manager during installation, Skype4Com library will be unavailable on that machine.

To register the DLL, as an administrator run

`regsvr32 Skype4COM.dll`

- .NET framework 3.5+

7.2.2 Using InkAndAudio Chat

- Start the Skype application
- Run the installed Skype-extra from `Tools->DoMore->InkChat`
- Select `users`
- Sketch and talk as desired
- Select "`SaveInk`" when done (save it as `example.rec`)

7.3 The PyInkAndAudio Chat

The PyInkAndAudio Chat application's source can be obtained from the project's homepage and run from the command line. PyInkAndAudio Chat has been tested with Linux and Mac OS X operating systems with the following configurations:

- `Skype (Beta) Version 2.0.0.13` for Linux and `Skype Version 2.7.0.330` for Mac OS X
- `Ubuntu 7.10 (Gutsy Gibbon)` and `Mac OS X 10.4 (Tiger)`
- `Python 2.5.1`
- `python-dbus`
  (Python interface to the D-BUS interprocess messaging system)

- `python-pmw`

  (Package that provides Python MegaWidgets or the Pmw modules for building high-level compound widgets in Python using the Tkinter interface to the Tk graphics library)

- `Skype4Py 1.0.31.0`

7.4 The InkAMP Player

The InkAMP player needs `JRE 1.5+` environment. The application can be downloaded from the project's homepage and `InkAMP.jar` file can be run to play the saved sessions.

7.4.1 Using InkAMP Player

The InkAMP player may be used to playback recorded ink and audio files. It may be installed as follows:

- Download `InkAMP.zip` from the project home page
- Run `InkAMP.jar` from the `InkAMP\jlGui3.0` folder after unzip
- "`open file`" and locate the stored `example.rec`

Bibliography

[1] Manoj Prasad A, Muthuselvam Selvaraj, and Sriganesh Madhvanath. Peer-to-peer ink messaging across heterogeneous devices and platforms. In *Compute '08: Proceedings of the 1st Bangalore annual Compute conference*, pages 1–5, New York, NY, USA, 2008. ACM.

[2] Gregory D. Abowd, Jason Brotherton, and Janak Bhalodia. Classroom 2000: a system for capturing and accessing multimedia classroom experiences. In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, pages 20–21, New York, NY, USA, 1998. ACM.

[3] Richard J. Anderson, Crystal Hoyer, Steven A. Wolfman, and Ruth Anderson. A study of digital ink in lecture presentation. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 567–574, New York, NY, USA, 2004. ACM.

[4] S. A. Baset and H. G. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, April 2006.

[5] Tom Berson. Skype Security Evaluation. *Tech. Rep., ALR-2005-031, Anagram Laboratories*, 2005.

[6] Yi-Min Chee, Max Froumentin, and Stephen M. Watt (editors). Ink Markup Language (InkML), October 2006. `http://www.w3.org/TR/InkML/` (valid on January 18, 2009).

[7] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Dis-

tributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, Volume - 2009, Year - 2001.

[8] CodeProject. Free source code and programming help. `http://www.codeproject.com/KB/cs/WAVE_Processor_In_CSharp.aspx` (valid on January 18, 2009).

[9] Philip Cohen, David McGee, and Josh Clow. The efficiency of multimodal interaction for a map-based task. In *Proceedings of the sixth conference on Applied natural language processing*, pages 331–338, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[10] Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen, and Josh Clow. QuickSet: multimodal interaction for distributed applications. In *MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia*, pages 31–40, New York, NY, USA, 1997. ACM.

[11] Microsoft Corporation. Ink Serialized Format (ISF) Specification, 2007. `http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/InkSerializedFormat(ISF)Specification.pdf` (valid on January 18, 2009) .

[12] Slate Corporation. JOT - A specification for an Ink Storage and Interchange Format. `http://unipen.nici.kun.nl/jot.html` (valid on January 18, 2009).

[13] Richard C. Davis, James A. Landay, Victor Chen, Jonathan Huang, Rebecca B. Lee, Frances C. Li, James Lin, III Charles B. Morrey, Ben Schleimer, Morgan N. Price, and Bill N. Schilit. NotePals: lightweight note sharing by the group, for the group. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 338–345, New York, NY, USA, 1999. ACM.

[14] Simon Dobrisek, Jerneja Gros, Bostjan Vesnicer, France Mihelic, and Nikola Pavesic. A

Voice-Driven Web Browser for Blind People. In *TSD '02: Proceedings of the 5th International Conference on Text, Speech and Dialogue*, pages 453–460, London, UK, 2002. Springer-Verlag.

[15] Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim Mccall, Elin Pedersen, Ken Pier, John Tang, and Brent Welch. LiveBoard: a large interactive display supporting group meetings, presentations, and remote collaboration. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 599–607, New York, NY, USA, 1992. ACM Press.

[16] Jon Ferraiolo, Fujisawa Jun, and Dean Jackson (editors). Scalable Vector Graphics (SVG) 1.1 Specification, 2003. `http://www.w3.org/TR/SVG` (valid on January 18, 2009).

[17] Gerald Friedland, Lars Knipping, Raúl Rojas, and Ernesto Tapia. Teaching with an intelligent electronic chalkboard. In *ETP '04: Proceedings of the 2004 ACM SIGMM workshop on Effective telepresence*, pages 16–23, New York, NY, USA, 2004. ACM.

[18] Simson L. Garfinkel. VoIP and Skype Security, 2005. `http://www.tacticaltech.org/files/tacticaltech/Skype_Security.pdf` (valid on January 18, 2009).

[19] The W3C Multimodal Interaction Working Group. `http://www.w3.org/2002/mmi/` (valid on January 18, 2009).

[20] Saikat Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. *5th Intl. Workshop on Peer-to-Peer Systems, Santa Barbara, CA.*, Feb. 2006.

[21] Isabella Guyon. Unipen 1.0 Format Definition. The Unipen Consortium, 1994. `http://www.unipen.org/dataformats.html` (valid on January 18, 2009).

[22] IBM. e-Brochure: IBM Multimodal Toolkit. `http://www-01.ibm.com/software/pervasive/demos/multimodal/mm_toolkit_flash_demo.html` (valid on January 18, 2009).

[23] Apple Inc. Cocoa: The Objective-C Programming Language. `http://developer.`

apple.com/ (valid on January 18, 2009).

[24] JavaZOOM. jlGui : Music Player for the Java Platform, 1999. http://www.javazoom.net/jlgui/jlgui.html (valid on January 18, 2009).

[25] Peter Kalmstrom. Skype Development Techniques and Tools An Overview. *eBay Developers Program 2007*.

[26] Matthew Kam, Jingtao Wang, Alastair Iles, Eric Tse, Jane Chiu, Daniel Glaser, Orna Tarshish, and John Canny. Livenotes: a system for cooperative and augmented note-taking in lectures. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 531–540, New York, NY, USA, 2005. ACM.

[27] KaZaA. http://www.kazaa.com (valid on January 18, 2009).

[28] B. Keshari and S. Watt. Streaming-Archival InkML Conversion. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, pages 1253–1257, Washington, DC, USA, 2007. IEEE Computer Society.

[29] Kazutaka Kurihara, Masataka Goto, Jun Ogata, and Takeo Igarashi. Speech pen: predictive handwriting based on ambient multimodal recognition. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 851–860, New York, NY, USA, 2006. ACM.

[30] HP Labs. Sharing Digital Ink in Heterogeneous Collaborative Environments. http://www.hpl.hp.com/techreports/2008/HPL-2008-54.html (valid on January 18, 2009).

[31] Michael G. Lamming and William M. Newman. Activity-based Information Retrieval: Technology in Support of Personal Memory. In *Proceedings of the IFIP 12th World Computer Congress on Personal Computers and Intelligent Systems - Information Processing '92 - Volume 3*, pages 68–81, Amsterdam, The Netherlands, The Netherlands, 1992. North-Holland Publishing Co.

[32] A. P. Lenaghan and R. R. Malyan. XPEN: An XML Based Format for Distributed Online Handwriting Recognition. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 1270, Washington, DC, USA, 2003. IEEE Computer Society.

[33] Yan Li. IBM White Paper on Multimodal Application Design Issues, December 2003. `http://www.ibm.com/developerworks/websphere/library/techarticles/0312_li/0312_li.html` (valid on January 18, 2009).

[34] Skype Limited. Skype Developer Zone. `https://developer.skype.com/Docs/Skype4COM` (valid on January 18, 2009).

[35] Skype Limited. Skype Developer Zone. `https://developer.skype.com/wiki/Skype4Py` (valid on January 18, 2009).

[36] Thomas P. Moran, Leysia Palen, Steve Harrison, Patrick Chiu, Don Kimber, Scott Minneman, William van Melle, and Polle Zellweger. "I'll get that off the audio": a case study of salvaging multimedia meeting records. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 202–209, New York, NY, USA, 1997. ACM.

[37] Jonathan Neddenriep and William G. Griswold. RiverInk–An Extensible Framework for Multimodal Interoperable Ink. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 258b, Washington, DC, USA, 2007. IEEE Computer Society.

[38] Hai Ning, John R. Williams, Alexander H. Slocum, and Abel Sanchez. InkBoard - Tablet PC Enabled Design oriented Learning. In *CATE*, pages 154–160, 2004.

[39] Groove Home Page Microsoft Office Online. `http://office.microsoft.com/groove` (valid on January 18, 2009).

[40] S. Oviatt, P. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Wino-

grad, J. Landay, J. Larson, and D. Ferro. Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions for 2000 and beyond. 2000.

[41] Sharon Oviatt. Ten myths of multimodal interaction. *Commun. ACM*, 42(11):74–81, 1999.

[42] Sharon Oviatt and Erik Olsen. Integration themes in multimodal human-computer interaction. pages 551–554, 1994.

[43] Elin Ronby Pedersen, Kim McCall, Thomas P. Moran, and Frank G. Halasz. Tivoli: an electronic whiteboard for informal workgroup meetings. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 391–398, New York, NY, USA, 1993. ACM.

[44] Skype Affiliate Program. `http://Skype.com/business/partners/affiliate/` (valid on January 18, 2009) .

[45] The LAME Project. LAME Ain't an Mp3 Encoder. `http://lame.sourceforge.net/` (valid on January 18, 2009).

[46] Python.org. Tkinter - Python interface to Tcl/Tk. `http://docs.python.org/library/tkinter.html` (valid on January 18, 2009).

[47] Wolfram Research. A Practical Introduction to Mathematica. `http://documents.wolfram.com/mathematica/` (valid on January 18, 2009).

[48] Ray Rischpater. Creating Multimodal Applications Using the IBM Multimodal Toolkit, 2005. `http://www.devx.com/wireless/Article/28367` (valid on January 18, 2009).

[49] J. Rosenberg, C. Huitema, and R. Mahy. TURN: traversal using relay NAT. *Internet Engineering Task Force.*, Jul. 2004.

[50] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN: Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). *RFC 3489,*

IETF., Mar. 2003.

[51] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Commun. ACM*, 30(1):32–47, 1987.

[52] Lisa Stifelman, Barry Arons, and Chris Schmandt. The audio notebook: paper and pen interaction with structured speech. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 182–189, New York, NY, USA, 2001. ACM.

[53] Microsoft Mobile Platforms Division Team. The Evolution of Tablet PC Technologies in Microsoft Windows Vista, 2005. `http://download.microsoft.com/download/3/9/b/39b003b3-64e8-41ac-9b4a-08bbbaef888f/2005_WhitePaper_6B.pdf`(valid on January 18, 2009).

[54] Rapyd-Tk Team. An Open-source application development environment for Python/Tkinter. `http://www.bitflipper.ca/rapyd/` (valid on January 18, 2009).

[55] S. Watt. New Aspects of InkML for Pen-Based Computing. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 1*, pages 457–460, Washington, DC, USA, 2007. IEEE Computer Society.

[56] Steve Whittaker, Patrick Hyland, and Myrtle Wiley. Filochat: handwritten notes provide access to recorded conversations. In *CHI '94: Conference companion on Human factors in computing systems*, page 219, New York, NY, USA, 1994. ACM.

[57] Lynn D. Wilcox, Bill N. Schilit, and Nitin Sawhney. Dynomite: a dynamically organized ink and audio notebook. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 186–193, New York, NY, USA, 1997. ACM.

VITA

NAME: Amit Regmi

EDUCATION:

- **University of Western Ontario**

  London, Canada
  September 2007 to January 2009
  M.Sc. (Computer Science)
  Supervisor: Dr. Stephen M. Watt

- **National Institute of Technology**

  Durgapur, India
  October 2001 to July 2005
  B.E. (Computer Science and Engineering)

- **Kathmandu University**

  Dhulikhel, Nepal
  1998 to 2000
  Intermediate of Science

WORK EXPERIENCE:

- **Software Engineer**

  Texas Instruments India Pvt. Ltd.
  Bangalore, India
  March 2006 to August 2007

- **Teaching and Research Assistant**

  University of Western Ontario
  Dept. of Computer Science
  September 2007 to December 2008