Enabling a Voice Modality in Mobile Games through VoiceXML

Michael J. Zyda
GamePipe Labs
University of Southern California
zyda@usc.edu

Dhruv Thukral
GamePipe Labs
University of Southern California
thukral@usc.edu

James C. Ferrans
Motorola Labs
Schaumburg, IL
james.ferrans@motorola.com

Jonathan Engelsma
Motorola Labs
Schaumburg, IL
jonathan.engelsma@motorola.com

Mat Hans
Motorola Labs
Schaumburg, IL
mat.hans@motorola.com

Abstract

The use of speech recognition in gaming applications is not entirely new. The growth of voice as a part of gaming has exploded largely due to the popularity of online player matchmaking services such as Xbox Live. Yet, the majority of its use is only limited to communications between players to coordinate game play activities through the use of a headset and a microphone.

To further explore the possibilities of using speech recognition to affect game play directly, Motorola has partnered with GamePipe Labs at the University of Southern California. This collaboration aims at leveraging the capabilities of VoiceXML (VXML), and use interactive voice dialogues to directly affect game play on mobile phones. This short paper describes the efforts taken under this initiative, and the results of this collaboration.

Author Keywords:

Mobile gaming, voice recognition, VoiceXML, multi-modal interaction, speech recognition, GamePipe Labs, Motorola Labs, multiplayer multimodal mobile games.

ACM Classification Keywords

H.5.2. [Information interfaces and presentation]: Multimedia Information Systems--*Audio input/output*; H5.2. User Interfaces---*interaction techniques, input devices and strategies, Screen design* H.5.3. Group and Organization Interfaces---*collaborative computing;* J.5 [Arts and Humanities] *Fine Arts, Performing arts (e.g., dance, music*); K.3.2 [Computer and Information Science Education]: computer science education, mobile games curriculum.

I. Introduction

Over the past few years traditional forms of input in PC and console gaming have evolved dramatically. The recent advent of the Wii Remote is just one of the many changes we have witnessed when it comes to novel ways of interacting with games. Before the Wii Remote came into the picture, legacy input devices such as the keyboard, mouse, game controllers and joysticks, have also had to contend with another addition; the headset and the microphone. Voice is increasingly becoming the preferred way of hardcore gamers to give them the edge in a tight situation where complex keyboard shortcuts can be replaced by simple to remember voice commands.

Such is the force of this new demand that every Xbox Live Starter Kit comes with a headset and microphone as standard. Xbox Live is an online multiplayer gaming service created and operated by Microsoft Corporation, and has the highest subscription base of any console based online service. The inclusion of the headset and microphone to the kit is meant to cater only to online multiplayer games, where teams of players use voice to coordinate game related activities by talking to each other instead of having to tediously type inside chat windows.

This trend however, has led to interesting experimentation in trying to incorporate voice during game play and introducing it into various aspects of gaming not limited to the above usage scenario. As mentioned already, this is mostly limited to PC based gaming right now. Games such as Unreal Tournament and SOCOM3. U.S. Navy Seals use speech recognition to affect the in game behavior of Non Playable Characters or NPC's. For instance, every SOCOM3 game package includes a Logitech Headset that allows for real time in-game voice communication in single player mode. And it does so quite effectively by allowing players to completely side step layers of command menus and button presses to control their AI teammates. Based on professional game reviews this piece of voice technology is considered the best and most user friendly mode of interaction in the entire game.

One of the most recent advocates of voice in gaming is Nintendo, with the introduction of the voice component in their popular Nintendo DS portable game console. The most encouraging development that came out of this is the fact that two of Nintendo's most successful games for the above platform, BrainAge and Nintendogs, use simple voice commands as an inherent part of their game play.

To expand on the use of voice in gaming, Motorola Labs has formed a partnership with GamePipe Labs at the University of Southern California to provide a platform that will enable the use of voice modality in mobile games. This platform was built on top of Motorola Linux framework for software development to develop applications on their mobile phones.

The introduction of voice modality in mobile games is of a genuine interest to determine how one overcomes the bottleneck created by the tiny input interfaces on mobile devices. It makes a lot of sense to leverage the existing audio interfaces of these phones and give game players a choice of both visual and voice modalities while playing mobile games. This can be an attractive alternate to the tiny keypad that is the primary form of interaction in almost all of today's mobile games.

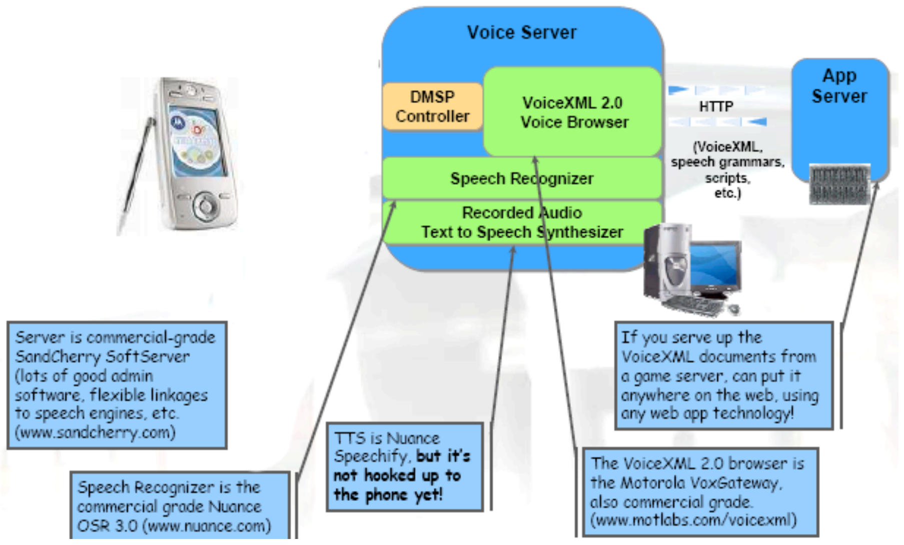
Voice Server
DMSP Controller
VoiceXML 2.0 Voice Browser
Speech Recognizer
Recorded Audio
Text to Speech Synthesizer
HTTP
(VoiceXML, speech grammars, scripts, etc.)
App Server
Server is commercial-grade SandCherry SoftServer (lots of good admin software, flexible linkages to speech engines, etc. (www.sandcherry.com)
Speech Recognizer is the commercial grade Nuance OSR 3.0 (www.nuance.com)
TTS is Nuance Speechify, but it's not hooked up to the phone yet!
The VoiceXML 2.0 browser is the Motorola VoxGateway, also commercial grade. (www.motlabs.com/voicexml)
If you serve up the VoiceXML documents from a game server, can put it anywhere on the web, using any web app technology!

Figure 1: Multimodal Server Architecture

II. The Technology

Our multimodal, multiplayer, mobile games are built on top of a proprietary Motorola Linux platform provided to us by Motorola Corporation. This platform is Motorola's framework for software development to develop applications on their mobile devices.

The Motorola Linux platform is built upon a Linux Kernel 2.4 core. The framework comes with a range of APIs for standard telephony applications such as communications and multimedia.

This framework was modified with the aid of Motorola for the specific needs of this class. The framework comes with Qt, a C++ based Graphical User Interface (GUI) toolkit that has been used to build complex interfaces such as the KDE desktop environment on Linux. Although not originally intended for use in game programming, the powerful ability of Qt to build widgets and other GUI based elements has led to a branch of interested developers to use Qt for game development. The Qt API consists of various useful graphics related classes that, when used effectively, can be used to build 2D games. However Qt has no support for perspective 3D with the version that came pre-deployed on the Motorola SDK.

For games that require networked capabilities, the Motorola SDK has support for the BlueZ Bluetooth stack, which is licensed under the GNU General Public License (GPL). For 3D and multimedia support, The Motorola Linux platform provides the use of the Simple Direct Media Layer (SDL) API. SDL is a multimedia library written in C that abstracts over various graphics platforms and is widely used for many Linux based games. SDL along with SDL Mixer, SDL's audio component, were ported by Motorola and us to their platform to allow the option of 3D development which was not available on QT.

The speech and voice recognition backbone of all our games is the Qt + VoiceXML framework which consists of a small Qt/C++ based client (or mobile phone) that communicates with a VoiceXML 2.0 Server (Figure 1). The VoiceXML 2.0 server is a commercial grade Multimodal voice server. The server consists of a speech recognizer in the form of Nuance's OSR 3.0 and Motorola's VoxGateway, a VoiceXML 2.0/2.1 interpreter.

The client side consists of a Qt/C++ based framework modified to incorporate voice and hence named the Qt + V framework. This provides us the ability to add voice functionality to any Qt based application. Each voice enabled Qt application or game communicates with the voice server through a WiFi link. The phones were modified to provide support for standard SDIO Wi-Fi cards which would then connect to the same subnet that the voice server is connected to. The application would read in a Qt+V properties configuration file which is used to pass a configuration object to the Qt+V application. A typical file would look like the following. For readability purposes, we are showing the configuration file as an image.

```
## The voice server's IP address
and DMSP port number.
serverIp=192.168.1.102
serverPort=6060

## The starting VoiceXML
document and dialog.
starturl=http://localhost:9292/d
efault.vxml
startdialog=hello

## Log level (see Log.h --
"debug" prints the most details)
loglevel=info

## Silence padding at end of
each audio utterance in
milliseconds
## (So speech recognizer knows
it's done.)
silencepadding=3000
```

Figure 2: A sample VoiceXML configuration file.

The above would then enable the client to create a session to connect to the voice server. Once the session is created it would allow the application to load VoiceXML documents and also create various dialogs for the documents loaded. Each game had a single session with the voice server with a maximum of four parallel sessions allowed on the voice server itself due to licensing restrictions.

III. The Games

The games created had to take into consideration the unique position of mobile games in the commercial market. Mobile games tend to be a spare time medium and require a lot of design considerations such as a small learning curve and short play times. Adding the element of voice creates more challenges when you factor in voice recognition accuracy and latency from the voice server which can affect game play. In addition to testing the technical maturity of the platform, we also tried to analyze the results of acceptance when using voice for playing games, and whether they were a value addition or a deterrent to the medium.

The results of our months of efforts led to the creation of the following three games.

GunPowder

Gunpowder is a Wild West Duel and Shoot em up Game. The game includes two modes: Multiplayer mode and Single Player mode. Multiplayer mode is a duel which takes place between two phones over a Wi-Fi network. Single player is a traditional wild west shooting game between the player and simulated Game AI.

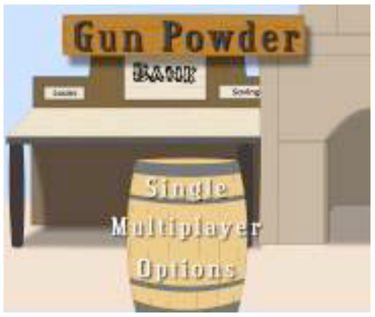

Figure 3: Gunpowder main screen

For the purpose of relevance we will discuss the multiplayer version in which two players play this voice enabled game against each other. The game consists of the following steps:

1. Players are notified that the game countdown is about to begin.
2. "1" appears on the screen. Players must say "1" in to the microphone. When the game detects that the number has been said, it can proceed to the next step.
3. "2" appears on the screen. Players must say "2" in to the microphone.
4. "3" appears on the screen. Players must say "3" in to the microphone.
5. After the word "3" is detected, the player turns around, and sees the other player on the phone screen.
6. Players must tap the screen to shoot the opponent.
7. After a shot has been made on the opponent's body, bullets flies in slow motion (Matrix style) while Wi-fi synchronization is completed. The games then waits for an OK signal that both games have ended.
8. Time is compared for both games to determine the winner. Once a winner is decided, words "You Lose!", "You Win!" or "Draw!" appear on the screen.

This game simulates the countdown style of old Wild West shootouts and makes it an inherent part of the game play. Therefore once voice is factored in the, the game provides a really engaging and challenging medium because the player who speaks more clearly gets past the 3-2-1 countdown faster and had more time to shoot the opponent. Hence the game becomes a competition of accuracy and speed which was found challenging by many players who played this game.

Bejeweled

Figure 4: Bejeweled game screen

Bejeweled is a classic puzzle game and one of the most popular games on today's mobile phones. For the unfamiliar, the purpose this game is to swap and line up jewels of the same color to make combinations of three or more in row or column. The jewels will disappear and the player will earn points according to the jewels destroyed, the combos made, and the level of difficulty. The player will advance to the next level when an in-game progress bar is full and the player will lose if there is no time left or no moves can be made.

Bejeweled also has a feature to provide the players hints on their next move. For the purpose of our demonstration, we decided to highlight the use of voice input to activate in game hints and cheats. The game play element that made this decision interesting was the fact that requesting a hint usually took around three seconds to get a feedback from the voice server, and in a fierce multiplayer competition three seconds can make a huge difference, thus making the player think twice before requesting a hint.

The game also added a basic speech to text engine in which players would use the voice functionality to record and hurl comments/taunts at each other while playing the game, which added another interesting social aspect of using voice in games. Many online multiplayer puzzle games provide the above mentioned feature in the form of chat windows or predefined templates, but none of them have ever used voice to enable this function.

Deja-Hue

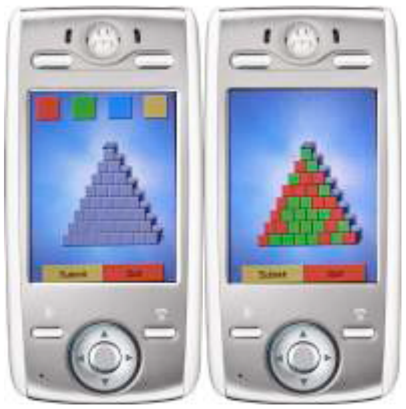

Figure 5: Deja-Hue multiplayer game screen

Deja Hue is memory based puzzle game in which players are given a pyramid of colored blocks such as the one shown above.

The purpose is to match the pyramid blocks with the exact same colors. The player with the most accurate representation of the original colored pyramid wins the game.

The role of voice was to choose the various colors from the pyramid and try to match the original pyramid representation. It was an interesting experiment to note the role of voice in memory games, and to see a player remembers more when he/she says out the pattern loud or when he/she is just using the stylus. There was no conclusive evidence because the players had an equal win/loss ratio whether they were using voice or not.

IV. The Challenges

Despite the success of having created three voice enabled multiplayer mobile games, there were many obstacles in reaching to that point.

Firstly our current licensing restrictions only allowed up to four parallel sessions with the voice server and we could not scale the games to include a vast multiplayer base. This of course, would not be an issue if we were to use voice applications in a more commercial environment.

The applications we created were also very sensitive to background noise and therefore in game music or sound effects would effectively come in the way of proper voice recognition. Therefore all the games we created had to do away with voice effects or background music to make way for more accurate recognition.

Finally, limitations in the audio module were inherited from another project and integrated into the Qt + VoiceXML client framework on the E680i mobile phone introduced an approximate three second delay into the recognition latency. Had we enough time to go back in and re-code parts of the audio module, this issue could have been eliminated. The latency issue in our implementation further emphasized a fact we already were quite aware of – in order to be used in competitive multiplayer game, speech recognition must be very fast as well as accurate! Fortunately, researchers at Motorola have already demonstrated multimodal "+ VoiceXML" client frameworks on other devices that can consistently deliver recognition results over wide area networks in less than one second. Hence we are not aware of any significant technical barriers preventing the implementation and deployment of networked-based speech modalities for mobile games on existing mobile phones and wireless networks, using the approach we have described in this paper.

V. Conclusion

In comparison to what we have achieved, the above challenges look very trivial because they are mostly a matter of optimization and some smart design decisions on the part of mobile game designers and developers.

We are very happy with the results of our efforts and have proved the maturity of this platform by creating three games which in a sense are very commercially viable. In fact when these games were showcased to various industry representatives, they responded with a very positive and enthusiastic feedback which we find very encouraging.

Having proved out the viability of incorporating a standardized network-based speech recognition modality into a mobile game, and having established a platform upon which we can do further experimentation, in the future we intend to explore how this enabler can be incorporated into mobile game design in more creative and novel ways. In particular, with the increased bandwidth and affordability of emerging wide area networking technologies, speech-enabled mobile clients to 3D massive multiplayer online games is one area we are interested in pursuing.

We plan to take our results further and explore the combination of multimodality and context awareness in the mobile games of tomorrow. This also would make for some very interesting use cases, which hopefully are also very much viable in a commercial environment.

VI. Acknowledgements

We would like to thank Dr. Fred Kitson, Vice President of Applications Research at Motorola Corporation for providing the initial support to get the mobile component of the GamePipe Program running. Under his support, we were provided the entire gamut of resources, financial and technical available at Motorola Corporation and come up with such a strong technological framework.

We would also like to thank Mitch Lasky and Zack Norman from EA Mobile who helped us kick start the mobile component of our program and provide us real world industry wisdom, which has driven the design and development of every project in this course. These principles have helped us establish credibility in all our projects when we showcase them to the industry and prepare our students for real world challenges.

Finally we would like to thank Gabi Artzi from Nuance who provided us the much needed licensing and support for their speech recognition products which have helped us make these games possible.

References

Zyda, Thukral et al., “Educating the Next Generation of Mobile Game Developers” for IEEE Computer Graphics and Applications, March 2007

J.R. Engelsma et al., "Ubiquitous Mobile Gaming," *Proc. System Support for Ubiquitous Computing Workshop (UbiSys),* 2006; http://www.magic.ubc.ca/ubisys/positions engelsma.pdf.

D. Pearce et al., "An Architecture for Seamless Access to Distributed Multimodal Services," *Proc. 9th European Conf. Speech Communication and Technology,* Int'l Speech Communication Assoc., 2005, pp. 2845–2848.

S. McGlashan et al., "Voice Extensible Markup Language (VoiceXML) Version 2.0," World Wide Web Consortium (W3C)recommendation,Mar.2004, http://www.w3.org/TRvoicexml20/

Jim Ferrans and Jonathan Engelsma. “Software architectures for networked mobile speech applications", in Automatic Speech Recognition on Mobile Devices and over Communication Networks, Zeng-Hua Tan, ed., Springer Press, 2008.