# Progress in Spoken Programming

Benjamin M. Gordon
George F. Luger
Department of Computer Science
University of New Mexico

**Abstract**

The dominant paradigm for programming a computer today is text entry via keyboard and mouse, but there are many common situations where this is not ideal. For example, tablets are challenging the idea that computers should include a keyboard and mouse. The virtual keyboards available on tablets are functional in terms of entering small amounts of text, but they leave much to be desired for use as a keyboard replacement. Before tablets can can become truly viable as a standalone computing platform, we need a programming environment that supports non-keyboard programming.

An introduction to this research was presented at the UNM CS Student Conference in 2011 [8]. In this paper, we describe progress and lessons learned so far.

## 1 Introduction

The dominant paradigm for programming a computer today is text entry via keyboard and mouse. Keyboard-based entry has served us well for decades, but it is not ideal in all situations. People may have many reasons to wish for usable alternative input methods, ranging from disabilities or injuries to naturalness of input. For example, a person with a wrist or hand injury may find herself entirely unable to type, but with no impairment to her thinking abilities or desire to program. What a frustrating combination!

Furthermore, with the recent surge in keyboard-less tablet computing, it will not be long before people want to program directly on their tablets. Today's generation of tablets are severely limited in comparison to a desktop system, suitable for viewing many forms of content, but not for creating new content. Newly announced products already claim support for high-resolution screens, multicore processors, and large memory capacities, but they still will not include a keyboard. It is certainly possible to pair a tablet with an external keyboard if a large amount of text entry is needed, but carrying around a separate keyboard seems to defeat the main ideas of a tablet computer.

What is really needed in these and other similar situations is a new input mechanism that permits us to dispose of the keyboard entirely. Humans have been speaking and drawing for far longer than they have been typing, so employing one of these mechanisms seems to make the most sense. Products such as Apple's Siri have demonstrated the usefulness of systems built around non-keyboard inputs. In this research, we consider the problem of enabling programming via spoken input.

Successful dictation software already exists for general business English, as well as specialized fields like law and medicine, but no commercial software exists for "speaking programs." Visual and multimedia programming has been an active research area for at least 30 years, but systems for general-purpose speech-based programming are rare. Several researchers have attempted to retrofit spoken interfaces onto existing programming languages and editors [2,3,5], but these attempts have all suffered from the same problem: existing languages were designed for unambiguous parsing and mathematical precision, not ease of human speech.

This research addresses the topic in two specific ways: through the creation of a new spoken programming language, and through the creation of an editing environment for the language.

# 2 Related Work

This research has been previously described in [8] and [9].

In terms of other research, the idea of adding speech support to an existing language is not new. In 1997, Leopold and Ambler added voice and pen control to a visual programming language called Formulate [11].

More recently, Désilets, Fox and Norton created VoiceCode [5] at the National Research Center in Canada. Begel and Graham studied how programmers verbalized code [3]. Based on this study, Begeland Graham developed a spoken variant of Java called Spoken Java. In addition to the Spoken Java language syntax, Begel and Graham developed a suitable plugin (SPEED) for the Eclipse development environment to enable speech input [3, 4].

Arnold, Mark and Goldthwaite proposed a system called VocalProgramming [2]. Their system was intended to take a context free grammar (CFG) for a programming language and automatically create a "syntax-directed editor," but the system appears to have never been implemented.

Shaik et al. created an Eclipse plugin called SpeechClipse to permit voice control of the Eclipse environment itself [12]. They permitted dictation of "well-known programming language keywords," but primarily concentrated on providing access to the menu and keyboard commands available in Eclipse.

Outside the realm of traditional programming languages, Fateman considered the task of speaking mathematical expressions [7]. He created a system that produced TEX output from a spoken form of equations.

# 3 Progress

This research project is made of two components: The programming language itself, and the development environment.

## 3.1 Spoken Programming Language

The language is a simple imperative language with English-like syntax for the supported constructs. It supports simple loops, conditionals, functions, and variables of a few basic data types. The syntax and supported features are more completely described in [10].

A mostly-complete compiler has been created that takes the textual form of the language as input and produces programs that run on the Java JVM as output. It uses ANTLR [1] as the parser engine and produces Java as an intermediate language. All of the major constructs of the

language are implemented (assignments, loops, function calls, etc), and the compiler is capable of compiling a large number of programs that solve real (though small) classic problems in computer science.

As originally anticipated, the syntax appears extremely verbose compared to traditional programming languages, but experience has shown that the extra verbosity is largely mitigated by the rhythms of comfortable English speech in practice. As an example, here is a tiny, "Hello, World"-style program that prints the number 42 and exits:

```
define function main taking no arguments as
    print 42
    return 0
end function
```

As a slightly larger example, this program prints $n!$ for $n$ ranging from 0 through 9 using a naive recursive factorial implementation:

```
define function factorial taking arguments N as
    if N < 2 then
        return 1
    else
        set X to N - 1
        set Z to the result of calling factorial with X
        return N * Z
    end if
end function

define function main taking no arguments as
    set I to 0
    while I < 10 do
        set Y to the result of calling factorial with I
        print Y
        new line
        set I to I + 1
    end while
    return 0
end function
```

The primary component missing from the proposed compiler features is type inference. Implementation of this feature is underway; currently, the compiler infers that every variable is an integer (notice that the examples above are compilable under this restriction).

## 3.2 Programming Environment

Unlike a traditional text-based programming language, a spoken programming language isn't much good without an environment to support entry and editing. The environment for this programming language is built as a plugin for the Eclipse IDE [6], using CMU Sphinx [13] as the voice recognition component.

The Eclipse plugin currently allows basic dictation of program text. It does not yet support editing or debugging commands. Additionally, due to the way Sphinx handles grammar-based recognition, the entire program must be spoken without a pause for it to be successfully recognized. The "print 42" program above can be dictated, but the second example is too long for normal humans to get through without needing a breath. Correcting this deficiency is the current primary focus of development in this area.

# 4 Future Work

The compiler and Eclipse plugin are expected to be finished in summer of 2012. After some initial sanity testing and feedback from early users, the effectiveness of the complete system will be evaluated through a user study. We anticipate beginning this study in fall of 2012.

# 5 Conclusion

The idea of programming a computer through voice input is not a new one, but the rise of tablet computing has made it more relevant than ever. The creation of a new programming language and an associated environment for voice input was proposed in spring 2011. Implementation of this idea is proceeding apace and will soon be ready for testing. Upon completion of the editing environment, we expect that these additions will result in a measurable improvement in the speed and accuracy with which code can be produced via speech.

# References


[1] ANTLR Parser Generator. `http://www.antlr.org/`, Retrieved April 13, 2012.

[2] Stephen C. Arnold, Leo Mark, and John Goldthwaite. Programming by voice, vocalprogramming. In *Proceedings of the fourth international ACM conference on Assistive technologies*, Assets '00, pages 149–155, New York, NY, USA, 2000. ACM.

[3] Andrew Begel and Susan L Graham. Spoken programs. *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 99 – 106, 2005.

[4] Andrew Begel and Susan L Graham. An assessment of a speech-based programming environment. *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on*, pages 116–120, 2006.

[5] A Désilets, DC Fox, and S Norton. Voicecode: an innovative speech interface for programming-by-voice. *CHI'06 extended abstracts on Human factors in computing systems*, pages 239–242, 2006.

[6] Eclipse: The eclipse foundation open source community website. `http://www.eclipse.org/`, Retrieved January 10, 2011.

[7] R Fateman. How can we speak math? *Journal of Symbolic Computation*, Jan 1998.

[8] Benjamin M. Gordon. Developing a Language for Spoken Programming. In *UNM Computer Science Student Conference*, pages 3–10, `http://www.cs.unm.edu/~csgsa/unm-cs-conf7.pdf`, April 2011.

[9] Benjamin M. Gordon. Developing a Language for Spoken Programming. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1847–1848, August 2011.

[10] Benjamin M. Gordon. Developing a Language for Spoken Programming (Dissertation Proposal). `http://www.cs.unm.edu/~bmgordon/proposal-bmg.pdf`, May 2011.

[11] J.L. Leopold and A.L. Ambler. Keyboardless visual programming using voice, handwriting, and gesture. In *Visual Languages, 1997. Proceedings. 1997 IEEE Symposium on*, pages 28 –35, September 1997.

[12] S Shaik, R Corvin, R Sudarsan, F Javed, Q Ijaz, S Roychoudhury, J Gray, and B Bryant. Speechclipse: an eclipse speech plug-in. *Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, pages 84–88, 2003.

[13] CMU sphinx - speech recognition toolkit. `http://cmusphinx.sourceforge.net/`, Retrieved January 15, 2011.