THE UNIVERSITY OF ALABAMA
COMPUTER SCIENCE

Automation of VUI to GUI Mapping

SIGCHI 2013 Doctoral Consortium

Amber Wagner,
Dept. of Computer Science
University of Alabama
ankrug@bama.ua.edu

Motivation

The Association for Computing Machinery (ACM) code of ethics states that all individuals should have equal opportunities to benefit from computer usage. By not providing alternative means of access to new environments for teaching computer science, users with disabilities are being denied learning opportunities that may allow them to explore career paths in computing.

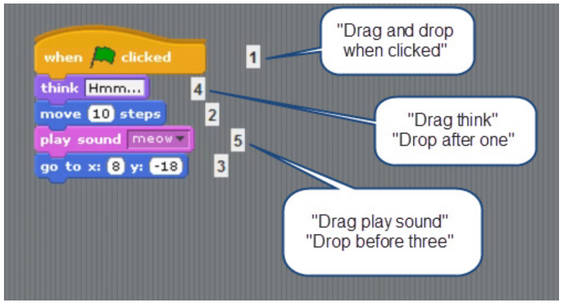
2.7 million kids (ages 5 to 17) suffer from a disability

The driving motivation for the underlying theme of this research is recognition that more children (or adults) should have the opportunity to learn about Computer Science using Initial Programming Environments (IPEs).

A possible solution to broaden the accessibility of GUIs is to use a voice-driven interface to assist as an input modality for those with motor impairments. Currently, children (or adults) with motor impairments are unable to take advantage of graphical IPEs (e.g., Scratch). The Vocal User Interface (VUI) described in this poster provides an opportunity for children with motor impairments to use graphical IPEs.

Myna

Myna (pictured below) is a Java program that runs parallel to Scratch. When Myna is started and Scratch is opened, the user can begin creating a program within Scratch solely through voice. The grammar for the user is relatively simple as the verbal commands match the commands on the screen with the exception of select action commands (e.g., "drop after").



```
when [flag] clicked    1
think Hmm...           4
move 10 steps          2
play sound meow        5
go to x: 8 y: -18      3
```

"Drag and drop when clicked"
"Drag think" "Drop after one"
"Drag play sound" "Drop before three"

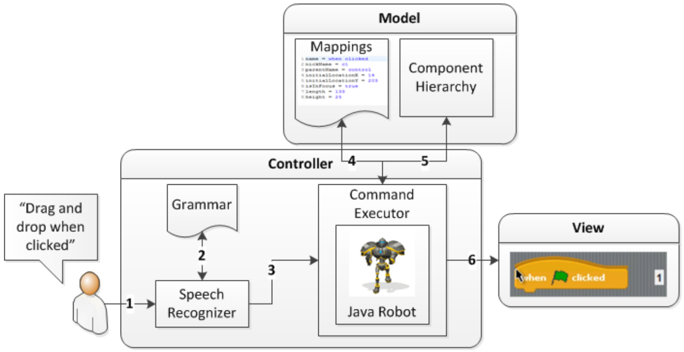
Myna Navigation

There are three key types of navigation in Myna:

1. *Drag and Drop Navigation*: This mimics the idea of clicking an object, dragging it to another location, and dropping it. Blocks added using drag and drop will be added to the end of the program.
2. *Transparent Frame Navigation*: The transparent frames allow small numbers to be placed next to commands within the program (see below image). The user can say, "drag", the desired command, "drop after" (or "drop before"), and the number of his/her choice.
3. *Continuous Navigation:* Allows the user to move the cursor to the right, left, up, or down mimicking the movement of the mouse. This is used sparingly to avoid vocal fatigue.

Myna Control Flow

The following is an example of how Myna maps voice commands to actions (pictured below):



```
name = when clicked
nickName = c1
parentName = control
initialLocationX = 14
initialLocationY = 203
isInFocus = true
length = 133
height = 25
```

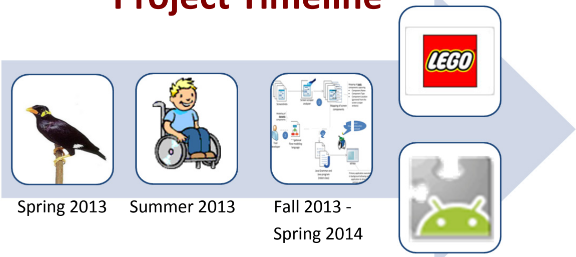
The user states, "Drag and drop when clicked." The speech recognizer identifies the command and checks it against the grammar. If the command exists, the command executor locates the mapping and verifies the state. Finally, the Java Robot performs the corresponding mouse/keyboard action.

Pilot Study

In April 2013, a pilot study was conducted with five CS graduate students. The participants each completed three programs with the mouse/keyboard and with Myna (voice). Testing showed that there is no significant difference in time (average of 13 seconds difference) between either modality.

Observation	Average Count
Participant stated incorrect Myna command	0.67
Participant stated incorrect Scratch command	0.33
Voice recognition error	3.67
Myna location error	0.93

Project Timeline



Phase I: Provide navigation support. Complete
Phase II: Expand existing implementation to create a 100% voice-controlled application (scroll-bar navigation, grammar customization). Spring 2013

Phase III: Conduct a formal user-study with United Cerebral Palsy of Birmingham. Summer 2013

Phase IV: Using model driven engineering, create a model to simplify the process of incorporating voice controls into applications built with the WIMP metaphor as the core such as Lego Mindstorms Labview and App Inventor. Fall 2013—Spring 2014

This project is supported by NSF grant IIS-1117940 with previous support from a Google Research Award.

Google research NSF