StaRt
A Biofeedback App

Heather Campbell, Helen Carey, Celine Wu, Dalit Shalom
Developing Assistive Technologies | NYU
October 2014
Name Development

The “StaRt” app is a work in progress. It was temporarily called the “aRt” app, and its name is still in development. The name “StaRt” is an acronym standing for “Speech Therapist’s application for R Treatment”.

History of StaRt

Apr 2013: started making LPC spectrum
Sep 2013: biofeedback app proposal
Jan 2014: first DAT team
Apr 2014: working version of LPC spectrum
May 2014: LPC and first UX mock-ups presented at Digital Media Student Showcase
Sep 2014: second DAT team
Team Roles

Heather Campbell
Project coordinator

Helen Carey
Development and implementation

Celine Wu
Designing character and logo

Dalit Shalom
User interface development
User Interface Development
Syllable Practice - User Practice v1

- Originally, we were to place color coded balls on the peaks of the user's LPC spectrum. These balls are color coded with respect to their desired locations on the model, which are represented as rings coded as the same color.
- See image below for the metaphor in use.

Syllable Practice - User Practice v2

- After receiving feedback from Tara that the user should not be forced to or encouraged to model the desired vertical (amplitude) locations, the balls are now placed on a horizontal line so all balls and ring would be aligned.
Consonantal front | 5 trials

Last practice: Sep 29, 2014
Accuracy last session: 89.6%

Visual representation (or any other name for this chart)
Potential Color Schemes
Development
What is a Hybrid App Anyway?

<table>
<thead>
<tr>
<th>Features</th>
<th>Native App</th>
<th>Web App</th>
<th>Hybrid App</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Languages</strong></td>
<td>Objective-C, Swift, Java, C, etc</td>
<td>HTML + CSS + JavaScript</td>
<td>Web</td>
</tr>
<tr>
<td><strong>Development Cycle</strong></td>
<td>Slower. You need to build a new app for each platform. Often different development tools are needed for each platform.</td>
<td>Quicker. Write once, publish once, view it anywhere. Multiple tools and libraries to choose from</td>
<td>Web</td>
</tr>
<tr>
<td><strong>Compatibility</strong></td>
<td>Platform-dependent, and occasionally hardware-dependent.</td>
<td>Platform-agnostic, content can be reformatted with CSS to suit any device</td>
<td>Web</td>
</tr>
<tr>
<td><strong>Deployment</strong></td>
<td>Generally must deployed through an app store, which often slows dev time and comes with its own set of bureaucratic hassles.</td>
<td>Just hit refresh.</td>
<td>Save instance from web to homescreen.</td>
</tr>
<tr>
<td><strong>Internet Dependence</strong></td>
<td>Stand-alone. Internet is not req’d for use. Use it anywhere at anytime.</td>
<td>Wifi or data connection generally needed.</td>
<td>Native</td>
</tr>
<tr>
<td><strong>Hardware Sensors</strong></td>
<td>Yes, all of them: camera, gyroscope, microphone, compass, accelerometer, GPS</td>
<td>Access through the browser is limited, though geolocation is common</td>
<td>Native-ish (can access most hardware thru APIs)</td>
</tr>
<tr>
<td><strong>User Experience</strong></td>
<td>Responsive and fully functional. Supports native OS gestures.</td>
<td>Can be sluggish, but new JS libraries for mobile are making this less of a problem. Written for screen &amp; mouse-based form factors, so gesture support reqs extra programming.</td>
<td>Web</td>
</tr>
<tr>
<td><strong>Animation/Graphics</strong></td>
<td>Fast and responsive.</td>
<td>Web apps are getting closer, especially with flat design and other developments in graphic optimization, but will never be as fast as native.</td>
<td>Web</td>
</tr>
<tr>
<td><strong>Data Storage</strong></td>
<td>Only if you build it, and it’s more difficult if data is disparate</td>
<td>Discussion is easy, all data is stored on a server</td>
<td>Web-ish (+ small amount of local storage)</td>
</tr>
</tbody>
</table>
How do Web Languages Work?

HTML5
Hypertext markup language is the backbone of your app or webpage. This where you define the structure of your document and provide semantic tags to assist in accessibility and semantic web functions. Your content (ex. text, video, animation, LPC code) is placed in the HTML structure.
How do Web Languages Work?

CSS

Cascading Stylesheets control the visual display of the structures and content in your HTML. Ideally, stylesheets are built in a cascade to account for various form factors (different devices) and media types (ex. screen, printer-friendly, mobile, synthetic speech reader etc). Additionally, the CSS layer often includes text to ‘modernize’ or adapt your styles to the idiosyncrasies of each of the major browsers.
How do Web Languages Work?

JavaScript
This is the behavior layer which affords real-time user interaction with objects in the app, through defined functions which run-on event triggers. This is your client-side scripting language, and it can run anything from simple animations to more sophisticated frameworks that make your app or site ‘behave’ like ordinary desktop programs.
How do Web Languages Work?

Data Layer
Languages like PHP, Python, SQL handle the communication of user data back and forth from the remote server. Obviously, this is a very important layer, especially for research, but it is outside of the scope of our DAT UI/UX project. When building out our HTML and JS, we will ensure that our design supports future data handling code.
How Are Apps Typically Built?

• Webapps are typically built on frameworks, like Ionic, Bootstrap, Intel XDK etc.

• Native apps are built in a native language and then ported to other platforms.

• Hybrid apps can be built using whatever tools best support the needs of the user or the function of the app.
HOW ARE APPS TYPICALLY BUILT?

Currently I'm hand-coding, because I believe that simple, lean, human-written code will be the easiest to integrate with other code.

- Webapps are typically built on frameworks, like Ionic, Bootstrap, Intel XDK etc.
- Native apps are built in a native language and then ported to other platforms.
- Hybrid apps can be built using whatever tools best support the needs or the use of the app.

There will always be more than one way to solve a problem… and taking notes helps.

MY DEVELOPMENT APPROACH
Next Steps

• Learn more about coding for a touch environment

• Learning how to use APIs to access core device features, like the microphone

• Integrating LPC program into HTML5/mobile safari environment

• Learning about building back-end interaction for remote data storage into code
Tutorials - Character Animation

Tutorial  Story board

Do the Wave
Fade in words

Fade to/from white.
Session 1, Script 1 VO

"Watch how the wave moves when I talk."

To/from white.
"Now you try." App listens.

Session 1, #2
"Watch this... vowel sounds..."
/o/
/a/
/i/

Template choosing as part of tutorial!

The /r/ Sound
Fade in/out white

Back /r/

Transition /r/, Guy appears, "...and his..."
Creating a character for tutorials

• How to do a tutorial? What types of tutorials have we seen? What works?

• Interactive. Videos and user actions both.

• Storyboards from existing scripts

• Decision to give voice a face, but what face?
Character: Process

• Brainstorm and decide on a character/species (Seal, Aardvark, Orchid, Rooster, Oyster, Porpoise, Starfish)

• Sketch options in likely poses

• Digitize and animate the character

• Incorporate character into title and logo

• Bring in other components into Flash

• Organize layers and timeline

• Finalize and export!
Character: Oyster or Starfish?
Tutorial Layers

1. Screen Recordings of LPC
2. Voice
3. Animation Of Character
Template Generation
Tutorials - Recording Outline Scripts

1. **Intro to Experience**
   - Demo Injection
   - "Try" capture pic of speech
   - Any moment - try to match////

2. **Capture Moment of Speech**
   - Demo Injection

3. **Experience Capture Diff in Your Sudoku**
   - Demo Injection

4. **KeyCode**
   - Demo Injection

5. **Process Capture //**
   - Demo Injection

6. **Match Diff in Sudoku**
   - Demo Injection

7. **Changing Mix it Up**
   - Demo Injection

8. **Controlling Mark**
   - Demo Injection

9. **Free Practice w/ Template**
   - Demo Injection

10. **Wrap Up**
    - Demo Injection

Next week!!

**Recording Outline Scripts**

- **Filming of Tutorials Next Week!!**
Process For Figuring Out How To Generate Individualized Templates

- Hagiwara (1995) suggested that by taking 80% of the F3 of the four vowels /ae,aa,uw,iy/, one could find a threshold dividing correct from incorrect productions of /r/.

- Sarah Hamilton & Suzanne Boyce at the University of Cincinnati are currently investigating whether Hagiwara’s formula works at finding this threshold.

- Looking at errored /r/ in kids with /r/ distortion, so not enough correct tokens.

- Threshold seems to be helpful for distinguishing correct from incorrect, which will be useful for scoring.

- To make a template, we’ll need to generate a prototype /r/ F3 value that represents an ideal F3 value for correct /r/.

- Needs to be generated from normative data from children with typically developing speech sounds.
Can we approximate the F3 of a prototype /r/ using vowels?

What combination of vowels?
What percentage of F3?

Methodology:

• Use Lee et al. (1999) normative data for ages 5 through 19+
• F0-F3 for vowels
• aa “pot” ae “bat” ah “but” ao “ball” eh “bet” ih “bit” iy “bead” uh “put” uw “boot” er “bird”
• For each sex, age, and vowel combo in normative sample, run different variations of Hagiwara’s formula on normative data in R programming language trying to find good estimate of correct /r/
• Compare that estimate with the actual /r/ value for each kid
• Plot the differences for different % values to find best percentage
R Code

```r
# Load data
library(readr)

data <- read.csv("path_to_data.csv")

# Analyze data
summary(data)
mean(data$variable)

# Plot data
plot(data$variable, type = "l")

density(data$variable)
```
Hagiwara <- round(Hagiwara, digits=2)
diff <- Hagiwara - R
diff <- round(diff, digits=2)

myDF <- data.frame(sex, age, vowels, Hagiwara, R, diff)
str(myDF)
write.xlsx(myDF, "Lee Normative Data-Hagiwara.xlsx", sheetName="Sheet4")

library(ggplot2)

#plotting with females
FemalesMYOF <- subset(myDF, sex=="F")
FemalesMYOF$age <- as.factor(as.character(FemalesMYOF$age))
FemalesMYOF$age <- ordered(FemalesMYOF$age, levels=c(5,6,7,8,9,10,11,12,13,14,15,16,17,18,19))
p <- ggplot(FemalesMYOF, aes(x = age, y = diff, fill=vowels, title="Females at 80%"))
p + geom_bar(position = "dodge", stat="identity")

#plotting with males
MalesMYOF <- subset(myDF, sex=="M")
MalesMYOF$age <- as.factor(as.character(MalesMYOF$age))
MalesMYOF$age <- ordered(MalesMYOF$age, levels=c(5,6,7,8,9,10,11,12,13,14,15,16,17,18,19))
p <- ggplot(MalesMYOF, aes(x = age, y = diff, fill=vowels, title="Males at 80%"))
p + geom_bar(position = "dodge", stat="identity")
Graphical Results

• For females, 72% of /ae/, /aa/, and /uw/ may be appropriate target

• For males, 68% of F3 of /ae/, /aa/, and /uw/ may be appropriate target

• More testing of this hypothesis is needed!!

• App will be able to calculate F3 for /r/ prototype by averaging F3 of /ae/, /aa/, and /uw/ to build template!

• Then F3 values during sessions within a certain range from this value can potentially be automatically considered correct!
Thank you