ICP 11: Text-To-Speech

Joe Moon
Email: jmn5y@umsystem.edu
Github: https://github.com/joemoon-0/WebMobile-2022Spring/tree/main/2022Spring-Mobile/Mobile_ICP11

Nathan Cheney
Email: ncxn8@umsystem.edu
Github: https://github.com/nathancheney/Web-Mobile-Programming/tree/main/ICP-11/Text_to_speech

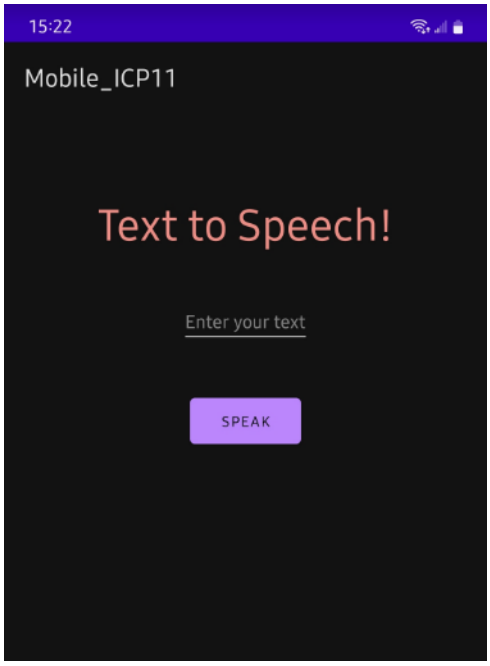
Introduction

Text-to-Speech applications allow users greater accessibility when using an application, particularly those who may be visually impaired.

Interface

A simple interface for this application was created to receive a text input followed by a button that would initiate the text-to-speech translation. The language set for this translation is US-English.

As shown on the right, the interface consists of a TextView, EditText, and a Button.



Text-To-Speech (TTS) Implementation

The TTS implementation utilizes the **android.speech.tts.TextToSpeech** library for the functions that allow for this feature. An event listener is used to capture the button click event which triggers the code to translate the text to speech.

The app first sets the language to US-English and then conducts checks for the TTS implementation as shown in the figure below in Figure 1.1. If a language is not supported, an error message is reported to the Log, otherwise the **speak()** method is called to translate the text.

```
34                      if (status == TextToSpeech.SUCCESS) {
35                          int result = tts.setLanguage(Locale.US);
36                          if (result ==TextToSpeech.LANG_NOT_SUPPORTED || result == TextToSpeech.LANG_MISSING_DATA) {
37                              Log.e("message", "Language is not supported");
38                          } else {
39                              speak();
40                          }
```

Figure 1.1: MainActivity.java - TTS implementation checks

The **speak()** method begins by capturing the text inputted by the user, sets the speech rate (an optional parameter), and then processes the text through the library's

method to translate the text. The **QUEUE.ADD** parameter is used to playback the complete text however it can be replaced by **QUEUE.FLUSH** to stop the playback before ending.

```
50      private void speak() {
51          String text = editText.getText().toString();
52          tts.setSpeechRate(1.0f);
53          tts.speak(text, TextToSpeech.QUEUE_ADD, null);
54      }
```

Figure 1.2: MainActivity.java – speak method

Finally, the **onPause()** and **onDestroy()** methods are implemented to complete the application. **onPause()** is set to stop the playback if the app is brought out of focus (minimized) while **onDestroy()** is set to release the TTS resources once playback stops so that other applications may use it if they request for it.