Introduction

When developing an application that records sound, it may be useful to display the sound amplitude, or volume, to the user. Since Android does not offer this feature, this article demonstrates one method of implementing it. Following is a process for creating a simple Pulse Code Modulation (PCM) Wave (.wav) recorder, which displays the signal amplitude.

The Android AudioRecord class allows you to read captured audio samples into a buffer. In this example, we store each sample using encoding with 16 bits per sample, in a short array buffer of two bytes. We calculate the amplitude using the root mean square (RMS), where the calculated value is equal to the square root of the mean of the squares of the sample values. Then, to display the result, we create a progress bar with a green-to-red gradient showing the volume. When the recoding is complete, the application savs the raw data in PCM Wave format, in a .wav file.

Drawable Resource for the Progress Bar

To define the appearance (curved rectangular boarder) and color (green-to-red gradient) of the progress bar, create an xml file in the **res/drawable** folder. Here is one example of how this xml file might look:

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="8dp" />
            <gradient
                android:endColor="#000000"
                android:startColor="#000000" />
            <stroke
                android:width="1dp"
                android:color="#FFFFFF" />
        </shape>
    </item>
    <item android:id="@android:id/progress">
        <clip>
            <shape>
                <corners android:radius="8dp" />
                <gradient
                    android:endColor="#FF0000"
                    android:startColor="#00FF00" />
            </shape>
        </clip>
    </item>
</layer-list>
```

Layout

Use the drawable resource above in the application layout. Choose the progress bar’s maximum value. In this example, we have set a maximum RMS value of 4000:

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:max="4000"
    android:progressDrawable="@drawable/progressbar" />
```

Application Code: Progress Bar Control

The sample application uses a threat to read from the recorder buffer and write to the output stream. We can compute the RMS value and set the progress bar value during this operation.

```
AudioRecord recorder; // our recorder, must be initialized first
short[] buffer; // buffer where we will put captured samples
DataOutputStream output; // output stream to target file
boolean isRecording; // indicates if sound is currently being captured
ProgressBar pb; // our progress bar recieved from layout
while (isRecording) {
        double sum = 0;
        int readSize = recorder.read(buffer, 0, buffer.length);
        for (int i = 0; i < readSize; i++) {
                output.writeShort(buffer [i]);
                sum += buffer [i] * buffer [i];
        }
        if (readSize > 0) {
                final double amplitude = sum / readSize;
                pb.setProgress((int) Math.sqrt(amplitude));
        }
}
```

Application Code: PCM Wave Format

To save the recording to a wave file, you will need a header as well as your captured samples. Learn more about header file specifications at the following URL: http://ccrma.stanford.edu/courses/422/projects/WaveFormat/ Keep in mind that intergers and shorts must be written in little endian byte order

Summary

The code samples in this article define a simple layout with one button, to start or stop audio recording, and a progress bar displaying the sound volume. The application initializes the recorder at the beginning. When the user presses the START button, the application begins to record sound and write it to the buffered output, while showing the sound volume with a dynamically updated progress bar. When the user presses the STOP button, the application stops recording, saves the audio data in a PCM Wave format (a .wav file) and resets. When the application is terminated, it releases the recorder.

Ref:http://developer.samsung.com/android/technical-docs/Displaying-Sound-Volume-in-Real-Time-While-Recording