```
 1 //
 2 // Created by daran on 1/12/2017 to be used in ECE420
    Sp17 for the first time.
 3 // Modified by dwang49 on 1/1/2018 to adapt to
   Android 7.0 and Shield Tablet updates.
 4 //
 5
 6 #include <jni.h>
 7 #include "ece420_main.h"
 8 #include "ece420_lib.h"
 9 #include "kiss_fft/kiss_fft.h"
10
11 // Declare JNI function
12 extern "C" {
13 JNIEXPORT void JNICALL
14 Java_com_ece420_lab3_MainActivity_getFftBuffer(JNIEnv
    *env, jclass, jobject bufferPtr);
15 }
16
17 // FRAME_SIZE is 1024 and we zero-pad it to 2048 to
   do FFT
18 #define FRAME_SIZE 1024
19 #define ZP_FACTOR 2
20 #define FFT_SIZE (FRAME_SIZE * ZP_FACTOR)
21 // Variable to store final FFT output
22 #define PI 3.141592653589793
23 float fftOut[FFT_SIZE] = {};
24 bool isWritingFft = false;
25
26 // initialize a hamming window once to save
   comutation time
27 float hammingWindow[FRAME_SIZE];
28 bool hammingWindowInitialized = false;
29
30 // initalize kiss_fft parameters
31 kiss_fft_cfg kcfg;
32 kiss_fft_cpx fin[FFT_SIZE];
33 kiss_fft_cpx fout[FFT_SIZE];
34
35
36 // function declarations
```

```
37 void generateHamming();
38
39 void ece420ProcessFrame(sample_buf *dataBuf) {
40     isWritingFft = false;
41
42     // Keep in mind, we only have 20ms to process
   each buffer!
43     struct timeval start;
44     struct timeval end;
45     gettimeofday(&start, NULL);
46
47     // Data is encoded in signed PCM-16, little-
   endian, mono channel
48     float bufferIn[FRAME_SIZE];
49     for (int i = 0; i < FRAME_SIZE; i++) {
50         int16_t val = ((uint16_t) dataBuf->buf_[2 * i
   ]) | (((uint16_t) dataBuf->buf_[2 * i + 1]) << 8);
51         bufferIn[i] = (float) val;
52     }
53
54     // Spectrogram is just a fancy word for short
   time fourier transform
55     // 1. Apply hamming window to the entire
   FRAME_SIZE
56     // 2. Zero padding to FFT_SIZE = FRAME_SIZE *
   ZP_FACTOR
57     // 3. Apply fft with KISS_FFT engine
58     // 4. Scale fftOut[] to between 0 and 1 with log
   () and linear scaling
59     // NOTE: This code block is a suggestion to get
   you started. You will have to
60     // add/change code outside this block to
   implement FFT buffer overlapping (extra credit part).
61     // Keep all of your code changes within java/
   MainActivity and cpp/ece420_*
62     // ********************* START YOUR CODE HERE
    ********************** //
63
64     /* initialize hamming window if not done so
   already */
65     if (!hammingWindowInitialized) {
```

```
 66          /* fill in Hamming array */
 67          generateHamming();
 68          /* zero pad fin */
 69          for (int i = FRAME_SIZE; i < FFT_SIZE; i
    ++) {
 70              fin[i].r = 0.0;
 71              fin[i].i = 0.0;
 72          }
 73          kcfg = kiss_fft_alloc(FFT_SIZE,0, nullptr,
    nullptr);
 74          hammingWindowInitialized = true;
 75      }
 76
 77      /* apply window */
 78      for (int i = 0; i < FRAME_SIZE; i++) {
 79          fin[i].r = bufferIn[i] * hammingWindow[i];
 80          fin[i].i = 0.0;
 81      }
 82
 83      /* compute FFT */
 84      kiss_fft(kcfg,fin,fout);
 85
 86      // thread-safe
 87      isWritingFft = true;
 88      // Currently set everything to 0 or 1 so the
    spectrogram will just be blue and red stripped
 89      float max_val = 0.0;
 90      for (int i = 0; i < FRAME_SIZE; i++) {
 91 //         fftOut[i] = (i/20)%2;
 92          fftOut[i] = log10(fout[i].r*fout[i].r + fout
    [i].i*fout[i].i);
 93          if (fftOut[i] > max_val)
 94              max_val = fftOut[i];
 95      }
 96      /* linearly scale the values so that it is
    between 0 and 1 */
 97      for (int i = 0; i < FRAME_SIZE; i++) {
 98          fftOut[i] = fftOut[i] / max_val;
 99      }
100
101      // ********************* END YOUR CODE HERE
```

```
101  ************************* //
102     // Flip the flag so that the JNI thread will
    update the buffer
103     isWritingFft = false;
104
105     gettimeofday(&end, NULL);
106     LOGD("Time delay: %ld us, buf size %d, cap size
     %d",  ((end.tv_sec * 1000000 + end.tv_usec) - (
    start.tv_sec * 1000000 + start.tv_usec)), dataBuf->
    size_,dataBuf->cap_);
107 }
108
109 void generateHamming() {
110     for (int i = 0; i < FRAME_SIZE; i++) {
111         hammingWindow[i] = 0.54 - 0.46 * cos((2 * PI
     * i)/(FRAME_SIZE - 1));
112     }
113 }
114
115 // http://stackoverflow.com/questions/34168791/ndk-
    work-with-floatbuffer-as-parameter
116 JNIEXPORT void JNICALL
117 Java_com_ece420_lab3_MainActivity_getFftBuffer(
    JNIEnv *env, jclass, jobject bufferPtr) {
118     jfloat *buffer = (jfloat *) env->
    GetDirectBufferAddress(bufferPtr);
119     // thread-safe, kinda
120     while (isWritingFft) {}
121     // We will only fetch up to FRAME_SIZE data in
    fftOut[] to draw on to the screen
122     for (int i = 0; i < FRAME_SIZE; i++) {
123         buffer[i] = fftOut[i];
124     }
125 }
126
127
```