CS12 Bonus Assignment: GuitarHero on Android

Pair programming is not allowed on this assignment. You must work on your own. The maximum number of points you can get on the bonus assignment is 10 points. If you don’t like porting GuitarHero, please check out alternative ideas at the end of this document.

Creating the Project:

1. File >> New >> Android Project
2. Select “Create Project from Existing Source”
3. Browse to the directory where you have downloaded the unpacked .ZIP containing the “Starter” directory to
4. NOTE: Eclipse will give you issues if the directory you have unpacked the .ZIP into is the same as what you title the project (Error: Invalid project description)
5. Select the “Starter” directory as your existing code source
6. Select “Android 2.2” in your build target
7. Click “Finish”
8. Click “Run GuitarHero” (Green arrow at top of screen)
   - This may run, but should not function. We will add functionality so that it actually works as part of this assignment

Porting Assignment 7:

In order to port assignment 7, which we coded in DrJava, we need to do a couple of things.

- Copy RingBuffer.java and GuitarString.java to Starter/src/edu/cs112/guitar_hero/
- Add a line to the top of GuitarString.java and RingBuffer.java to “package edu.cs112.guitar_hero;”
- Change SAMPLE_RATE (44,100) in GuitarString.java to “GuitarHero.SAMPLE_RATE”

Bonus Assignment:

The portions of this assignment are organized from the top of GuitarHero.java to the bottom. If you proceed in that order, your life will be much easier.

PART 1: Initializing GuitarString Objects (1 Point)

Take a look at line 58 of the code. It asks you to initialize a set of GuitarString objects that represent individual frets for each string on the guitar. Make sure that each fret is offset properly from the corresponding first fret note represented in the NOTE_FREQUENCIES array. Also, ensure that each object is placed in the appropriate slot of the mStrings array based on its string and fret position.

PART 2: MotionEvents (3 Points)

Take a look at line 71 of the code. This is the place where we handle touch events that are created by the user. The method signature has already been written for you, and each event is represented by the parameter “event”. The parameter “event” is of type MotionEvent, documentation about which can be found at http://developer.android.com/reference/android/view/MotionEvent.html.

Your job is to take that event, determine whether or not it was a “press down” (as opposed to “press up” or “move”), and if so convert the X,Y location of it into a plucking of a single string/fret. Your method should do nothing besides create the actual plucking (i.e. it should not sample the frequencies to determine the digital output).
**PART 3: Calculating Frequency (1 Point)**

Take a look at line 96. Using summation, as in the original GuitarHero, determine the audio frequency that should be output and store it in the “sample” variable. In addition, each string should be advanced a single step in the simulation as was done in the original GuitarHero assignment. This should be no more than six lines.

**PART 4: Masking and Converting Frequency (3 Points)**

The frequency we just output in the “sample” variable is of type double, which is 32-bits long. Bit stands for “binary digit” and can be either a 0 or a 1. For example, 15 written as a 32-bit binary number is 00000000000000000000000000001111. Note the 2 at the end to denote that it is in binary.

Take a look at line 102. For writing out digital audio, we don't care about the top 16 bits of the double so the first step we take is to cast the double to a new variable of type short. Short is a 16-bit data type that we typically use to represent small integers. In this case, we use it to represent a small double. Make sure that before casting you clip the frequency to be between -1 and 1.

Next, we have to store the 16-bit short as two 8-bit byte variables inside a byte array, mOutputFrequencies, at the location given by mOutputIndex. We add the bottom 8-bits first by casting our short variable to be of type byte. Once we have done that, we increment the mOutputIndex. Next, we store the high-order 8 bits of the short variable by using a bit-shift operator (>>>) and shifting it to the right 8 bits. The result of this is cast to a byte and added to mOutputFrequencies at position mOutputIndex. Finally, mOutputIndex is incremented.

**PART 5: Utilizing the AudioTrack API (2 Points)**

Lastly, we want to delve a little deeper into how to use the Android API. In order to do this, we will modify the onResume() and onPause() methods (on lines 139 and 149, respectively) to make calls on the audioTrack member object instantiated at line 33.

The onResume() and onPause() methods in an Android application are called when (not surprisingly) the Android application resumes from a suspended state or is put into a suspended state. So, in these scenarios, we will want to play and pause the audio track (i.e. register and unregister the audio device for producing output). Take a look at the AudioTrack API found at http://developer.android.com/reference/android/media/AudioTrack.html. Replace the comments in those two methods with the appropriate calls on the audioTrack object for resuming and suspending digital audio output.

**Concluding Assignment**

If you press the green button now, the code should launch the simulator and go off without a hitch. When you touch different portions of the screen, audio should play. If it does not, please ask for assistance or revisit the steps above to determine where your error occurred.

Note that you may hear the sound quality of the digital audio on Android is not great. This is due to the fact that we lowered the sampling rate so drastically that the resultant quality is not good. In order to improve this, we could have you increase the sampling rate. However, due to software issues, the digital audio output in Android is so bad that the resultant latency from a high sampling rate would make the app unusable. If you find ways to improve the quality without sacrificing latency, please let us know. Otherwise, do not be upset that your application is imperfect.

**Run on an Actual Device**

Lastly, for those who are interested we have access to actual Android hardware that you can connect to your computer and load with the GuitarHero application (Google: “Android USB Debugging”). Please make sure you sign up for access to these devices well in advance as there will be only a short period of time to play with them.
Alternative Ideas

Pair programming is not allowed on this assignment. You must work on your own. The maximum number of points you can get on the bonus assignment is 10 points.

Creating the Project:
File >> New >> Android Project
Enter Project Name
Select “Create New Project In Workspace”
Click “Next >”, then select “Android 2.1 (API Level 7)”
Click “Next >”, then replace “your.package.namespace” with “edu.cs112.<your_proj_name>”
Click “Finish”
Click “Run <your_proj_name>” (Green arrow at top of screen)
An application should pop up in the emulator that says, “HelloWorld...”

Alternative Bonus Assignment:
If you are uninterested in porting assignment 7 to Android, there is still a way for you to explore the Android platform and get bonus points. However, this assignment will be far more open-ended and therefore the grading will be much less predictable.

Android provides an open-source mobile platform that allows third-party developers to write Java applications that can run on millions of mobile phones. Your task in this assignment will be to combine what you learned in class with the Android platform to do “something interesting”. Examples of interesting things include:

- (Approx. 2 pts.) Write a summary of the Android platform, what happens in the lifecycle of an application, how the Dalvik JVM approximately operates, and how to interface with this new API and programming paradigm
- (Approx. 5 – 10 pts.) Port a problem set that we did in class (except GuitarHero) onto the Android platform. Your code should include a coherent layout design and working application functionality
- (Approx. 7 - 10 pts.) Create an entirely new Android application that does... whatever you want! Obviously, since this is open-ended 7 points will not be rewarded for “Hello, World!” applications or things copied online from tutorials. Anything copied or trivial will receive 0 points. In order to receive the full 7-10 points a student will show:
  ⊲ Good design
  ⊲ Complex code that accomplishes a non-trivial task
  ⊲ Excellent commenting and explanation of the program's functionality
  ⊲ Incorporation of one or more Android-specific APIs