# Jam Sesh: Final Report

*Music to Your Ears, From You*


## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>2</td>
</tr>
<tr>
<td>Prior Work</td>
<td>2</td>
</tr>
<tr>
<td>APIs:</td>
<td>3</td>
</tr>
<tr>
<td>Goals</td>
<td>3</td>
</tr>
<tr>
<td>Application:</td>
<td>4</td>
</tr>
<tr>
<td>Stephen, Ben, Edward</td>
<td>4</td>
</tr>
<tr>
<td>Analysis</td>
<td>4</td>
</tr>
<tr>
<td>James, Thomas, Stephen</td>
<td>4</td>
</tr>
<tr>
<td>Synthesis</td>
<td>4</td>
</tr>
<tr>
<td>James, Thomas, Edward, Ben</td>
<td>4</td>
</tr>
<tr>
<td>Summary</td>
<td>5</td>
</tr>
<tr>
<td>References</td>
<td>6</td>
</tr>
</tbody>
</table>
Overview

The JamSesh project has been aimed at providing the user with real-time synthesised accompaniment to freestyled music to bridge the current void between music and technological aid. Our trajectory, however, has changed drastically since the inception of our concept in September. The final version of our efforts has been scaled back in relation to many of our lofty goals for the project. That said, the end project will be able to generate to a chord progression for the user based upon only a few parameters so that the individual can play along.

Prior Work

Auto-generated accompaniment programs have been made before, such as “Midi Utility” created by KH Midi Music Ltd. This program differs from our project goal, however, in that it requires the user to input the music as a midi file and select a number of options — the musical element is preloaded, not instantaneous. Our goal is to have a program respond in real time to live music, requiring only a few user options, including instrument selection. Unfortunately, we will not be able to use the prior work done in “Midi Utility,” as its code is not open source. Another important piece of prior work regarding real-time improvisation has been done by Al Biles of RIT; his project, GenJam, is based on a genetic algorithm that can learn to improvise jazz based upon the input of the user. This project is intriguing because it runs in real-time just as we hope to do, but our goal is to do so without a genetic algorithm and instead aim purely for improvisation. Another element of Genjam that is also seen in other works is the ability of the program to play a more melodic line back-and-forth “conversation” with the user. Our aim for this project is to empower the user and instead lean toward a user-centric application.
APIs:

To achieve the final product we originally decided to make use of three key API’s, namely TarsosDSP, TheEchoNest, and JFugue. Tarsos is designed to find the Pitch of a microphone in real time. Our application of this API is a core part of the analysis work, as having the users pitch is vital to synthesising accompaniment for them. TheEchoNest is an API that can analyse a .wav file for key, pitch, tempo and beat, which does not allow for real-time switches for synthesis, but it does allow us to change slightly after-the-fact for large necessary changes on the users part such as a key shift. For musical synthesis, we made use of the JFugue API. JFugue allows for easy synthesis and uses “music strings,” which are simple strings that contain notes and chords in their order to be played. JFugue has a play() method that takes these strings in as a parameter, parses them, and then plays them in one of the 128 instruments it has at its disposal.

After finding that pulling useful musical information from such raw pitch data posed a greater number of challenges than we originally anticipated, we decided to drop the idea of gathering real-time note detection using TarsosDSP. Although we were able to recognize pitches and their durations (the essential components needed to create a MIDI file), the process ran far too slowly and only worked if each note was sustained for an unreasonably long period of time. Because of its inability to detect notes in a more realistic environment, where pitches could be rapidly changing, we made the decision to drop TarsosDSP as well as TheEchoNest, which we found offered us no real use if we just had the user enter his or her desired musical key.

Goals

The goals of Jam Sesh are simple to understand but complex to achieve. Overall, the originally planned framework of our project centered around processing sound inputted by a user, analyzing its properties, and synthesizing supporting harmonies and background music for the soloist. Some high-level goals for this project included: a real-time element, allowing the selection of different instruments, synthesizing complex music structures. While we are happy to report some of our project goals have been achieved, we acknowledge that many of our initial goals were an overreach and would require much longer than a single semester — in addition to us gaining further knowledge in the subject area — to achieve them.

After a large amount of time spent researching musical analysis and synthesis, and after a number of test programs and we made the difficult but necessary decision to cut some features and redefine our goals. Our main goal now is to generate a unique chord progression taking only the desired key and BPM as parameters from the user.
Application:
Stephen, Ben, Edward
Application is responsible for the UI and passing information gathered from the user to the synthesis element. The application would have a “start” button and a “stop” button that act as a power switch of synthesis element. The application would also have a set of radio buttons for key signature, a set of radio buttons for time signature, and a text field for beats per minute (bpm). Values gathered from those objects will be passed to the synthesis class. There are also two JLabels in application that display current chord and next chord. The chord information is gathered from the synthesis class.

Analysis
James, Thomas, Stephen
Analysis will then receive pitch information from the API and send interpreted data to the synthesis part. Synthesis will decide the pitch for each instrument to play, export the pitches into a midi file, and send the file back to application, which has a midi player.

Synthesis
James, Thomas, Edward, Ben
Since we have abandoned the use of an analysis element, the synthesis element changed a lot. The synthesis element now is receiving the key signature, time signature and tempo in beats per minute (bpm) from the application element. The synthesis element then call a recursive method to recursively play chords. This recursive method would call a random chord progression generator class to get the chord progression. This generator would return a MusicString that contains chords that sound reasonable and different every time. The whole synthesis element runs in a separate thread because, otherwise, the application would stop responding when the synthesis element starts running.
Summary

Overall, working on Jam Sesh has been a memorable experience for each group member. Combining the successes achieved along with the adversity we have faced, we have learned quite a bit about the collision of music and technology. With all of the high-reaching goals we put together initially, we have learned the valuable lesson about sunk costs — that, after putting effort into a venture for an extended period of time, it is sometimes necessary to make the difficult decision to let go of a piece of the project so that the remainder of the project could still be possible. The unique experience of working on such a long-term project has allowed us to learn about working in a team on such a large scale in addition to what we learned about music and technology.
References


