**Musical Rendering Libraries** Gian Biondi  
  
The application we are developing needs to not only store sheet music but also display it.  There are many applications and even formats geared toward musical engraving.  Musical engraving is the art of drawing musical notation usually for mechanical reproduction.  Most tools render music in a static format, such as in PDF format.  The iOS application we are developing is meant to render scores dynamically, allowing for features like zooming and scrolling, so a static image isn’t enough.  A lot of applications exist that display music on a screen as we are seeking to do.  Most of them utilize a custom solution.  As a result there isn’t much in terms of selection for rendering libraries for musical score.  This paper will discuss some rendering libraries.  
  
**JMSL**  
 JMSL, or the Java Musical Specification Language, is a Java API for music composition and performance.  JMSL is used for making standalone applications and web applets.  JMSL is based on HMSL, or Hierarchical Music Specification Language. The notion of hierarchies is a key one in JMSL.  A hierarchy is simply a collection of parent/child relationships. A song form provides a simple example, where a parent called mySong might have four children: verse1, chorus, verse2, and chorus. In JMSL, mySong stored as a SequentialCollection.  Through the power of Java, JMSL has many features.  For our purposes it may be too much.  While it does render music visually, it has many more features geared for creating and playing music as well as for analyzing music and sounds.  Unfortunately it is not free.  The API costs $55 for students and $120 for developers. As such, cost may be a disqualifying factor along with the sheer size of the libary. Below is Java code demonstrating some JMSL functionality. The code here creates a mixer object and loads a musical score into it for editing, display or analysis.

JMSLMixerContainer mixer;  
   Instrument instrument;  
  
   MusicJob myMusicJob;  
  
   public void init() {  
       JMSL.setIsApplet(true);  
   }  
  
   public void start() {  
       synchronized (JMSL.class) {  
           initJMSL();  
           initMusicDevices();  
           buildMixer();  
           buildInstrument();  
           buildMusicJob();  
           launchMusicJob();  
       }  
   }  
  
   private void initJMSL() {  
       JMSL.scheduler = new EventScheduler();  
       JMSL.scheduler.start();  
       JMSL.clock.setAdvance(0.1);  
   }  
  
   private void initMusicDevices() {  
       JSynMusicDevice.instance().open();  
   }  
  
   private void buildMixer() {  
       mixer = new JMSLMixerContainer();  
       mixer.start();  
   }  


Figure 1: JMSL Example Score

**VexFlow**  
 VexFlow is an engraving engine for musical notation  designed for HTML5 and Javascript.  It is designed to be used as a back-end rendering engine for mainly online tools.  It requires no external libraries or dependencies although it can be used with other JavaScript libraries such as Raphael for SVG support and jQuery or easy DOM Manipulation.  While only in the “pre-pre-pre-alpha” stage, the library is very well documented including a step-by-step guide to getting started.  The library supports hardcoding of musical scores and since it is built in JavaScript it can use Javascrip’s built-in XML parser to read MusicXML and display any score loaded into it.  The problem with this technique is that this can be rendered only by an HTML5 compatible web browser.  Fortunately, in iOS it is possible to use a browser frame in a native app.  That browser frame acts as a small web browser embedded in an iOS application.  The browser uses Safari and its Webkit rendering engine, so it is compatible with the HTML5 specification. The Webkit rendering engine is extremely powerful but the browser-frame version used in native iOS apps suffers from reduced performance. This is why competing browsers like Google Chrome for iOS and Mozilla Firefox do not perform as well as on other platforms. This issue requires more investigation and testing to determine if it will hinder the use of VexFlow. Below is a simple example of using VexFlow to create a stave and populate it with notes.   
   
var canvas = $("div.two div.a canvas")[0];  
 var renderer = new Vex.Flow.Renderer(canvas,  
   Vex.Flow.Renderer.Backends.CANVAS);  
  
 var ctx = renderer.getContext();  
 var stave = new Vex.Flow.Stave(10, 0, 500);  
 stave.addClef("treble").setContext(ctx).draw();  
  
 // Create the notes  
 var notes = [  
   // A quarter-note C.  
   new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "q" }),  
  
   // A quarter-note D.  
   new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "q" }),  
  
   // A quarter-note rest. Note that the key (b/4) specifies the vertical  
   // position of the rest.  
   new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "qr" }),  
  
   // A C-Major chord.  
   new Vex.Flow.StaveNote({ keys: ["c/4", "e/4", "g/4"], duration: "q" })  
 ];  
  
 // Create a voice in 4/4  
 var voice = new Vex.Flow.Voice({  
   num\_beats: 4,  
   beat\_value: 4,  
   resolution: Vex.Flow.RESOLUTION  
 });  
  
 // Add notes to voice  
 voice.addTickables(notes);  
  
 // Format and justify the notes to 500 pixels  
 var formatter = new Vex.Flow.Formatter().  
   joinVoices([voice]).format([voice], 500);  
  
 // Render voice  
 voice.draw(ctx, stave);

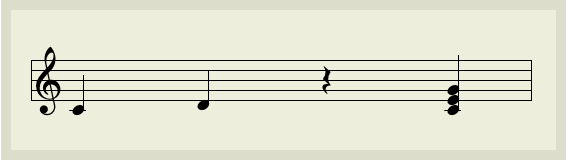


Figure 2. VexFlow Example

**DixShtix**

Another Library for rendering notation is DixShtix.  DixShtix is a Java music library for rendering notation as well as playing sound and MIDI.  It was designed with piano, harp and guitar players in mind.  It fully supports MIDI and RMF, NIFF, score editing and displaying, exporting as audio and creating instruments. It is unclear if MusicXML is fully supported or not.  DixShtix is fully open source but not very well documented with sample code or tutorials.  It seems very promising for use with our application, but Java isn’t terribly compatible with iOS.  iOS applications are written in objective-C and do not easily integrate with Java applications.  So it is a good source of information but probably not the best to use directly.

There are not too many choices with respect to rendering libraries for music notation. It seems that most of the applications which display notation have their own proprietary methods for doing so. In this paper I have listed some of the libraries that are available and that work well for quickly engraving music. Of them, I believe that the best and easiest method for use in an iOS application is VexFlow, because it is lightweight, easy to use, well documented and fully compatible with the platform. Also, due to Vexflow’s HTML5 implementation it will be easier to port to other platforms when our application transcends iOS to be ported to Android and other platforms.

**Bibliography**  
  
Dixshtix.  <http://dixshtix-midi.sourceforge.net/>  
  
Vexflow.  <http://vexflow.com/docs/tutorial.html>  
  
JMSL.  <http://www.algomusic.com/jmsl/>

WebKit. http://www.webkit.org/