SSScrollView scrolling MusicXML score display

***SSScrollView* provides a scrolling view of a MusicXML file using SeeScoreLib, with pinch-zoom as in SeeScore.**

SSScrollView is derived from *UIScrollView* and uses *SSSystemView*.

SSScrollView is compatible with iOS 5.0 and later.

Using SSScrollView in your project

- In your project add SSScrollView.m/.h and SSSystemView.m/.h.
- Add SeeScoreLib.framework
- Add CoreText and QuartzCore frameworks
- In "Build Settings", under "C++ Standard Library" select libc++ , and under "C++ language dialect " select C++11 (This is the default)

In your storyboard you should place a *UIView* and convert its type to *SSScrollView*. You can then hook up the *containedView* to the *UIView* inside the *SSScrollView* and *scrollDelegate* to your bar indicator.

The ViewController should call SSScrollView didRotate in didRotateFromInterfaceOrientation

The bar indicator should have a *SSBarControlProtocol* delegate which can be connected to the *SSScrollView*

To load a MusicXML file (.xml or .mxl) you should call

[SSScore scoreWithXMLFile:options:error]

or

[SSScore scoreWithXMLData:options:error]

with the MusicXML file or data and it returns an SSScore object. This can be passed to [*SSScrollView setupScore:]*.
NB *SSSystemView* is used by *SSScrollView* and should not be placed in Interface Builder

The interface to SeeScoreLib.framework which handles layout and rendering of MusicXML is in SeeScoreLib/SeeScoreLib.h

You will need
#import <SeeScoreLib/SeeScoreLib.h>
at the top of each file where you use the framework interface.

The main interface is in SSScrollView.h

How it works

SSScrollView organises layout and active placement of *SSSystemView* instances inside a scrolling view. *SSSystemView* draws a single complete system of music (ie a full or part set of parts for a range of bars).
Apple requires us to display only the visible part of a scrolling view in order to conserve memory, so we only keep enough drawn *SSSystemViews* to fill the screen at any time. When the *SSScrollView* is scrolled, disappearing *SSSystemView* instances are removed, redrawn in a background thread to ensure the scroll is not interrupted and replaced where they are due to appear. There is also a separate background thread which organises layout of all the systems, ie

automatic allocation of bars to systems and positioning of items within the system. As systems are laid out the contentSize of the scroll view is updated. Progress of this thread can be followed by a BarControl using the SSBarControlProtocol and SSUpdateScrollProtocol. The *SSBarControl* provides an example. Use of threading allows us to display the start of a long score immediately while layout continues perhaps for many seconds

Licensing terms