SeeScore Development for iOS
Hints and tips

The developer will need to understand C and Objective-C.

Important files:

SSSampleViewController.m

This contains most of the app logic, and is what you should concentrate on understanding.

• At the top is **`kPianoSamplesInfo.`** This defines parameters of the set of sample files for the instrument, including instrument name, file name pattern and various timings. Other samples info should be added here if you are using other instruments

• Event handlers are called by the synth when it is playing:

BarChangeHandler - notified at each new bar when playing. You may or may not need it

BeatHandler - notified on each beat. Used for count-in

EndHandler - notified when play reaches the end. This can change the pause button into a play button, and any other UI that needs to change when play is finished

NoteHandler - notified on the start and end of each note. You can use this to move your note cursor.

• Audio Session handling (**`setupAudioSession`** etc.) can be left unchanged. This is required by Apple

• **loadFile** contains all the logic for loading a file (.xml or .mxl)

Whenever you relayout the score you need to call abortBackgroundProcessing on SSScrollView to ensure it finishes any work it is doing in the background, and await the result by doing the following processing in the completion handler. You can see this at the top of loadFile.

showingParts is an array containing YES and NO for each part - YES if it is displayed, NO if not.

Note that some scores contain tempo indication (eg metronome marking) and some do not. This affects the tempo control which can either supply a default tempo (eg 80 beats-per-minute), or a scaling for the score-defined tempo. The tempo slider needs to adjust its scale accordingly. You use score.scoreHasDefinedTempo to know if the tempo is defined at the start of the score, and adjust the tempo control accordingly.

• **tap** moves the SSScrollView cursor to the tapped bar. There is also optional code to display the xml for the tapped item, and to colour the tapped item

• **stopPlaying** stops the synth playing.

• **longPress** shows how to change the displayed parts

• **transpose** shows how transpose (if you have the transpose licence) and also how to substitute clefs in a part (eg all treble clefs into alto clefs to play on a viola).

• **noteXPos** returns the x-position of a note (for the note cursor)

• **moveNoteCursor** is called from the note event handler and moves the SSScrollView cursor to the note

• **colourPDNotes** shows how to colour notes as they are played, also called from the note event handler

• **displayNotes** prints the contents of the PlayData in the console, and can be used for debugging

• **play** is called when the play button is tapped and starts the synth playing the score. It creates the synth, passing in the SSSyControls implementation (self, see below) sets up the sampled instruments (eg piano) and metronome, creates the PlayData passing in the SSUTempo implementation (self, see below), sets up the event handlers, synth setup with the playdata and starts the synth (after a short delay to allow setup to complete).

• **tempoChanged** is called when the tempo slider is moved. It updates the tempo readout beside the slider, and calls updateTempoAt on the synth. NB The synth reads the tempo directly from the slider using protocol SSUTempo defined at the bottom of this file

• **didRotateFromInterfaceOrientation** forces a relayout when the device is turned

• **protocol SSSyControls** link from UI controls to synth
partEnabled returns true if the part should be played
partInstrument returns the id of the instrument to be used to play the part
partVolume returns the volume to play the part - range [0..1.0]
metronomeEnabled returns true if the metronome should be played
metronomeInstrument returns the id of the metronome instrument
metronomeVolume returns the volume of the metronome - range [0..1.0]
partStaffEnabled (optional) can be used to play just the upper or lower staff of a 2-staff part (eg a piano part)

• **protocol SSUTempo** link from the UI tempo control to the synth
bpm returns the beats-per-minute value used if the score has no tempo markings
tempoScaling returns the scaling used for the tempo values defined in the score

SSScrollView/SSSystemView
These display the score in a UIScrollView, and will probably not need to be changed. There is some complication here which should not be changed without checking with me.

SSBarControl
Displays the 'ruler'-type control with cursor. Can be used or not as you wish

SettingsViewController
Displays settings. You won't need this.

InfoViewController
Displays header info. You won't need this

evaluation_key.c
You will replace this with your key.