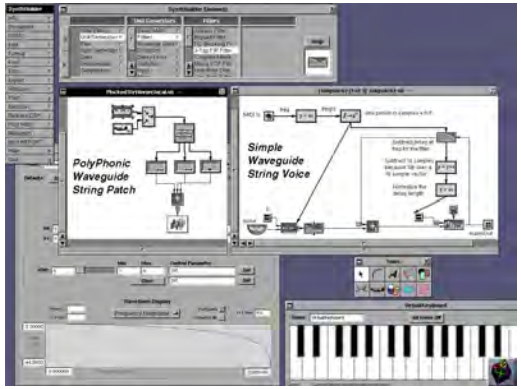
Use your JUCE card to escape UI layout jail

Nick Porcaro (Chief Scientist)
moForte Inc.

Background

- We've been building music systems in Objective C for many years dating back to the NeXT Music Kit (1988) and SynthBuilder (1994)
- moForte's products presently use Faust/C++ for DSP and UI is done with Objective C/UIKit
- Quick demo of GeoShred (written in Objective C).



Transition to JUCE

- About a year ago, we started doing prototyping using JUCE.
- We've also been experimenting with Faust to JUCE UIs.
- For the past several months, we have been working on standalone effect app in JUCE, moDhwani.
- The plan is to eventually convert our full code base to JUCE.

moDhwani

- moForte + Dhwani:
- One that produces beautiful musical sounds
- Name of a famous Bollywood musical.
- A girl's name meaning sound
- Music festival in Singapore
- A girl who wants to do something and does it
- But also something more rude, look it up ;-)

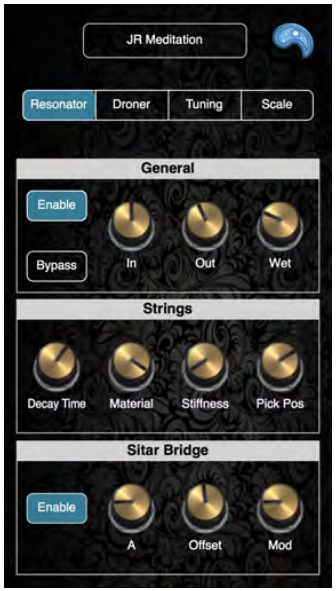
Objective C/UIKit

- Powerful, elegant, but not cross platform.
- Extensive use of Interface Builder for UI
- Extensive use @selector, used to map configuration XMLs to method calls.
- iOS autolayout is complex, hard to convert existing interfaces.

“Stompbox” and "Panel" Layouts.

- Built using the Interface Builder + Objective C code for complex UI behavior. Sliders, knobs, switches can send/respond to stacks of controls.
- In order to effectively reproduce these stompboxes/panels in JUCE, we use Grid layout in code, instead of manually laying out proportional rectangles or an XML format.
- Complex control behavior was ported from Objective C to C++/ JUCE.

Objective C/UIKit



C++/JUCE

Grid Layout


Example3
Resonator
Droner
Scale
Tuning
Vibrato
Vib Depth
Vib Rate
Whammy
Tuning
Fine
Coarse
Detune
Reverb
Enable
Level
Damping
Room Size

JUCE Grid Layout

- Flex/Box grid based on CSS standards
- Works like a responsive website.
- Not perfect, but special cases can be encapsulated.
- Better to use code as opposed to a custom XML file format because special case handling creates redundant or complex XML
- Grid was sufficient because the UI only needs to scale.

Overall layout of Tuning section

```
void resized() override {
  Grid grid;
  grid.templateColumns = { 1_fr };
  grid.templateRows = { 1_fr, 1_fr, 1_fr};
  grid.setGap (2_px);

  grid.items.addArray ({
    GridItem (vibrato).withRow ({ 1 }).withColumn ({ 1 }),
    GridItem (tuning).withRow ({ 2 }).withColumn ({ 1 }),
    GridItem (reverb).withRow ({ 3 }).withColumn ({ 1 })
  });

  grid.performLayout (getLocalBounds());
}
```

Vibrato

```
void resized() override {
  Grid grid;
  grid.templateColumns = { 1_fr };
  grid.templateRows = { 1_fr, 1_fr, 1_fr};
  grid.setGap (2_px);

  grid.items.addArray ({
    GridItem (vibrato).withRow ({ 1 }).withColumn ({ 1 }),
    GridItem (tuning).withRow ({ 2 }).withColumn ({ 1 }),
    GridItem (reverb).withRow ({ 3 }).withColumn ({ 1 })
  });

  grid.performLayout (getLocalBounds());
}
```

Tuning

```
void resized() override
{
  Grid grid;
  grid.templateColumns = { 12_fr, 25_fr, 25_fr, 25_fr, 13_fr};
  grid.templateRows  = { 1_fr };
  grid.setGap (2_px);

  grid.items.addArray ({
    GridItem (spacer1).withRow ({ 1 }).withColumn ({ 1 }),
    GridItem (fineTuneKnob).withRow ({ 1 }).withColumn ({ 2 }),
    GridItem (coarseTuneKnob).withRow ({ 1 }).withColumn ({ 3 }),
    GridItem (detuneKnob).withRow ({ 1 }).withColumn ({ 4 }),
    GridItem (spacer2).withRow ({ 1 }).withColumn ({ 5 }),
  });

  grid.performLayout (getLocalBounds());
}
```

Reverb

```
  void resized() override {
    Grid grid;
    grid.templateColumns = { 25_fr, 25_fr, 25_fr, 25_fr};
    grid.templateRows  = { 1_fr };
    grid.setGap (2_px);

    grid.items.addArray ({
      GridItem (enableButtonC).withRow ({ 1 }).withColumn ({ 1 }),
      GridItem (levelKnob).withRow ({ 1 }).withColumn ({ 2 }),
      GridItem (dampingKnob).withRow ({ 1 }).withColumn ({ 3 }),
      GridItem (roomSizeKnob).withRow ({ 1 }).withColumn ({ 4 }),
    });
    grid.performLayout (getLocalBounds());
  }
};
```

DSP parameter handling

- Objective C @selector is emulated using a HashMap that maps a string to a static member function of a class.
- This is done at app init time, as opposed to late binding
- Decided against updating all parameters out of ValueTrees on each render callback.
- Immediate updating works well enough for us - we have hundreds of parameters.

Defining a Dispatch Table

```
void StringVerbDSP::setupFloatControllerDispatchTable()
{
  floatControllerDispatchTable.set("stringVerbEnable", stringVerbEnable);
  floatControllerDispatchTable.set("stringVerbInGain", stringVerbInGain);
  floatControllerDispatchTable.set("stringVerbBypass", stringVerbBypass);
  floatControllerDispatchTable.set("stringVerbOutGain", stringVerbOutGain);
  ...
```

Defining a Dispatch Table

```
// typedefs for dispatch table

typedef HashMap<String,floatSetter> floatControllerDispatchTable;

class StringVerbDSP: public MFDSP
{
public:
  StringVerbDSP();
  ~StringVerbDSP() {};
  void setupDispatchTable();

private:
  static void stringVerbEnable(MFFloat val);
  static void stringVerbInGain(MFFloat val);
  static void stringVerbBypass(MFFloat val);
  static void stringVerbOutGain(MFFloat val);
  static void stringVerbDryWetPan(MFFloat val);
};
```

Custom Components

- Custom knob, segmented control and slider
- Custom Scale Editor
- Real time scale update with timers.
- Control Surface Editor: Combination of drag/drop and grid layout works well.
- Somewhat simpler than UICollectionView
- Outline views useful in control surface editor design, easier than drilling down.

Side by Side Demo, Sympathetic Resonator with moDhwani, Objective C (UIKit)/C++ (JUCE)

Summary

- Use Grid for UI layout, each part of the grid can be a custom component.
- If you need a “responsive” layout, then you can use a combination of FlexBox/Grid
- It’s not hard to write custom components if you need to for your product ID.
- Can use either bitmaps, scaleable vector graphics, or direct drawing for “skinning”
- DSP parameter mapping with HashMap