Trinity College Dublin

MorphOSC- A Toolkit for Building Sound Control GUIs with Preset Interpolation

Liam O'Sullivan

Electronic & Electrical Engineering,

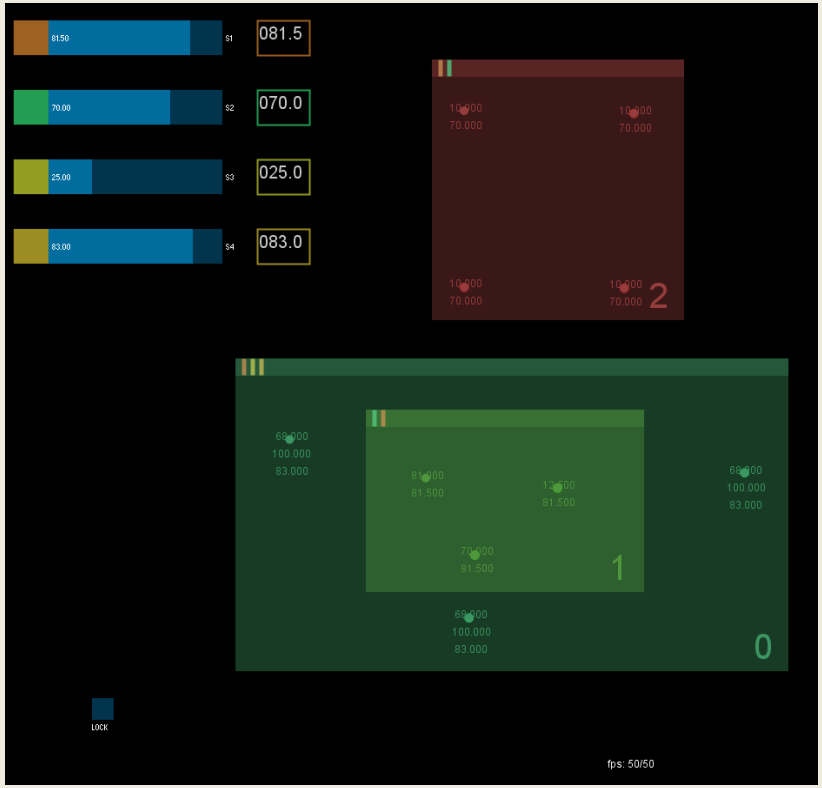
Trinity College Dublin.

Summary

- ***MorphOSC*** is a new toolkit for building graphical user interfaces for the control of sound using morphing between parameter presets.

- Processing class library:
  - Cross-platform: desktop and Android.
  - Open-source.

- Improved interaction:
  - Intuitive configuration of parameter spaces.
  - Layering metaphor for parameter subsets.



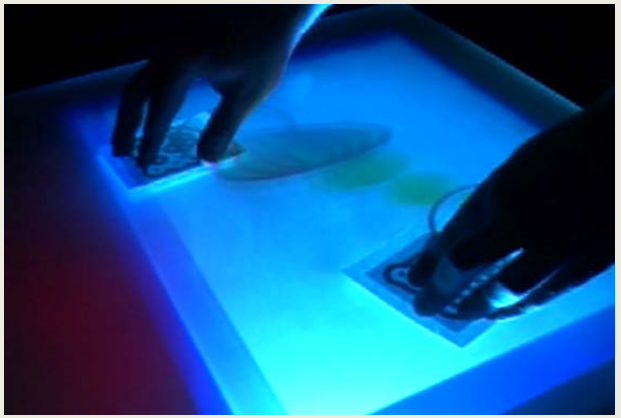
Presentation Outline

1. **Project origins, background, goals.**
2. Similar work: software controllers.
3. MorphOSC library implementation.
4. Discussion and future work.

Origins and Motivation

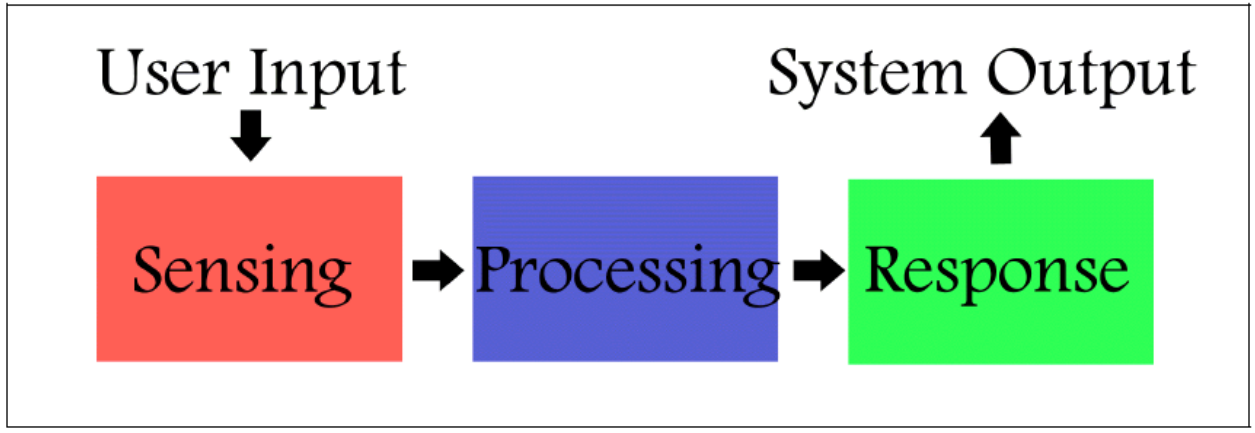
“Tailoring Multitouch Interfaces for Musical Control”

- Graphical User Interface.
- Sensor input.
- Interaction design.



Background: Interactive Music System

- Three basic stages linked by a communications protocol.
- Processing stage *maps* the controller inputs to the parameters of output.
- Mapping imparts character to an IMS (Drummond, 2009).



Simple Mapping

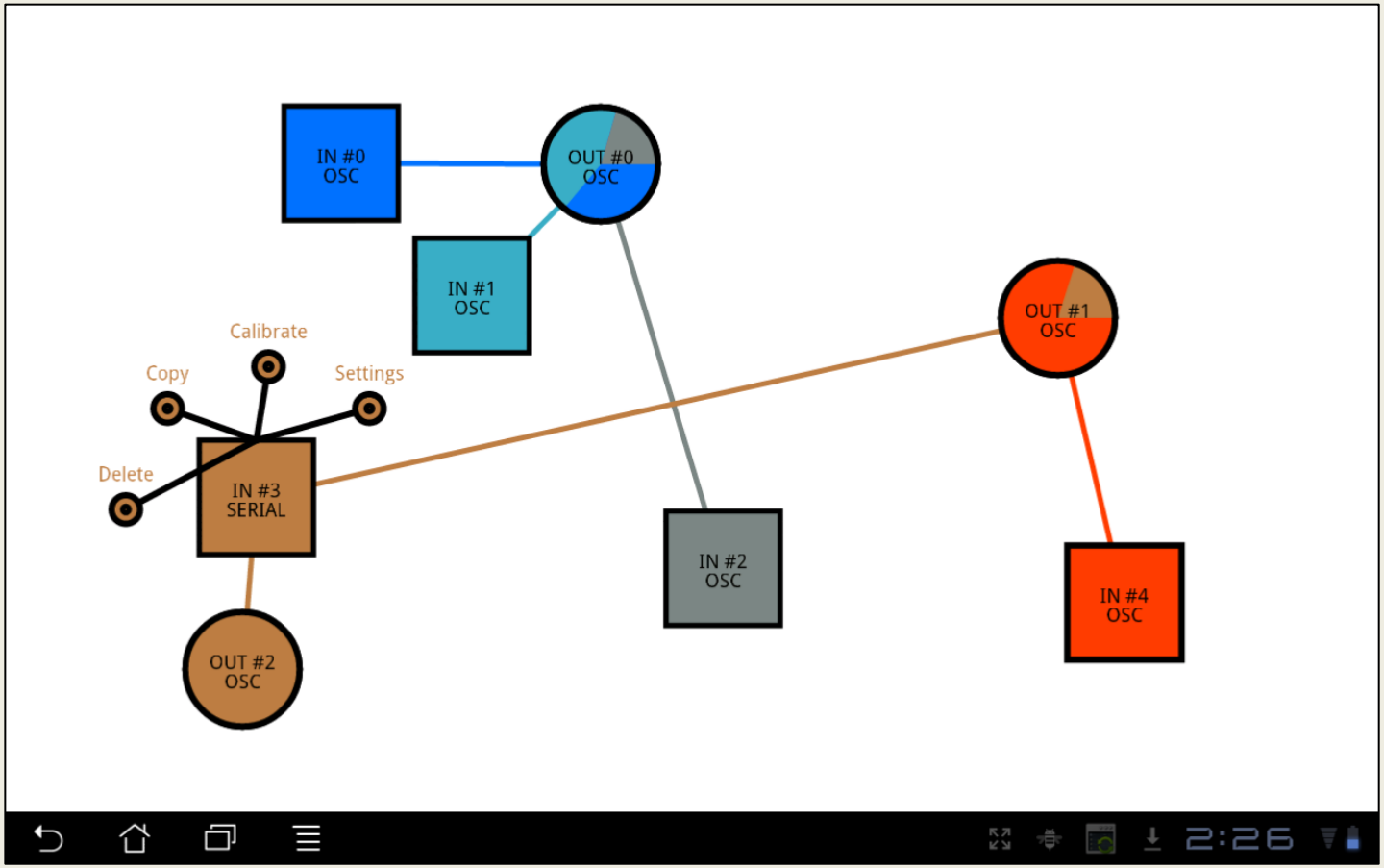
- Mapping may be a simple one-to-one arrangement.
  - Hardware synthesiser: a single control widget (e.g. slider) will control a single sound parameter.
  - But! A violin doesn't have a SOLITARY pitch control.

Complex Mappings

- Type: one-to-many, many-to-one, many-to-many.
- Trade-off between initial ease-of-use and potential for expressive play/ long term engagement.
- Subjective tests: users prefer complex mappings to simple ones, once learned (Hunt 2000).

Violin pitch = (w_{large} * finger position) + (w_{small} * bow pressure)

CrossMapper



CrossMapper on Android (O'Sullivan & Boland, 2012)

Interpolated Parameter Spaces

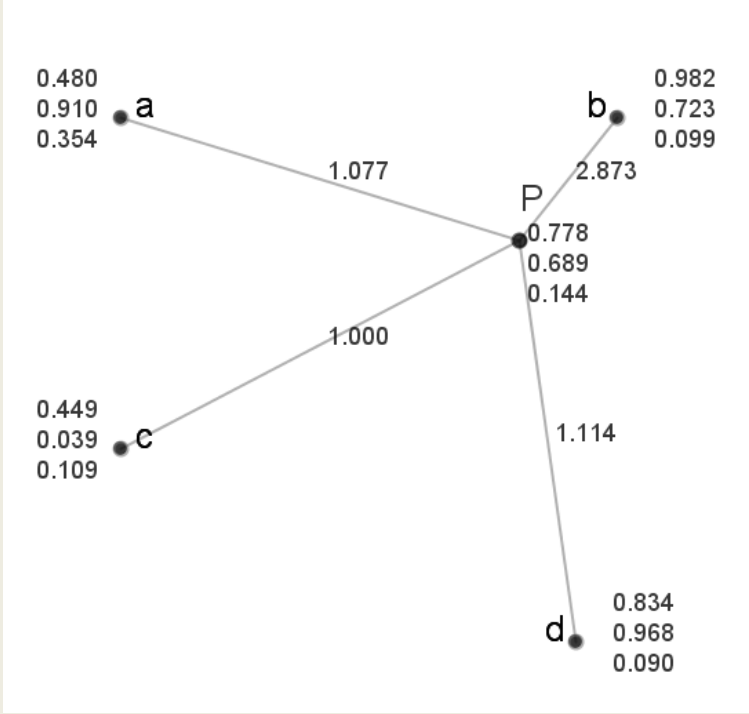
- Parameter space of output (e.g. timbre) is often greater than the available controls, necessitating a **few-to-many** mapping.

- A 2-D controller, for example, can control a high-dimensionality parameter space.
  - Anchor points represent ‘snapshots’ of parameters.
  - Can interpolate between anchors for output.
  - Sacrifices individual control of parameters for intuitiveness/ usability.
  - Suited to 2-D screen space.

Example Interpolation Method

- Shepard's Method uses simple **inverse distance weighting** to interpolate between values of a set of ordered parameters (Shepard, 1968).

$$p_i = \frac{\sum_{n=0}^{k} p_{ni} d_n^{-1}}{\sum_{n=0}^{k} d_n^{-1}}$$



Project Goals

- ‘Intuitive’ graphical interface offering dynamic configuration of complex two-to-many mappings.
- Open Sound Control formatted messaging over networks.
- Cross-platform, open-source code with portability to touch screens.

Presentation Outline

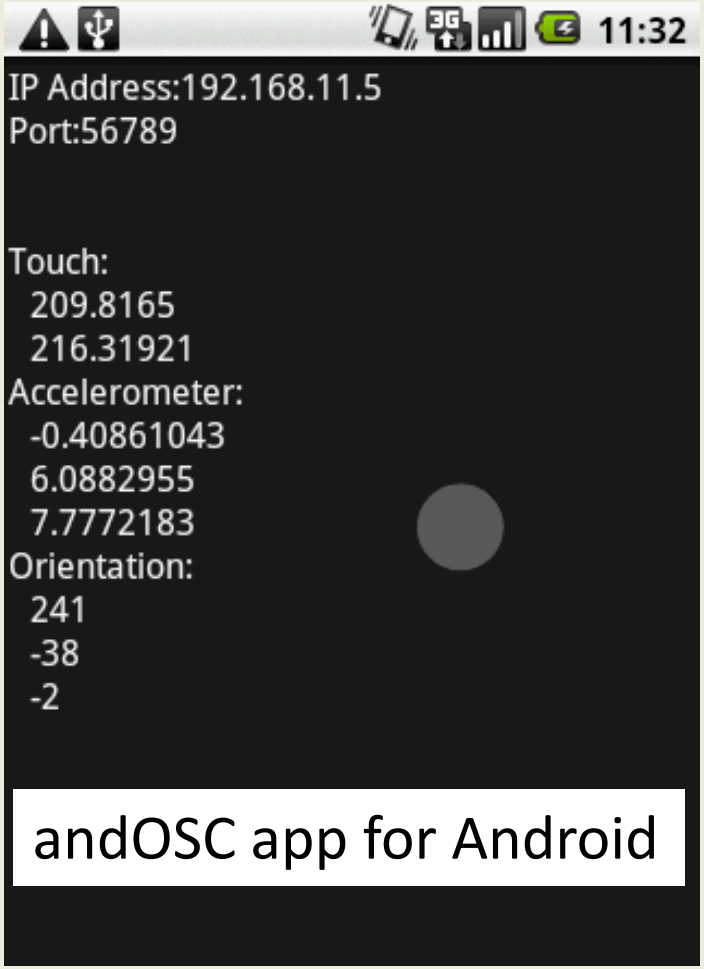
1. Project origins, background, goals.

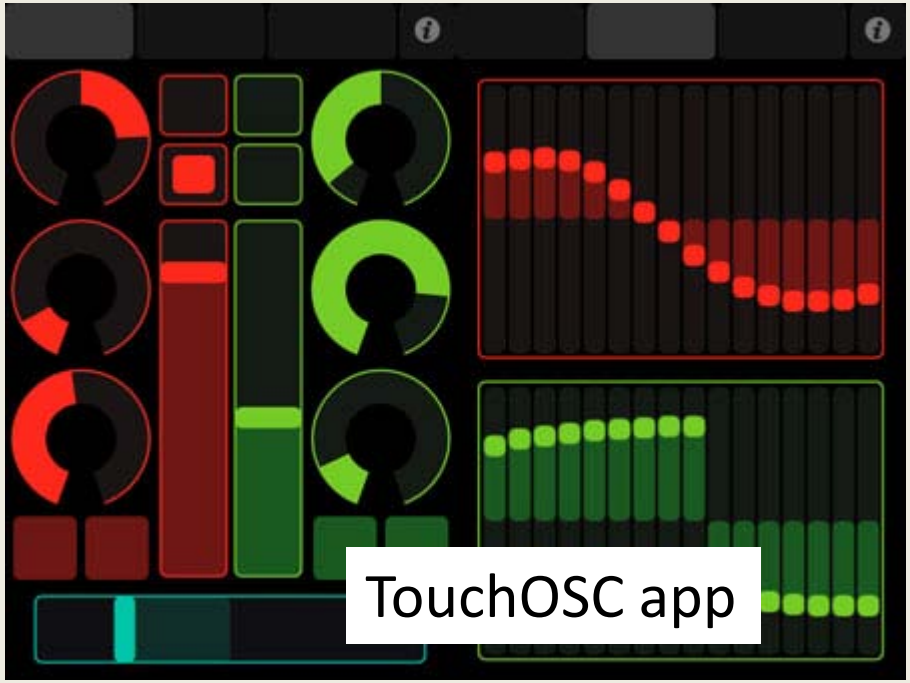
2. Similar work: software controllers.

3. MorphOSC library implementation.

4. Discussion and future work.

Similar Work: OSC Interfaces

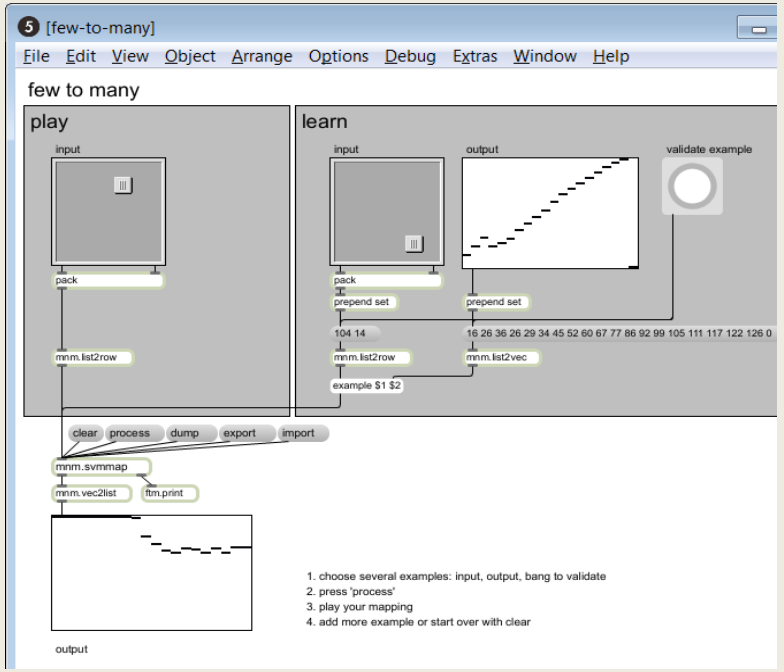




Similar Work: *MnM*

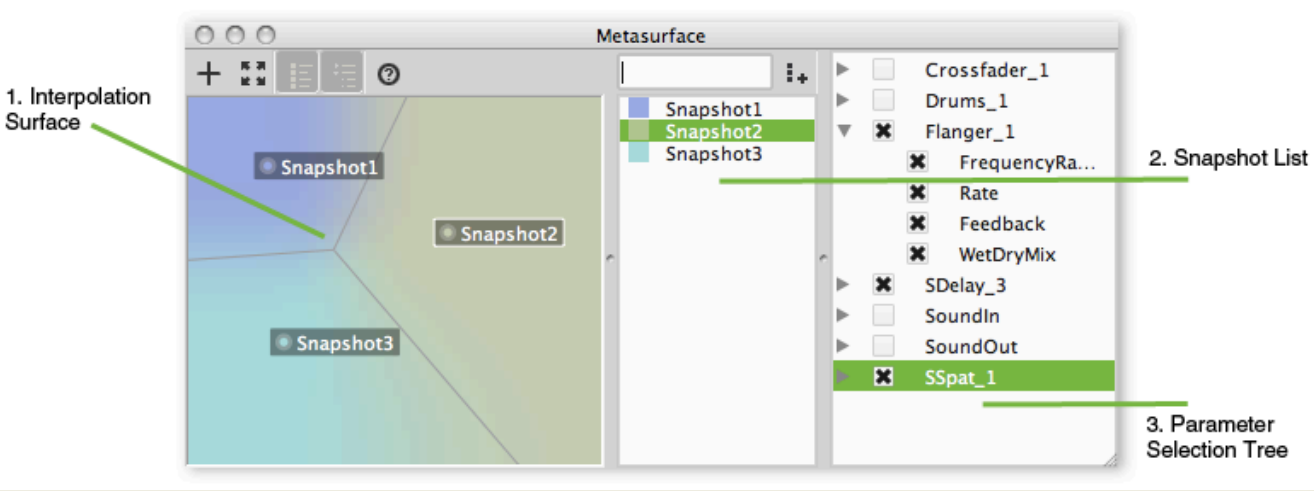
- IRCAM: MnM mapping toolbox.
  - Part of FTM external objects.
  - Requires Cycling'74 Max environment.
  - Limited to Max GUI elements.

Example Max patch using two-to-many mappings. The output parameter space is learned for a position of the X-Y controller. Moving it interpolates between the learned mappings.



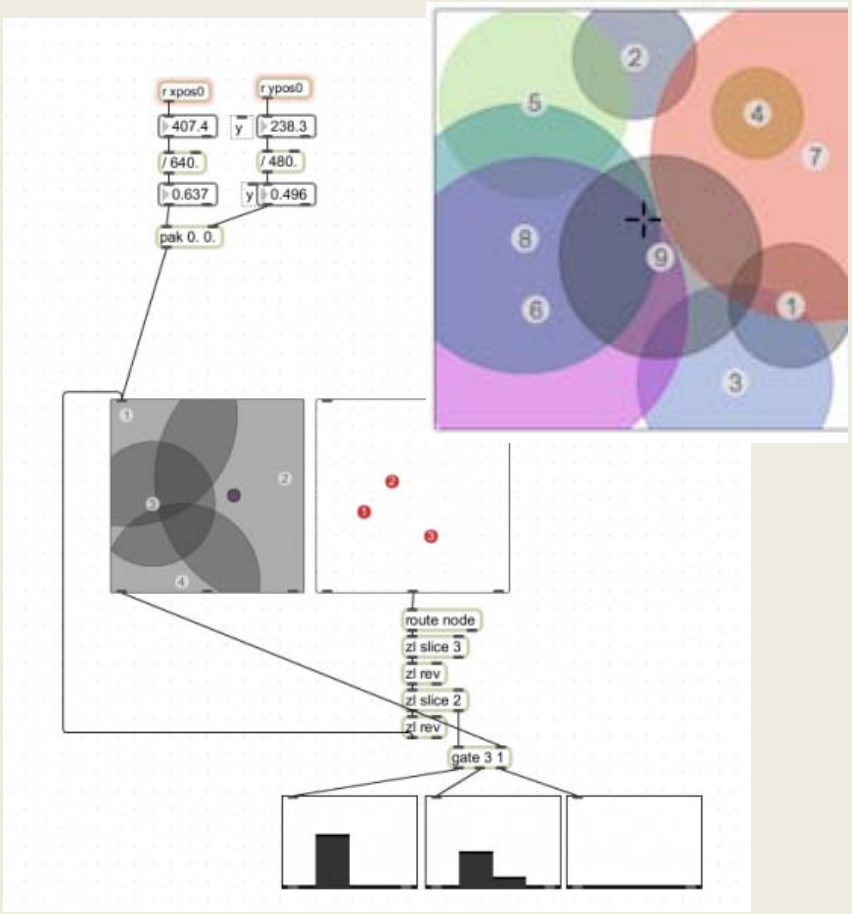
Similar Work: *MetaSurface*

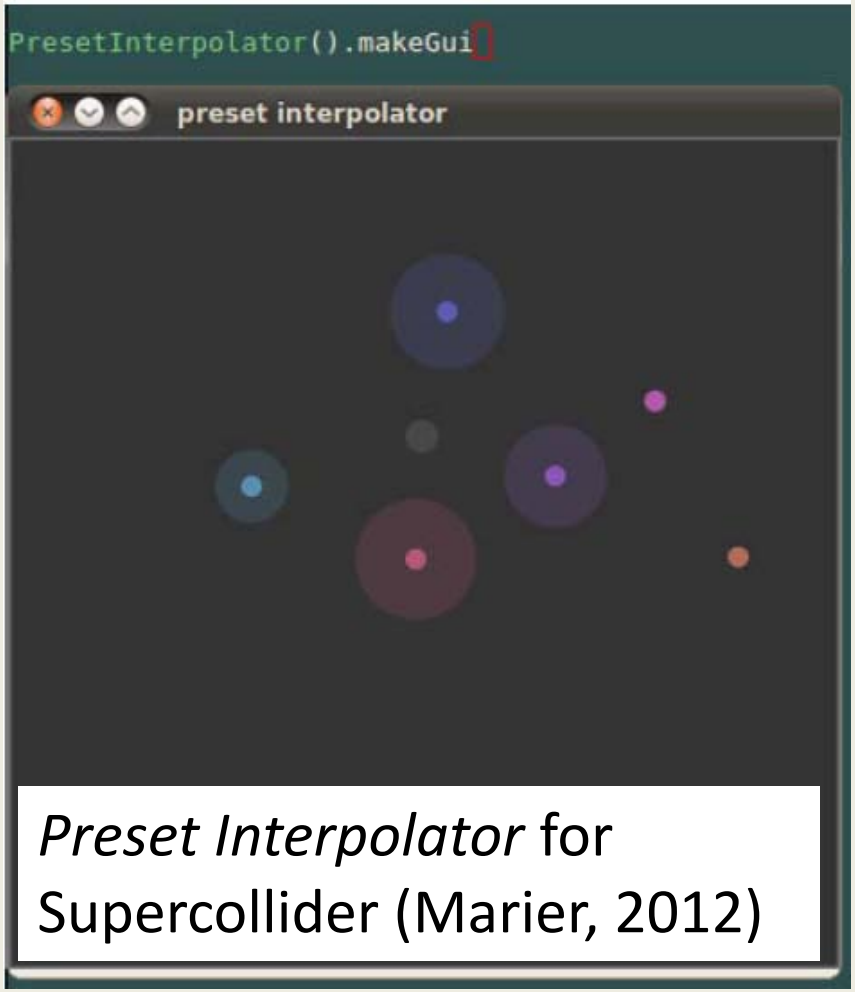
- AudioMulch MetaSurface.
  - Provides a control space for mappings using interpolation between snapshots.
  - OSC/MIDI output, control of softsynth parameters.
  - Commercial software bundle.
  - Sub-menus to include parameters in interpolation.



Similar Work: Environments

Cycling’ 74 Max 5 *nodes* object





Preset Interpolator for Supercollider (Marier, 2012)

Presentation Outline

1. Project origins, background, goals.

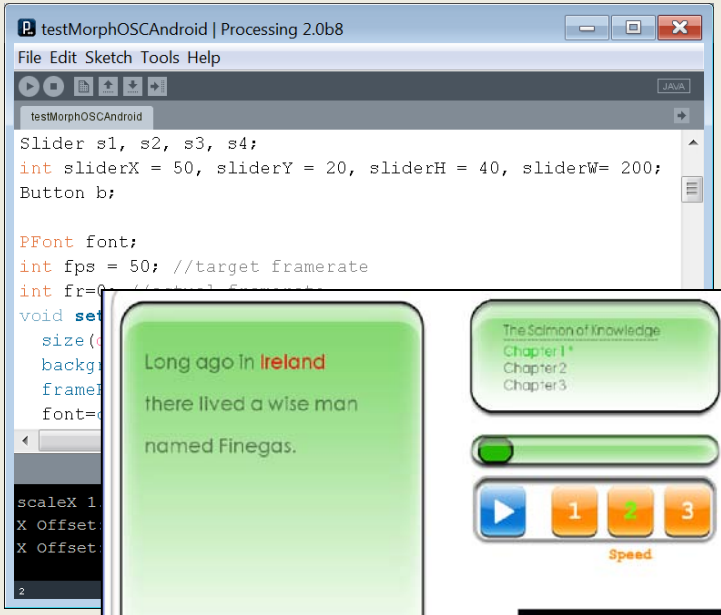
2. Similar work: software controllers.

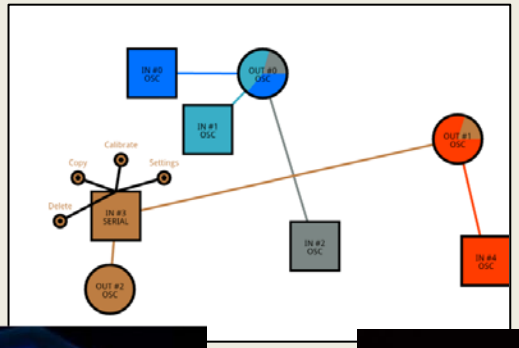
3. MorphOSC library implementation.

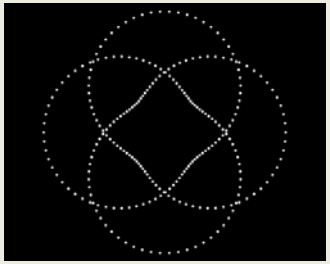
4. Discussion and future work.

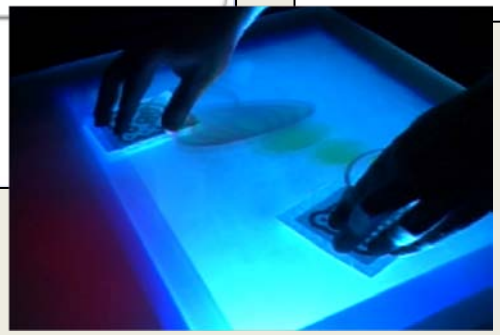
Processing Development Environment

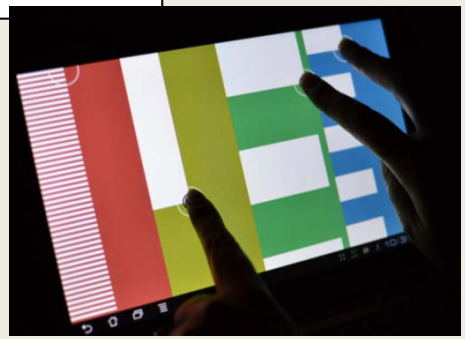
"Processing is an **open source** programming language and environment for people who want to create images, animations, and interactions."











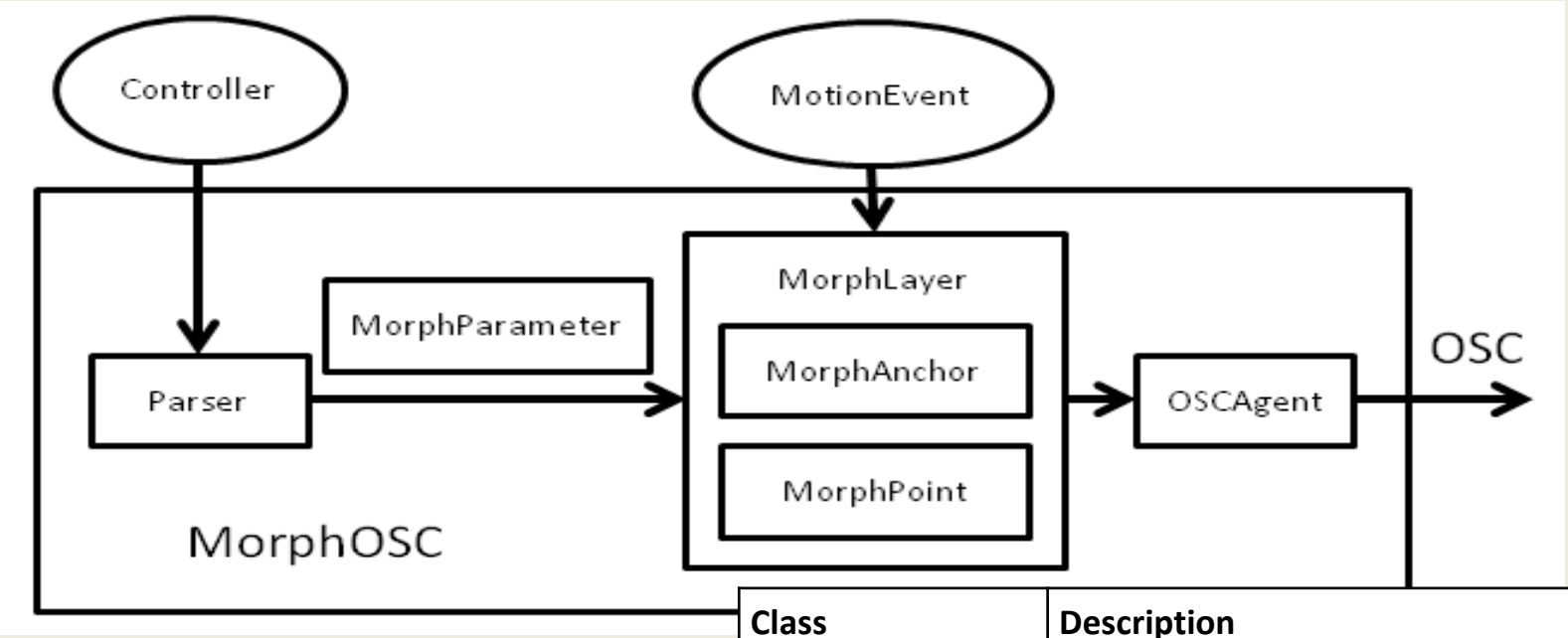
ControlP5 and *OscP5* Dependencies

- Libraries for Processing by Andreas Schlegel: www.sojamo.de

- Use these to keep code-base small, rapid development.

- Currently removing dependency on OscP5, ControlP5 is a long-term goal.

Class Structure



- Install by including JAR file in Processing path.

Class	Description
MorphOSC	Base class, manages interaction space.
Parser	Parses subset of widget fields.
MorphLayer	Interactive GUI element. Container for (i), (ii), (iii).
(i) MorphAnchor	Holds a set of parameter values.
(ii) MorphPoint	An interpolation point.
(iii) MorphParameter	Parameter value parsed from widget.
OSCAgent	Formats outgoing messages.

Usage

morphOSCUsage | Processing 2.0b8

File Edit Sketch Tools Help

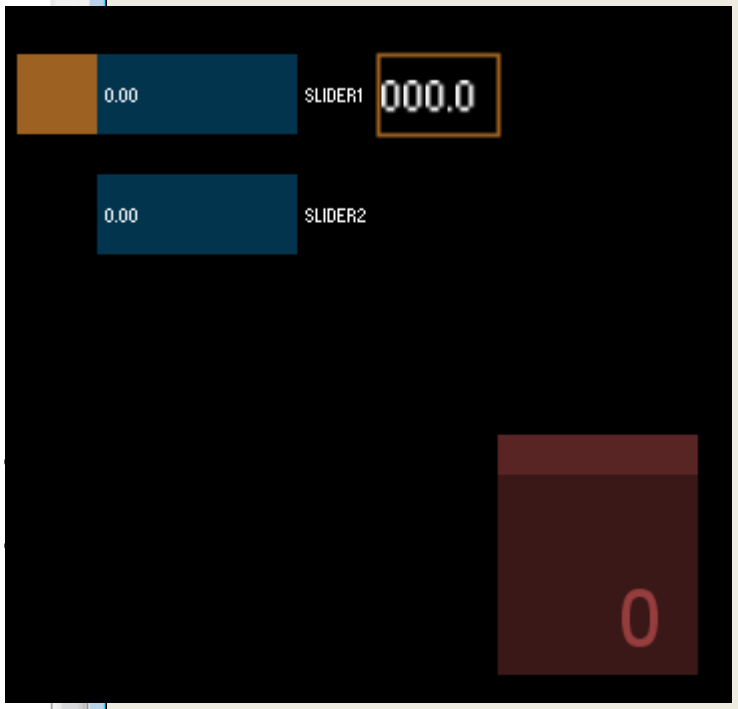
JAVA

morphOSCUsage

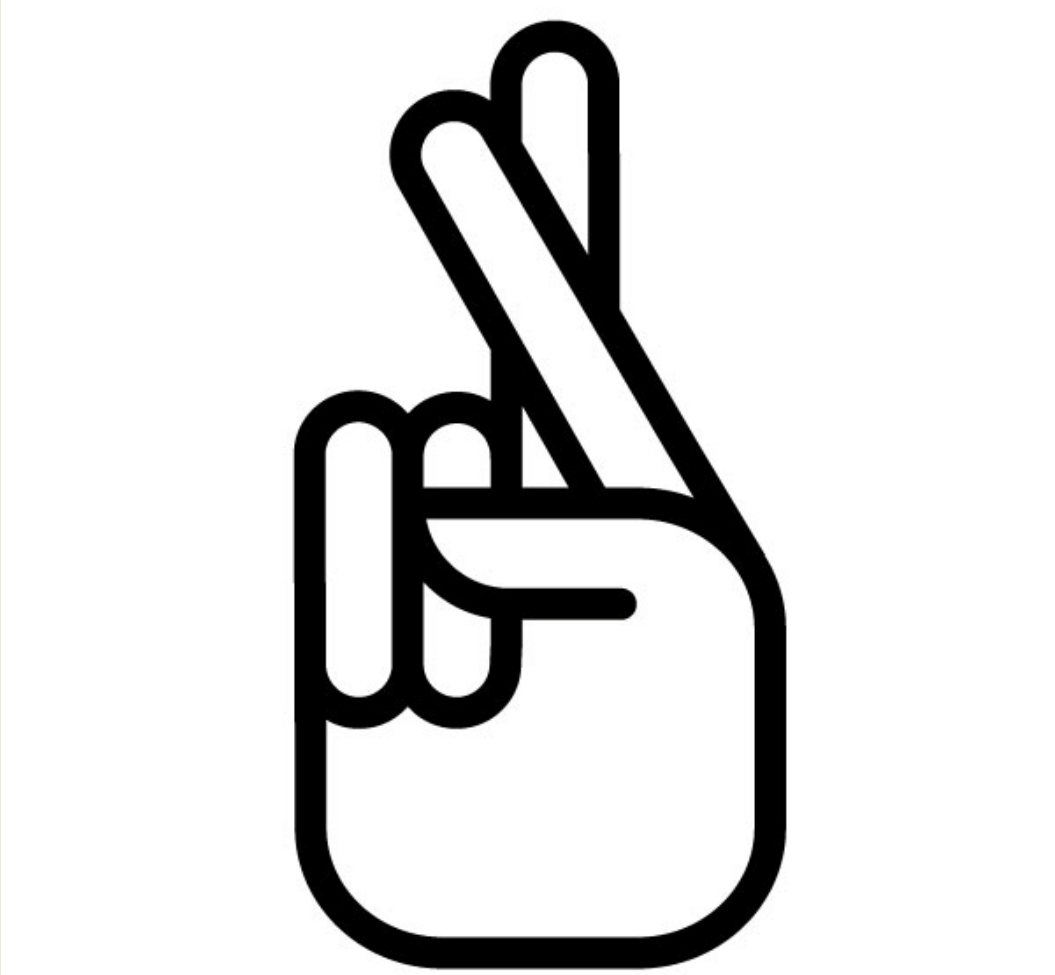
```
import net.liamosullivan.morphosc.*;
import controlP5.*;
MorphOSC m;
ControlP5 cp5;
Slider s1, s2;

void setup() {
  size(400, 400);
  background(0);
  cp5 = new ControlP5(this);
  m = new MorphOSC(this);
  s1 = cp5.addSlider("Slider1").setWidth(100).setHeight(40);
  s1.setPosition(50,40);
  s2 = cp5.addSlider("Slider2").setWidth(100).setHeight(40);
  s2.setPosition(50,100);
  m.addController(s1);
  m.getControllerInfo();
  m.addMorphLayer(width*0.75, height*0.75);
}

void draw() {
}
```



Demonstration



Presentation Outline

1. Project origins, background, goals.
2. Similar work: software controllers.
3. MorphOSC library implementation.
4. **Discussion and future work.**

Discussion

- Contributions:
  - ‘Intuitive’ graphical interface and interaction.
  - Dynamic, complex mapping in real time.
  - Cross-platform toolkit for Processing.

- ‘Barely beta’ version: basic features, buggy as hell.

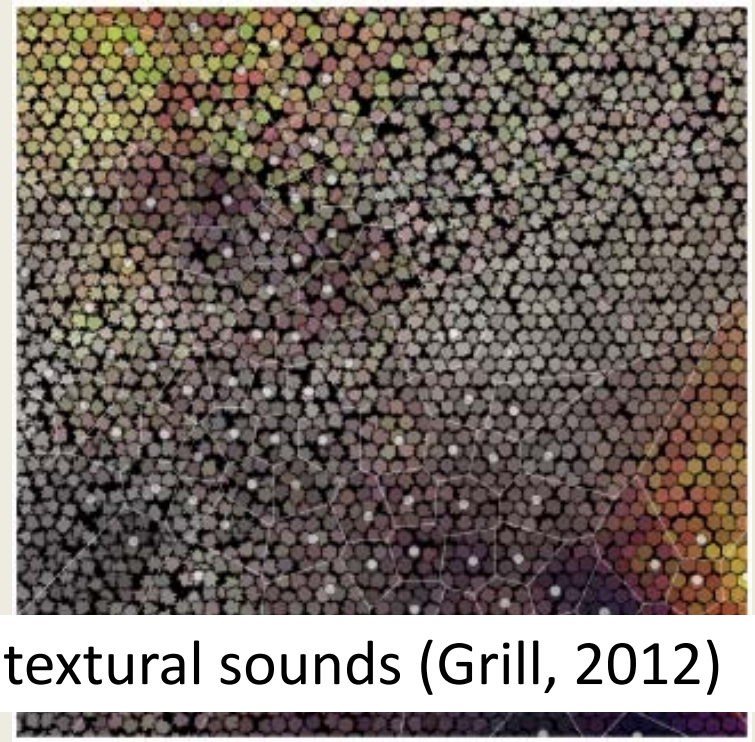
Immediate Future Work

- Attach to software development cycle: finalise feature set and move to full beta.
  - Save and recall scenes, options (xml).
  - Specification of OSC formatting.
  - Android GUI.

- Inclusion of gesture recording/ playback.

Future Work

- Is the layering metaphor effective?
  - Can it be exploited in other ways?
- Do visualisations make the interface more useable?

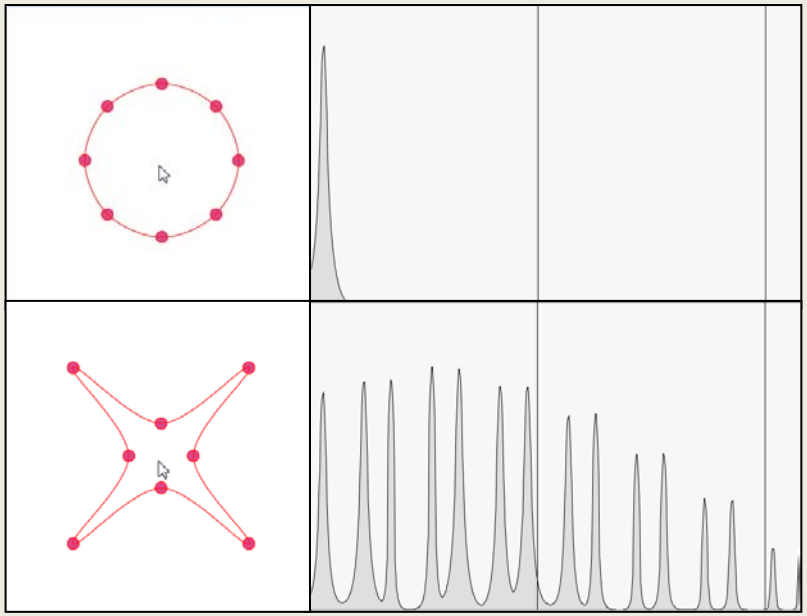


Visualisation of textural sounds (Grill, 2012)

Perceptual Anchors

- Graphical anchors used to represent the high-level aspects of the associated sounds.

An example relationship between the shape of a virtual controller (left) and output sound spectrum (right) for prototype audiovisual system (O'Sullivan & Boland, 2011).



Overlong Bibliography Slide

Drummond, J. 'Understanding Interactive Systems'. *Org. Sound* 14(2): 124–133, 2009.

Grill, T. and Flexer, A. Visualization of perceptual qualities in textural sounds. *Proceedings of the International Computer Music Conference*, 2012.

Hunt, A. and Kirk, R. ‘Mapping Strategies for Musical Performance’, in *Trends in Gestural Control of Music*, M. Wanderley and M. Battier, Editors. 2000.

Marier, M., Designing Mappings for Musical Interfaces Using Preset Interpolation. *New Interfaces for Musical Expression (NIME'12)*, 2012.

O’Sullivan, L. and Boland, F. Visualizing and Controlling Sound with Graphical Interfaces. *Audio Engineering Society 41st Conference: Audio for Games*, 2012.

O’Sullivan, L. et al. Introducing CrossMapper: Another Tool for Mapping Musical Control Parameters. *New Interfaces for Musical Expression (NIME'12)*, 2012.

Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. *Proceedings of the 1968 23rd ACM national Conference*, 1968, 517–524.

Software Resources

www.processing.org

ftm.ircam.fr/index.php/MnM

www.audiomulch.com/help/metasurface.htm

http://hexler.net/software/touchosc

Get It!

https://github.com/LiamOSullivan (Source)

www.mee.tcd.ie/~lmosulli/projects.html

Email: lmosulli@tcd.ie

Thanks!