KES Transactions on Innovation in Music: Vol 1 No 1
Special Edition - Innovation in Music 2013 : pp.32-40 : Paper im13bk-003

pMix-Touch

Oliver Larkin

Music Research Centre, University of York, YO10 5DD, UK
oliver.larkin@york.ac.uk

1. Abstract

pMix-touch is an extension of the author's pMix software [1], which is a composition, sound design and performance tool based on multi-layered preset interpolation that was originally created as a Max MSP library (int.lib) and presented at the ICMC in 2007 [2]. pMix facilitates the control of VST plug-in parameters from a 2D graphical interface that has been designed to provide intuitive real-time feedback and to allow the control of multiple parts of a signal processing graph from one abstract "Interpolation Space" (hence multi-layered preset interpolation).

pMix-touch allows The Interpolation Space to be controlled using a multi-touch interface and displayed remotely on the screen of a tablet or smart phone. This is implemented using an embedded web server, which runs inside the pMix application on a host computer. Client devices visit a webpage and data is exchanged with the server using the HTML5 websocket protocol [3]. The interface is drawn on the web page using JavaScript and the HTML5 canvas element [4].

The paper describes a new extension for Max MSP, which has been developed in order to provide a lightweight embedded web server solution with websocket support, allowing any Max MSP patch to be controlled remotely via a web page.

2. Introduction

Preset interpolation is available in many audio software packages and is a convenient way of controlling multiple parameters, discovering new hybrid sounds and generating smooth transitions [2]. Essentially preset interpolation acts as a few-to-many mapping system where a small number of parameters (e.g. X and Y coordinates on a two dimensional Cartesian space) control the many parameters of a synthesiser or effect.

pMix is a standalone application that was developed by the author using Max MSP and the int.lib library described in [2]. It uses four instances of the int.vst patch to provide four VST plug-in slots, which can be routed in series or parallel. In each plug-in slot the user can select parameters to use in preset interpolation, removing any parameters that do not interpolate smoothly. The application can be used during live performance, or in the studio with DAW software via Rewire or the

internal recording and playback functions. It comes with a collection of sound generation and processing plug-ins that have been developed with preset interpolation in mind. The plug-ins cover a range of experimental DSP techniques used in computer music (noise generators, resonators, frequency modulation synthesis, formant filtering, frequency shifting).

The main user interface used in pMix is a window called The Interpolation Space (abbreviated to iSpace), which once the user has designed their setup, can be maximized and presented as the sole interface used during performance. It shows an arrangement of presets represented as coloured circles (each layer represents a particular VST plug-in and has its own colour). The user navigates around the iSpace with the mouse (or via MIDI/ OSC control messages) and the parameters of the layer's associated plug-in are interpolated. An automation window allows breakpoint transitions to be sequenced or for the user to record and play back freehand gestures.

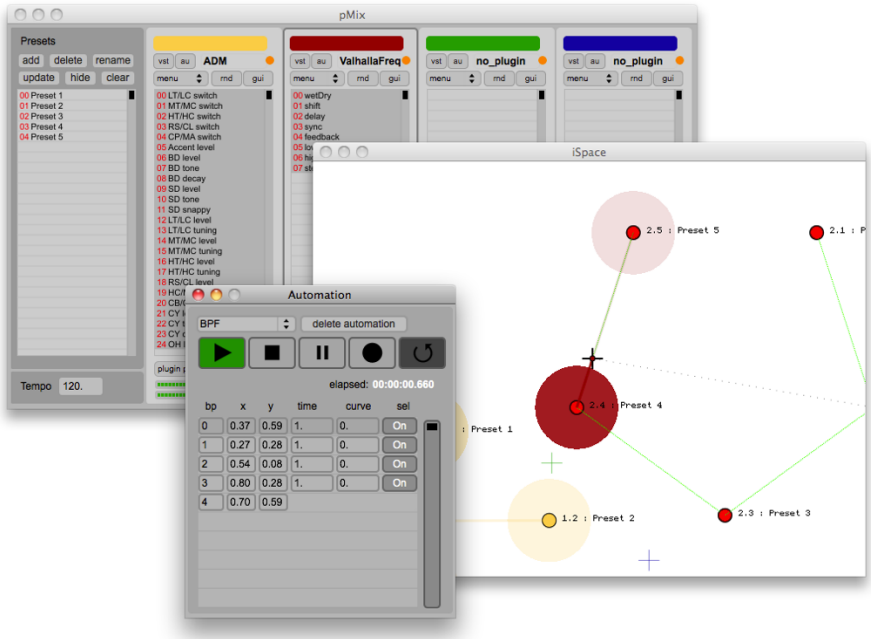

Figure 1. pMix User Interface

pMix was awarded the second prize at the LOMUS 2008 International Music Contest [5]. It is available for free on the Mac App Store [1] and has been downloaded over 5000 times since version 1.0 was released in January 2012. It was recently chosen as one of the interfaces used in the Inventor Composer Coaction [6] at the University of Edinburgh, and received positive feedback from users.

When pMix was first developed, multi-touch interfaces were in their infancy and were not widely available. The iSpace, with its two-dimensional map of multiple layers of circles, seems well suited to multi-touch control. A single touch could control each layer's interpolation point, or multiple touch devices could each be assigned a layer. Gestures could facilitate the placing of presets. The rest of this paper will discuss the technical approaches taken in order to make the iSpace multi-touchable.

3. Making pMix Touchable

Several options were considered:

- Write a script for an existing multi-touch control app
  (e.g. Hexler's TouchOSC [7] or Cycling74's Mira [8])
- Develop a bespoke iOS/Android app
- Develop a web-based interface and embed a webserver in pMix

None of the existing scriptable control apps offered enough flexibility to recreate the pMix interpolation space on a touch screen, and this option would not have provided a smooth user experience. It would also require purchasing a third party application. The closed nature of the iOS app store, and the fact that at least two apps would have to be made and maintained for the Android and iOS platforms made making a bespoke application un-appealing, although it would offer the best performance since it would allow low level access to accelerated graphics routines and operating system integration. After investigating the current browser support for recent HTML5 technologies, it was clear that the final option of a web site had the most promise since it could be viewed using any device with a web browser. Performance would probably not be an issue since browsers are very well optimized due to the strong competition in the marketplace. The ease of setup is a key issue and the simplicity of visiting a website is appealing, since providing that the desired TCP ports are not blocked on the network, getting connected should be as simple as typing in a web address in the browser. Apple document iOS web-applications and provide a simple method for saving a webpage bookmark to the home screen, which can make the user experience roughly comparable to using a native application [15].

4. Related work and technologies

For a detailed overview of previous implementations and a variety of visual approaches to preset interpolation the reader is directed to [2]. Notable developments since the work described in [2] include the "nodes" object included in Max MSP since version 5.16 (which is based on a similar gravity model as that used in int.lib), Martin Marier's implementation for SuperCollider [9] and Liam

O'Sullivan's MorphOSC for Processing [10], which features an implementation of multi-layered preset interpolation.

An early attempt at controlling Max MSP remotely via the web browser was Olaf Matthes' flashserver external [11], which allows bi-directional communication between Max MSP and Adobe Flash / ActionScript over local network or over the internet. It establishes a full-duplex TCP connection using a Flash XMLSocket, and allows communication with individual clients. This is a flexible solution and has been used in many projects since its creation over ten years ago [11].

Nowadays the web is moving away from closed-source proprietary technologies such as Flash, which is not supported on many popular devices such as the iPad. One alternative that the author explored in 2011 was to run an embedded web server in Max MSP [12] and for clients to continuously poll the server to check if new data is available (known as AJAX polling, a commonly used technique in the increasingly dynamic content of modern websites). This is obviously a fairly inefficient approach especially for low-latency control, although it has worked well for simple projects over a LAN and WLAN. In the author's implementation individual clients all receive the same data. With an increased number of clients, the demands on the webserver are dramatically increased due to a vast amount of polling, so this solution does not scale well. This issue has been recognized by the web development community, and has resulted in the creation of the HTML5 websocket protocol [3], which has achieved widespread adoption [13]. The websocket protocol allows full-duplex transmission of data over a single TCP connection, and importantly the server can send data to the client without the client requesting it. The data can be transmitted on Ports 80 and 443 (the standard ports used for HTTP and HTTPS traffic), which makes it significantly easier to access on the web and across networks where firewalls block non-web ports and network address translation (NAT) is a problem.

Many approaches to communicating with applications like Max MSP via the web browser use an intermediary application, which provides the web server functionality and forwards control information to Max using either OSC or inter-application MIDI. A popular application for this purpose is node.js [14] which allows web servers to be written in JavaScript and has been extended with OSC and MIDI modules. Several node.js users have created examples of communication between audio applications and the web browser using this approach. Another intermediary application that achieves similar results is Charlie Robert's Interface-Server [15], which has been designed as a bridge between his HTML5 canvas-based JavaScript library Interface.js and audio applications running on a host computer.
Using an intermediary application offers advantages in terms of flexibility, and perhaps reliability due to splitting responsibilities between applications; however, there are several reasons to prefer an embedded solution. By embedding the web server the application becomes self-contained (apart from the optional remote UIs) and is therefore easier to distribute and to configure. Embedded servers are also potentially slightly faster, since there are fewer endpoints in the chain. Lastly by embedding the server, it can provide functionality for reading and writing to data structures that exist within the memory space of the application.

5. Websocket support for Max MSP

Websocket support has been added to Max MSP via the development of an external object *ol.wsserver*, which embeds Civetweb [16] a lightweight, MIT licensed, open-source web server. The object currently supports text-based websocket communication which allows data to be sent and received between up to 32 individual clients (this is an arbitrary limit and could be increased). Numeric data is converted into a textual representation, which is sent between client and server via the websocket text opcode [3]. The Max message *tx* sent into the object with an argument specifying the message to transmit will be routed to all the connected clients – an optional integer argument specifies the index of a specific client to send to (see Figure 2). Messages sent from the object are prefixed with *rx* for data received from clients, and *cx* for information about clients connecting and disconnecting.
The object has been programmed to enable multiple webservers to be run within Max at any one time (providing a different port number is provided as the first argument in the ol.wsserver object box).

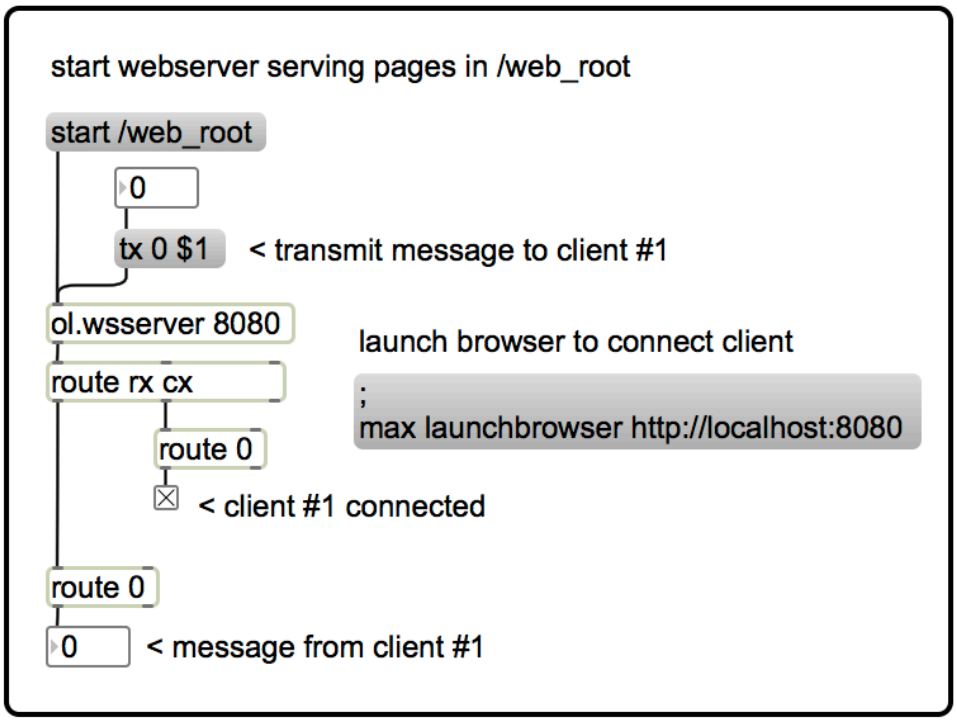

Figure 2. ol.wsserver Max object - basic usage

The textual data sent between client and server can be formatted arbitrarily, for instance a user could use OSC style syntax to target different aspects of the client or server, providing that the correct parsing was implemented on either end. For more complex applications it may be desirable to send the entire state of a Max MSP patch to/from the clients. In this case it's possible to serialize the state using JavaScript Object Notation (JSON). This is particularly convenient since Max MSP has its own support for JavaScript since version 4.5. A single JavaScript object can be used to encapsulate the state of the patch in server-side and client-side representations. In fact pMix/int.lib is mostly written using JavaScript, so this provides a relatively simple way of mirroring the data in the remote user interface. An example of how this might be handled on the client side can be seen in Figure 3.

```
var gConnection; // the websocket connection
var gState; // state of our patch/interface

function connectWebsocket() {
  if ('WebSocket' in window) {
      //create connection
      gConnection = new WebSocket('ws://'
                                  + window.location.host
                                  + '/maxmsp');

      // called when connection established
      gConnection.onopen = function(ev) {
          // send message to server
          gConnection.send('hello from client');
      };

      // called when connection closed
      gConnection.onclose = function(ev) {};

      // received message from server
      gConnection.onmessage = function(ev) {
        msg_selector = ev.data.substr(0, 5);
        // handle certain messages
        if(msg_selector = "json ") {
          // update the state from server's message
          gState = JSON.parse(ev.data.substr(5));
          //do something with state
        }

        // could handle more messages from Max MSP here
      };

      // something wrong
      gConnection.onerror = function(ev) {
          alert("WebSocket error");
      };
  }
  else {
    alert("Websocket not available");
  }
}
```

Figure 3. **Client-side JavaScript code for websocket handling**

6. A new remote Interpolation Space using Interface.js and the HTML5 canvas

To recreate the iSpace in the browser, a new widget was developed for Interface.js. Integrating with an existing, mature UI library seemed logical, allowing use of existing widgets for buttons and sliders if necessary. Other reasons for this choice are that Interface.js already supports multiple input modalities (mouse interaction and multi-touch), handles widget layout and renders all its widgets using the HTML5 canvas without using CSS, meaning that the code can be self-contained.

7. Conclusion and Future Work

The new version of pMix - pMix-Touch will enable the exploration of new methods of navigating and designing the iSpace using multi-touch gestures. This paper has detailed the groundwork required to get the application to that stage, namely the development of a new embedded web server object for Max MSP. It has provided commentary on the existing approaches to making remote interfaces, and described advantages of using an embedded websocket server.

Developing ol.wsserver has opened up lots of possibilities for connecting Max MSP patches to the web browser and for communicating with multiple remote users in real-time. The advances in HTML5 provide exciting possibilities for advanced graphical interfaces that can be operated remotely and are relatively platform independent. The current functionality of the object could be enhanced to expose more generic HTTP server features that are available in Civetweb, such as direct handling of HTTP GET/POST requests. Websocket support could be enhanced to enable secure connections and to enable transmission of binary data, which should be faster and more suitable for certain types of data. One option is to support Max MSP 6's dictionaries as a way of sharing state between server and client in an organized, hierarchical way. Another idea is to make the object respond to *jit_matrix* messages and to publish the binary data from these matrices to connected clients (*jit_matrix* is the data holder used to represent matrices and images in Max MSP/Jitter). The next step in this direction will be an object capable of streaming/receiving real-time audio and video to/from multiple clients, which is something that should be possible using other emerging technologies such as WebRTC and HTML5 Audio. Also, when WebGL is widely supported, this should be an interesting alternative to the HTML5 canvas, and should provide some advantages in terms of rendering speed, with potential for more sophisticated graphics explicitly rendered on the GPU.

Revisiting pMix after several years since it was first developed, many limitations of the original Max MSP implementation have been discovered, prompting work on a complete rewrite of the application in C++. The main aspect that can be improved is that the application can become more dynamic, with a potentially limitless number of plug-ins and more sophisticated use of screen space, supporting a

modular audio graph, like that of Max MSP, rather than a static configuration of four plug-ins in series or parallel.

pMix v1 is available for free on the Mac App Store[1].

ol.wsserver for Max MSP is free and Open Source, under the MIT license. Source code and binaries are available at [17].

8. References

[1] pMix – Mac App Store
https://itunes.apple.com/gb/app/pmix/id492576037?mt=12
[2] Larkin, O. “Int.lib - A Graphical Preset Interpolator for Max MSP”. Proceedings of the International Computer Music Conference (ICMC), 2007. Copenhagen, Denmark
[3] Internet Engineering Task Force (IETF) RFC6455 Websocket Protocol
http://tools.ietf.org/html/rfc6455
[4] W3C HTML5 Canvas Element Specification
http://www.w3.org/TR/2dcontext/
[5] LOMUS 2008
http://concours.afim-asso.org/2008/
[6] Inventor Composer Coaction
http://people.ace.ed.ac.uk/students/s9809024/icc/pMix.htm
[7] TouchOSC
http://hexler.net/software/touchosc
[8] Tarakajian, S, Zicarelli, D, Clayton, J. “Mira: Liveness in iPad Controllers for Max/MSP”. Proceedings of the 13th International Conference on New Interfaces for Musical Expression (NIME), 2013. Daejeon, Republic of Korea.
[9] Marier, M., “Designing Mappings for Musical Interfaces Using Preset Interpolation”. Proceedings of the Conference on New Interfaces for Musical Expression (NIME), 2012, Michigan, USA.
[10] O'Sullivan, L. "MorphOSC - A Toolkit for Building Sound Control GUIs with Preset Interpolation in the Processing Development Environment." Linux Audio Conference, (LAC) 2013
[11] Flashserver
http://www.nullmedium.de/dev/flashserver/
[12] Max Web Control
http://cycling74.com/toolbox/max-web-control
[13] Can I use Web Sockets?
http://caniuse.com/websockets

[14]node.js
http://nodejs.org/
[15]Roberts, C, Graham W, and Matthew W. "The Web Browser As Synthesizer And Interface" Proceedings of the 13th International Conference on New Interfaces for Musical Expression (NIME), 2013. Daejeon, Republic of Korea.
[16]Civetweb on SourceForge:
http://sourceforge.net/projects/civetweb/
[17]ol.wsserver source code/binaries
https://github.com/olilarkin/wsserver