An Internet Browser Plug-in for Real-time Sound Synthesis using Pure Data

Marcos Alonso, Günter Geiger, Sergi Jordà
Music Technology Group
Universitat Pompeu Fabra
Ocata, 1
Barcelona, Spain
malonso,ggeiger,sjorda@iua.upf.es

Abstract

In this paper we present an Internet browser plug-in for real-time sound synthesis. The plug-in uses the Pure Data (Pd) sound synthesis engine, and allows flexible and transparent sound generation and control embedded into web-pages. Pd is a general sound synthesis language that is based on the MAX graphical programming paradigm. It can be used to construct and implement a broad range of sound synthesis algorithms and paradigms, ranging from standard sample playback to interactive sound generation and manipulation. The plug-in is cross-platform and runs on Windows, OSX and Linux platforms. Versions for Windows Internet Explorer as well as for Netscape/Mozilla have been developed. The plug-in opens a wide range of new possibilities for the presentation of dynamic multimedia content, where sound plays an important role and the playback of static sound files is not sufficient.

The data format used by the plug-in (the pd patch), has to be transmitted over the net, is very small in size and thus perfectly suitable for low bandwidth connections. Examples of potential applications are games, adaptive sound backgrounds, interactive sound installations, web sites, collaborative music on the web, education and e-learning. The first implementation of the Pd plug-in has been written for a project in collaboration with the Catalan theater group "La Fura dels Baus".

1. Introduction

Internet browser plug-ins are commonly used to enhance the display possibilities of internet browsers. Most of these plug-ins mainly concentrate on graphics, sound is only playing a secondary role. Designers of multimedia web pages are mostly limited to the playback static sound samples (wave files) or to use a common set of precooked sounds via a sample-based synthesizer. This affects the amount of data that has to be transmitted as well as the flexibility and quality of the sound generation. By implementing a software based synthesis engine in a plug-in we can get rid of these limitations.

The idea of having finer grained control over the auditive content of multimedia and interactive web pages is not new. As early as 1995, Ossenbruggen and Elïens propose the use of client-side sound synthesis techniques in order to reduce the amount of resources needed for high quality music on the WWW [van Ossenbruggen and Eliëns, 1995]. Jsyn [Burk, n a], developed by Phil Burk in 1998 [Burk, 1998], also provides a real-time unit generator based synthesis for Java applets in a web page, in which sound synthesis is performed by 'C' code hidden in a Netscape plug-in or DLL beneath Java native methods. A similar approach is also taken by the JASS system [van den Doel and Pai, 2001]. All these systems match our approach and goals pretty closely; the main difference is that while these applications are written using a new API, we have chosen to bring to the web one of the most widespread interactive computer music and sound synthesis environments: Pure Data.

1.1. The Pure Data System

Pure Data [Puckette, 1996] is a computer music system that was written by Miller Puckette and implements the "MAX" paradigm of graphical programming languages for sound processing [Puckette, 1988]. The Pd system is freely available on the Internet with open source code. Because of its openness and extensibility, it enjoys an ever growing users and developers community.

Its broad user base and widespread use, together with cross platform compatibility (Linux, Windows and Mac OS) and a relatively small code base and memory fingerprint, make it the ideal system to be used as a general sound synthesis extension for client based sound rendering. What we are bringing with this plug-in is the possibility of running patches created with Pd in a web browser.

2. Design and Implementation

One big advantage of Pd is its graphical and interactive programming style. Algorithms and sound synthesis can be implemented and tested at the same time. Pd is a complete computer music system that comes with a graphical editor, which is the essential tool for working and performing with Pd. Once the synthesizer has been built with Pd, the performance part can be replaced by alternative GUI's, like Paradiddle [Lindsay and Parkes, 2004] or GriPD [Sarlo, 2004]. The Pd plug-in is in that sense, another way of controlling Pd.

The flexibility of Pd allows steering the Pd engine from almost every networked application, either through the Open Sound Control [Wright and Free, 1997] protocol, or through the simple Pd built in netsend/netreceive protocol. Pd has already been used together with Internet browsers, either as a server side streaming engine [Barbosa et al., 2003] and [Ritsch and Frauenberger, 2003] or on the client side as a stand-alone application that is communicating over a local loop-back network with a browser side java or JavaScript applet.

Implementing Pd as a browser plug-in makes this technology accessible to all users, by just installing the plug-in and by communicating via the browsers Document Object Model (DOM)[Le Hégaret et al., na] as depicted in Figure 1.

For this to be possible, the Pd engine had to be separated from the GUI and implemented as a loadable library that offers a script-able browser plug-in interface. Through this interface, the engine that runs within the browser in a separate thread, can be controlled and any of the synthesizer parameters can be changed at any moment.

Besides the common required interface, the Pd plug-in has three additional methods.

```
start()
stop()
send(name,value)
```

The `start` and `stop` methods are used to start and stop the Pd engine. The `send` command is used to communicate with any of the Pd methods and symbols that are defined in the Pd patch.

3. Usage

3.1 Loading a patch

This section shows how the plugin can be embedded in web-pages. The first thing that has to be done is to load a Pd patch into the plugin when the web-page is shown in the browser. This is done with the `<embed>` tag as follows.

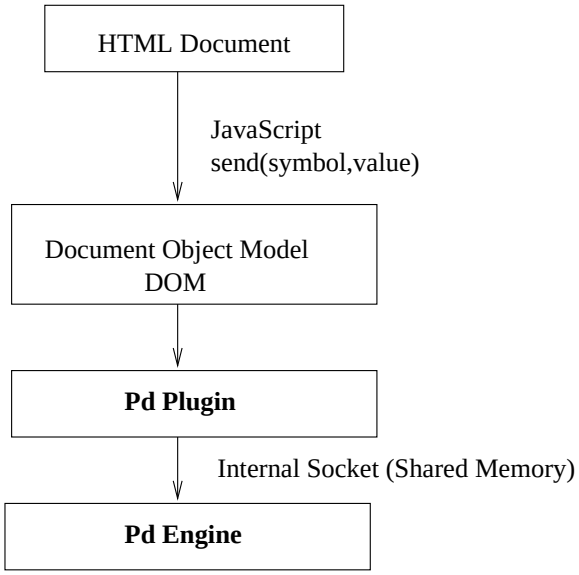

Figure 1. Pd plug-in structure

```
<embed type="audio/pd"
       src="mypatch.pd"
       hidden="true">
```

This code starts the plug-in using the Pd patch named mypatch.pd. If the Pd patch is self generating, this can already be used to supply a web-page with self generating background sound. Most applications probably want the sound to be interactive. This is done with a scripting language like JavaScript or Flash Actionscript.

3.2 Controlling the patch

If we want to control what is going on in the patch we have to communicate with the Pd engine. On the Pd side we use a receive object, which controls one or more aspects of the synthesis. Here is a simple example how to control a sine oscillator.

Figure 2. Pd receive symbol example

On the JavaScript side we first have to assign the loaded

Pd instance to a variable in order to be able to access it and being able to call its methods.

```
<script language="JavaScript">
  var pd;
  if(document.all) pd=window.document.pdx;
  else pd=document.embeds[0];
</script>
```

Once we have the handler we can start sending messages using JavaScript via the Document Object Model (DOM)[Le Hégaret et al., na]. Here is an example of a link that will send a freq message to Pd when clicked.

Sending a message to the plug-in using JavaScript:

```
<a href="javascript:pd.send('freq',440)">
 Set frequency to 440hz</a>
```

The first argument of the send method is the name of the receive object in the Pd patch (freq), the rest is the message that should be sent to the Pd receiver.
The same using Flash Actionscript:

```
getURL("javascript:pd.send('freq',440)")
```

4. Applications

The generality of the Pd language and the ease of use of the plug-in technology give way to a wide range of applications. In this section we will describe some of them.

- **Interactive Music:** From the musician's point of view the most obvious application is interactive music. This includes applications like synthesizers, sequencers, sound installations or multimedia installations in general.

  The decoupling of the GUI from the sound engine expands the control possibilities that are natively offered by Pd, and allows the GUI interface to be designed according to the application, instead of having to stick to predefined GUI elements. The simultaneous use of the Pd plug-in and the Flash plug-in becomes for example an extremely simple and natural issue, which definitely blows all the sound restrictions and limitations of Flash-based web pages and opens a new standard for web applications in the style of SoundToys [Stanza,].

- **Low bandwidth delivery of music:** Generative music, encoded as Pd patches can be transmitted over the net using only a few bytes (the size of the Pd patch). The idea is that instead of generating music, encoding it as sound-file and transmitting the sound-file, only the instructions on how the music is generated get send over the net, and the music itself is generated in real-time on the user's machine. This generative approach has already been implemented in some commercially available web browser plug-ins such as the "Koan Vector Audio" system by Sseyo [SSEYO, na]. The main difference between Koan's plug-in and ours is that we are using an open format, which can be generated with freely available software (Pd) and which already profits from a big and widespread community of users and developers.

- **Auditory User Interfaces:** The Pd engine can be used to give auditory, dynamic feedback to the user. This not only includes the playback of triggered event sounds, but also the possibility of rendering user interface information in real time, such as auditory progress bars, mouse positioning and vicinity cues. The graphical user interface can be augmented or replaced by an auditory user interfaces [Kaltenbrunner, 2002].

- **Sonification** Another application of the Pd plug-in technology is sonification of data. The problem with standard web technologies is that they only allow for static sonification, this means that the sound-files have to be calculated on the server, and the data can only be viewed from one angle. With the dynamic possibilities of the plug-in it is possible to offer an unlimited amount of viewpoints and angles over the data and the adjustment of sonification parameters.

- **Collaborative music on the web** Collective creation and the production of open and continuously evolving works are indeed two of the most appealing artistic breakthroughs the Internet can offer to music composers and creators in general, and this is an area in which this research team has been largely working in the last years, specially within the scope of FMOL [Jordà, 1999][Wüst and Jordà, 2001] and the Public Sound Objects projects [Barbosa et al., 2003]. The idea of musical computer networks is by no means original, since earlier implementations (although on a local area scale) date back to the late 1970s with performances by the League of Automatic Music Composers [Bischoff et al., 1978], to later become the Hub [Gresham-Lancaster, 1998]. The late nineties saw the breakthrough of several collaborative projects and sites such as ResRocketSurfer, MIT's Brain Opera, William Duckworth's Cathedral [Duckworth, 1999] or the same FMOL, but the truth is that more than twenty five years after the League's first experiments, collective and real time music creation and improvisation on the net are still at a burgeoning state [Barbosa, 2003]; a situation that could start changing with this plug-in, which finally manages to bring to the web one of the interactive computer music de facto standards.

5. Future Work

By the time of writing this document the first beta versions of the plug-ins for Mozilla/Netscape (for Windows and Linux) and Internet Explorer for Windows are available on the plug-in website [Alonso,]. The primary milestones for the first public versions are improving the stability and latency since all the basic functionalities have been already implemented. Beyond providing a stable version and a series of examples with which others can experiment, there are numerous other features that have been envisioned:

- Low latency versions using DirectX or ASIO. The current versions work with the MME standard windows audio output. Even though the latency is not too high some applications may need a faster time response.
- Data flow is presently only one-way, from browser to plug-in. A way to obtain data from the Pd plug-in in the web-application still has to be added.
- The current versions of the plug-in can only load a single ".pd" file at a time. A new compressed format for a whole Pd project should be developed to overcome this limitation.

As Pd can be a very powerful language, there are system security issues that arise. These issues have not yet been closely investigated, but we hope to overcome them by restricting some of Pd's built-in features such as sound file writing.

6. Conclusions

We have presented the Pd web browser plug-in, a piece of software that brings to the Internet the power and the unlimited possibilities of Pd, one of the most versatile, powerful and widespread programming environments for real time interactive music and sound synthesis. Although the project is far from being finished it is already completely useful, and we believe its use can boost the creation of interactive and collective music sites and projects, and bring new sonic standards to Flash-based interactive web pages.

This research has been partially funded by the EU-FP6-IST-507913 project SemanticHIFI.

References

[Alonso,] Alonso, M. Pd plug-in. www.iua.upf.es/~malonso/pdplugin.

[Barbosa, 2003] Barbosa, A. (2003). Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, 13:53–59.

[Barbosa et al., 2003] Barbosa, A., Kaltenbrunner, M., and Geiger, G. (2003). Interface decoupled applications for geographically displaced collaboration in music. In *Proceedings, International Computer Music Conference*, pages 199–202, Singapore. International Computer Music Association.

[Bischoff et al., 1978] Bischoff, J., Gold, R., and Horton, J. (1978). Music for an interactive network of computers. *Computer Music Journal*, 2(3):24–29.

[Burk, 1998] Burk, P. (1998). Jsyn - a real-time synthesis api for java. In *Proceedings, International Computer Music Conference*, Ann Arbor, USA. International Computer Music Association.

[Burk, n a] Burk, P. (n-a). Jsyn - java audio synthesis. http://www.softsynth.com/jsyn.

[Duckworth, 1999] Duckworth, W. (1999). Making music on the web. *Leonardo Music Journal*, 9.

[Gresham-Lancaster, 1998] Gresham-Lancaster, S. (1998). The aesthetics and history of the hub: The effects of changing technology on network computer music. *Leonardo Music Journal*, 8:39–44.

[Jordà, 1999] Jordà, S. (1999). Faust music on line: An approach to real-time collective composition on the internet. *Leonardo Music Journal*, 9:5–12.

[Kaltenbrunner, 2002] Kaltenbrunner, M. (2002). Y-windows: Proposal for a standard aui environment. In *Proceedings of the 8th International Conference on Auditory Display*, Kyoto, Japan.

[Le Hégaret et al., na] Le Hégaret, P., Whitmer, R., and Wood, L. (n/a). Document object model (dom). http://www.w3.org/DOM.

[Lindsay and Parkes, 2004] Lindsay, A. T. and Parkes, A. P. (2004). Paradiddle: a code free meta-gui for musical performance with pure data. In *Proceedings, International Computer Music Conference*, pages 423–426, Singapore. International Computer Music Association.

[Puckette, 1988] Puckette, M. (1988). The patcher. In *Proceedings, International Computer Music Conference*, pages 420–429, San Francisco. International Computer Music Association.

[Puckette, 1996] Puckette, M. (1996). Pure data: another integrated computer music environment. In *Second Intercollege Computer Music Concerts*, pages 37–41, Tachikawa, Japan.

[Ritsch and Frauenberger, 2003] Ritsch, W. and Frauenberger, C. (2003). A real time audio rendering system for the internet (iars), embedded in an electronic music library (iaem). In *Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, London, UK.

[Sarlo, 2004] Sarlo, J. (2004). A graphical interface editing tool and run-time environment for pure data. In *Proceedings, International Computer Music Conference*, pages 305–307, Singapore. International Computer Music Association.

[SSEYO, na] SSEYO (n/a). Koan vector audio. http://www.sseyo.com/showcase/vectoraudio/index0.html.

[Stanza,] Stanza. Soundtoys. http://www.soundtoys.net.

[van den Doel and Pai, 2001] van den Doel, K. and Pai, D. K. (2001). Jass: A java audio synthesis system for programmers. In *Proceedings of the 2001 International Conference on Auditory Display*, Espoo, Finland.

[van Ossenbruggen and Eliëns, 1995] van Ossenbruggen, J. and Eliëns, A. (December 1995). Bringing music to the web. In *Proceedings of the Fourth International World Wide Web Conference, The Web Revolution*, pages 309–314. O'Reilly and Associates, Inc.

[Wright and Free, 1997] Wright, M. and Free, A. (1997). Open sound control: A new protocol for communicating with sound synthesizers. In *Proceedings, International Computer Music Conference*, Thessaloniki. International Computer Music Association.

[Wüst and Jordà, 2001] Wüst, O. and Jordà, S. (2001). Architectural overview of a system for collaborative music composition over the web. In *Proceedings of the 2001 International Computer Music Conference*, San Francisco. International Computer Music Association.