PDradio

http://pdradio.iem.at

Georg Holzmann

Oct. 2005, Graz

PD Webradio is a self-regulating virtual radio station using a Content Management System.
Users can listen to the music-stream, vote for songs they would like to hear, upload their own music and listen to the comments of the automatic radio moderator.
The whole system is designed for the PD-community, a (not so small) group of people using the realtime audio-, videosystem *Pure Data*[1] to create their music.

1 concept

PD webradio is a self-regulating internet radio station for the Pure Data community.
Registered users can upload PD-related music and share them with the community.
At the beginning the whole system was made for study purposes only at the Institute of Electronic Music and Acoustics (IEM) [2] in Graz, Austria, by Winfried Ritsch and Georg Holzmann.

1.1 radio stream

Listeners of the PDradio can hear the latest Pure Data radioprogram.
Additional to the music there is an automatic radio moderator, who speaks a little bit about the songs and the composers - in a very funny english.

1.2 interaction

If you want to hear specific songs you have the possibility to vote for them. After each vote a new playlist will be generated and you can see it immediately in the browser.
As registered user you are allowed to upload your own songs as ogg or mp3 files.
With each song you should also upload additional metadata, which is only one text where you can describe yourself, the song, contact information or whatever you want.
That text will be read by the automatic radio moderator before your song is played - so it should be in a "readable" english for the speech synthesizer.

[1]PD, aka Pure Data, by Miller Puckette and others: http://puredata.org
[2]Institute of Electronic Music and Acoustic, Graz: http://www.iem.at

1.3 archive

All the songs and metadata are stored in the PDradio archive.
You can also download these files, but again only if you are registered.

1.4 license

The current license text looks as the following:

> *You are resplonsible for the content and you are not allowed to upload files prodected by copyright laws! All files of the PDradio archive are downloadable for registered users and are transmitted by the internet radio for free.*
> *By clicking "Upload file" you agree on all conditions of PDradio!*

which is indeed a little bit "incomplete".
So we might switch to a specific Creative Commons License [3] in future.

2 technical realisation

PDradio is implemented with Open-Source software and runs on a Linux server with Zope/-Plone [4].
The audio streaming, speech synthesis, radio-moderator, DJ, ... is implemented in PD and Python for the communication to plone.

2.1 CMS System

As already mentioned we use an Apache server on a Debian stable Linux system with Zope.
Plone is built using Zope, an object oriented application server. The language that drives Zope and Plone is Python [5], so it is easy to extend the system with new "Products" written in Python.
The Plone-portal, running in Zope, does all the work which is needed for member management and registration of new users, uploading/downloading files (with the Product *LocalFileSystemNG*), etc.

2.2 the webinterface

In the webinterface the user can see a link to the radio stream, the current track, the actual playlist and a list of all available songs with links to vote for each one.
We made a new Product *PdControl* for Plone, which consists of a Python-script and a HTML-template. This Product manages the user-interaction and communicates with the sound engine (PD).
As an example you can see the template code to display the *current track*, which is played at the moment:
calling the Python method *get_playing_info* (see table 1) is just calling the link *here/get_playing_info* (which results in `http://pdradio.iem.at/radio/get_playing_info` as absolute URL).

[3]Creative Commons: `http://www.creativecommons.org`
[4]Plone: `http://www.plone.org` , Zope: `http://www.zope.org/`
[5]Python: `http://www.python.org`

```
<u><b> current track: </b></u>
<div class="PdControlFile"
     tal:content="here/get_playing_info">
  current song
</div>
```

Table 1: HTML template example

In table 2 you can see the called Python method, which is part of the class *PdControl*. It makes a connection to the sound engine (if not already open) and returns the current track.

```
def get_playing_info(self):
    """get the current song from PD."""

    server = self.connect()
    song = server.get_playing_info()
    return song
```

Table 2: corresponding Python code

2.3 XML-RPC

The communication between the webinterface and the sound engine is managed using the XML-RPC protocol [6], a remote procedure call between various progamming languages using HTTP as the transport and XML as the encoding.
The XML-RPC server is running in the sound engine and responds to calls from the webinterface.

2.4 the sound engine

The sound engine is implemented in PD and is responsible for the program logic, speech synthesis (radio moderator), soundfile player and the audio streaming (see figure 1).

2.4.1 program logic

All the program logic (ranking of the songs, current playlist, next song, scanning for new files) is again implemented in Python, but embedded in PD via the pyext[7] external.
Also the XML-RPC server is running in that pyext external in a separate thread, so the information of the webinterface can finally be accessed in PD.
The ranking algorithm is rather simple:
If a user votes for a song, the ranking increases by one. Always the song, with the highest ranking is on top of the playlist (or if there are more songs with the same ranking it will be choosen randomly). When the song is playing its ranking is reseted to 0.

[6]XML-RPC: http://www.xmlrpc.com/
[7]pyext by Thomas Grill, http://www.grrrr.org

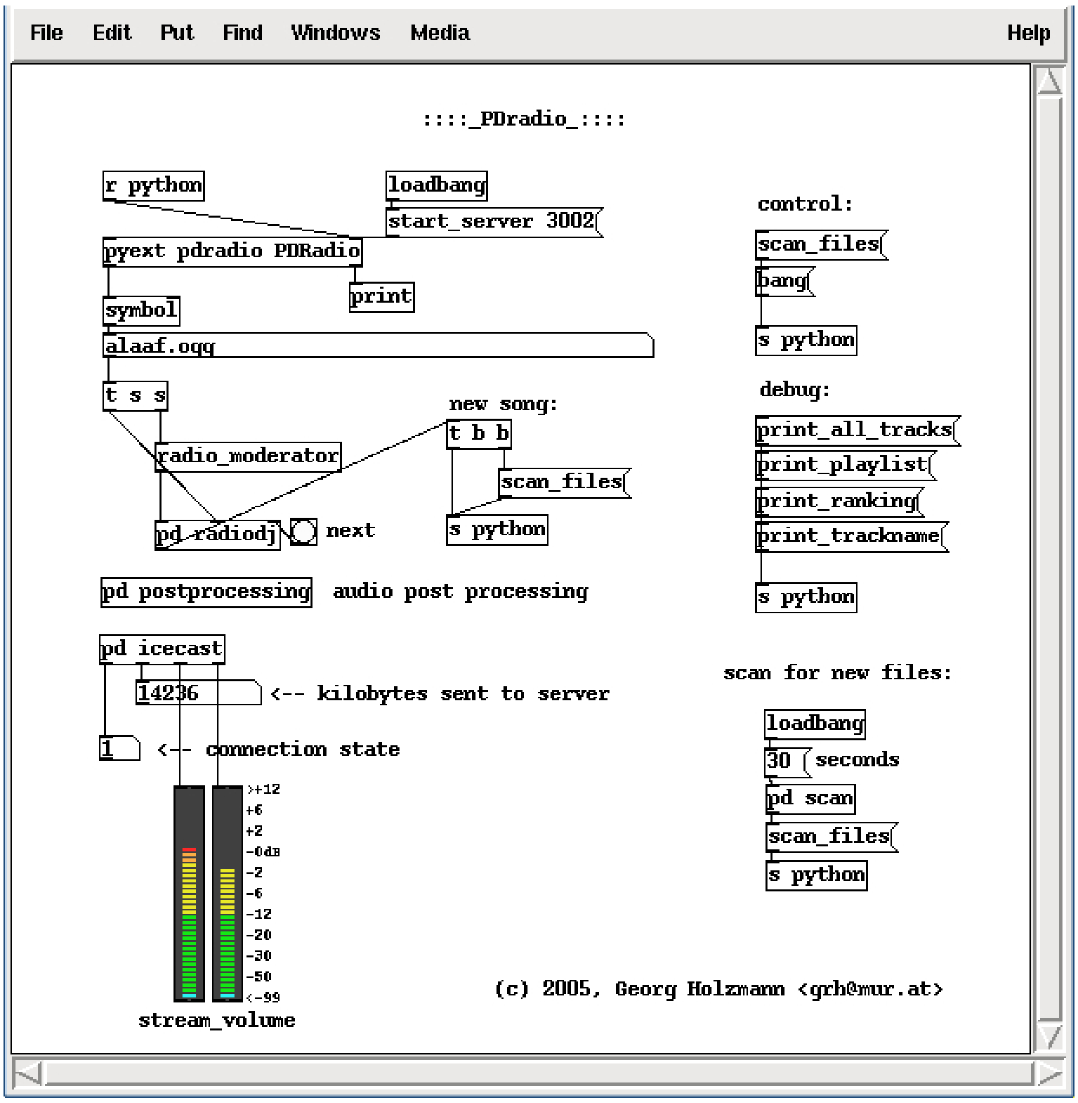

Figure 1: main PD patch of the sound engine

2.4.2 radio moderator

With each song also additional metadata is uploaded and is read by the automatic radio moderator.
For speech synthesis we use Flite (festival-lite) and its wrapper for PD [8].
There are also additional expletive words inbetween metadata and songs, which are changing all the time, so the moderator sounds a little bit more "natural" or "intelligent".

2.4.3 post processing, streaming

The post processing of the uploaded ogg or mp3 audio files consists of a simple limiter and compressor combination.

[8]Flite: http://fife.speech.cs.cmu.edu/flite and PD external by Bryan Jurish

Finally an icecast2 server [9] is streaming the resulting audio and the speech of the moderator.

3 future plans

Some ideas for the future, to keep the project alive:

- switch to an ogg stream
- a newscast at each whole hour (with RDF [10])
- announcements of PD related events:
  a (registered) user can upload a specific announcement text, which will be read after the news each hour for e.g. one week
- live streaming of events:
  registered users can reserve one or two hours to stream their live shows (they get a password+username and can stream directly from PD to our icecast server)
- PDtv - a video stream with PD related audio-visual works

4 conclusion

So if you want to listen to the stream, vote for songs or you have music related to PD and want to share it - follow the link: http://pdradio.iem.at !

[9] Icecast: http://www.icecast.org and shoutcast external for PD by Olaf Matthes: http://www.akustische-kunst.org/puredata

[10] RDF, Resource Description Framework: http://www.w3.org/RDF/