RjLib

(For discussion of future improvements see **RjLibnew**)

RjLib contains all the additional abstractions and externals used in **RjDj** on top of the vanill

Check it out from svn with:

```
svn co http://svn.rjdj.me/scenes/trunk/rjlib
```

Or use the snapshot at **http://rjdj.me/~fbar/rjlib.zip** (date: 12.12.2008)

Also useful: The rjutils from **http://rjdj.me/~fbar/rjutils.zip** (includes accelerometer da **SceneTemplate?**)

soundinput

This is the generic soundinput abstraction. Use this instead of the `adc~` object.

soundoutput

The soundoutput abstraction is used instead of the `dac~` object from Pd.

Externals

On top of that a restricted set of externals can be used:

- `fiddle~` pitch detection
- `bonk~` attack detection, spectral analysis

Externals for environment audio analysis:

- `rj_accum` change detector, compares short-term values with long-term values. Value message from e.g. rj_centroid. This externals are not meant for imediate reaction tim configure them to accumulate change in the audio signal over a given duration. We a are provided with fixed frequency (e.g. each block 1024)
- `rj_centroid~` compute spectral centroid value (e.g. low frequency, high frequency) - rj-centroid~-help.pd
- `rj_senergy~` compute spectral energy --> needs FFT, see rj-centroid~-help.pd
- `rj_barkflux_accum~` compute spectral change between short term average spectrum spectrum. Useful for catching changes in the sound envirnonment (Indoor vs street)
- `rj_zcr~` compute zero crossings, does not need fft. Can be useful for speech detectio talking)

If you want to use these externals on your computer for scene testing, the source code of t provided in the /src folder along with a makefile to compile on macosx and linux (makefile) (makefile_mingw). You can also download the binaries for win32 and darwin :

date: 12.12.2008

http://rjdj.me/~mahm/rj_externals_win32.zip

http://rjdj.me/~mahm/rj_externals_darwin.zip

Text and Images

Additionally we have two externals that only make sense on the **RjDj** mobile client: `rj_ima`

They let you put images and text on the screen and they are attentionally kept really simple with an argument which for [rj_image] is an image name - jpg and gif with transparency ar png as well, don't know ATM). For [rj_text] the argument is a string (a "symbol" in Pd).

You can move the center of images and text with "move X Y" messages, and for text, you c "size <FONTSIZE>" and the text with "text <SYMBOL>" messages. It's a bit similar to GEM Text positioning is a bit, well, strange. It's all Apple's fault!!! :)

Rjlib contains two helper abstractions that should make working with text and image a bit e and [g_showtext]. They have helpfiles. [g_showtext] also converts lists with spaces in it to quite handy.

Currently the images and text are z-stacked in the order of object creation. Yes, that's not g better solution.

Probably it's easiest to get a grip of image and text by looking at some examples. I wrote tv images and text: "Moogarina.rj", which implements a simple "Preferences" panel with image "Showtime.rj", which is just a demonstration scene without sound.

You can get them both in our public scene repository here: **http://trac.rjdj.me/browser** or with SVN like:

```
$ svn co http://svn.rjdj.me/scenes/trunk/rjdj_scenes/Showtime.rj
```

Hint: [rj_text] is very useful in connection with [u_loadmeter] to show the machines load o Everything below 80 is good, everything above is risking crackles. :)

Deprecated objects

The following two are abstractions that have counterparts in the **new "rj" library**, so you s anymore:

accelerate

Accesses the raw accelerometer data from the iphone and sends them parsed in certain forr accelerate]

Values that can be routed out:

- pitch: pitch angle of the iphone in radian (-pi/2 to pi/2). 0=iphone lying on a table
- roll: roll angle of the iphone in radian.
- magnitude: amount of the overall acceleratio
- x: acceleration on the x axis
- y: acceleration on the y axis
- z: acceleration on the z axis

touch

Accesses the raw touchpad data and provides it in different formats (sent to [send touch])

Values that can be routed out:

- tap: this message is sent whenever a finger touches the pad
- speed: gives you the speed of which each finger is drawing on the touchpad
- updown: gives 1, when finger is on touchpad, 0 when it leaves

- direction: gives the direction angle of which the finger is moving (radian: 0 -> pi)