21M.380 Music and Technology
Sound Design

Pd assignment 3 (pd3)
Channel strip abstraction

Due: Wednesday, March 9, 2016, 9:30am
Submit to: MIT Learning Modules ▸ Assignments
10% of total grade

1 Instructions

Complete the `[ch~]` abstraction provided with this assignment, such that it can be used as an input channel strip in a Pd software mixer. Follow the specifications in this document and those implicitly given through the available elements in `ch~.pd`. A corresponding help patch `ch~-help.pd` is provided for testing purposes.

2 Specifications

Your `[ch~]` channel strip abstraction should provide the following features:

- Mono audio input
- Fader controlling the output level from –100 dB to 0 dB
- Mute button
- Post-fader mono effect (FX) send bus
- Stereo panpot, switchable between cosine, square root, and linear panning laws
- Stereo audio output

3 Guidelines

- Start from the incomplete `ch~.pd` patch provided with this assignment. The patch already contains the final patch's basic structure, which you should leave as is. You are expected to 'fill in the gaps' by completing the following subpatches:
  - `[pd fade_mute]`

  - `[pd fx]`
  - `[pd pan]`

  Do *not* add any elements outside of these subpatches!

- Do not remove, change, or ignore any of the existing elements anywhere in `ch~.pd`! They provide the specification details according to which your final patch is expected to work. In particular, leave the graph-on-parent GUI elements intact, including the numeric ranges of all faders, number boxes, etc.

- Note that you will not need to add any `[inlet]`, `[outlet]`, `[inlet~]`, `[outlet~]`, `[s]`, or `[r]` objects anywhere in `ch~.pd` for the patch to comply with this assignment's specifications. So don't. ☺

- Only objects from Pd vanilla are allowed, and no externals will be required to comply with the assignment's specifications. In particular, usage of the `[expr]` and `[expr~]` externals that ship with Pd vanilla should be avoided.

- You can use the `ch~-help.pd` patch to test your `[ch~]` abstraction as you develop it. The help patch includes two instances of the abstraction that are fed by two different test signals. It also includes a simple distortion effect by means of which you can test your abstraction's effects (FX) send bus. Note that you will *not* include the `ch~-help.pd` patch with your final submission, though!

- When you test a first draft of `ch~.pd` patch in `ch~-help.pd`, pay particular attention to smooth transitions. For example, dragging faders or number boxes should not result in crackling audio, switching between panpot laws should not result in audible dropouts, etc.

- Also, the GUI elements in `[ch~]` should at all times reflect the actual state of audio processing. For example, the mute button will be deactivated on load, so the channel should actually be unmuted by default.

- The patch can be implemented entirely using techniques from Farnell (2010) that are referenced in the provided version of the `ch~.pd` patch. After you have consulted the corresponding book chapters (which will greatly simplify things), you will find that the main challenge is not so much how to build a fader, cosine panner, mute button, etc., but rather to implement a patch from given specifications. This should be the focus of your efforts.

- It is important to know that one can set the send and receive symbols of any GUI object (slider, toggle, etc.) via ctrl+Properties (Mac) or right-click Properties (Linux, Windows). For example, after setting a slider's receive symbol to `foo`, you can use a `[send foo]` to send it messages. Vice versa, you can receive data from a slider into a `[receive bla]` after setting its send symbol to `bla`.

4 Assessment criteria

Functionality according to the provided specifications

Documentation of the code using informative comments

Your patch will be evaluated by means of the `ch~-help.pd` patch, *in the help patch's version that is provided with this assignment*. Ensure that the `ch~-help.pd` patch does soemething meaningful, even if you don't manage to implement `ch~-help.pd` in its entity. For example, if you manage to implement the fader, mute button, and FX send, but not the panpot, connect the `[inlet~]` and `[outlet~]` objects in `[pd pan]` in a meaningful manner, so that at least *some* signal will be heard in `ch~-help.pd`.

5 Submission format

Submit a single `.zip` archive that contains *only* your final `ch~.pd` file. The reason why I ask you to zip the patch is that Stellar (and probably also LMOD) automatically replaces any tilde character (~) in a file name with an underscore (_). There is no need to submit the `ch~-help.pd` (to which you should actually not make any changes).

References and useful resources

Farnell, Andy (2010). *Designing Sound*. Cambridge, MA and London: MIT Press. 688 pp. ISBN: 978-0-262-01441-0. MIT LIBRARY: 001782567. Hardcopy and electronic resource.

21M.380 Music and Technology: Sound Design

Spring 2016