Assignment 1

TASK

Design a real-time audio application, consisting of a Pure Data patch running on Bela and generating sound, controlled via an external device.

DESIGN GUIDELINES

Sound:

- Bela has be the only source of sound in the application;
- the patch has to synthesize sound in real-time, by means of oscillators and/or samples loaded from audio files;
- the patch has to include some kind of audio processing, for example filtering, delay, distortion, modulation, waveshaping, amplitude enveloping, etc.

Control:

- the patch should be controlled via one of the following:
  - MIDI messages coming from a MIDI controller connected directly to Bela;
  - MIDI messages coming from an application running on the host (e.g., a Pure Data patch, a DAW);
  - OSC messages coming from a Pure Data patch running on the host;
  - a combination of the above;
- regardless of the chosen control strategy, the application may be designed to rely upon one of the following:
  - human interaction (i.e., you using the MIDI controller or the MIDI/OSC control patches);
  - automations and prescribed sequences (e.g., MIDI clips, list of MIDI/OSC messages, automatic/algorithmic generation of MIDI/OSC messages);
  - a combination of the two.

Structure and components:

- the patch can include any Pure Data object, beyond the list of those we covered in class;
- the application has to be showcased in class (no MIDI controllers that cannot be brought to class, nor setups that cannot be replicated in class).

I strongly recommend to browse the built-in help files to understand how objects work. Help patches often times also include links to similar objects that may be useful in your project.
You are free to find inspiration from other example patches, like the ones that I shared with you, the ones that you find on Bela and also the one published by the large Pure Data community that exists online.

However, you are required to design something personal, so any inspiration must be accompanied by a personal interpretation and further development. In class, you will be requested to argument what your sources were and where your contribution lies. So, please do not just copy one of the hundreds of additive synthesis example patches out there, but try to be creative!

Furthermore, in class you will be requested to describe in detail how the application works. If asked to, you must prove that you exactly understand how every object in the patch works, how it interacts with the specific objects it is connected with, as well as its role in the synthesis/processing/ control pipeline.

RUBRIC

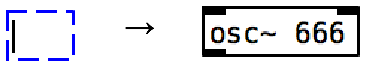
The assignment will evaluated in accordance to the following criteria (in order of importance):

- correctness of the patch (no useless/redundant connections, no unexpected/inexplicable behaviors);
- creativity and originality;
- complexity of the synthesis/processing;
- complexity of controls and mapping (between control parameters and synthesis parameters);

The grading rationale will reflect your previous knowledge of audio programming (e.g., Max/MSP). This means that if you have formally gained skills in Max/MSP already, the requested effort will be higher compared to the case of students with no experience in the field.

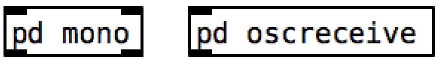
Appendix

In class we covered a variety of objects. Most of them can be created by first putting in the patch an "empty" object (ctrl/cmd+1) and then typing in their name (and parameters).

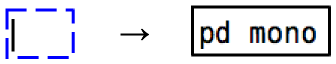


This will generally allow you to create the object in any of your patches and place it where you need.

However, we encountered some objects whose names start with "pd", like for example:



If you try to create these objects from scratch, by first putting an empty object in the patch and then adding typing in the name, it won't work!



(note the different number of inlets/outlets)

We will see the reason of this soon, but for now just make sure to remember that, when you need any of these "pd" objects in your patches, you will have to copy them from other patches. We did this in class for "pd mono", "pd oscreceive" and "pd oscsend", we copied them from some example patches I shared with you (mono.pd, osc_receive.pd, osc_send.pd, respectively). You can copy these "pd" objects from my patches, or from any of the patches we built in class.