Computer Music - Languages and Systems
JUCE - Homework #2

Group 11 - BeetleJUCE:
Adriano Farina, Alessandro Gorni, Giuseppe Risitano, Enrico Regondi, Rebecca Superbo

May 2022

1 Goal

This Project aims at creating a plugin that implements a flanger effect. This plugin takes an audio waveform from a microphone, an instrument or an audio clip and outputs the modified waveform, and it is controllable by the user through an user interface (GUI). Specifically, it is implemented with the help of the JUCE framework and it is intended to be used within a Digital Audio Workstation (DAW) as a VST3.

2 Flanger

Flanging is an audio effect produced by mixing two identical signals together, one signal delayed by a small and gradually changing period: the result of such operation is a so-called "comb filter" which is in charge of the very peculiar effect perceived. It is a delay-based effect, meaning that one or several copies (in the case of the feedback) of the signal are created and stored. By slightly delaying in time the copies of the input signal and controlling the delay with a Low Frequency Oscillation (LFO), the distinctive flanger effect sound is generated. In particular, the LFO controls the total delay time, thus causing constructive/destructive interference between the original signal and its copies.

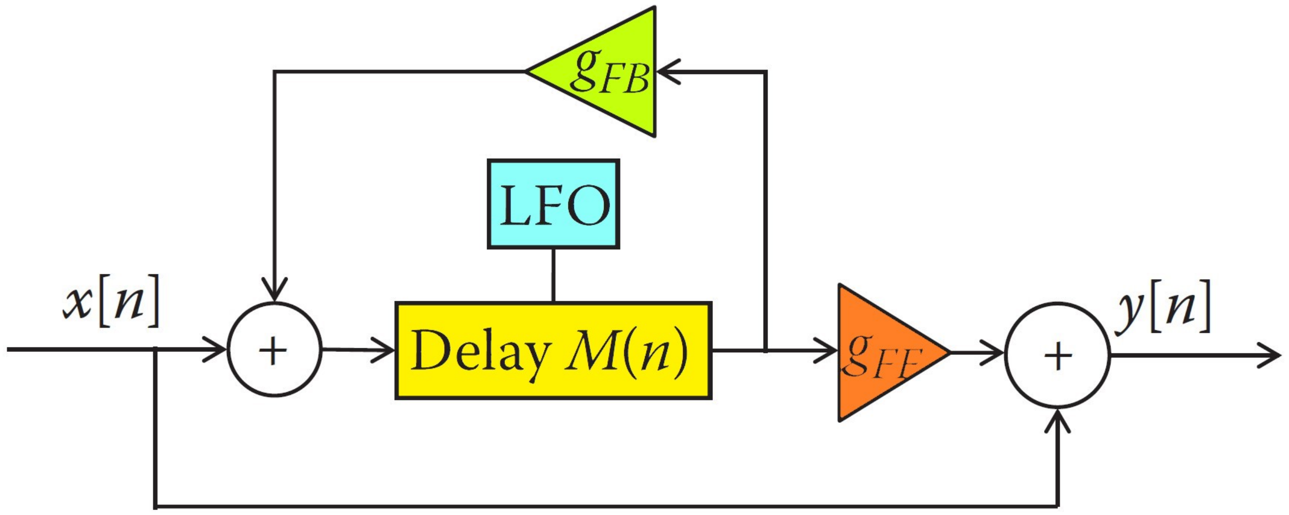


Figure 1: Block diagram of a flanger with feedback

2.1 LFO

The LFO is essential for the correct functioning of the flanger effect. This Low Frequency Oscillation modulates the total delay time, and the maximum value of delay time is the sum of the delay and the LFO width (”sweep” in our implementation). Changing the waveform of the LFO, the behavior of the flanger changes. The two copies of the signal, original and delayed one, are mixed together, and constructive and destructive interference will be present, with the creation of some resonances and a series of notches in the spectrum of the mixed signal. This type of effect is called comb-filtering. Since the delay time is varying, these points of interference will change during the time evolution of the signal (the comb-filter notches are constantly moving).

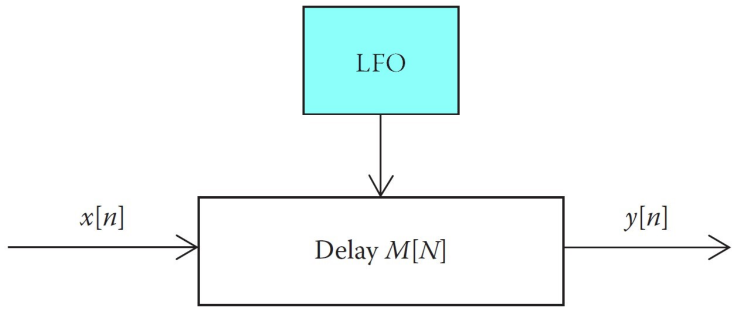


Figure 2: LFO block

2.2 Delay

Delay time is a crucial parameter in the implementation of the flanger. Humans cannot distinguish two copies of the same signal that have a time delay ¡ 50 milliseconds. Flanging exploits this concept setting the delay time at about 25 milliseconds and lower (5-25 milliseconds in our case), totally avoiding the perception of the echo. Chorus effects exploit the same principle of the flanger but the time delay is constant (not controlled by an LFO) and a slightly higher value (around 15-35 milliseconds).Phasers work, instead, is frequency-based and works with a phase delay, whereas flanger is time-based and works on a time delay.

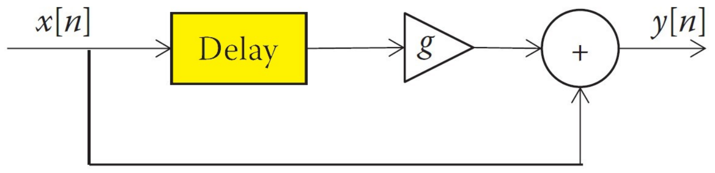


Figure 3: Delay block

2.3 Feedback

This flanger has also a feedback option, that controls the amount of delay that is fed back to the input, causing several successive copies to gradually decay over time. Technically, feedback sharpens the notches and peaks that are present in the frequency spectrum of the signal, increasing the intensity of the effect. Feedback value must be strictly minor than 1. Higher values modify the colour of the sound.

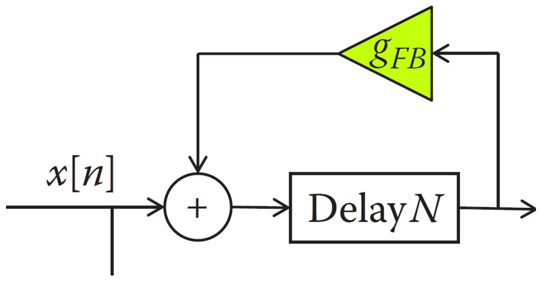


Figure 4: Feedback block

2.4 Mix (Delay Gain)

Delay gain affects the amount of delayed signal that is mixed in with the original. If its value is 0, no effect is produced, while the maximum flanging effect is obtained for the value of 1.

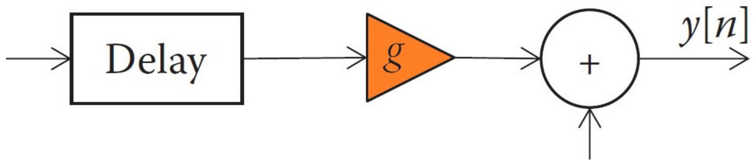
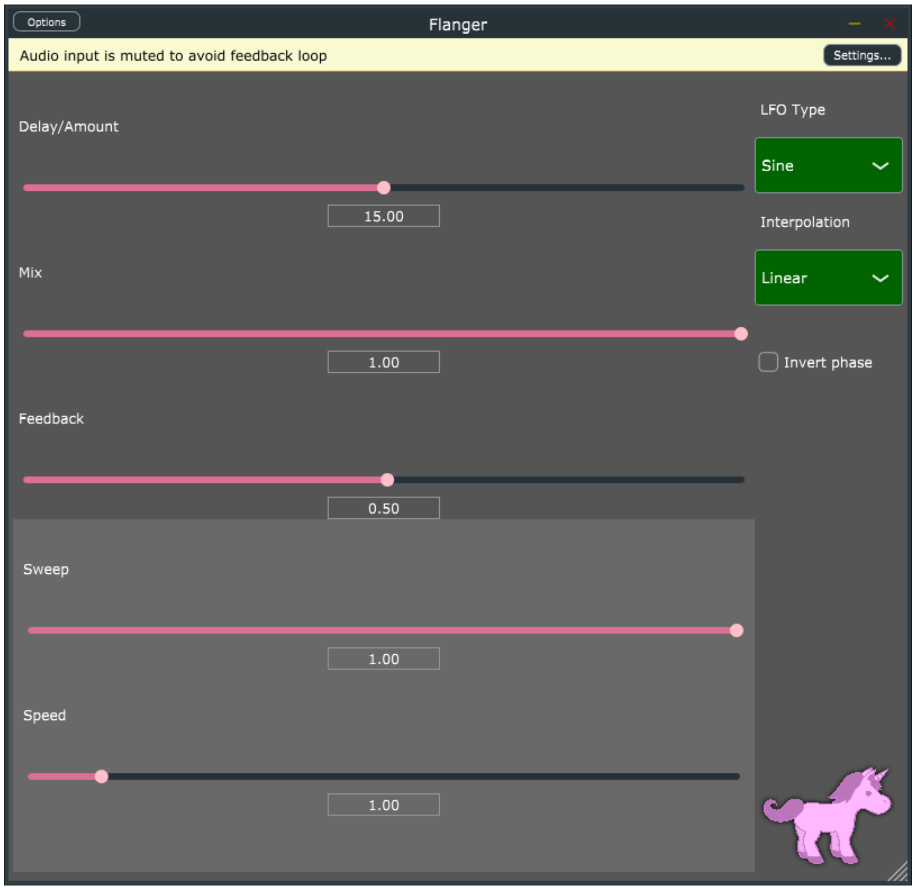


Figure 5: Delay gain block (mix parameter)

3 Implementation

3.1 GUI

Audio plugins tend to feature *skeuomorphic* graphic interfaces that mimic old-school physical equipment. They tend to rely on fixed window sizes, textures that emulate materials, and unreadable fonts.
We wanted to create a usable and accessible interface, suitable for use with modern computer systems, not for satisfying somebody's nostalgia. Some real consideration was paid to Human Factors parameters, even if we didn't have the time for performing real user testing.



Audio input is muted to avoid feedback loop

Delay/Amount 15.00
Mix 1.00
Feedback 0.50
Sweep 1.00
Speed 1.00
LFO Type Sine
Interpolation Linear

Figure 6: GUI

3.1.1 Types of variables, types of controllers

Our usability requirements are the possibility of directly manipulating the parameters quickly, both with a mouse, with a keyboard, or with a touchscreen device.
Categorical parameters, the type of waveform driving the LFO, the time interpolation algorithm, and phase inversion, are relegated to a sidebar. Quantitative parameters are presented as a column of linear sliders. These are superior to knobs for two main reasons.
First of all, knobs look like they should be twisted, but as rotation is not particularly suited to the interaction models possible on computers users are actually asked to click and drag. Depending on the chosen implementation chosen by the plugin developer, one can be required to drag vertically, horizontally, or both, leading to frequent "gimbal-locks". These are cases in which a user attempts a motion that results in no effect on the controlled variable, or sometimes even the opposite effect than they expect.
Conversely sliders offer a multi-modal direct manipulation paradigm that offers no doubt to their usage. They can be dragged, clicked, scrolled, and set with a keyboard. As they are mono-dimensional interfaces

with mono-dimensional interaction paradigms their operation follows Fitts' Law.

$$P = C_1 + C_2 log_c(2D/W)$$

Where P is the positioning time, D is the distance between the cursor and the object, W is the size of the object, and the C parameters are specific to the pointing device. 2D interactions are therefore much slower than 1D ones.

3.1.2 Limits

The limit of our approach is that we can only control one parameter at a time. It would be possible to group two or more semantically parameters into a single multi-dimensional control, allowing for the control of more parameters at the same time. However this would require developing a new ad-hoc component, which is not particularly time consuming, but presents accessibility obstacles. Sticking to default components ensures that everything can be used with with assistive devices and voice assistants.
Both knobs and linear sliders share limits in the selection of precise values, and in the relationship between the range of motion of the controller and the range of values that makes semantic sense for the controlled variable. Knobs make sense for controlling variables that are intrinsically rotational, like angles. Knobs used for gain are a mere historical artifact. For solving the precision limits the obvious solution is a text box, in which the user can manually type whatever exact value they need. At the same time sliders allow for the rapid exploration of the variable space, allowing the user to quickly narrow down the range of values they need, and then adjust the fine details textually.

3.1.3 UI Scalability

The final issue we took care of is the scalability to different screen formats. We implemented a dynamic layout in which each element is scalable, with sensible minima and maxima to always ensure the visibility of all elements. This allows out plugin to be used at literally every screen resolution found in the computers of the past twenty years, supporting anything from VGA to the up-scaling of modern retina displays.

3.2 Implemented features

- Arbitrary number of channels:

  All the processBlock() works inside a for cycle that iterate all the available input channels
- Different wavevorms for LFO:

  Four wave form types are available for the user: Sine, Triangle, Square and Sawtooth
- LFO parameters:
  - SWEEP: wave amplitude
  - SPEED: wave frequency
- Other parameters:
  - DELAY: initial delay
  - MIX: amount of effect (wet/dry)
  - FEEDBACK: presence
  - INVERT PHASE: toggle phase inversion
- Different interpolation methods:

  Three interpolation modes are implemented: Linear, Quadratic and Cubic. They are available and, even though it doesn't really change perceptually, Cubic and Quadratic interpolations are slightly brighter.

4 Conclusions and future work

The VST Plugin GUI is no much more than a parameter setter. It might be useful to make the interface react to sound (thanks to a simple oscilloscope or a fancy visualiser) or create a different input method. Presets can be added.