Automatic Parameter Smoothing in RackAFX v6.8+

Will Pirkle

This document explains how to use the new Automatic Parameter Smoothing (APS) option in RackAFX v6.8+. This is an optional treatment you may apply to your GUI controls to prevent “zipper noise” or other clicks/discontinuities when rapidly moving controls around. This only applies to GUI linked variables of the continuous nature - *float* and *double* data types. Smoothing the *int* and *UINT* based controls is probably not what you want, and is not implemented, as these controls usually operate in a switching manner to begin with.

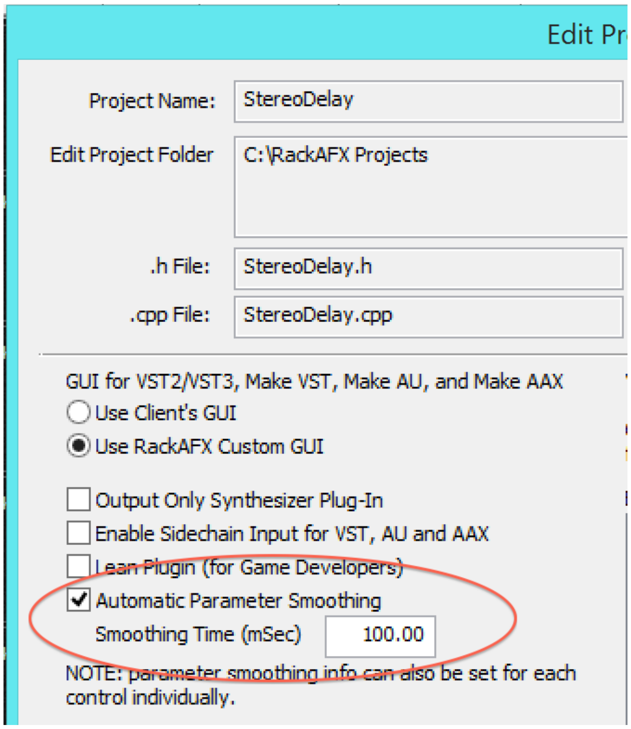
Usage

Many users have requested the ability to have the GUI controls “smoothed out” so that clicking or other side effects are eliminated. In the new RackAFX API for v6.8, parameter smoothing is implemented for you as an option. There are multiple schemes for accomplishing parameter smoothing. The MIDI Manufacturer’s Association (MMA) recommends using a ramp (linear) smoothing method by ramping the current GUI control value up or down by some small increment on each sample interval. This may be done with a linear interpolation operation which can be implemented with a first order feed-forward interpolating filter. The “smoothing time” which is the time it takes for a suddenly changed parameter to eventually reach the desired value, is controlled by the increment size.

Another option is to use a first order feedback filter in a lowpass configuration with a time constant that is equal to the desired smoothing time. RackAFX v6.8 uses this method, based on code that a user posted on the Forum several years back. The computational requirements are no different from the linear interpolator, and the GUI control changes sound smooth and continuous. The smoothing time only needs to be calculated once during setup and is easy to control or change if needed.

Setting up and Implementing Parameter Smoothing

RackAFX v6.8+ makes it simple to setup parameter smoothing on continuous GUI controls. You may enable parameter smoothing at a global project-level, or for each control individually. To turn on global parameter smoothing, use the Edit Project button (or File -> Edit Project) and check the Automatic Parameter Smoothing button. You can also set the smoothing time, which defaults to 100 milliseconds, which works well for most types of controls. Figure 1 shows the Edit Project panel:



Edit Pr

Project Name: StereoDelay

Edit Project Folder: C:\RackAFX Projects

.h File: StereoDelay.h

.cpp File: StereoDelay.cpp

GUI for VST2/VST3, Make VST, Make AU, and Make AAX

○ Use Client's GUI

● Use RackAFX Custom GUI

□ Output Only Synthesizer Plug-In

□ Enable Sidechain Input for VST, AU and AAX

□ Lean Plugin (for Game Developers)

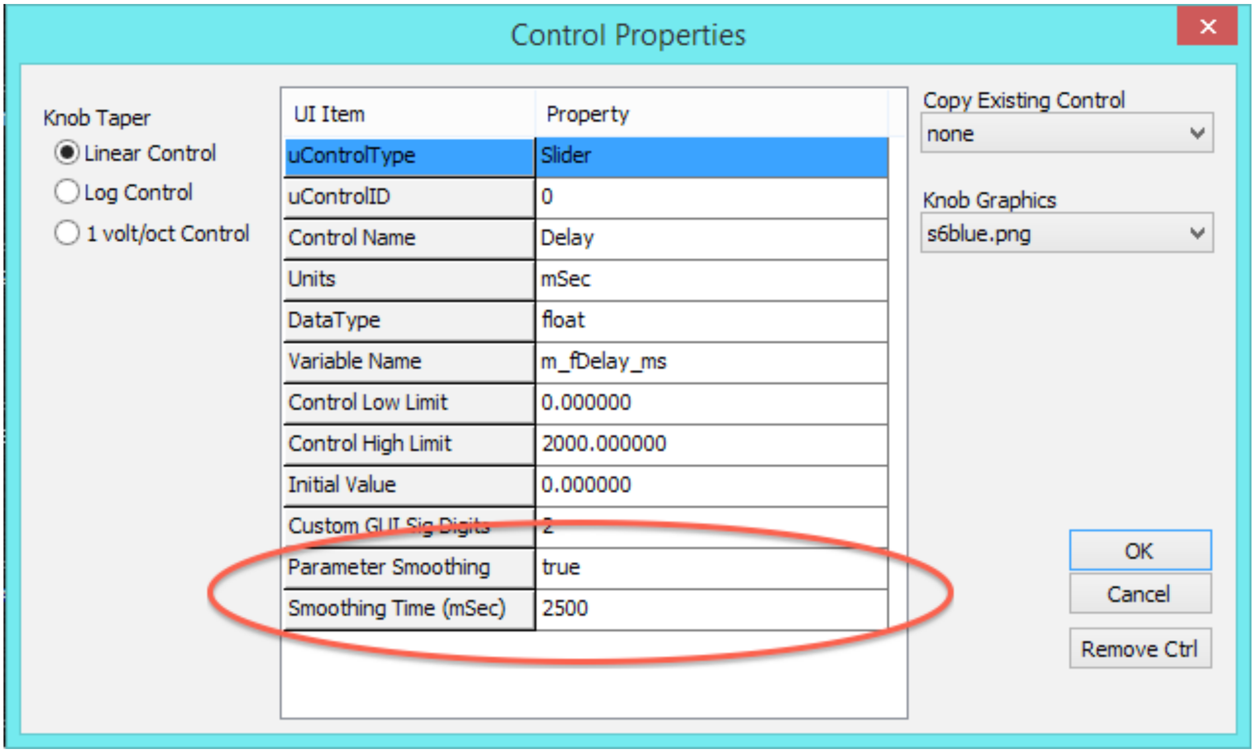
☑ Automatic Parameter Smoothing

Smoothing Time (mSec) 100.00

NOTE: parameter smoothing info can also be set for each control individually.

Figure 1: Global Parameter Smoothing on the Edit Project Panel

You may also setup parameter smoothing on a control-by-control basis; the Global Smoothing method will enable or disable all GUI controls at once, but you can also do that individually on the Control Properties panel that you get when you right-click on a GUI knob or slider (depending on whether you are using the Classic or Modern mode of RackAFX v6.8+). In this case you will see some new data entry rows that allow you to enable/disable and setup the Parameter Smoothing Time individually as shown in Figure 2.



UI Item	Property
uControlType	Slider
uControlID	0
Control Name	Delay
Units	mSec
DataType	float
Variable Name	m_fDelay_ms
Control Low Limit	0.000000
Control High Limit	2000.000000
Initial Value	0.000000
Custom GUI Sig Digits	2
Parameter Smoothing	true
Smoothing Time (mSec)	2500

Figure 2: Setting up the parameter smoothing for individual controls

In the example above, I've changed the Smoothing Time from the 100 mSec default to 2.5 seconds (2500 mSec) because this control is for a Stereo Delay (Delay Time); the 100 mSec smoothing time does not reduce the clicking enough as the GUI control is moved rapidly since the maximum delay time is a full 2 seconds.

Where is the Code?

The Parameter Smoothing manipulation code is implemented mostly on the *CPlugIn* base class. There are only two pieces of code in your main plugin object. The first is in *prepareForPlay( )* where the return value now calls the base class implementation:

```
bool __stdcall CStereoDelay::prepareForPlay()
{
       // Add your code here:
       setDelayVariables(false);

       m_DDL_Left.prepareForPlay();
       m_DDL_Right.prepareForPlay();

       // --- this sets up parameter smoothing, if enabled in RackAFX - DO NOT ALTER
       return CPlugIn::prepareForPlay();
}
```

The base class *prepareForPlay( )* sets put the parameter smoothing object and initializes everything for you.

The smoothing itself is done with a call to the (new) base class method *smoothParameterValues( )* which occurs **on each sample interval** so that the GUI control value can morph smoothly. If you use *processAudioFrame( )*, the call is made at the top of the function before any other processing is done:

```
bool __stdcall CStereoDelay::processAudioFrame(…)
{
        // --- smooth params (if enabled) DO NOT REMOVE THIS CODE
        smoothParameterValues();

        …finish processing here
```

If you use either of the buffer processing functions, the call to the smoother is done inside the appropriate loop, on a per-sample-interval basis:

```
bool __stdcall CStereoDelay::processRackAFXAudioBuffer(…)
{
        for(UINT i=0; i<uBufferSize; i++)
        {
                // --- smooth params (if enabled) DO NOT REMOVE THIS CODE
                smoothParameterValues();

                … finish processing for this sample interval
```

or:

```
bool __stdcall CStereoDelay::processVSTAudioBuffer(…)
{
        // --- setup pointers (not shown)

        // loop and process
        while (--uNumFrames >= 0)
        {
                // --- smooth params (if enabled) DO NOT REMOVE THIS CODE
                smoothParameterValues();

                … finish processing for this sample interval
```

The parameter smoothing implementation is done with a new object called *CFloatParamSmoother* which can be found in pluginconstants.h. With a little diligence, you can also create your own custom smoothing object and hook it into the code, however that is outside the scope of this document. In future versions of RackAFX, we may implement other smoothers to give you more options.

Note that the smoothing operation is a first order lowpass filter so it will add some processing: 2 multiply and 1 addition operation along with some simple logic (the same as linear interpolation code) during each sample interval. For plugins with GUI control counts greater than 50 or so, you may want to pick and choose which controls to smooth if you think that there is excess processing being used.

The parameter smoothing operation calculates the updated (smoothed) control value, then calls *userInterfaceChange( )* as if the user was actually moving the control, so there is nothing else for you to do and no extra code for you to write to take advantage of this new feature.

Old vs. New Projects

New projects already have the parameter smoothing code in them. When you open old (pre v6.8) projects in RackAFX, part of the updating process will add the code for you - if you use *processAudioFrame( )* there is nothing else to do or add; the code is written for you. If you use either of the buffer processing functions instead, a comment block has been added instructing you on where to insert the single line of code for the parameter smoothing, as in the examples above. I needed to do this as I do not know how you may have implemented your processing loop.

Ported Projects

All derivative projects (Make VST, Make AU, Make AAX) have been updated and overhauled in RackAFX v6.8 to include any other necessary code for parameter smoothing. This was easy to implement since most of the work is being done on your plugin object itself. The RAFX-as-VST-DLL option also includes the same parameter smoothing code as Make VST, so it is also implemented for you.