TClapInstrument API

Version 1.0 19 November 2022

In the past I created a little framework to create VST2, VST3 and FruityPlug instruments.

See here:

Now it is time to move on to CLAP: I don’t think anyone still uses FruityPlugs and since Steinberg forced me to delete support for VST2 it is a good time to ditch support for VST3 as well.

There are two main class: TClapInstrumentBase and TClapInstrument.

TClapInstrumentBase creates a Class which makes it easy to create a plugin without knowing the internals of Clap. You can use this for your development, or use TClapInstrument:

TClapInstrument is a small layer above TClapInstrumentBase which allready implements a few things needed: Parameter, Midi Handling, State, Editor handling.

TClapInstrument is almost fully compatible with TVSTInstrument in the past. Only a few renames are necessary.

As an example there is TmyClap which is almost the same as the previous TmyVstPlugin. Only a few renames, mostly replacing ‘vst’ with ‘clap’ !

In this document the API of TClapInstrument is discussed.

In the document Description SimpleSynth the working of the simplesynth is explained.

You can read this and that document in any order.

To make the TMyClap available as plugin, you have to describe it in a ClapInstrumentInfo record and make it available through the function GetClapInstrumentInfo. If all goes well your plugin will be instantiated in your DAW. From there you have control over it with this API.

Please use the PrjClap projects as a starting base as they include the startup code.

Declaration:

The function GetClapInstrumentInfo must be implemented delivering the following information:

clapid: A unique string for a CLAP plugin

cl The class name : here: TMyClapPlugin. It must me derived from TComponent.

Ecl The class name for your editor: here: TFormMyClap

isSynth true if this is a synth.

softMidThru if you want to pass all midi events to the next plugin

and a few other information fields.

API. The API is split into two parts:

**DSP**

Methods concerned with the audio processing part. You should not update the GUI from a DSP call.

**GUI**

Methods concerned with the parameters, presets and editor

| **DSP** | All methods are virtual and optional for processing |
| --- | --- |
| proc OnSysexEvent(s:string); | s starts with $F0 and ends with $F7 |
| proc OnMidiEvent(byte0,byte1,byte2:byte); | Remark: NOT called from UI Thread |
| proc MidiOut(const b1, b2, b3: byte); | Remark: NOT called from UI Thread |
| proc Process32(startsample,samples,channels:integer;  inputp, outputp: PPSingle); | Here you process your audio. Inputp and outputp are arrays of array with the first subscribt the channel and the second the sampleposition. See example.  New: Since CLAP is sample accurate, you must fill the buffer starting at startsample. |
| proc OnSampleRateChanged(samplerate:single); |  |
| proc OnPlayStateChanged(playing:boolean;ppq:inte); | Called when the DAW changes play state, or position |
| proc OnTempoChanged(tempo:single); |  |
| proc updateProcessorParameter(id,value) | Virtual method called when a Host parameter changes (NON-UI thread). You should update your ‘Processor’ or ‘Model’ |

| **GUI** |  |
| --- | --- |
| proc OnInitialize; | Called after creation. Here you should add your parameters |
| proc AddParameter(id:integer;title, shorttitle,units:string;min,max,val:double) | Adds a parameter to the system. Each parameter has an unique id and some other values |
| func ParamGetText(id:integer;....):string; | Value is between 0 and 1! |
| proc updateHostParameter(id,value) | Updates a parameter on the Host |
| proc updateEditorParameter(id,value) | Virtual method called when a Host parameter changes (UI thread). You should update your UI. |
| proc ResendParameters | Resends all parameters through UpdateEditorParameter (must be called from UI thread, e.g. when opening your plugin editor) |
| function getEditorClass:TformClass; | Virtual method. You can override the default creation of the editor class (as defined in GetCLAPInstrumentInfo) here (normally not needed) |
| proc OnEditOpen | Virtual method. Called when the editor opens. It could be wise to call ResendParameters and to create a mechanism so when a UI element changes you send the changed value to the host. |
| proc OnPresetChange(prgm:integer); | Virtual method. Only needed if you want to show the preset number in the UI. |
| proc OnEditClose | Virtual method. For cleanup, but normally not necessary. |
| proc OnFinalize | Virtual method. For cleanup, you should release all resources, especially timers. |
| Proc OnEditIdle | Called regularly if the Editor is open |
| procedure DoMidiEvent(byte0, byte1, byte2); | NoteOn/Off, ControllerChange |
| EditorForm: TForm | your Editor. can be NIL |

Missing from the API (will be implemented on request)

* Setting/Getting preset names. Setting the preset number (to host)
* MidiOut/MidiThrough (for)
* MidiIn stuff like MPC, aftertouch (for VST3)