FluidSynth

Bug and patch for fluid_voice_off()

Jean-jacques ceresa

- First writing: 14/02/2016: Fluid_voice_off-0001.patch
- Correction : 11/06/2017: Fluid_voice_off-0002.patch
  A bug reported by Reinhold Hoffmann from Notation Software (see 2.1.3).

1. Bug in fluid_voice_off()

1.1. Demonstration

On console type the following commands:
This command displays the number of actives voices, using fluid_synth_get_active_voice_count() API.

```
> voice_count
voice_count: 0
```

Now play a note

```
> noteon 0 60 127
```

Then display again the numbers of actives voices until it reach 0

```
> voice_count
voice_count: 1
> voice_count
voice_count: 1
…..
….
> voice_count
voice_count: 0
```

This is the normal behavior.When voice amplitude reach 0, the voice is finished by the audio rendering API.

Now play a note, and quickly type **reset** to cancel any sound (don't wait the note cancel by itself) and than display the number of actives voices.

```
> noteon 0 60 127
> reset
> voice_count
voice_count: -1
>
```

The voice count is incorrect.

1.2. Context of the bug

The bug is a bad behavior in **fluid_voice_off()** function and a bad behavior in the context of the function who call fluid_voice_off().

The function who call **fluid_voice_off()** are alls FluidSynth API on the application task side (i.e shell task or MIDI task side). Theses are:

- (1)delete_fluid_synth(), (API)
- (2)fluid_synth_all_sounds_off_LOCAL(), (API MIDI)
- (3)fluid_synth_system_reset_LOCAL., (API MIDI)
- (4)fluid_synth_update_polyphony_LOCAL, (API)
- (5)fluid_synth_free_voice_by_kill_LOCAL, API MIDI (on noteon)
- (6)fluid_synth_set_sample_rate, (API)

- (7)fluid_synth_check_finished_voices(), is call on any API. The function stop voices that have been finished by the audio rendering API.

When an application make call to one API (i.e fluid_synth_system_reset_LOCAL) these API call fluid_voice_off() one time to request voice cancellation . Then later on any API call (i.e fluid_synth_get_active_voice_count()), fluid_synth_check_finished_voices() call fluid_voice_off() one more time. fluid_voice_off is called 2 times. This is why ***active voice count*** become –1 (see 1.1). Really fluid_voice_off() needs to be call only one time.

2. Solution and patch

2.1.1. The actual fluid_voice_off (v 1.1.6)

```
/*
 * fluid_voice_off
 *
 * Purpose:
 * Turns off a voice, meaning that it is not processed
 * anymore by the DSP loop.
 */
int
fluid_voice_off(fluid_voice_t* voice)
{
	fluid_profile(FLUID_PROF_VOICE_RELEASE, voice->ref);

	voice->chan = NO_CHANNEL;
	UPDATE_RVOICE0(fluid_rvoice_voiceoff); /* request to finish the voice */

	if (voice->can_access_rvoice)
	  fluid_sample_null_ptr(&voice->rvoice->dsp.sample);

	voice->status = FLUID_VOICE_OFF;
	voice->has_noteoff = 1;

	/* Decrement the reference count of the sample. */
	fluid_sample_null_ptr(&voice->sample);

	/* Decrement voice count */
	voice->channel->synth->active_voice_count--;

	return FLUID_OK;
}
```

- fluid_synth_check_finished_voices() needs to call fluid_voice_off(), but the "request to finish the voice" (**UPDATE_RVOICE0(fluid_rvoice_voiceoff))** is not need has the voice is already finished.
- The other API (see 1.2 (1) to (6)) needs only call the "request to finish a voice". These other API need not to call the entire fluid_voice_off() job.

2.1.2. Patch **fluid_voice_off-0001.patch**

```
diff -Nur ./Fluid_1.1.6   ./Fluid_1.1.6_patch >  0001-Fluid_voice_off-0001.patch
```

The patch propose these 3 steps in files fluid_voice.c,flluid_voice.h, fluid_synth.c

Step1: fluid_voice.c

1.1)Dissociate the "request to finish the voice" in a separate function called fluid_voice_off().

```
/** void fluid_voice_off
 Force the voice into finished stage. Useful anywhere a voice
 needs to be cancelled from MIDI API.
*/
inline void fluid_voice_off(fluid_voice_t* voice)
{
        UPDATE_RVOICE0(fluid_rvoice_voiceoff); /* request to finish the voice */
}
```

1.2)In the initial fluid_voice_off() function,

- The "request to finish the voice" part code has been removed.
- Also the name is changed to **fluid_voice_stop()** for convenience as this function works the reverse of fluid_voice_start().
- The unuseful return value has been removed.

```
/*
 * fluid_voice_stop
 *
 * Purpose:
 * Turns off a voice, meaning that it is not processed
 * anymore by the DSP loop.
 */
void
fluid_voice_stop(fluid_voice_t* voice)
{
        fluid_profile(FLUID_PROF_VOICE_RELEASE, voice->ref);

        voice->chan = NO_CHANNEL;

        if (voice->can_access_rvoice)
          fluid_sample_null_ptr(&voice->rvoice->dsp.sample);

        voice->status = FLUID_VOICE_OFF;
        voice->has_noteoff = 1;

        /* Decrement the reference count of the sample. */
        fluid_sample_null_ptr(&voice->sample);

        /* Decrement voice count */
        voice->channel->synth->active_voice_count--;

}
```

Step2: fluid_voice.h

```
void fluid_voice_off(fluid_voice_t* voice);
void fluid_voice_stop(fluid_voice_t* voice);
```

Step3: fluid_synth.c,
fluid_synth_check_finished_voices() needs to call fluid_voice_stop().

2.1.3. Patch **fluid_voice_off-0002.patch**

A bug in fluid_voice_off-0001.patch has been reported by Reinhold Hoffmann from Notation Software. The bug is in delete_fluid_synth() that must use **fluid_voice_stop()** to insure unloading SoundFont data (avoiding a serious memory leak).

The fluid_voice_off-0002.patch corrects this bug and replace fluid_voice_off-0001.patch.

```
/* turn off all voices, needed to unload SoundFont data */
  if (synth->voice != NULL) {
    for (i = 0; i < synth->nvoice; i++) {
      fluid_voice_t* voice = synth->voice[i];
      if (!voice)
        continue;
      fluid_voice_unlock_rvoice(voice);
      fluid_voice_overflow_rvoice_finished(voice);
      if (fluid_voice_is_playing(voice))
          {
                fluid_voice_off(voice);
                /* If we use fluid_voice_off(voice) only it will trigger a delayed fluid_voice_stop(voice)
                via fluid_synth_check_finished_voices(). But here, we are deleting the
                fluid_synth_t instance so fluid_voice_stop() will be never triggered resulting
                in SoundFont data never unloaded (i.e a serious memory leak).
                So, fluid_voice_stop() must be explicitly  call to insure unloading SoundFont data */
                fluid_voice_stop(voice);
          }
    }
  }
```

End of patch