FINAL PROJECT

VST/AU HYBRID SOUND SYNTHESIZER

Maciej Zygmunt PIECHOCZEK

Promoter:

dr inż. Marek KOPEL

Wrocław, 2015
TABLE OF CONTENTS

1. INTRODUCTION ................................................................. 4
   1.1. Aim of the project ............................................... 4

2. THEORY .............................................................................. 5
   2.1. Elements of synthesizer ........................................... 5
       2.1.1. Oscillator ..................................................... 5
       2.1.2. Filter .......................................................... 6
       2.1.3. Amplifier .................................................... 6
       2.1.4. Envelope Generator ....................................... 6
   2.2. Types of sound synthesis ........................................... 7
       2.2.1. Subtractive Synthesis ...................................... 7
       2.2.2. FM Synthesis ................................................. 7
       2.2.3. Hybrid Synthesis .......................................... 8

3. IMPLEMENTATION ............................................................... 9
   3.1. Hybrid Synth’s structure .......................................... 9
   3.2. UML class diagram .................................................. 10
   3.3. Tools ........................................................................... 10
       3.3.1. RackAFX ..................................................... 10
   3.4. Synth engine implementation .................................... 11
       3.4.1. Oscillators .................................................... 11
       3.4.2. Envelope Generator and Operators .................. 11
       3.4.3. Algorithms ..................................................... 12
       3.4.4. Filter ............................................................ 12
       3.4.5. Modulation Matrix .......................................... 12
       3.4.6. Polyphony ...................................................... 12
   3.5. Graphical User Interface implementation .................. 13
   3.6. VST plugin implementation ...................................... 14
   3.7. AU plugin implementation ....................................... 15

4. TESTS AND PRACTICAL USAGE ........................................... 16
   4.1. Compatibility .......................................................... 16
   4.2. Performance .......................................................... 16
       4.2.1. One voice at a time, sharp release .................... 17
4.2.2. One voice at a time, long release time ....................................................... 18
4.2.3. Two voices at a time, sharp release .......................................................... 18
4.2.4. Two voices at a time, long release time .................................................. 18
4.2.5. Three voices at a time, sharp release ....................................................... 18
4.2.6. Three voices at a time, long release time ............................................... 18
4.2.7. Conclusion ............................................................................................... 19

4.3. Sound creation ............................................................................................ 19

4.3.1. E-Piano .................................................................................................... 19
4.3.2. Synth Strings .......................................................................................... 20
4.3.3. Bell .......................................................................................................... 21
4.3.4. Brassy Lead ............................................................................................ 22
4.3.5. Sci-fi FX .................................................................................................. 23

5. CONCLUSIONS ............................................................................................ 25
5.1. Further expansion opportunities .............................................................. 25

6. BIBLIOGRAPHY .......................................................................................... 26
1. INTRODUCTION

The rapid development of professional audio technology along with the improvement of computer hardware has increased the popularity of home music production. Computers these days have fast processing units, spacious RAM memory, plenty of hard drive space, MIDI protocol and 24bit audio interfaces. All these functionalities make the computer a very sophisticated music production machine. A whole song or even album can be made entirely on the laptop or even tablet computer nowadays without any lack of professional studio quality.

Music can be made using recorded real instruments or vocals; samplers; or sound synthesizers. The purpose of a sound synthesizer is to produce sound electronically or mechanically. Software synthesizers are instruments which are used entirely in a computer environment. Software plugins allow us to extend the functionality of existing software, so such synthesizer plugin can be used in any modern Digital Audio Workstation software as a live or session instrument.

In this project the effort was put forth to make such software synthesizer plugin. The synthesizer we made is hybrid, that means it combines two types of sound synthesis.

1.1. Aim of the project

There are many synthesizer plugins to choose from nowadays, so anybody can pick one that suits their needs the best. However, we decided to make another one. Most of currently existing synthesizers are very complex and hard to use, so we decided to make one that is easy and fun to use while allowing the user almost infinite soundshaping possibilities.
2. THEORY

Sound synthesis is the process of producing sound. It can reuse existing sounds by
processing them, or it can produce sound electronically or mechanically. Sounds are
synthesised using a sound synthesizer. We can call any instrument a synthesizer, since
all of them produce sound electronically or mechanically, however, recently the word
“synthesizer” has come to only mean an electronic instrument that is capable of
producing a wide range of different sounds.

2.1. Elements of synthesizer

Synthesizer is a complex device which consists of some “building blocks”. In every
synthesizer, despite of its type, we can distinguish some common elements.

2.1.1. Oscillator

Oscillators are the components that render pitched notes or act as controllers. There
are two basic types of Oscillators in a synthesizer:

- Low Frequency Oscillators (LFOs)
- Pitched Oscillators.

The LFOs potentially produce mathematically perfect waveforms since they are only
used as modulators. As their name says, they usually operate at the frequency rate lower
than 20Hz.

The Pitched Oscillators will be used for the synthesis of the notes and therefore
cannot be mathematically perfect. We need to mitigate the so-called aliasing effect.

The Oscillator is capable of producing different kinds of waveforms. Most
commonly used are:

- Sinusoid
- Triangle
- Sawtooth
- Pulse (square)
2.1.2. **Filter**

Filters are used to sculpt the harmonic content of the Oscillator output by narrowing its bandwidth. Filters may also be used in effects or their elements.

We can distinguish:

- Lowpass Filters (LPF)
- Highpass Filters (HPF)
- Bandpass and Bandstop Filters (BPF and BSF)

2.1.3. **Amplifier**

The Amplifier is an element responsible for synthesizer’s volume and pan control. Both parameters can be controlled by several modulation sources like LFO, Envelope Generator, MIDI controls like Key Velocity, Modulation Wheel, or MIDI Volume. Its volume (gain) level is most commonly controlled by Envelope Generator. In analog synthesizers Amplifier is typically referred to as Voltage-Controlled Amplifier (VCA), while in its digital applications Digitally-Controlled Amplifier (DCA).

![Figure 2.1: Detailed diagram of Digitally-Controlled Amplifier (DCA).](image)

2.1.4. **Envelope Generator**

Envelope Generators are used to shape the amplitude, pitch, and frequency domain content of the synthesized signal.

Envelope generator cycles through a set of stages depending on keyboard input events such as:

- Key Pressed
- Key Released

Most common design of Envelope Generator makes use of the following stages:

- Attack - the time between Key Pressed event and attainment of the peak level
- Decay - the time from attainment of peak level to sustain level

![Figure 2.2: Detailed diagram of Envelope Generator (EG).](image)
• Sustain - the level of modulated value in which it stays until Key Released event occurred
• Release - the time that passes between we release the key and value falls to zero.

2.2. Types of sound synthesis

There are many techniques which can be used to synthesise sound. Many of them use a “source and modifier” model as a metaphor for the process which produces the sound: a raw sound source produces the basic tone, which is then modified by in some way to create the final sound. The use of this model can be seen most clearly in analog subtractive synthesizers, but it can also be applied to digital synthesizers as well.

2.2.1. Subtractive Synthesis

Subtractive Synthesis takes a “raw” sound from Oscillator, and filters it by the Filter to remove some of the harmonic content. These Oscillators traditionally produce simple mathematical waveshapes, although Oscillators found in modern subtractive synthesizers are able to produce a wide range of sounds. Modulation of the cut-off frequency of the Filter produces the characteristic sweep sound, which is strongly associated with Subtractive Synthesis.

2.2.2. FM Synthesis

FM stands for Frequency Modulation. In this type of synthesis, the output frequency of one Oscillator called Modulator modulates the frequency of another called Carrier. In FM synthesizers, its basic components: Oscillator, VCA (DCA), and EG are grouped into blocks called Operators. Complex frequency mirroring, phase inversions and cancellations happen that can produce a wide range of timbres. The main problem with FM is that it is not possible to program it intuitively without a lot of practice, but an

Figure 2.3: Modulator - Carrier pair of Operators.

Figure 2.4: Output waveforms of the Carrier depending on the Modulators output.
experienced user can create superb sounds which are hard or even impossible to obtain by other synthesis methods.

FM as a means of producing audio sounds was first described by John Chowning at Stanford University in a paper entitled: “The Synthesis of Complex Audio Spectra by Means of Frequency Modulation”, published in 1973. This was patented in 1975 and later licensed to Yamaha. This method was the basis of the company’s flagship DX7 synthesizer introduced in 1983, which started a whole new line of digital synthesizers.

2.2.3. Hybrid Synthesis

The word “hybrid” can be considered very broad in the context of sound synthesis. It may be a combination of digital and analog elements in one hardware unit, or a combination of more than one synthesis method in one synthesizer.

In our case, this will be the combination of subtractive and FM synthesis methods, that is, sound will be shaped using FM first, and then filtered.
3. IMPLEMENTATION

In this chapter we will describe the technical details of our Hybrid Sound Synthesizer’s implementation. We are going to refer to it as “Hybrid Synth” in the later part of the thesis.

3.1. Hybrid Synth’s structure

![Hybrid Synth's internal structure diagram](image)

Our synthesizer consists of the following elements:

- Four Operators, where each one consists of Oscillator and EG. Fourth Operator has feedback loop, that means it can modulate itself.
- Noise Generator, which is an Oscillator specialized in the noise rendering
- Algorithm section which combines Operators in several ways, which results in different timbres
- Filter with EG connected
• DCA for output signal control
• Two LFOs for sound modulation

3.2. UML class diagram

![UML class diagram](image)

Figure 3.2: Hybrid Synth’s UML class diagram

3.3. Tools

We used several tools to create Hybrid Synth:

• Visual Studio 2012 Professional
• RackAFX 6.5 with the new VSTGUI4 designer
• Adobe Photoshop CS6 (for GUI background creation)

3.3.1. RackAFX

RackAFX is a handy tool which makes plugin creation a lot easier. It:

• integrates with Visual Studio and creates solutions and project templates for us
• provides GUI prototyping view
• provides tool for Graphic User Interface creation
• exports ready plugin to VST 2.4, VST 3 and AU projects
• provides analysis tools
3.4. Synth engine implementation

The instrument was written completely in C++ programming language using Steinberg’s VST and Apple’s AU APIs for plugin implementation. We also used RackAFX program for project handling and Graphical User Interface (GUI) design.

3.4.1. Oscillators

There are three types of Oscillators in this project:

- Low Frequency Oscillator - which is used as a modulation source
- Pitched Oscillator - the main sound source
- Noise Generator.

All of these are separate classes that all inherit from the Oscillator base class.

All the oscillation is done in the doOscillate() function.

3.4.2. Envelope Generator and Operators

There is single EnvelopeGenerator class in the project. It is of ADSR type. There are 6 identical objects of that class. One of them is used to control the filter’s cut-off. Another one controls Noise Generator’s amplitude. Four other are used to control pitched oscillators’ amplitude. Pitched Oscillator and EG together form the Operator - the basic block of the FM synthesizer.
3.4.3. Algorithms

![Figure 3.4: All Hybrid Synth’s Algorithms](image)

Our Operators are arranged into eight different combinations, which are called “Algorithms” here. Starting from the first, we come from bright and distorted sounds, to mellower and organ-like. Operators in Algorithm 8 are mixed together equally like in a traditional subtractive synthesizer.

Noise Generator’s output is mixed equally with all carriers’ output in each Algorithm, so it allows us to add some “air” to the result sound.

The Algorithm block is contained in the `doVoice()` method of `HybridSynthVoice` class and is implemented as switch statement.

3.4.4. Filter

We use implementation of Moog Ladder Filter in our project. This is probably the most popular type of filter because of its distinctive sound, high resonance, and self-oscillating capabilities.

It comes in six variants:

- Second and fourth order LPF
- Second and fourth order BPF
- Second and fourth order HPF

3.4.5. Modulation Matrix

We use Modulation Matrix to centralize all synthesizer’s modulation in one place. This not only makes the code easier to maintain, but also allows modulation routing “on the fly”.

3.4.6. Polyphony

Polyphony is implemented by using the array of voices. In the very beginning of the `HybridSynth.h` file we define the array’s size by the constant. Each voice is like a
separate synthesizer and consists of all synthesizer elements. If we press three keys on
the keyboard, it is like playing three synthesizers at the same time.

3.5. Graphical User Interface implementation

Hybrid Synth’s GUI was intended to be user friendly and visually appealing. It
makes use of color coding and each synthesizer’s section is logically planned.

The GUI mockup was first created in Adobe Photoshop then all knobs were placed
using RackAFX’s GUI Designer.

![Figure 3.5: Hybrid Synth’s background graphic creation in Adobe Photoshop](image)

When all knobs were placed, aligned, and tested, we removed placeholder knobs
from the mockup in Photoshop, and exported obtained background to RackAFX. Thus,
we had the GUI done.

RackAFX 6.5’s GUI Designer uses VSTGUI4 for handling the User Interface. This
standard allows us to build plugins with GUls made in GUI Designer to both VST and
AU formats. It saves us a lot of work.
3.6. VST plugin implementation

VST stands for Virtual Studio Technology. This is an audio software plugin introduced by Steinberg in 1996 for audio effects and instruments (sometimes referred to as VSTi) in 1999. It quickly became an industrial standard as other companies implemented VST support in their products.

For Hybrid Synth we used the newest version of Steinberg’s API: VST 3.6.0.

We created the VST version in the following way:

1. We used RackAFX’s Make VST function to export our project as a VST3 project
2. We downloaded VST 3.6.0 API from Steinberg’s website and unzipped it
3. We placed the project in the correct place in VST3 API’s folder hierarchy:

   ...\VST3_SDK\public.sdk\MyProjects\SynthProjects\Windows\HybridSynthVST

4. We compiled the plugin. Obtained HybridSynthVST.vst3 can be also used as older VST 2.4 format if we just change the file extension to .dll.
3.7. AU plugin implementation

AU stands for Audio Units and is Apple’s native audio plugin format. This is however very similar to Steinberg’s standard.

We created the AU version in the following way:

1. We used RackAFX’s Make AU function to export our project as a Audio Units Xcode project
2. We copied obtained project to OS X operating system
3. We opened the project and compiled the plugin in Apple Xcode environment. It was built to OS X’s default plugin directory and immediately showed up in all installed DAWs.
4. TESTS AND PRACTICAL USAGE

In this chapter we are going to test the finished product and create some sounds.

4.1. Compatibility

The compiled product is available in the following formats:

- VST 2.4 (Windows 32-bit)
- VST3 (Windows 32-bit)
- AU (Macintosh 32-bit)

VST for Mac is not supported.

This means, that Hybrid Synth can run in the following DAW programs (not all mentioned):

- Windows
  - Cubase (VST 2.4/3)
  - Nuendo (VST 2.4/3)
  - FL Studio (VST 2.4)
  - Ableton Live (VST 2.4)
  - REAPER (VST 2.4)
  - ACID Pro (VST 2.4)
  - Orion (VST 2.4)
- Macintosh (OS X 10.7 and above)
  - Logic Pro (AU)
  - MainStage (AU)
  - Ardour (AU)
  - Ableton Live (AU)
  - REAPER (AU)
  - Digital Performer (AU)

4.2. Performance

Hybrid Synth’s performance will be measured in percentage of CPU usage. We are going to test the synthesizer with different settings and active voice numbers. Ableton Live 9 (Mac) will be our test platform. We are using AU version of the plugin.
Testing machine parameters:

- MacBook Pro (15-inch, Late 2011)
- Processor: 2.2 GHz Intel Core i7
- RAM memory: 8 GB 1333 MHz DDR3
- Hard drive: 256GB 2.5-inch SSD840 Pro SATA III
- Graphics: AMD Radeon HD 6750M 512 MB
- Operating system: OS X 10.10.1 Yosemite

Size of Ableton’s audio buffer: 256 samples.

4.2.1. One voice at a time, sharp release

In an empty project, with no plugins loaded our CPU usage remains on 1% level. When we load our plugin, this level grows to 3% - 5%.

When we play one note at a time, the CPU usage oscillates between 15% and 25%. It goes back to the previous level immediately after key release. All Envelopes’ release times are set to 0mS.
4.2.2. **One voice at a time, long release time**

When we play one note at a time, the CPU usage oscillates between 15% and 25%. It stays at this level after key release as long as the longest carrier, Noise Generator, or Filter release time.

When we play another note right after we release the key, we notice a momentary jump to 28% - 35% of CPU usage. That is because the Hybrid Synth allocates new voice in the array before the first one is completely finished.

4.2.3. **Two voices at a time, sharp release**

When we play two notes at a time, the CPU usage oscillates between 28% and 35%. It goes back to the previous level immediately after key release. All Envelopes’ release times are set to 0mS.

4.2.4. **Two voices at a time, long release time**

When we play two notes at a time, the CPU usage oscillates between 28% and 35%. It again stays at this level after key release as long as the longest carrier, Noise Generator, or Filter release time.

When we play another two notes right after we release the previous keys, we notice a momentary jump to 55% - 61% of CPU usage. At this level we start to hear crackling.

4.2.5. **Three voices at a time, sharp release**

When we play three notes at a time, the CPU usage oscillates between 41% and 47%. It goes back to the previous level immediately after key release. All Envelopes’ release times are set to 0mS.

4.2.6. **Three voices at a time, long release time**

When we play three notes at a time, the CPU usage oscillates between 41% and 47%. It again stays at this level after key release as long as the longest carrier, Noise Generator, or Filter release time.

When we play another three notes right after we release the previous keys, we notice a momentary jump to 80% - 90% of CPU usage. Again, we can notice the crackling.
4.2.7. Conclusion

This version of Hybrid Synth is unfortunately very CPU expensive and needs significant optimization. However, performance has been sacrificed for the sake of code readability.

4.3. Sound creation

In this section we will test Hybrid Synth’s sound creation capabilities in practice by synthesizing both well known and completely new sounds.

4.3.1. E-Piano

FM E-Piano is one of the most distinctive sounds in the entire history of music. Due to the popularity of Yamaha DX series in the 1980’s, and the fact that these synthesizers were too hard to program for a vast majority of musicians, this stock sound quickly became a cliché.

We can very easily recreate this sound with Hybrid Synth:

1. We choose Algorithm 5. Operators A and B became our carriers, while C and D are now modulators.
2. All waveshapes are sine waves.
3. We detune Operator A slightly.
4. We shape envelope of carriers with very low sustain levels, sharp attacks, slow decays and not too long release. This gives a good base for any piano sound.

5. We increase the ratio of Operator D to maximum and add some feedback. Next, we set the output level to 56% to add brightness to the sound.

6. We increase the level of Operator C to 70% to add more middle tones to the sound.

7. We shape envelopes of modulators to 0 attack and sustain values, shorter decays than carriers’ but the same release times.

8. We leave filter untouched, since original Yamaha DX doesn’t have filters.

9. Compensate the volume level.

10. We add subtle vibrato effect (pitch modulation) from LFO1 to carrier operators, to enrich the piano sound.

4.3.2. Synth Strings

String ensemble sound is a good example of the synthetic sound inspired by real instruments which can be used in almost any of the existing music genres. This time we are going to use the full power of the hybrid synthesis. The Synth Strings sound core will be made using FM synthesis, and shaped with Hybrid Synth’s filter, to add that distinctive analog swipe sound.

Figure 4.3: Synth Strings patch
We can very easily create this sound with Hybrid Synth:

1. We choose Algorithm 1. Operator A is our carrier, while D, C and D are stacked modulators.
2. We set Operator B’s waveshape to sawtooth, Operator D’s to square, and we leave A and C’s as sine waves.
3. We detune Operator D slightly and set Operator C’s ratio to minimum.
4. We shape envelope’s of A and B to slow attacks, slow decay in A, decay of B to 0, sustain levels between 50% and 75%, and both release times to 2 seconds.
5. We add some feedback to Operator D. Next, we set the output level to about 40%.
6. We increase the level of Operator C to 30% and level of B to about 65%.
7. We shape envelopes of Operators C and D untouched.
8. We set the Filter cut-off to about 5kHz and resonance value to about 6.
9. We shape the envelope of the Filter. Slow attack and decay times, sustain to 30%, and release time to 2 seconds. We set Envelope Generator’s intensity to 80%.
10. Compensate the volume level.
11. We add subtle filter cut-off modulation from LFO1 to make sound more vibrant.

4.3.3. Bell

Bell sound is another example of distinctive FM sounds. Due to lack of base frequency this type of sound is hard to recreate in purely subtractive synthesizers. Hybrid Synth is capable of recreating any kind of bell, gong or mallet sound.

In order to synthesize the bell sound:

1. We choose Algorithm 5. Operators A and B became our carriers, while C and D are now modulators.
2. All waveshapes are sine waves.
3. We set Operators’ A and B levels to maximum.
4. We detune Operator A and B slightly.
5. We set the ratios of Operators C and D to 2.66, then we set their levels to 50% and 80%.
6. We set all Operators’ attack and sustain values to 0, decay and release times to 5 seconds.

7. We leave filter untouched, since original Yamaha DX doesn’t have filters.

8. Compensate the volume level.

4.3.4. Brassy Lead

This lead sound is an example of hybrid synthesis. It can be used in solo parts of the song, because it occupies wide spectral range.
In order to synthesize the Brassy Lead:

1. Open monophonic version of Hybrid Synth.

2. We choose Algorithm 7. Operators A, B and C became our carriers, while D is modulator of C.

3. We set A and C’s waveshapes to square, B and D’s to sawtooth.

4. We set Operators’ A, B and C’s levels to maximum and D’s level to 77.5%.

5. We set D’s feedback to 1.6 to shape the unique timbre.

6. We set the ratio of operator B to 0.5, to lower it by the one octave.

7. We set all release times to 100mS.

8. We create the fast vibrato effect on Operators A and C by setting LFO1’s rate to 35.6 and amount to 54%.

9. We set the Filter cut-off to about 7.5kHz and resonance value to 3.87.

10. We shape the envelope of the Filter. Fast attack and slow decay. Sustain to 37%, and release time to 211mS. We set Envelope Generator’s intensity to 81%.

11. We make slow swipe effect by modulating Operator D’s amplitude and Filter cut-off value by LFO2. We set FCut modulation value to -0.17%, LFO2’s rate to 0.2 and its amount to about 70%.

12. We turn on the legato mode and set glide time to 70mS. This will glue subsequent notes together.

13. Compensate the volume level.

4.3.5. Sci-fi FX

Finally, we are going to make a special effect sound using subtractive synthesis only. This sound is a special effect known from very old science fiction movies. This kind of sound was originally created using Moog’s modular synthesizers.

In order to make this effect:

1. We make sure that all Operator’s levels are set to 0.

2. We set Noise Generator level to maximum.

3. We set Filter type to one of band pass variations, cut-off to about 2kHz and resonance value to maximum.
4. Since the high resonance value causes very loud self oscillating effect, we want to lower the main volume as much as we can to prevent sound of clipping and protect our hearing.

5. Now we set LFO1’s waveform to random, FCut modulation level to 100% and we can play around with different rates and amounts.

6. We can also try different filter cut-off frequencies.

Figure 4.6: Sci-fi FX patch
5. CONCLUSIONS

The project’s goal have been reached.

Our hybrid synthesizer:

- combines two types of synthesis: FM and subtractive
- is polyphonic
- allows many sound modulation possibilities
- is easy and fun to use in comparison to contemporary existing software synths which are most often very complex and require some learning time from the user
- has appealing user interface
- arouses creativity
- can serve as the main and only synthesizer during whole production process
- is multiplatform
- works in vast majority of modern Digital Audio Workstation programs

To summarize, the program meets the requirements.

5.1. Further expansion opportunities

Hybrid Synth can be subject of further expansion.

To make it even better and more universal, we can for example add:

- pulse width modulation
- stereo shaping controls to operators
- unison section
- second filter with parallel/serial connection switch
- filter keytracking
- polyphony voice number selector and/or mono button
- modulation section which will allow us to use such sources as velocity, keytrack, modulation wheel, aftertouch, breath control, and expression pedal to modulate parameters of the synthesizer
- patch selector and memory

If we add all of these features, Hybrid Synth would become a real alternative for commercial synthesizers.
6. BIBLIOGRAPHY

