Documentation for the piece “I solens flint 1000 floder” by Mads Kjeldgaard

Source material and production

“I solens flint 1000 floder” is an electronic composition by Mads Kjeldgaard. It was composed in 7th order ambisonics, partly at NOTAM in Oslo, Norway, and partly at EMS, Stockholm, Sweden. The title (which is in Danish) can be translated to something along the lines of “In the shard of the sun 1000 rivers”. The piece explores concepts of environment and potential and uses a mixture of concrete and synthetic material as it’s source material.

The spatialization in the piece was achieved using a combination of the Reaper DAW, IEM Plugins and custom scripting. The main spatialization technique was the encoding of mono or stereo material using IEM’s StereoEncoder and multi channel material using IEM’s MultiEncoder.

Structure

The piece has two overall major components that intersect with eachother in the beginning and the end of the piece.

The first component (audible in the intro and outro) consists in the higher parts of the spectrum of granular structures which are contrasted and reactive to a low frequency whooshing sound. From here on referred to as “The granular component”.

The second (and arguably main) component, situated in the middle part of the composition, consists of a vortex of flutelike sounds. From here on referred to as “The vortex component”.

A third component is the sound of an icey river recorded in a norwegian forrest in winter time (see cover photo).

Component one: The granular component

The source material for this component consits mainly of hydrophonic recordings from a thawing lake in central Copenhagen, Denmark. These recordings originally contained a lot of background noise, partly due to the recording equipment but most of all due to traffic sounds. The noise was removed using the Izotope RX software and what was left was a crackling sound existing in a vacuum of sorts with no background. These were then edited further both manually and using custom scripting, which will be covered later on in this document.

The low, whooshing sound in this component is a processed recording of a hand paper dispenser from a bath room.

Figure 1: The author gathering material for this piece

Component two: The vortex component

The source material for this component was synthesized and recorded in Studio 3 at Elektron Musik Studion in Stockholm, Sweden.

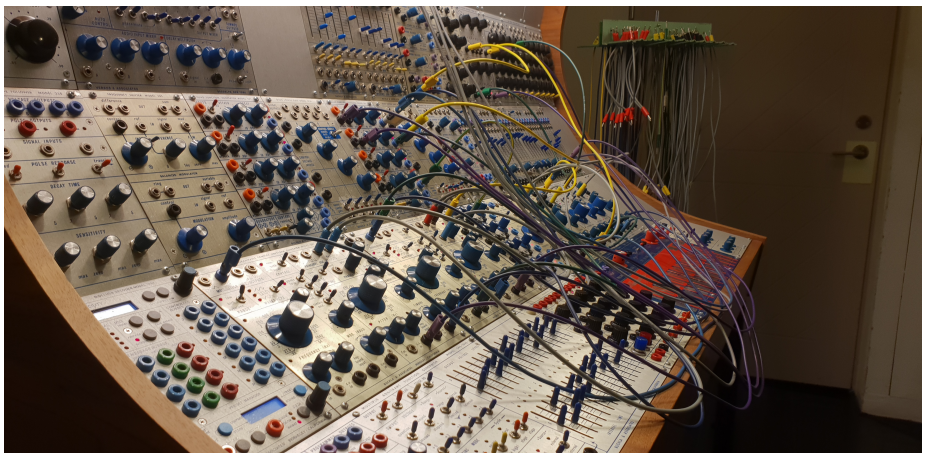

Figure 2: The Buchla 200 at EMS. Photo: Mads Kjeldgaard

During a residency there in 2018, I experimented with self playing feedback system patches on the Buchla 200 synthesizer available in this studio. The outcome of this was more than 25 hours of material.

Scripting

Random envelope points

A central script in the composition of this piece was one that would let the user select a range of items on a track and an envelope point (in this case the azimuth and elevation of an ambisonic encoder) and then at each starting point of each item insert a random value. The interpolation curve between these points can then be selected using different versions of the script and then edited further like any other envelope point in Reaper. In the end, in this composition this technique was mostly used to automate small deviations in volume on items that were copied several times on the same track.

There is a problem though: Many composers of ambisonic sound seperate the source and the encoding into two different tracks. The encoder being on the parent track and the source on the child track. This poses a problem in Reaper when moving material around (especially using Ripple Editing) since the source track's items and the parent track's envelope points are not connected. This can be solved by inserting an empty dummy track on the parent track containing

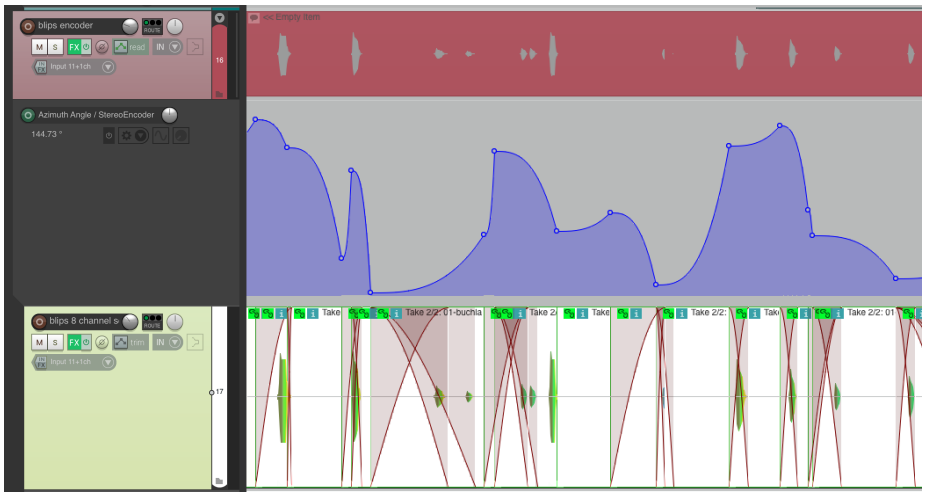

Figure 3: Envelope points inserted using mk_Insert random envelope points for selected items and track envelope (exp interpolation).lua

the encoder and automation. This will make Reaper move the envelope points during Ripple Editing.

The full source code for this script can be found at github.com/madskjeldgaard/ReaScripts.

Below is the code for the central part of the script written in Lua.

```lua
local ran_env_lib= {}

-- Insert random envelope points for selected items and track envelope (no interpolation)
num_items = reaper.CountSelectedMediaItems( 0 )
--
-- num_items = reaper.CountTrackMediaItems( reaper.GetSelectedTrack( 0, 0) )

-- MAIN
function ran_env_lib.main(envelope_interpolation)
    -- Set random seed from operating system's time
    math.randomseed( os.time() )

    -- Loop through all selected items and perform the action
    -- Argument is interpolation amount
    items_loop(envelope_interpolation)

    -- For some reason this is needed to update the arrangement
    reaper.UpdateArrange()
end
```

```lua
-- Insert envelope point
function insert_point(in_env, position, interpolation)
    if in_env then

        -- Generate a value
        local env_val = math.random(0, 1000) / 1000
        env_val = env_val * 1.0 -- Envelope values are 0.0-2.0

        -- Delete old point
        reaper.DeleteEnvelopePointRange( in_env, position, position+1 )

        -- Insert new point
        reaper.InsertEnvelopePoint( in_env, position, env_val,
                interpolation * 10.0, 0, false, false)

    else
        reaper.ReaScriptError( "No envelope selected" )
    end
end

-- Go through all selected items and insert points
function items_loop(master_interpolation)
    if num_items then

        for i_num=0, num_items-1 do

            -- Item
            local item = reaper.GetSelectedMediaItem( 0, i_num )

            if item then
                -- Position
                local i_pos = reaper.GetMediaItemInfo_Value( item, "D_POSITION" )

                -- Selected envelope
                local sel_env = reaper.GetSelectedTrackEnvelope( 0 )

                -- Set new point
                insert_point(sel_env, i_pos, master_interpolation)

                -- Sort the new points in time
                reaper.Envelope_SortPoints( sel_env )
            end
        end

    else
        reaper.ReaScriptError( "No item(s) selected" )
```

```
    end
end

return ran_env_lib
```

Recursive processing

Another central part of the script writing process for this piece was the development of a range of scripts to help automate a process of electro acoustic manipulation in Reaper. These were spread out over a range of small scripts, making them easy to incorporate into other custom actions in Reaper (these are collected in the fx subfolder of the ReaScripts repo).

The core part of this technique was a modified version of a script originally written by Michael Pilyavskiy aka. mpl which is available here. This original script was written with an incorporated gui to help the user make sophisticated "morphing" gestures between random fx parameters.

This was in my case shaved down to a core script that randomizes all fx parameters of a focused vst plugin in Reaper and combined with a script that would apply the randomized fx parameter settings to the item and render it as a new take, making it possible to browse between original and manipulated versions of the item using Reaper's take functionality.

Another script was then developed to do all of this recursively and repeatedly so that I could automate the creation of many different versions of the same item. And this was then finally developed into more sophisticated versions that would stretch or reverse the items on each iteration and then either feed the manipulated version or the original version to the next iteration of the process (the user can chose between different versions of the script to faciliate this).

An example of this can be heard in the granular component of the piece, where the high pitched sounds of the crackling ice source material was run through this process using a delay that on each iteration was randomized, the item's playback speed changed and then potentially reversed.

The full source code for this script can be found at github.com/madskjeldgaard/ReaScripts.