



Sequenc0r

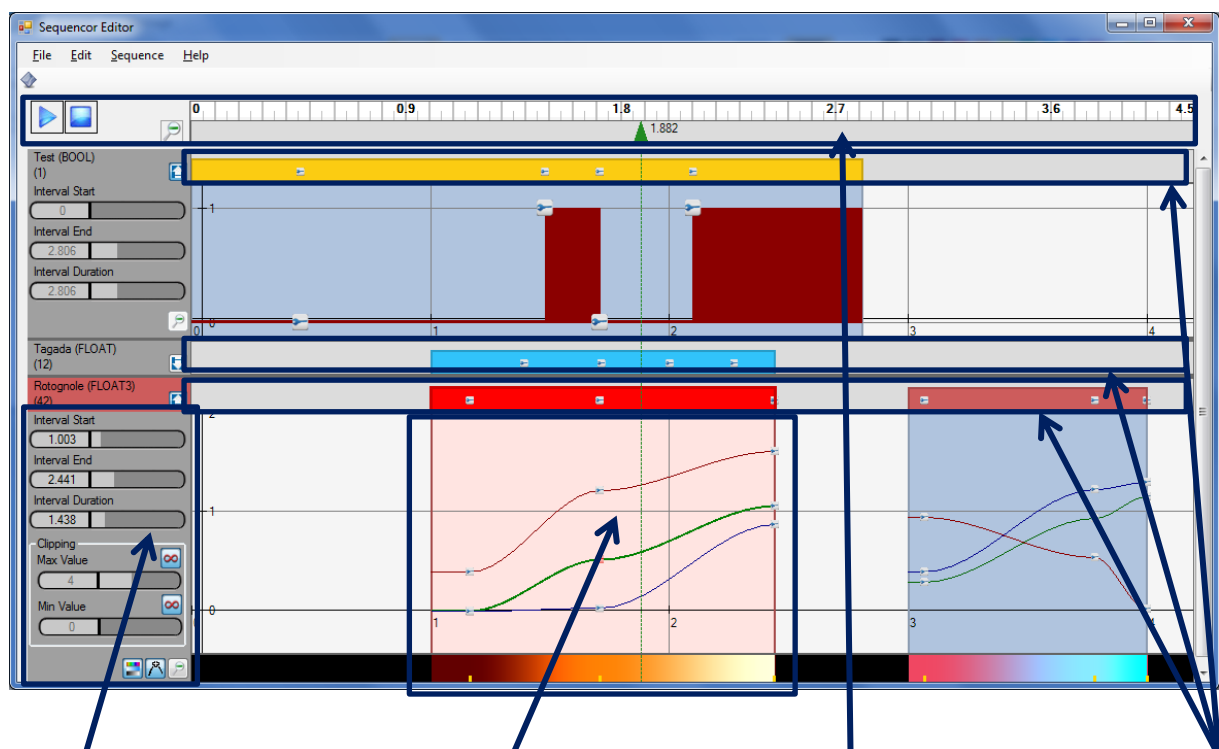
A cute sequencer for demomaking.

Creating visual effects is easy but synchronizing and animating the parameters of the effect with a music is the real effort. As I am lazy, I wrote a simple sequencer software to help us out.

The Frontend

Table of Contents

Time Line	2
Edition Mode	2
Play Mode.....	2
Creating a new Project	2
What's in a Project ?	3
What are Sequencing Data ?	3
Parameter Tracks	3
Intervals.....	4
Animation	4
Left Control Panel.....	5
Main Control Panel.....	5
Tangents Edition.....	7
Gradient Control Panel.....	7



Interval Editor

Animation Editor



Time Line


Parameter Tracks

Time Line

The time line is the central component of the sequencer as it drives the time simulation. All objects in the sequencer (intervals or keys) can anchor or be set to the current cursor time to facilitate synchronization. The cursor time can also be set to a precise time through the context menu « Set Cursor at Time » option.

Edition Mode

You can play/pause the sequencer by pressing the  toggle button in the top left corner or the SPACE bar. You can stop the sequencer and rewind the cursor to 0 by pressing the  button or BACKSPACE.

Pressing the  button will either zoom out to fit the entire sequence if not interval is selected, or fit only the selected interval in the time line.

If you left click and drag the upper part of the time line (i.e. the white part with graduations), then you can scroll the time line.

If you left click and drag the lower part of the time line (i.e. the gray part with the cursor) then you move the cursor within the time line's boundaries.

Using the mouse wheel will zoom in and out.

Play Mode

In « play mode », you can press the left arrow to rewind 1 second or the right arrow to fast forward 1 second.

COOL FEATURE : If you have selected an *EVENT* parameter, pressing **RETURN** will drop an event key. The *EVENT GUID* of the dropped key is the one last edited in the Key Editor Form (cf. chapter on Key Editing). This feature helps you created synchronized keys when you hear a specific beat in the music.

Creating a new Project

You can create a new project by pressing Ctrl+N or through the File menu.

You can open an existing project by pressing Ctrl+O or through the File menu.

What's in a Project ?

A project contains user informations like the last position of the cursor, whether tracks are unfolded or not or the colors of the tracks but also and mainly, the music that was last used and the sequencing data.

What are Sequencing Data ?


The sequencor library is organized in the following way :

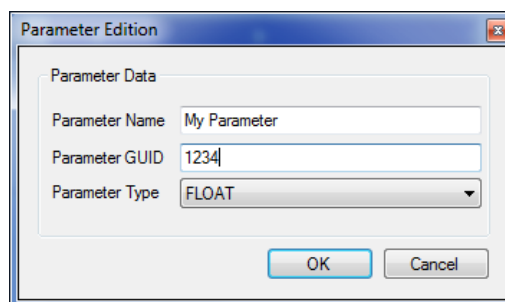
- Parameter Track #0
 - Interval #0
 - Animation Track #0
 - Key #0
 - Key #1
 - Key #2
 - (some more keys...)
 - Animation Track #1
 - Key #0
 - Key #1
 - (some more keys...)
 - (some more animation tracks...)
 - Interval #1
 - Intervals and keys...
 - (some more intervals...)
- Parameter Track #1
 - Some Animation Tracks and Intervals and Keys within them...

Parameter Tracks

Basically, that means from an empty project, you can create « **Parameter Tracks** ». The following parameter types are supported :

- **BOOL**, a boolean parameter that you can animate as true or false values
- **EVENT**, a parameter that will trigger events every time a key is crossed over
- **INT**, an integer parameter
- **FLOAT**, **FLOAT2**, **FLOAT3**, **FLOAT4**, scalar or vector floats that can also be used as colors in the case of **FLOAT** (grey level), **FLOAT3** (RGB) and **FLOAT4** (RGBA)
- **PRS**, Position/Rotation/Scale animation keys to drive the animation of objects

You can create a new parameter from the « Sequence » menu or through the toolbar by clicking the  button. You are then shown with a form waiting for data :



The image shows a 'Parameter Edition' dialog box with the following fields:

- Parameter Name:** My Parameter
- Parameter GUID:** 1234
- Parameter Type:** FLOAT (selected from a dropdown menu)

At the bottom, there are 'OK' and 'Cancel' buttons.

You can enter an optional name to clarify the usage of the parameter and the **GUID** of the parameter that will be supplied to you at runtime to identify the parameter when you receive notifications.

Finally, you must choose the type of parameter to create. Once you choose the parameter type, you cannot change it !

Intervals

Once you have created a parameter, a default interval is created for you :



You can create additional intervals by Copy/Pasting them through the context menu or by dragging them while pressing CTRL or through the « Create Interval » context menu.


You can select intervals by clicking on them, you can drag them around using left click-dragging, or expand/shrink them by left click-dragging their boundaries. Pressing ESCAPE while manipulating intervals will cancel manipulation.

Finally, you can delete intervals by selecting them and pressing DEL, or through the context menu.

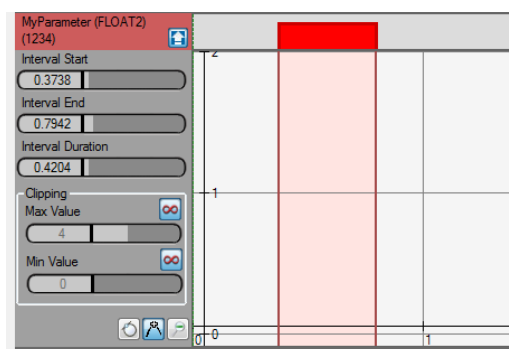
NOTE : *Intervals are automatically re-ordered when you drag them around other intervals. Their boundaries also « stick » to each other, possibly with intervals in other tracks for easy synchronization. The boundaries also stick to the current cursor position. You can disable the sticking behavior by pressing ALT while dragging.*

Left click dragging the mouse when not on an interval will scroll the time line, while mouse wheel will zoom in and out, exactly the same as with the time line described above.

Animation



You can unfold the animation editor either by double clicking an interval or by clicking the little  button in the left part of the parameter track.


You are then showed with the animation editor :







Left Control Panel

On the left side of the animation editor, you can precisely change the selected interval's start and end time, as well as its duration.

If your parameter is an INT or any of the FLOAT types or POSITION or SCALE, then you can set the minimum and maximum clipping values. By default, the  symbol is activated, meaning that clipping does not occur and key values are free. Uncheck the  symbol to set your own min/max clipping values. Parameter values that you received when notified at runtime are then guaranteed to be in the [ClipMin, ClipMax] range.

Click on the  button in the lower right corner of the animation editor's left panel to perform a vertical fit of your keys (cf. Time Line chapter to perform a horizontal fit).



If your parameter is any of the FLOAT types, or a PRS type, then you can switch between *linear* and *cubic* interpolation mode by clicking the  /  toggle button. When in *cubic* interpolation mode, you can show or hide the tangents visualization mode by clicking the  button.

If your parameter is a FLOAT, a FLOAT3 or a FLOAT4 type, then you can also show or hide the color gradient representation of the parameter by clicking the  toggle button.

Main Control Panel

The main control panel shows the keys and curves and graduations as well as various other informations about the animation of your parameter.

The visualization of the parameters depends on its type :

- Booleans are shown as bulky rectangles either 1 unit high if the boolean is true, or flattened if the boolean is false.
- Float/Integer/Position/Scale types are shown with their interpolation curves and smaller key icon  to enable clear manipulation.
- Event/Rotation types are shown with their keys set at the proper time but at height 0 as they have no manipulable value per se (although you can spawn the Edit Key form on rotation keys to edit them as Angle/Axis values). Their key icons  are also larger than usual Float/Integer types to pick them more easily.
- PRS parameters in general are a bit « messy » as you visualize position, rotation and scale keys altogether in the same edition box. That can amount to 6 different curves + large rotation keys icons. You can use the tooltips to get informations on the keys you are about to manipulate though...

Moving within the animation panel

- Panning within the animation panel is achieved using the middle mouse button.

- The mouse wheel, as always, zooms in and out horizontally (time-wise).
- SHIFT + Mouse Wheel zooms in and out vertically (value-wise).
- CTRL + Mouse Wheel zooms in and out both horizontally and vertically.

Key Editing

For most types, you can double click within an interval to create an *interpolated key* that will take the best value at the clicked position (or 0 if it's the first key to be created).

You can left click-drag a key to move it around. Pressing SHIFT while dragging keys will constrain the motion horizontally or vertically. Pressing ESCAPE while manipulating keys will cancel manipulation.

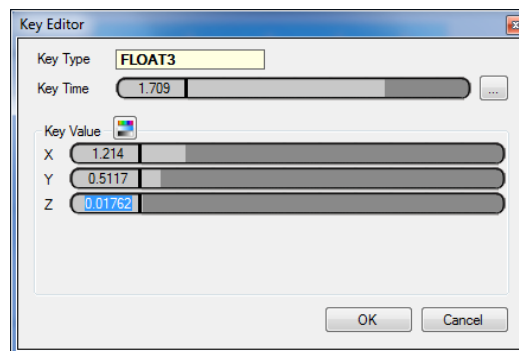
NOTE : *Keys are automatically re-ordered when you drag them around. They also « stick » to each other, possibly with intervals and keys from other tracks for easy synchronization. The keys also stick to the current cursor position. You can disable the sticking behavior by pressing ALT while dragging.*

You can create precise keys (or special type keys like PRS) through the context menu « Create Key » menu flavours.

You can delete keys by selecting them and pressing DELETE, or through the context menu « Delete Key » option.

You can copy/paste keys through the context menu, or you can also copy a key by dragging it while pressing CONTROL.

You can edit individual keys by double clicking them, or through the context menu's « Edit Key » option. This will then open a key edition form :



This is a precise way to edit time and values for a key.

Pressing ESCAPE in the animation editor while not manipulating any key will fold the editor again.

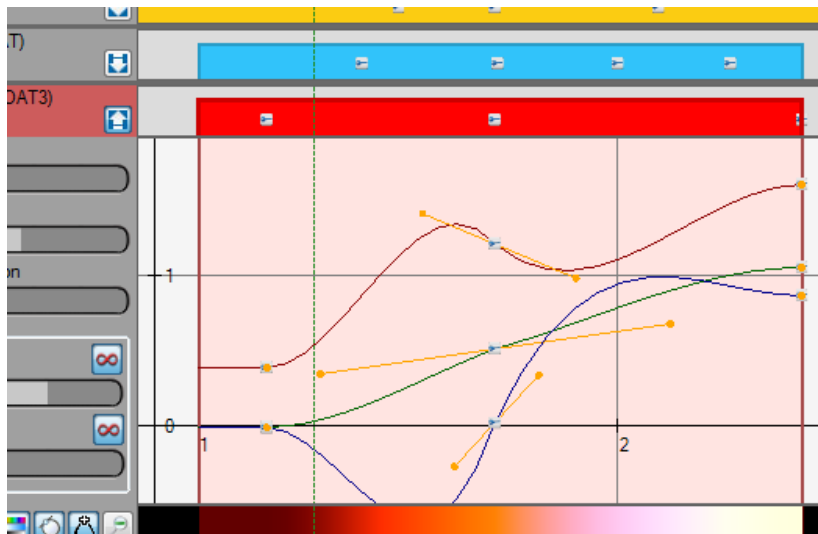
NOTE : *When creating EVENT type keys, you are asked for an EVENT GUID. This is simply an integer value that will be supplied at runtime when you subscribe to events notification. Combining the parameter's GUID and event GUID will allow you to identify and trigger the*

proper runtime event (like starting an animation, playing a sample, performing a camera shake or anything really).

Also, the last value entered as GUID will be the one used when dropping « play mode event keys » if you press RETURN during play (cf. chapter on the Time Line for an explanation on event keys quick drop).

Tangents Edition

When editing FLOAT types in *cubic* interpolation mode, if you show the tangents then you can edit them on top of standard key edition :




When simply dragging a tangent without modifier keys, the motion of IN and OUT tangents are symmetric.

If you press ALT while dragging, then you can break tangent symmetry and move either the IN or OUT tangent individually.

If you press ALT + SHIFT while dragging, this helps you to constrain the dragged tangent in the axis of the other tangent. Also, there is an anchor point when the tangent is exactly the length of the opposite tangent. This helps to re-align tangents together...

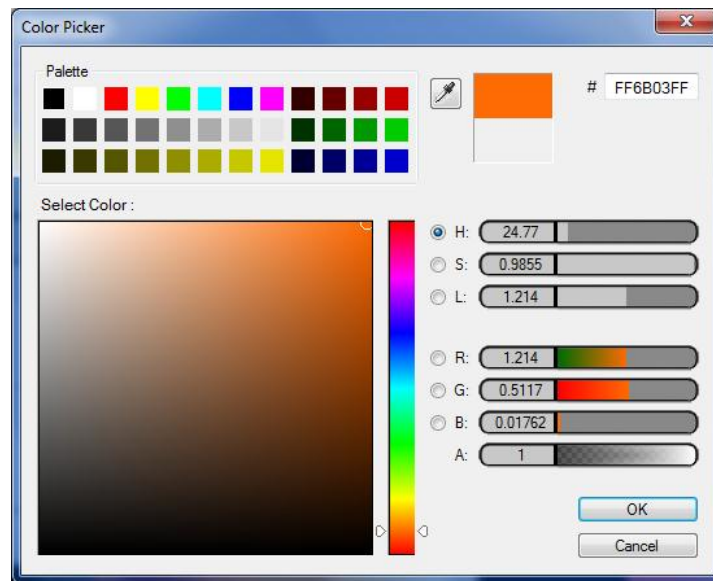
Pressing ESC while manipulating tangents cancels manipulation.

Gradient Control Panel

For color parameter types (i.e. FLOAT, FLOAT₃ and FLOAT₄), a color gradient editor can be shown by pressing the  button :



The small yellow pins represent the color keys. They cannot be moved in that editor but double clicking them will open a colour picker « à la Photoshop » :



Sequencing Runtime Library

The sequencing library used at runtime is currently only available in C# but solutions are being examined to interface it with C++ (or even port it to C++, as it's not such a humongous job after all).

The sequencing runtime library is fed with a binary that can be exported from the sequencer front end application using the “Export as Binary” option in the File menu, generating a *.sqc file.

You only need to write the following lines of code to create and subscribe to the sequencer's events :

```
SequencorLib.Sequencor Sequencor = new SequencorLib.Sequencor();
m_Sequencor.TagNeeded += new SequencorLib.Sequencor.TagNeededEventHandler( Sequencor_TagNeeded );

// Load the sequence file
System.IO.FileInfo SequencerFile = new System.IO.FileInfo( "./SomeSequence.sqc" );
m_Sequencor.LoadFromBinaryFile( SequencerFile );

// Subscribe to the sequencor events
m_Sequencor.ParameterChanged += new SequencorLib.Sequencor.ParameterChangedEventHandler(
Sequencor_ParameterChanged );
m_Sequencor.EventFired += new SequencorLib.Sequencor.EventFiredEventHandler( Sequencor_EventFired );
```

At each frame, you simply need to update the sequencer time from the music (assuming you have a music player whose position is given in milliseconds) :

```
m_Sequencor.Time = m_MP3Stream.Position * 0.001f;
```

And finally, the events are handled using that kind of code :

```
// This event is received when the sequencer needs to reconstruct its tags : associate any value to the
// parameter, like a pointer to the object driving the parameter for example...
// Retrieve the tag afterwards by using _Parameter.Tag
```

```

object Sequencor_TagNeeded( SequencorLib.Sequencor.ParameterTrack _Parameter )
{
    // Attach tags to parameters...
    switch ( _Parameter.GUID )
    {
        case 2:
            return m_MyRenderingTechnique;
    }

    return null;
}

// Occurs whenever a parameter's value is updated
void Sequencor_ParameterChanged( SequencorLib.Sequencor.ParameterTrack _Parameter,
SequencorLib.Sequencor.ParameterTrack.Interval _Interval )
{
    switch ( _Parameter.GUID )
    {
        case 1: // The FLOAT parameter driving light intensity
            m_MyRenderingTechnique.LightIntensity = _Parameter.ValueAsFloat;
            break;
        case 2: // The FLOAT3 parameter driving some color (this demonstrates the use of a tag)
            (_Parameter.Tag as RenderTechnique).WallColor = _Parameter.ValueAsFloat3;
            break;
    }
}

// Occurs whenever an EVENT type parameter's event is fired
void Sequencor_EventFired( SequencorLib.Sequencor.ParameterTrack _Parameter,
SequencorLib.Sequencor.ParameterTrack.Interval _Interval, SequencorLib.Sequencor.AnimationTrack _Track, int
_EventGUID )
{
    switch ( _EventGUID )
    {
        case 1234: // The events of GUID 1234 correspond to a sudden flash
            Flash();
            break;
    }
}

```

This is a basic usage demonstration but you can also subscribe to more intricate events, like the ones that are triggered if you step in or out of an interval (these are useful to enable or disable an effect altogether for example).

You can also subscribe to specific changes on specific animation tracks of a parameter with multiple animation tracks (like FLOAT2, FLOAT3, FLOAT4 or PRS).

Or event subscribe to the events of a single interval, discarding other intervals... The choice is yours really.

NOTE : When you receive a parameter change event from an interval, you can also query the interval's relative time (using the [SequencorTimeRelative](#) property) and normalized time (using the [SequencorTimeNormalized](#) property) which give you respectively the time from the beginning of the interval and a time in [0,1] within the interval.

Embedding SequencOr in your Renderer

TODO (only works in C# for now, and it's still work in progress anyway, but working fine already ☺ !)