URALI: A PROPOSAL OF APPROACH TO REAL-TIME AUDIO SYNTHESIS IN UNITY

Enrico Dorigatti
Conservatorio di Musica “F. A. Bonporti”
enricodorigatti@rocketmail.com

ABSTRACT

This paper aims to give a basic overview about the URALi (Unity Real-time Audio Library) project, that is currently under development. URALi is a library that aims to provide a collection of software tools to realize real-time sound synthesis in applications and softwares developed with Unity.

1. INTRODUCTION

Unity is a developing environment used for the creation of software applications, in particular videogamens and mobile apps. Today is well recognized due to the amount of instruments it offers within its environment, as well as for the great quality of the final product in general, and of the graphic in particular. It has, also, a huge community of users all over the world, made of people helping each other in a collaborative atmosphere.

Unity is well known for the visuals quality, also if everyone knows that an app -and also more a game- needs not only high level graphic, but also a great audio to result very effective and addictive. Pre-made audio clips are very well supported within Unity, with a variety of effects to manipulate it, and a virual mixer where to mix together various sources.

However, there is no only the case where scenes are built by the developer and there is a story to follow. In fact, there is also the case of the generative and algorithmic art[1], today quite popular, as well as the sonification one, where there are little or no rules, and the application evolves autonomously during time, based on algorithms and random events, as well as on rules. For those cases, a developer will likely not use a premade clip, but baybe will search for something mor organic and maybe uncommon, and, most of all, that can evolve during time related to the application's visuals.

Of course there are way to control synthesis-dedicated softwares like Supercollider and Max/MSP via OSC protocol, but going that way means the use of external tools that can work (interpret and send/receive) OSC data, and, more important, knowledge about how to operate and program an audio synthesis dedicated environment like the ones mentioned above.

Natively, inside Unity, there are no istruments that can produce sound and that can be driven by an algorithm, instruments that are a must to carry out an evolving and unpredictable sonification; instead, there is a base where to start to build a sound synthesis chain, a very interesting and useful native function called OnAuioFilterRead. It is called from a separate thread and it aims to fill a buffer with samples that are going to represent sounds when played by the speakers. However, there are no classes providing objects with which easily synthesize basic waves or carry out other types of sound synthesis; plus, creating a synthesis chain within OnAuioFilterRead is not so immediate and can easily lead to confused code, as well as a loss of performance. From here, the decision to built some personal instruments in order to obtain an algorithmic sonification, ended in building from scratch a solid and optimized library containing instruments for the so called “academic” audio synthesis.

2. GENESIS

The first project of URALi was a simple sinusoidal oscillators class meant to carry out a precise and specific task, that was providing sinusoidal waves's data ready to use in various synthesis processes (ring modulation, additive and FM), to create the algorithmic, real-time and visual-s-related sonification of *Life*, an audiovisual generative software based on a custom version of the famous J. Conway's Life algorithm.

However, the project grew up very quickly, eventually becoming the base where to start the building of a bigger library to be used for the procedural and academic-like sonification of any Unity project using it, bringing also an added value that comes from the proven extreme simplicity to integrate and correlate, in this fully multimedial environment, audio data with visual ouputs and/or algorithms, and vice-versa.

3. INSIDE URALI

URALi is, at the moment, a full-working audio library that offers basic objects for the audio synthesis. In this paragraph is proposed an overview of the actual state of completion of the library, as well as the main improvements scheduled to be implemenented as soon as possible, since there is still a lot of work to do in order to make URALi a really powerful instrument.

[1]E. Dorigatti, *“Life”*; G. Albini, *“Memoriale”*

3.1 Actual state of completion

At the moment, URALi provides an implementation of the following object and features:

- sinusoidal, triangular, sawthooth and square oscillators;
- lookup oscillator;
- PolyBLEP algorithm to antialiase harmonic waveforms;
- multi-segment (not limited to) ADSR envelope;
- granulator;
- white noise source;
- convenience objects (enveloped oscillator, additive module);
- multithreading.

3.2 Scheduled features

Among other minor features, and improvements, the prioritay ones are:

- wavetable to play user's own waveforms;
- GUI to tweak the parameters inside Unity editor and not only inside the code;
- extension of the convenience objects section, adding, among the others, modules for other types of synthesis;
- miscellaneous utility features, like the possibility to record to a wav file and to view the spectrogram and the waveform generated.

4. THE LOGIC OF URALI

It is worth to distinguish between how URALi works in the background and how is meant to be used on the final user side, to have at last an idea of the whole process.

4.1 Internal synthesis logic and design

The idea benind URALi is to avoid the saturation of the OnAudioFilterRead function with any kind of wave calculation or wavetable's sample interpolation. So it leaves all the calculation tasks to a separate and much more faster background thread, that eventually returns the result of all operations, allowing OnAudioFilterRead to carry out minor tasks, like amplitude or frequency control, without any performance loss, that would mean discontinuities in the waveform.

4.2 User side design

On the user side, URALi needs the user to write its own audio synthesis chain inside a function that has to be passed as the argument for another function that starts the audio engine. Then, various parameters -such as frequency or amplitude- of the objects can be modified in runtime thanks to public properties, changing the acoustic characteristics of the final output. To achieve an audible result, then, one have to recall inside of OnAudioFilterRead the function of the library that returns the samples, making possible to hear a sonic result at last.

5. CONCLUSIONS

URALi is still in development and that leads to having, right now, an instrument that provides only the very essential tools for the audio synthesis, also if those works perfectly. The hope is that the project (as well as the generative art as discipline) can interest more artists and Unity developers everyday, supported by a very powerful and friendly development environment.

Last note is that URALi, first of all, is my personal approach and solution to the lack of instruments for generative audio that I found in Unity. That means that it may not be ther perfect solution, nor the only one. It works very well but, as the title says, it is a proposal.