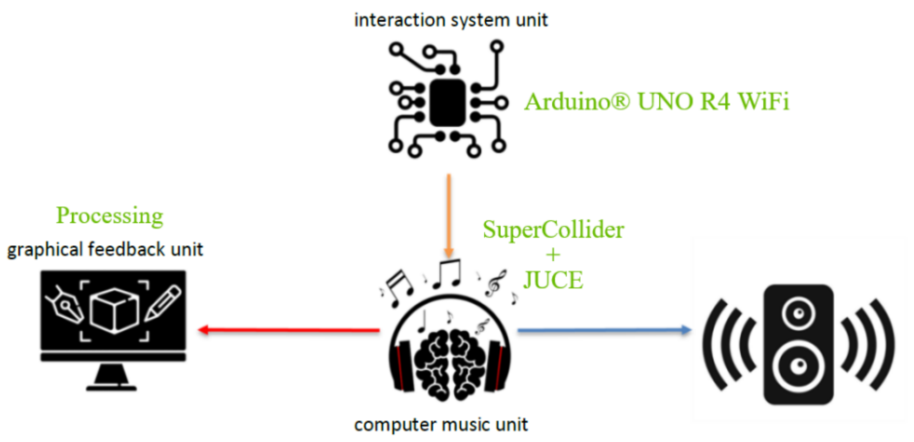
CMLS Homework Assignment

Grain of Life

Design and implementation of a Computer Music System

Francesca Benesso (248523), Matteo Benzo (259322), Filippo Garofalo (934771), Diego Pini (250641)

Abstract – The aim of this homework assignment is to design and implement a computer music system using an interaction unit (*Arduino® UNO R4 WiFi*, a device able to capture the user inputs), a computer music unit (we used *SuperCollider* to perform the synthesis and *JUCE* to implement the effects), and finally a graphical unit, to provide some graphical feedback to the user, implemented in *Processing*.
For this project, we wanted to create a system that was able to create sounds using a granulator controlled by the evolution of Conway's Game of Life. Our goal was to create mesmerizing soundscapes for a hypothetical art installation.



1. The idea: Granular synthesis based on Conway's Game of Life

1.1 What is Game of Life?

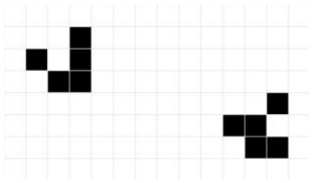
The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970.
The universe of the Game of Life is an infinite, two-dimensional orthogonal grid of square *cells*, each of which is in one of two possible states, live or dead (or populated and unpopulated, respectively). Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbors die, as if by underpopulation.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by overpopulation.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The initial pattern constitutes the seed of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed, live or dead; births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a tick. Each generation is a pure function of the preceding one. The rules continue to be applied repeatedly to create further generations.

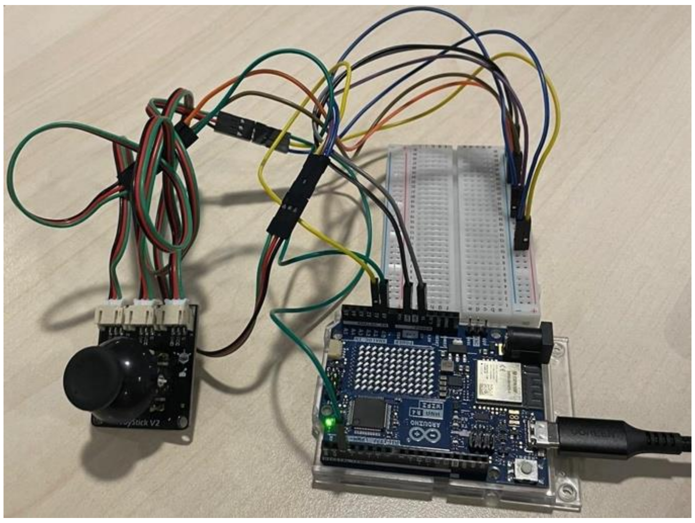
1.2 Game of Life in our project

Game of Life is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input.
So how can an external user interact with the game in our project, and consequently modify the sound generated by the granulator? Interaction is possible by entering the initial condition to start the propagation of the cells: we create an initial configuration and observe how it evolves. In addition, we can modify the evolution of the system (and consequently of the sound) also by entering other conditions during the development of the Life.
In this way we obtain a peculiar sound result: something that can evolve by itself during time, based on the cells propagation, but also influenced by the interaction of the user when he wants to alter the natural development of the Life.
We slightly modified a famous initial pattern frequently used in Game of Life: the 2-glider mess. We chose this pattern because it has a good independent time evolution, but also its shapes allow us to maintain control of the evolution without creating complete chaos, when a new 2-glider mess is added.
Finally, to perform the interaction, we used a hardware component: Arduino® UNO R4 WiFi, equipped with a Joystick module v2.



2. Interaction unit: Arduino® UNO R4 WiFi

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of a programmable microcontroller (a small computer on a single integrated circuit) and a development environment for writing software (sketches) to run on the microcontroller.
These boards, like the one we used (Arduino® UNO R4 WiFi), can read inputs from sensors and control outputs such as motors, lights, and other actuators.
A joystick module typically features a thumbstick like those found on game controllers. The module consists of two potentiometers that measure the displacement of the stick in the horizontal (X) and vertical (Y) directions, converting these movements into variable voltage signals that can be read by Arduino (analog inputs). Additionally, the joystick v2 includes a push-button feature, which is activated by pressing down on the stick, providing a digital input signal.
With X and Y coordinates we control the position of the 2-glider mess into the two-dimensional orthogonal grid of square cells. With the push-button connected to a digital pin the user can drop the pattern into the grid. When the button is pressed for 3 seconds the grid is resettled.



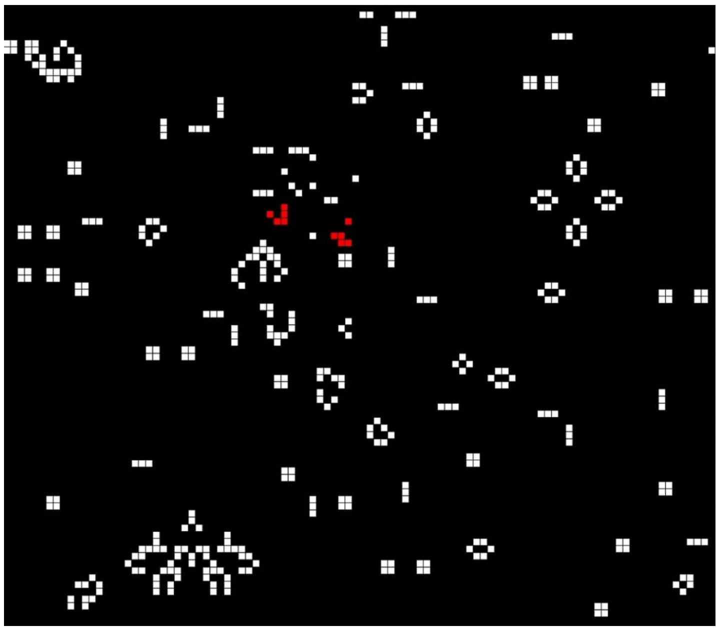
3. SuperCollider: creation of the granulator and mapping

How does the granulator interact with the Game of Life? We implement granular synthesis in SuperCollider using the GrainBuf UGen, able to perform granular synthesis with a sound stored in a buffer.
The user can insert his favorite track and adjust some parameters as his wants:

- Number of output channels: if 1, mono is returned.
- Type of the trigger of the sound (how periodically the sound is activated in an amount of time).
- Grain dimension: size of the grain (in seconds).
- The buffer containing the sound chosen.
- Playback rate (a parameter that controls the pitch).
- Position of the grain inside the track contained in the buffer (0 is beginning, 1 is end of file).
- Model of interpolation used for pitchshifting grains: 1 = no interpolation, 2 = linear, 4 = cubic interpolation (more computationally intensive)
- Panning: determines where to pan the output.
- Shape of the envelope: -1 uses a built-in Hann envelope.
- Maximum number of overlapping grains: This value is set at the UGens init time and can't be modified. Defaults to 512. This can be set lower for more efficient use of memory.

Some of these parameters give a better performance when settled at a standard value, but others can be controlled dynamically by the user. We chose some of these to be controlled by the joystick directly and by the global configuration of the two-dimensional orthogonal grid of square cells: the number of alive pixels is going to affect the overall grain density, the X coordinate is used to set the pitch in the range +/- 1 octave, the Y coordinate will increase the grains' duration randomness.
We chose to have the user control of these variables in relation to the type of sound we desire for our synthesis, but of course other configurations and experimentations are also valid as well: mapping possibilities are almost endless.

4. Processing: a powerful graphic interface



When the idea of creating a soundscape of a hypothetical art installation came up, we asked ourselves: what is the fundamental feature of a serviceable installation of this kind? Nice-looking graphical feedback to the user was our response.
We chose to use Processing to achieve such a task. Processing is an open-source programming language and integrated development environment (IDE) built for electronic arts, new media art, and visual design communities. Created with the goal of teaching the fundamentals of computer programming in a visual context, Processing is designed to simplify the creation of visual applications and interactive graphics. Its syntax and functionality make it particularly well-suited for creating graphic interfaces, animations, and interactive

installations. With a focus on ease of use, it allows artists and designers to produce complex visual effects with minimal code.
We created the two-dimensional orthogonal grid of square cells, we initialized the 2-glider mess, and finally implemented the logic for interactive game of life.

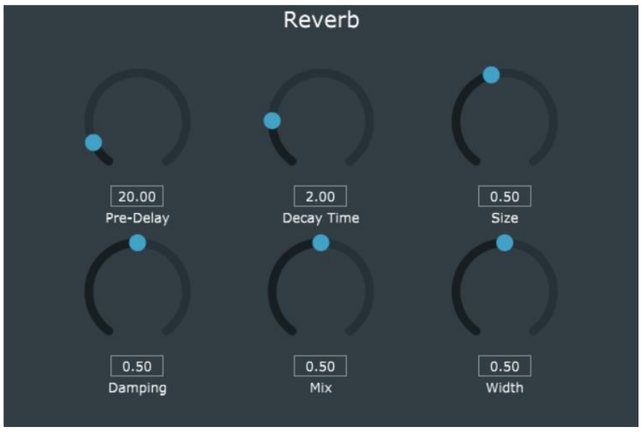
5. VST Plugins

To add depth and possibilities to our system, we chose to develop four plugins with JUCE (both the processing and the GUI are implemented with it). We opted for a reverb, an equalizer, a delay and a bitcrusher. These additions significantly enhance the versatility and expressiveness of our audio system, unlocking all its potential.

5.1 Reverb

Reverb is an essential effect for creating atmospheric ambient music.
The implementation we choose to go with has six parameters: pre-delay, decay time, size, damping, mix and width.
Here's a brief description of how each of these parameters change the sound:

- Pre-delay: time of the first reflection. A low value will make the sound muddier, while a higher value will help the clarity of the sound.
- Decay time: time required for the reflections to die away. The higher the value, the more ambience is achieved. On the contrary, a lower value will create a more intimate atmosphere.
- Size: perceived size of the virtual space simulated. A high value will make the sound appear as if it was played in a very big room, for example a cathedral. A low value will resemble smaller rooms, like a living room.
- Damping: regulates the absorption of high frequencies. A high value will darken the sound, a low one will make it brighter.
- Mix: percentage of the reverb level present in the sound. 0% means the reverb is not present. 100% that only the reverb will be heard.
- Width: regulates how wide the reverb appears in the stereo field. A wide reverberation provides a more immersive experience.


Reverb

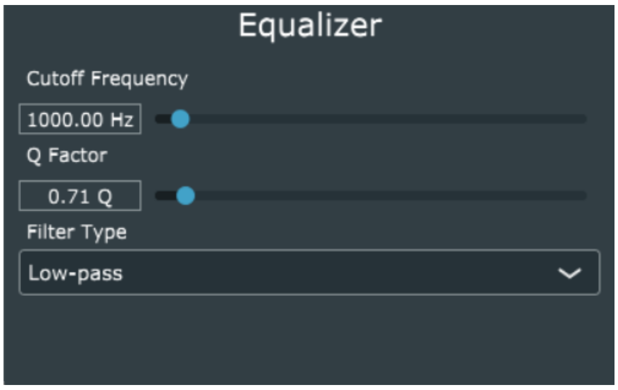
20.00 Pre-Delay
2.00 Decay Time
0.50 Size
0.50 Damping
0.50 Mix
0.50 Width

5.2 Equalizer

This plugin implements a low-pass filter and a high-pass filter. It lets you choose whether you want to use just one of them or both, resulting in a band-pass filter.
The parameters are kept minimal: cutoff frequency, q-factor and a drop-down menu that lets you choose the type of filter.
Here they are briefly explained:

- Cutoff frequency: reference frequency of action of the filter. The attenuation at that frequency depends on the q-factor.

- Q-factor: regulates the resonance of the filter. A typical value would be 0.71, that would approximate a Butterworth filter in which the cutoff frequency is attenuated by 3 dB.



Equalizer

Cutoff Frequency

1000.00 Hz

Q Factor

0.71 Q

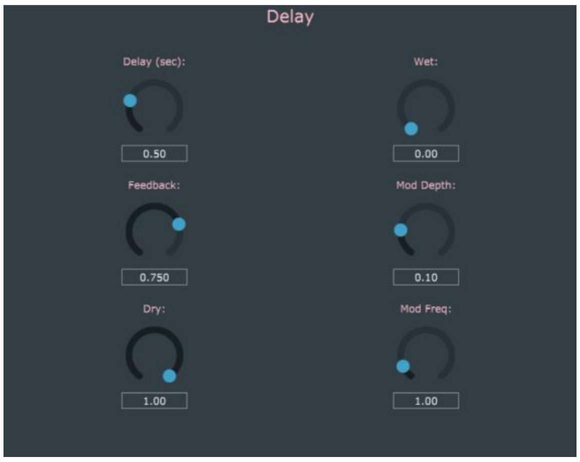
Filter Type

Low-pass

5.3 Delay

A delay was an obligatory choice for our objective. Its usage generates a hypnotic feeling to the experience. We also added the possibility to modulate the delay time in order to add some unpredictability and achieve interesting results.
The plugin lets you play with 6 parameters: delay, feedback, dry, wet, mod depth and mod freq.
Here's a brief explanation:

- Delay: time between the signal and each repetition. Measured in seconds.
- Feedback: amount of signal fed back into the input. In practice controls the volume of the next repetition compared to the one before.
- Dry: amount of the original signal at the output. 1 means 100%.
- Wet: amount of the delayed signal at the output. 1 means 100%.
- Mod depth: controls the depth of the modulation applied to the delay time. This modulation creates a varying delay time, producing a flanging effect.
- Mod freq: controls the frequency of the modulation applied to the delay time. This determines how fast the modulation oscillates.



Delay

Delay (sec): 0.50

Wet: 0.00

Feedback: 0.750

Mod Depth: 0.10

Dry: 1.00

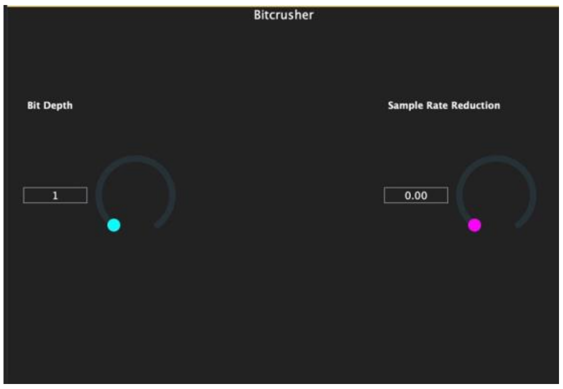
Mod Freq: 1.00

5.4 Bitcrusher

To add texture and grit we opted for a bitcrusher instead of a more conventional distortion.
It is possible to modify two parameters: bit depth and sample rate reduction.
Here's how they work:

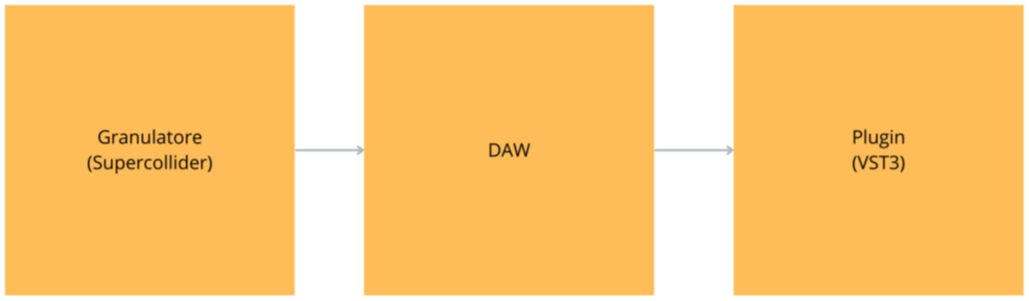
- Bit depth: controls the bit depth reduction of the audio signal by limiting the number of bits used to represent each sample. Lower bit depth values result in more quantization noise and a more pronounced bitcrushing effect.

- Sample rate reduction reduces the effective sample rate by skipping samples. Lower values result in more aliasing and a more pronounced bitcrushing effect.



6. Routing and Communications

The graphics and the code of the game of life are implemented in processing, in order to communicate with the granulator created in supercollider we use the OSC message in local machine.
The OSC message is also used to send information between the Arduino and processing (like the position of initial conditions), but in this case the OSC message is sent through the network.
OSC messages helped to put in communications different types of environment and programming languages thanks to its versatility and platform independence.
How do the granulator (made in Supercollider) and the effects (VST3) communicate? Through BlackHole (a MacOs virtual audio loopback driver), the output of the granulator is redirected to a DAW. There the audio is processed by the plugins. No latency is added in this passage.



7. Conclusions

Grain of Life finally culminated in a mesmerizing soundscape installation that seamlessly integrates various technologies to deliver an immersive interactive experience. Utilizing SuperCollider for granular synthesis, the soundscapes generated were both intricate and dynamic, laying a robust auditory foundation. The implementation of audio effects, including an equalizer, delay, reverb, and bitcrusher through JUCE, added layers of depth and texture to the sound. The graphical interface, in which Conway's Game of Life is implemented and developed in Processing, provided a captivating visual counterpart to the auditory experience, enhancing user engagement. The inclusion of an Arduino-controlled joystick allowed for intuitive user interaction, enabling participants to influence the system's output in real time. This interdisciplinary approach not only showcased the harmonious blend of sound, visuals, and user input but also highlighted the potential of interactive installations in creating evocative and participatory forms of art.