Amalgam: A live coded visualaudio performance

Timo Hoogland
HKU University of the Arts Utrecht
info@timohoogland.com

Saskia Freeke
Creative Coding Utrecht
info@sasj.nl

ABSTRACT

This paper describes Amalgam, a live coded visualaudio performance and research by Timo Hoogland and Saskia Freeke. The system uses visuals by capturing, downsampling and analysing them in multiple rows for brightness and hue to generate musical information such as rhythms, melodies and other musical parameters. With this approach we attempt to make coding of music and visuals in a collaborative setting more coherent whithout sacrificing aesthetic.

1 Introduction

Live coding performances, especially within the Algorave scene, in many cases involve two performers. A musician, live coding the music, and a visualist, live coding the projected image. Usually the formed duos are ad-hoc collaborations, for which a visualist is randomly paired with a musician and will have to match their visuals with the music being created. The result is a performance where visuals are more a supporting, complementary act than a shared main act with the musician (Street et al. 2019). In this paper we introduce the performance Amalgam. Amalgam is a live coded visualaudio performance and research by Timo Hoogland and Saskia Freeke. The performance is a result of our ongoing research in exploring the relationship between sound and visuals in a live coding context. The system proposes a way of using visuals to generate musical information such as rhythms, melodies, dynamics and other musical parameters. With this approach we attempt to make coding of music and visuals in a collaborative setting more coherent whithout sacrificing aesthetic.

In the past there have been various examples in the electronic music scene of systems designed where the sound is generated from an image, such as the Oramics Machine by Daphne Oram (Manning 2012) and software like Xenakis' UPIC system (Thiebaut, Healey, and Bryan-Kinns 2008), SoundPaint (Reuter 2005) and MetaSynth, famously used in the ending of the track Formula by Aphex Twin where you can hear and see Richard's face (Aphex Twin 1999). Within the live coding scene there have also been explorations of collaborative performances such as the work "Circulo e Meio" by Joana Chicau and Renick Bell sending OSC-messages between their systems (Bell and Chicau 2018).

This paper describes some of the early experiments as well as the most recent setup. We elaborate on the technical aspects and the various (live coding) languages, environments and technologies used, such as Mercury and Hydra, the Flok collaborative editor and MaxMSP visual programming environment. Furthermore we reflect on the current setup, what the challenges are and what we would like to work on for future improvements.

2 Early experiments

Our first exploration started out as an experiment for the Art Machines festival organised by Creative Coding Utrecht (CCU) in 2019 for which we were asked to live code visuals together. A fork of the Mercury live coding environment (designed by Timo Hoogland) that only uses the Jitter visual programming of MaxMSP as its library was created. Freeke prefers to code using Hydra, a web-based visual synth created by Olivia Jack (Jack 2019). The 2-dimensional visuals were send over a network as a texture with the NDISyphon application, using the NDI protocol. Next the visuals were mapped on 3-dimensional shapes in the Jitter OpenGL environment. In this collaboration Freeke focussed on the patterns, colors and geometry of the texture, while Hoogland arranged multiple textured cubes in a 3D world and let them react to the sound. After this performance the decision to continue with this visual collaboration was made and we performed with this setup at other events such as Github Satellite in 2020 (Satellite 2020).

Since Hoogland mainly focuses on live coding of electronic music the decision was made to have Freeke program the Hydra textures and also control the 3D environment where pre-programmed configurations of the cubes could be selected with a Launchpad. In a second iteration of the project we experimented using the brightness values of

the Hydra textures as a heightmap for a single cube in the 3-dimensional space. This version was performed at the Iterations Symposium by CCU in Amsterdam (Utrecht 2021) and at Ohme in Brussels (Ohme 2021). In order to take the networked performance further we started experimenting with the Flok web-based collaborative editor designed by Damián Silvani (https://flok.clic.cf) (Silvani 2022). This allowed us to live stream a performance for which we would not have to be in the same room (Joana Chicau 2021).

3 Reflection and moving towards a more collaborative audio and visual system

While the versions of the project up till this point did combine the visuals from Hydra in the 3D visual environment of Mercury to extend the visual output, it was still only a "one way street" audiovisual performance where the visuals react to the sound, but not vice versa. During rehearsals and experimentation we made similar observations as addressed in a visualists meetup in Sheffield in 2018 (Street et al. 2019). We found that in many instances visuals are a secondary aspect to the live coding performance, as a complement to the music. Also for the musician it is usually difficult to see the visuals and adjust the music based on that, since the projection is behind them. At the same time it is also difficult for visualists to anticipate what the musician will do next and adjust the visuals based on that. We think it could be interesting for the visualist to have a more direct impact on the music that also allows to make changes in the composition during a live performance without sacrifising their visual aesthethic. We came to the conclusion that we wanted to experiment with making a system that makes an analysis of the visuals and allow Hoogland to use the analysis for musical aspects.

4 Visualaudio? Is this a typ-o?

The title of this paper reads: Amalgam: A live coded visualaudio work. The observant reader will notice that the words audio and visual are swapped. Usually a performance is described as audiovisual when the work consists of sound or music and a complementary visual aspect. By swapping the words we want to emphasize that this work focuses on letting visuals control parts of the sound and composition and therefore make the music visual-reactive as opposed to the more common practice of audiovisual works where the visuals are audio-reactive.

Furthermore the title Amalgam was chosen because this refers to the alloys that can be formed through metallic bonding of a metal with the element Mercury. With the title we suggest that the live coding language Mercury formed a bond with the other language (in this case Hydra) and out of it originated a new combined material.

The visual style of Freeke has an interesting connection with the style in which Hoogland programs music. Similar to many other live coding languages, Mercury is based on the composition technique of (total) serialism (Hoogland 2019). An approach to composition that originated from the twelve-tone music, for which every aspect of the composition is described as series of values. Separate series for pitch, rhythm, dynamics and other musical aspects are created in different lengths and sequenced over time to create an algorithmic composition. Freeke's visual style also consists of series, but in the form of basic geometry; displayed patterns with bright colors and high contrast resulting from switching between dark and bright regions. By creating three seperate rows in the visuals we create the option to map different patterns of visuals to different musical aspects in the composition (see figure 1). The high contrast in visuals translates well to musical phrases Hoogland tends to use, alternating between high and low pitches rhythmically, which find their roots in the Metal genre. Following is a technical description of the working of the system and the various tools involved.

5 Current setup

For the current setup we use the Flok web-based collaborative editor. The Flok editor allows for running multiple editors in parallel. An extension for the Mercury language was added by Hoogland to allow execution Mercury code that can then be received by the Flok REPL and transmitted over OSC to the Mercury MaxMSP based interpreter and sound engine. Next to the Mercury editor sits a Hydra editor that allows for displaying the evaluated visuals behind the written code.

The visuals currently consist of three rows with each uniquely coded patterns. The three separate rows are created by masking the various rendering outputs with a rectangular shape on different heights. The following code illustrates the masking:

```
// create three sources rendered to different outputs
osc().out(o1)
```

Figure 1: Live performance at Algopolis, Ljubljana (RS) 2022, depicting the three pattern rows (photo by Katja Goljat)

```
noise().out(o2)
gradient().out(o3)

// create a solid black screen and add the masked sources positioned at different heights
solid(0)
  .add(src(o1).mask(shape(4).scale(1,5,1)).scrollY(0.33))
  .add(src(o2).mask(shape(4).scale(1,5,1)))
  .add(src(o3).mask(shape(4).scale(1,5,1)).scrollY(-0.33))
.out(o0)
```

This masking allows Freeke to create three different visual patterns rendered to `o1`, `o2` and `o3` respectively.

The visual output is capture with OBS on Hoogland's computer, and by enabling the OBS Virtual Camera the texture can be opened in a MaxMSP patch that analyses the visuals. The patch imports the visuals with the `jit.grab` object and averages the pixel values to a smaller desired width, for example `32` by `3`, with `cv.jit.resize` from the cv.jit package. From this downsampled pattern a single row is selected, converted from RGB (Red, Green, Blue) to the HSL (Hue, Saturation, Lightness) colorspace with `jit.rgb2hsl`. The HSL model is often used in computer vision because it more closely matches human perception. Hue contains information about the color and lightness captures the brightness of the color (Cavaco et al. 2013). The seperate rows of HSL values are output as individual lists with `jit.spill` and sent out as OSC messages in the form of `/visual/r<number>/<color|light>` where `number` is the row `0`, `1` or `2`. A patcher in figure 3 depicts a simplified version of the system.

In Mercury the OSC messages are used as lists for creating melodies, controlling dynamics, modulating effects and much more. The OSC values can be scaled to a different range by adding the low and high output range between curly brackets. This expects the input-range to be between `0` and `1`. Consider the following code that creates a synthesizer and uses the color of the 0th row as a tone-row, the brightness of row one as note length and the brightness of row two as amplitude.

```
new synth saw name(bass)
  // semitones between 0 and 24
```

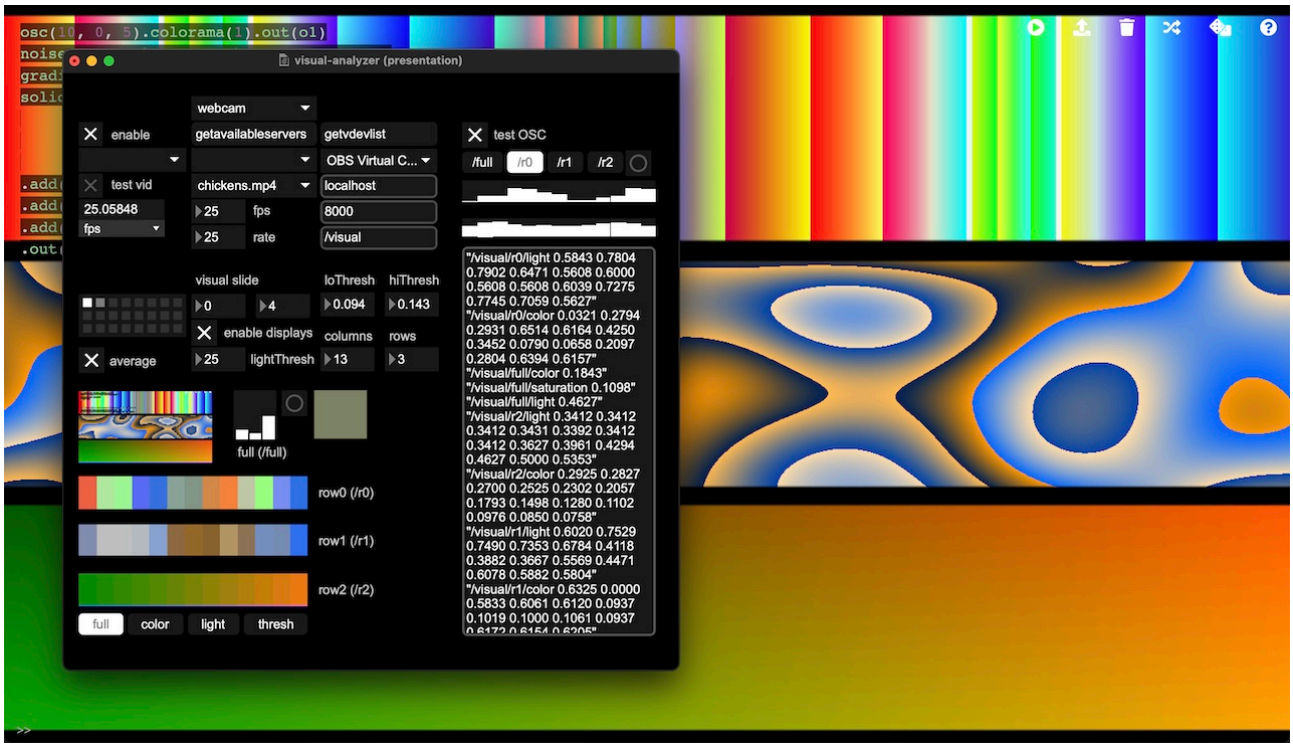

Figure 2: User interface of MaxMSP patch making a visual analysis and browser with Hydra visuals in background

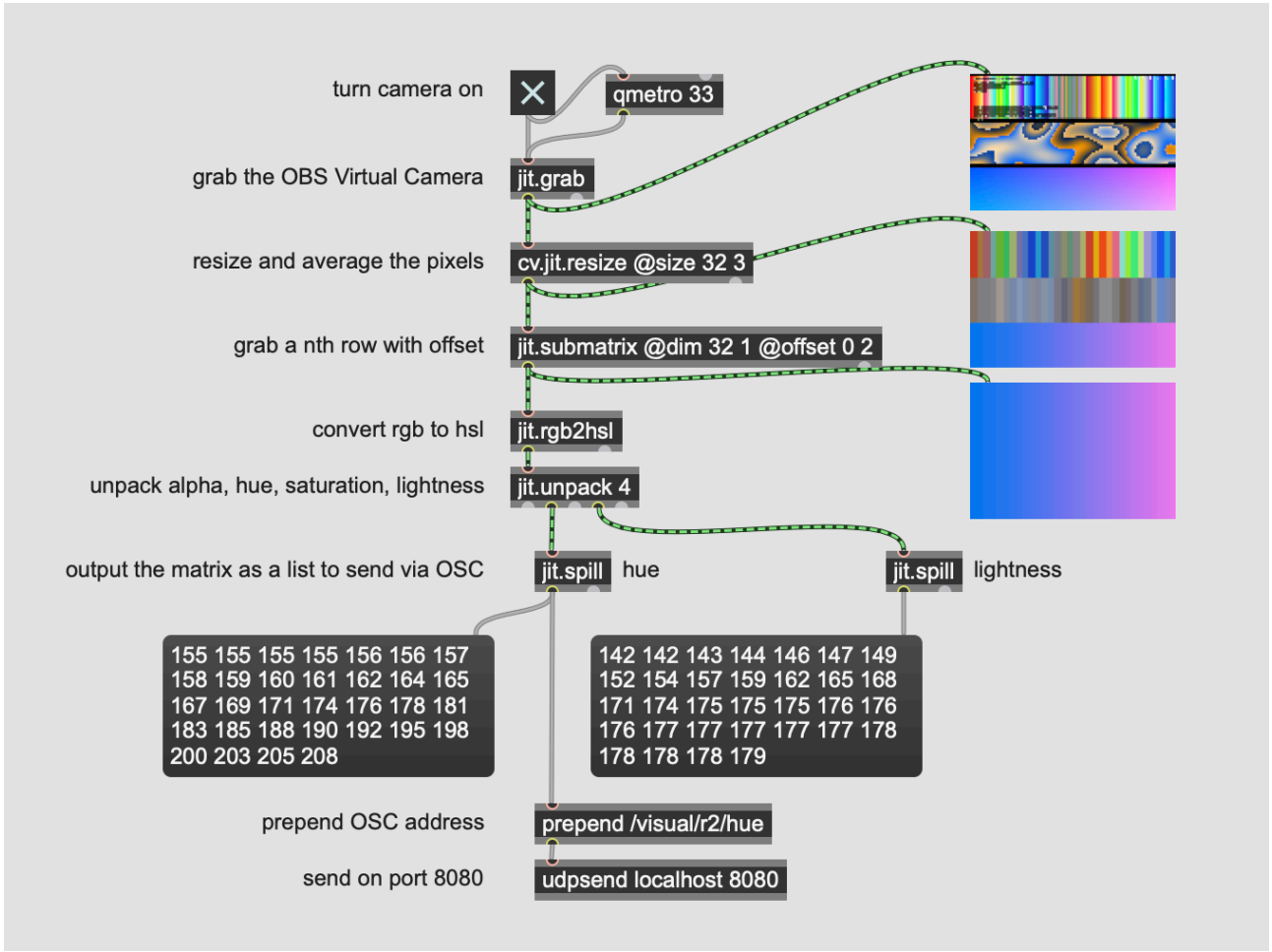

Figure 3: Simplified MaxMSP patcher of visual analysis system outputting OSC messages

```
set bass note(/visual/r0/color{0:24})
// release time between 50 and 800 milliseconds
set bass shape(1 /visual/r1/light{50:800})
// amplitude between 0.2 and 0.9 (± -13 to -1 dBFS)
set bass gain(/visual/r2/light{0.2:0.9})
```

In order to make sure that we both see the same visuals during the performance we synchronize the start of the Hydra visual time. This can be achieved by evaluating `H.synth.time = 0;` in the Flok editor. We use `setResolution(960, 540)` to adjust the visual resolution and put less stress on the graphics card to assure a smooth and high framerate while preserving a good picture quality when projecting.

6 Reflection

Moving the performance from audiovisual towards visualaudio creates a closer collaboration between the performers, but at the same time also adds challenges and cognitive load. The three rows of visual patterns give extra options that can be used as compositional material. During solo live coding performances most of the musical material is coded with algorithms in Mercury, but now Hoogland has to pay closer attention to the visuals programmed by Freeke and make decisions on which rows can be applied to what instruments and which parameters.

During the performance Freeke is mainly occupied with keeping up the pace of changing the visuals. Because the three rows are three separate Hydra renders it therefore requires more effort to code the visuals. Even though the music reacts to the visuals there is still some work to be done to make the visual coding feel closer related to the musical output. This is due to a couple reasons. For instance there is no agreement beforehand what visual row will be used for what musical aspect, this is still decided on-the-fly, meaning Freeke has to pay close attention to what the musical changes are when the visuals change, making it harder to anticipate what will happen musically when adjustments are made. Furthermore an entire row of the visual is used as a sequence to adjust musical parameters over time. While this gives interesting results it adds more complexity to be able to expect the outcome because the music will loop over the visual content left to right, while at the same time visuals are also scrolling, rotating and scaling over time. There is currently no visual indication to see what part of the visual is used to generate a note.

In terms of audience-code comprehension it might not be directly obvious how the visuals are modifying parts of the music. Even though we project our code and it could be clear from the code that parts of the visuals are used due to the writing of the OSC addresses in functions to access the visual rows' brightness or color.

Lastly there are still some technical aspects that can be improved. Currently a full screen capture is done of the visuals including the code of the editor, resulting in the code being added to the averaging of the visual colors. On top of that there seems to be a little lag between the screencapture with OBS and the visual analysis in MaxMSP.

7 Conclusion

In this paper the reader is introduced to Amalgam. A live coding visualaudio performance in which the visuals influence musical parameters in real time to create a closer collaboration between musician and visualist. We've seen that collaboratively writing code in the Flok editor allows both musician and visualist to view the visuals and eachothers code. By capturing the visuals via OBS, downsampling and analysing them in multiple rows for brightness and hue, the resulting arrays can be used as a sequence of notes, amplitudes or for instance control other musical effects. This allows the visualist to program graphics in their own aesthetic while still being able to change musical outcome.

8 Future Work

Amalgam is an ongoing collaboration and the project will certainly see adjustments and improvements or new explorations in the future. First of all we want to work on a different way to capture the visuals from the Flok editor that allows us to analyse the image without having the code from the editor interfere with the analysis. Furthermore we will plan more rehearsals to continue investigating how we can make the visual programming more closely related to the audible output. This investigation will also include working on ways to communicate this more clearly to the audience by for instance embedding an explanation of the system in the performance.

References

10 Aphex Twin. 1999. “Formula.” https://youtu.be/wSYAZnQmffg?t=327. 1999.

Bell, R, and J Chicau. 2018. “A Trans-Disciplinary Tool for Collaborative, Choreographed, and Embodied Audio-Visual Live Coding.” In *Proceedings of the Third International Conference on Live Interfaces (ICLC)*, 173–80.

Cavaco, Sofia, J Tomás Henriques, Michele Mengucci, Nuno Correia, and Francisco Medeiros. 2013. “Color Sonification for the Visually Impaired.” *Procedia Technology* 9: 1048–57.

Hoogland, Timo. 2019. “Mercury: a live coding environment focussed on quick expression for composing, performing and communicating.” Medialab Prado / Madrid Destino. https://doi.org/10.5281/zenodo.3946338.

Jack, Olivia. 2019. “Hydra: Live Coding Networked Visuals.” In *Proceedings of the Fourth International Conference on Live Coding*, 353. Madrid, Spain: Medialab Prado / Madrid Destino. https://doi.org/10.5281/zenodo.3946269.

Joana Chicau, Renick Bell. 2021. “Choreographing Coding - Tales from Real-Time Collaborative Audio-Visual Coding.” https://youtu.be/r4ws5evLO_g?t=565. 2021.

Manning, Peter. 2012. “The Oramics Machine: From Vision to Reality.” *Organised Sound* 17 (2): 137–47.

Ohme. 2021. “Order of Operations.” Ohme; https://youtu.be/2wwkDZAAEl8?t=1297. 2021.

Reuter, Jürgen. 2005. “Soundpaint–Painting Music.” *LAC2005 Proceedings*, 79.

Satellite, Github. 2020. “Art Machines - Fostering Digital Creativity Through Live Coding and ML.” Github; https://youtu.be/ECdxifljjE4. 2020.

Silvani, Damián. 2022. “Flok - a Web-Based P2p Collaborative Editor for Live Coding Music and Graphics.” https://github.com/munshkr/flok. 2022.

Street, Zoyander, Alejandro Albornoz, Renik Bell, Guy John, Olivia Jack, Shelly Knotts, Alex McLean, et al. 2019. “Towards Improving Collaboration Between Visualists and Musicians at Algoraves.” In *Proceedings of the International Conference on Livecoding, Medialab Prado, Madrid, Spain*, 16–18.

Thiebaut, Jean-Baptiste, Patrick GT Healey, and Nick Bryan-Kinns. 2008. “Drawing Electroacoustic Music.” In *ICMC*.

Utrecht, Creative Coding. 2021. “Iterations Symposium.” Creative Coding Utrecht; https://youtu.be/VZr92KzaqC4?t=10210. 2021.