URack: Modular audio-visual composition with Unity and VCV Rack

Matthew Hughes
Animal Logic Academy,
University of Technology Sydney
Australia
matthew.d.hughes@student.uts.edu.au

Andrew Johnston
Animal Logic Academy,
University of Technology Sydney
Australia
andrew.johnston@uts.edu.au

ABSTRACT

This demonstration presents URack, a custom-built audio-visual composition and performance environment that combines the Unity video-game engine with the VCV Rack software modular synthesiser. In alternative cross-modal solutions, a compromise is likely made in either the sonic or visual output, or the consistency and intuitiveness of the composition environment. By integrating control mechanisms for graphics inside VCV Rack, the music-making metaphors used to build a 'patch' are extended into the visual domain. Users familiar with modular synthesizers are immediately able to start building high-fidelity graphics using the same 'control voltages' regularly used to compose sound. Without needing to interact with two separate development environments, languages or metaphorical domains, users are encouraged to freely, creatively and enjoyably construct their own highly-integrated audio-visual instruments.

This demonstration will showcase the construction of an audio-visual patch using URack, focusing on the integration of flexible GPU particle systems present in Unity with the vast library of creative audio composition modules inside VCV.

Author Keywords

audio-visual, audio-visual composition, modular synthesis, real-time graphics, game engine

CCS Concepts

•Applied computing → Sound and music computing; •Human-centered computing → Interactive systems and tools;

1. AUDIO-VISUAL SOFTWARE SYSTEMS

A common method for creating audio-visual art with software is to construct the system using a multimedia programming environment such as Max [3] or Pd [8]. These are graphical environments in which applications are 'patched' together by drawing connections between independent nodes. Component libraries allow practitioners to combine audio and graphical development in the same environment, achieving a high level of integration between the sound and the visuals.

NIME'20, July 21-25, 2020, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom.

Environments like these typically have their roots in sound-sculpting, and often lack graphical fidelity and flexibility compared to alternative environments focused on graphics. Tight coupling between the audio and visual faculties is important for creating a convincing multi-modal work, so some forego higher-fidelity for a greater level of integration and immediacy [10].

Artists who desire more depth in graphics often create their own visual engines using creative coding libraries such as OpenFrameworks and Cinder [7, 2], and use these to accompany separate audio engines. This approach however, requires an advanced programming skill set.

Alternatively, video game engines offer artists a high fidelity, real-time graphics pipeline and feature-rich tooling. Engines such as Unity [9] and Unreal [5] provide high-level control over meshes, lighting, physics, particle systems, fluid and cloth simulations, with huge communities (and corporations) furthering the state-of-the-art in these technologies and their interfaces.

Audio-visual artworks that utilise video game engines for graphics typically don't use the game engine for audio as well. The maturity of environments like Max and DAWs like Ableton Live, with greater focus on sound manipulation make them far better candidates for being creative with audio than anything built into a game engine itself. Therefore, it is common for this software to be used as an audio backbone bound to a game engine handling the graphics. A common way to unify the two engines is with a (local area or internal) network connection, as demonstrated in [4], and often through use of the Open Sound Control (OSC) protocol [6].

However, approaches that rely on combining separate sound and graphical engines often require the user to program in two distinct environments—fragmenting the creative process with the use of multiple languages or contrasting compositional metaphors.

2. WEARING TWO CAPS

A system with a separated sound and graphical engine forces the artist to split their process into its modalities and alternate between composing for both. The artist must put on their 'programmers cap' to make meaningful changes to the visual world, and reason about their composition within the metaphorical systems present in their graphical engine. Then they can switch to their audio engine, put on their 'audio-production cap', and reason about their composition using musical metaphors.

Some solutions attempt to bridge this gap, pushing for high-quality composition of both modalities within the same environment. [1] embeds the 'Chuck' audio programming language inside Unity, in order to streamline the composers workflow: Unity and Chuck code can share memory and

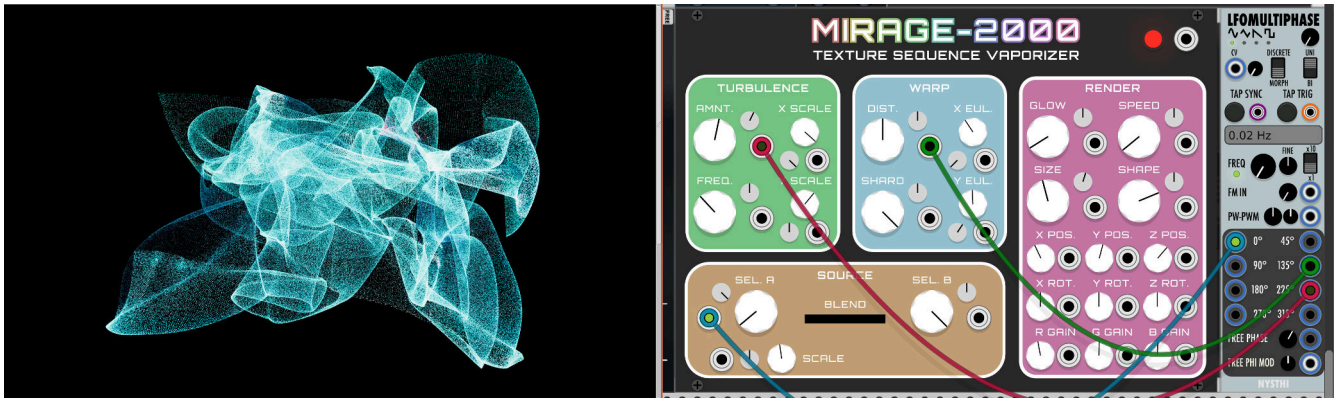

Figure 1: URack's 'Mirage-2000' module projects a sequence of images onto a point-cloud that can be morphed and mangled.

events, and be edited within the same text editor (albeit in two different languages). [7] abstracts the control of a fluid simulation built in OpenFrameworks behind a series of custom Pure Data modules, opening it up for interfacing within Pd. These projects are similar in intent to URack, but their interfaces are geared towards creative-coders rather than uninitiated artists.

3. VCV RACK

VCV Rack is a software modular synthesiser. Functionally, it is similar to environments like Max and Pd; modules inside VCV are 'patched' together using virtual cables, however its core metaphor is one based on the 'Eurorack' modular synthesis paradigm, ultimatley making it an interface geared more for electronic musicians than creative programmers. A patch in VCV is built by combining together modules from many different manufacturers—each with distinct design aesthetics and interaction mindsets that make them more comparable to 'instruments' than programming objects.

4. URACK

URack is a custom-built software solution that combines the feature-packed, flexibly deployed Unity game engine with the modular music-making metaphors of VCV Rack. Users familiar with modular synthesis are able to create high-fidelity visuals by making use of the graphics-oriented modules that ship with URack. Conversely, users familiar with programming Unity are able to integrate sound design in their existing scenes by utilising the URack API for Unity.

Comparing it with a similarly integrated audio-visual composition environment like Max with Jitter, URack offers an overall higher-level interface, allowing artists to compose works using the metaphor of a modular 'instrument', as opposed to a programming environment. URack also exposes the benefits of working with a game-engine, including higher-fidelity simulations and straightforward deployment to a wider range of devices (such as smartphones and XR headsets).

The communication between VCV Rack and Unity works using Open Sound Control (OSC) over a UDP network connection. When OSC messages are received at the Unity instance, any publicly exposed script property or VFX-Graph property that is addressed by VCV will be updated on the next frame. Modules can target any number of Unity instances at different network addresses, allowing one module inside Rack to manipulate graphics on a number of presentation devices simultaneously. A synchronised scene can be displayed on multiple computers and projectors, and at the same time inside an XR headset, for example.

URack modules comply with the Eurorack 'voltage standards' (10 volts peak-to-peak), meaning any module that outputs voltage in VCV is now able to control Unity. Users have access to the constantly growing library of thousands of VCV modules, and are able to utilise LFOs, sequencers, other complex modulation sources and even audio signals to control visual systems.

Additionally, URack includes tools and an API for Unity developers that makes it easy to convert existing game-engine components into new modules. Using a python script, 'front-panel' designs in the form of vector graphics files can be converted into functioning URack modules without needing to write any VCV Rack plugin or OSC networking code.

This demonstration will showcase URack inside VCV, focusing on how this approach allows audio-visual artists to reason about graphical manipulation using sound-design metaphors, while relieving the friction present when an artist must alternate between separate compositional environments for each modality.

5. REFERENCES

[1] J. Atherton and G. Wang. Chunity: Integrated Audiovisual Programming in Unity. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Virginia, US, 2018.

[2] N. N. Correia. Prototyping audiovisual performance tools: A hackathon approach. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Baton Rouge, US, 2015.

[3] Cycling 74. Max/MSP. `https://cycling74.com`.

[4] A. Dolphin. Compositional applications of a game engine: Creative practical applications in sound art and music composition. In *Proceesings of the IEEE Consumer Electronic Society's Games Innovation Conference*, London, UK, 2009. IEEE.

[5] Epic Games. Unreal. `https://unrealengine.com`.

[6] R. K. Hamilton. UDKOSC: An immersive musical environment. In *Proceedings of the International Computer Music Conference*, Huddersfield, UK, 2011.

[7] A. Johnston. Fluid simulation as full body audio-visual instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, page 5, Daejeon, Korea, 2013.

[8] Miller Puckette. Pure Data. `https://puredata.info`.

[9] Unity Technologies. Unity. `https://unity.com`.

[10] G. Wakefield and W. Smith. Cosm : A toolkit for composing immersive audio-visual worlds of agency and autonomy. In *Proceedings of the International Computer Music Conference*, Huddersfeld, UK, 2011.