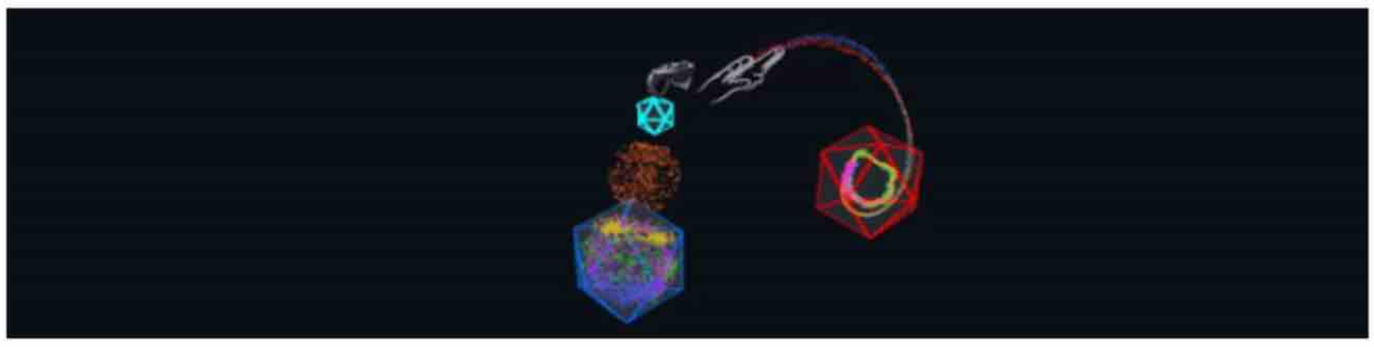
Home

Sam Bilbow edited this page 2 hours ago · 8 revisions

polygons~



Developing a Sound ARt Performance Practice

Platform Windows Environment Unity/Pd Performances Playlist Guide Wiki Project Blog Discord XRt Space

As a direct consequence of witnessing participants enjoyment and play with *polaris~*, I recognised for the first time looking from the outside in, that the system had a larger propensity for fostering learning, virtuosity, and depth of expression than I had originally thought; if only the experience was slightly more complex, the interactions more *nuanced*. The gestures, play, and expression by participants led to a kind of quasi-transhumanist dance, one that struck me as a technologically mediated dialogue between hybrid self and hybrid environment, a delicate balance of agency.

polygons~ is a fifteen-minute experimental audiovisual AR improvisation using an AR performance ecosystem including three instruments: ambi, click-+, and hands.

To get started with `polygons~`, follow the guide shown on the sidebar of this page.

MAH Showcase @ The ACCA

9 views • 6 months ago

emute_06 @ The Rosehill

10 months ago

Citation

Bilbow, S. (2022). polygons~ AR Performance. YouTube. Available Online: https://www.youtube.com/watch?v=9IErsDvhXjM

or with BibTeX

Clone this wiki locally

https://github.com/sambilbow/po

Components

Sam Bilbow edited this page 2 hours ago · 1 revision

- Software Companion for the Project North Star open-source AR headset that allows developing Unity scenes with MRTK/Leap Motion assets.
- LibPdIntegration: a wrapper for libpd that allows for the implementation of Pure Data patches into Unity
- Automatonism: a library of Pure Data Vanilla patches that emulate modules of a synthesizer.
- A set of example scripts and scenes that use the above components to demonstrate possible interactions between head/hand tracking and patch parameters in Pd, with the chief aim of creating a set of expressive multisensory AR instruments / experiences.

Features

Hardware

- Six degrees-of-freedom (3D position / orientation) head tracking via Intel T261
- 90 fps, 170° hand tracking via Ultraleap
- Single piece optical combiner allowing for up to 110° horizontal FoV
- 2x 120Hz displays per-eye for a total resolution of 2880x1600
- 2x 3-metre cables (1x miniDP, 1x USB-A 3.1)
- Spatial audio AR (the ability to hear localised sound whilst being able to hear your real audio environment) via Unity3D and Aftershokz Aeropex bone conduction headphones.

Engine

- The ability to create 3D scenes that contain 'GameObjects' that in turn can have visual attributes such as 3D meshes, material colours, and textural properties; physical attributes such as edges, position, mass, velocity and real-time parameterisation via C# scripting.
- Thanks to the Software Companion, the headset is created as a GameObject with real-time position / orientation.
- Thanks to LeapMotion, hands (all the way down to individual finger joints) are created as GameObjects with real-time position / orientation relative to the headset.

Audio

- LibPdIntegration uses **native Unity3D audio spatialisation**. This is great because it means that a GameObject can output the signal of a Pd patch whilst moving, rotating and scaling. The effect of these can perceived in real-time because the AudioListener is anchored to the real-time headset position. This, for example, means that the volume of a Pd patch whose signal is being transmitted from a GameObject located in space is automatically scaled dependent on its distance to the participants head (quieter as it gets further away, louder as it is brought closer).
- LibPdIntegration can **'instance' Pd patches**, meaning it can use one patch on multiple GameObjects, but maintain processes like randomness within them as they are technically different 'instances' or versions of the patch.
- Pure Data allows **extended audio techniques** through an extensive library of algorithmic 'objects' that can create and manipulate audio signals.
- LibPdIntegration allows **real-time parameter control** in Unity of any object in a Pd patch via "receive" objects and a specific C# method.
- The combination of "Play Mode" toggling in Unity, and the quick visual patching style of Pure Data means that audio-visual interactions can be **prototyped very rapidly**

Clone this wiki locally

https://github.com/sambilbow/po

Software Requirements

Sam Bilbow edited this page 2 hours ago · 4 revisions

Compatibility

- Tested with Windows 10, 11
- Tested with PureData 0.51-0

Installation

- Download the repository as a .zip or via `git clone https://github.com/sambilbow/polygons` in your terminal emulator

Unity Editor

- 2020.3.18f1 available @ Unity3D Archive

Ultraleap SDK

- Gemini 5 or later available @ LeapMotion

Intel Realsense

`Note:` this device is now end-of-life, running the project with other 6DoF sensors will require Software Companion compatibility with them first.

- Intel® RealSense™ SDK 2.0 available @ GitHub Release

PureData

- Pure Data 0.51-0 (tested) available @ PureData
- Pure Data Latest available @ PureData

Clone this wiki locally

`https://github.com/sambilbow/po`

Hardware Requirements

Sam Bilbow edited this page 2 hours ago · 5 revisions

PC Specifications

- Minimum system requirements for the North Star headset are found on the Project Northstar Documentation Website. I used a Macbook Pro 16,1 with an i9 CPU, AMD 5500M GPU, running Windows 11 in Bootcamp with custom Radeon drivers, and a custom CPU thermal throttling profile.
- A DisplayPort connection that can output 4k@60Hz for the headset.
- A USB (A) 3.2 Gen 2 connection (10Gb/s) for the headset
- A USB port for the audio interface
- An HDMI port for the projector

Visual

This project guide assumes you have a calibrated Project North Star headset. If you don't, check out the below link to get started, and make use of their Discord server!

- Project North Star Headset (used Combine Reality Deck X)
  - Ultraleap Stereo IR 170 info
  - Intel Realsense T261 info
    - `note:` this device is now end-of-life, Xvisio SeerSense XR50 is now recommended, ask on Project North Star's Discord Server for more details.
- A projector
- Enough space to perform without standing in the throw of the projector (causes strobe)

Audio

- A Stereo USB Audio Interface (stereo headphone jack should work too)
- PA Speakers

Clone this wiki locally

https://github.com/sambilbow/po

Instruments

Sam Bilbow edited this page 1 hour ago · 3 revisions

Unity Scene

1. Add the Unity project in `polygons/unity/` to Unity Hub
2. Open `polygons/unity/EskySettings.json`, to include your headset offsets and update your `"displayWindowSettings"` to suit your monitor + headset display layout.
3. Open the Unity project.
4. Open the polygons~ scene: `polygons/unity/Assets/Scenes/Experiences/polygons.unity`

ambi

ambi, a red wire-frame icosahedron, serves as a dual-oscillator drone synthesiser that the performer can call on my taking and bringing it closer to them with their hands. Its voice is contingent eleven real-time parameters, which are sent to the PureData patch attached to it. At specified distance two particle systems are activated from the palms of the performers hands, and the drone from ambi is activated.

click+-

click, a set of two blue wire-frame icosahedra, form a feedback system between two low- frequency oscillator controlled impulse generators. The feedback can be altered by bringing the smaller icosahedron (click-) closer to the larger one (click+). click+- makes use of LibPdIntegra- tion's Unity send functionality; each time there is an impulse generated, a message is sent to Unity, which triggers a shower of particles to be emitted from click- to click+ forcefield-guided cone. As the performer intervenes and creates the feedback system, the closer they bring the icosahedra together, the faster the visual particle shower becomes, and the more erratic the impulse generators feedback on each other. A further sound parameter, connected to a reverb, is mapped to the amount spin click- let go with by the performer. The sound palette explores a gradient from static electricity crackles to a sound akin toa car motor catching and revving up as the feedback increases.

hands

hands constitutes the final AR instrument in polygons ; each of the performers hands, when a virtual button besides their palm is toggled on, generate a highly resonant filtered white noise. The cutoff, resonance, and amplitude of this generator-filter system is controllable independently per hand, through several dynamic movement-based attributes. The filter cutoff per hand is mapped to that hand's distance to the performer's face; the filter resonance per hand is mapped to the angle from that hand's palm angle towards the centre of the performance space; and the amplitude of each is mapped to the stretching of the fingers outwards away from the palm. The sounds delivered by hands are harsh and unpleasant at certain parameter combinations and purposefully loud; from a performative standpoint, this engenders specific gestures, movements, and stances, as the performer grapples with a sonic experience akin to howling and shrieking winds.

More detailed information about musical parameter mappings can be obtained by viewing the performances I've done so far with polygons~. Detailed tables of parameter mappings will be added at a later date.

MAH Showcase @ The ACCA

9 views • 6 months ago

emute_06 @ The Rosehill

10 months ago

Clone this wiki locally

`https://github.com/sambilbow/po`

Performance Setup

Sam Bilbow edited this page 1 hour ago · 3 revisions

Visual

For translating my visual experience (I deemed this necessary for the audience to understand my gestures and their links to musical outcomes) I would project the real-time hybrid composite feed (made up of a black and white video feed from one of the head-mounted sensors on the headset with an overlay of the Unity scene) on the wall behind me

Audio

For the audio, I replaced my bone-conduction headphones (sacrificing some of my own 3D audio immersion) with the PA system of the venue. The panning would be reversed so that my gestures and movements would translate realistically to the audiences auditory perspective, i.e. the movement of my hand during the performance of hands across my body from left to right would transfer to the sound panning house-right to house-left rather than stage-left to stage-right

Clone this wiki locally

https://github.com/sambilbow/po