Kessel Run

Shawn Lawson
Rensselaer Polytechnic Institute
lawsos2@rpi.edu

Ryan Ross Smith
Rensselaer Polytechnic Institute
ryanrosssmith@gmail.com

ABSTRACT

From slick to glitch, we create an aural-ocular intermixing of finely polished to glitchy accidents. Within the live manipulation of audio signals and generative coding exists an environment of contrasts: experimentation/accidents, surface/internals, and success/failure. Code is visibly projected, integrated into the visuals, the artist's thought processes are on display, where all of their successes, struggles, and failures become part of the performance - an embodiment of signal and logic.

1 Visuals

The visuals for Kessel Run are performed in Google Chrome, which loads a local webpage. The webpage has a text editing area for writing GLSL fragment shader code, a stats and menu bar, a texture loading pane, an Open Sound Control pane, and the OpenGL framebuffer. A 4-band FFT is accessible from a uniform variable, while a 512-band FFT and waveform are accessible as a texture in the fragment shader. An external python script, running in a terminal, forwards incoming Open Sound Control data from a UDP port to a connected websocket created in the webpage's Open Sound Control pane. Fragment shader code is written in text area. The shader code is automatically compiled after 200ms have lapsed with no key or mouse input. If the shader compiles successfully it is automatically swapped onto the graphics card and used to draw the window's framebuffer. If the shader does not compile, error lines will appear showing where to check for problems; meanwhile, the most recent successfully compiled shader will continue to draw to the framebuffer.

2 Audio

The audio for Kessel Run consists of the largely on-the-fly assemblage of previously defined sonic elements in Ableton Live. These elements were created using a diverse collection of homegrown software and hardware solutions, and pulls from a variety of locales in the sound artist's aesthetic output. The physical methods by which these elements are assembled also have an (in)direct influence on the visual component, actions that result in not only a marked sonic disruption, but provide data streams (continuous and momentary) that are made available to the visual artist for further (dis)use.

3 Integration

The integration of Open Sound Control and FFT allow both performers to exert control over timing. The visual artist listens to and responds to simplified FFT data when coding; as well, the sound artist can see the visuals and use their sound to modify the visuals. Hidden or entirely different visual context changes can occur when the visual artist designs within conditional statements, which can be selected/executed/walked through when the sound artist sends Open Sound Control cues when launching sounds. In fact, individual sound cues and intensity values can be sent and used to create tracks for each sonic element, creating a tight integration between visual and aural components.