SCREAM – SuperCollider Resource for Electro-Acoustic Music

Michael Leahy*

*EGR & Recombinant Media Labs

michael@egregious.net

Abstract

SuperCollider3 is a major achievement for programmatic real time audio synthesis. However, the adoption of SuperCollider3 has been limited to a small community due to it being a domain specific language/environment and the difficulty of using the tools provided in the default distribution. The SuperCollider3 language is a powerful tool to interact with the SuperCollider3 server, but requires the user to engage SuperCollider3 through a programming language. Scream is a high level component oriented framework built in Java to interact directly with the SuperCollider3 server. Scream enables complete applications to be built with sophisticated GUIs that are accessible to users of all skill level while maintaining an API for developers to create new software.

1 Introduction

Scream is a cross platform component oriented Java environment and API to utilize the SuperCollider3 DSP audio engine and language among other OSC/MIDI enabled applications. My initial goal with Scream is to establish a framework to use with SuperCollider3 (SC3) to create new audio tools for studio, live performance, and sound installation applications. SC3 is the most advanced real time audio programming environment available today. Since the summer of 2002 SC3 has been an open source project under GPL with a fairly small, but dedicated community. SC3 is very powerful, however it is not directly aimed at the hobbyist or typical musician. Currently, other real time audio engines such as Max/MSP and PureData offer the user a graphical interface to create synthesis patches. The default tool to interface with SC3 is SCLang which is an interpreted programming language similar to Smalltalk. SCLang is a client to the SC3 Server and communicates with it via OSC (Open Sound Control).

2 Scream Description

Scream is a high performance real time engine that facilitates advanced software development and multimedia interaction with SC3 through a component oriented framework. This framework provides many services to Scream applications such as an accurate callback clock/scheduler, custom event system and data models, efficient client/server based networking system for OSC communication, and hardware accelerated 2D/3D graphics library support.

2.1 High Level Language Support

Scream is constructed in the Java programming language. There are many benefits to using a high level language to connect to the SC3 Server. The first and foremost benefit is that Java is an easy to understand and mature object oriented language that supports rapid component oriented development. Java also provides several auxiliary APIs for networking, MIDI, graphics (Java2D, JOGL, LWJGL), and cross platform support.

An immediate concern of using Java is the ability to create a real time system appropriate for music performance and creating multimedia interfaces. JFC/Swing is the standard GUI toolkit available, but is not necessarily a real time capable system. Scream provides a custom GUI library that is renderer agnostic. Components in the GUI library may have multiple hardware accelerated renderers constructed in Java2D/Volatile Image API, JOGL (OpenGL) and/or LWJGL (OpenGL). Swing is still accessible in Scream GUIs when using a Java2D or JOGL renderer, but is not relied upon to display real time data.

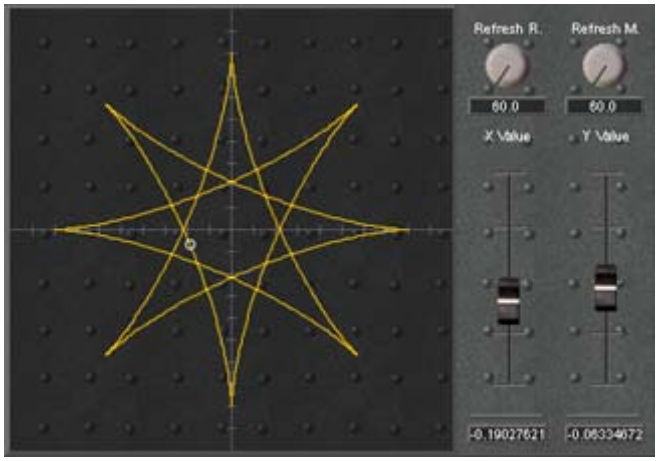

Figure 1. A partial screen image showing components from the Scream GUI library. The ScreamGraph component (using a Java2D renderer) is on the left displaying a hypotrochoid shape. It may be used for spatialization or as a geometric oscillator. Knobs and faders are visible on the right.

2.2 Scream Engine

Scream is an engine that provides services to all executing modules. For instance there is only one clock/scheduler that is shared between all executing modules. This allows accurate event scheduling to occur on a regular basis without interruption. Another subsystem that is shared between all modules is MIDI. Data models from any module may register to receive MIDI events that are dispersed from a central interface. A further Scream subsystem is the lag and modulation engine. Most Scream controls may be lag enabled or connected to modulation sources such as various LFO waveforms. The ScreamGraph component pictured in **Figure 1** is used for spatial data, but also can be used as a complex geometric oscillator/modulation source.

On the API side for developers Scream integrates functionality such as loading and saving state automatically. Developers do not need to worry on how to implement load/save functionality when the Scream component library is used to construct new software.

One of the most compelling features of Scream is its component orientation. Scream utilizes a plugin engine that facilitates extensibility. All Scream modules are essentially plugins that utilize the functionality of the Scream engine. There is a lot of freedom in designing a new audio system. Unlike other audio tools where plugins are basically limited to audio effects or instruments, each component in Scream may define its own plugin interface. An example of a current Scream component that utilizes five different types of plugins is the ScreamGraph component. ScreamGraph plugins include generators (parametric math, polar, splines, etc.), transforms (image processing on the resulting generated geometry), positional plugins, time movement (linear time, random, brownian), and data filters.

Data filtering is an interesting aspect of Scream components. A generic component may have a data filter attached to its output. The ScreamGraph component includes data filters to translate graph coordinates to the proper format for SC3s ambisonic spatialization functionality. On the fly coordinate data is transformed into amplitude and angular data.

3 Current Scream Progress

Fulltime development of Scream commenced in April 2003. I have overcome most of the major development issues to providing a high performance environment for Scream. Most of my work has been focused on the Scream engine itself, but presently (February 2004) there are three completed applications available. The first application is a granular sampler. This application allows the loading of sample data into SC3 from local files or over the network and enables the user to scan (forwards/backwards), alter the sampling rate, pitch shift, and define an envelope applied to the sample. The second application is an ambisonic mixer utilizing the 1st order b-format functionality available in SC3 which is suitable for spatializing audio on a 2D plane with N-speakers. The final application is a combination of the previous two, the AmbiGranusampler which enables the Granusampler with ambisonic spatialization.

Scream was used for a large scale installation at Burning Man during August 2003. The Sol System collective constructed an eight point sound array erected around a 120' diameter circle. Each point was an 5' by 10' full range stack of speakers. This was a perfect setup to utilize the Scream ambisonic spatial mixer and it sounded amazing on the open desert with no confined space causing reflections.

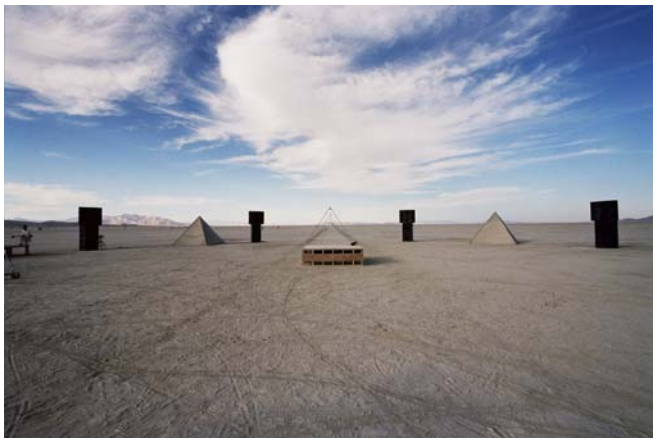

Figure 2. Sol System 8 channel sound array at Burning Man 2003 (picture by: Andrew Polson)

4 Future Scream Development

Work on the Scream engine continues rapidly. Core engine functionality will be well defined, implemented, and tested by summer 2004. This functionality includes an IDE editor and compiler for SC3 synth definitions. Beyond Scream engine development the standard component library will be extended. These extensions include further modular environments such as a standard mixer that maps to SC3 functionality. Other components will include 3D graphics/spatialization tools. Eventually, 2nd and 3rd order ambisonic plugins will also be developed making Scream/SC3 capable of controlling a large scale speaker array setup in a 3D arrangement.

4.1 Middleware Integration

Scream will integrate with various middleware software available on the Java platform. In particular Scream will integrate with the Xith3D and OpenMind 3D scene graph engines, JavaODE physics bindings to ODE (Open Dynamics Engine), the SimBionic AI engine, and JMSL.

4.2 Domain Specific Language Integration

With OpenGL 1.5 support and beyond Scream will utilize OGLSL (OpenGL Shading Language). OGLSL is a domain specific language (DSL) oriented towards extending the fixed hardware pipeline of modern accelerated graphics cards with programmable functionality called shaders.

Scream will form a bridge between OGLSL and SC3 enabling advanced audio/graphics applications. It will be possible for SC3 to provide control data that can be mapped to particular functions of an OGLSL shader and vice versa. Scream will provide an IDE for shader development that will allow easy integration with the Scream engine and SC3. This shader development tool will be similar to ATI/3DLabs RenderMonkey, but have the added benefit of being able to interactively access the Scream engine and SC3. The Scream shader IDE will be compatible with the RenderMonkey XML file format, so that shaders may be exchanged freely between either program.

4.3 Interactive Audio for 3D Simulations

Beyond constructing a framework for creating music tools my focus has extended to creating a multimedia distribution for Java where Scream/SC3 provides the unifying audio technology. Scream will enable advanced multimedia games and 3D simulations that include dynamic real time audio. I believe that all the advances in real time computer graphics are meaningless without dynamic real time audio. The gaming industry has mostly ignored real time audio up to this point. Scream and SC3 will provide the missing link. Scream will utilize the emerging Interactive XMF standard that should be publicly available by summer 2004. Scream will provide an IXMF soundtrack manager and will extend IXMF with Scream/SC3 functionality.

Scream will offer a robust fail safe API for advanced audio by also supporting FMOD and finally OpenAL. FMOD & OpenAL are cross platform audio engines for 3D simulations, however both lack the real time audio synthesis capabilities of SC3. If SC3 is not available on any given platform or for some reason fails to run then the Scream engine will fallback to FMOD or OpenAL and a static sound library if provided by the game/simulation.

4.4 Ms. Pinky

Another interesting area of development is the integration of Ms. Pinky with the Scream environment. Ms. Pinky is a tool to use vinyl/turntables as a control source. Not only will advanced DJ tools be available through Scream, but this control data will be available generically to any Scream component and even within the SC3 engine itself. It will be possible to control spatialization from turntables and use Ms. Pinky to send control data to OGLSL shaders creating dynamic graphic effects driven through turntablism.

For an accurate snapshot of completed and future Scream development visit the web site available in the references section.

5 Acknowledgments

My family and friends for continued support.
Naut Humon for material and facility support.
Gérard Pape / CCMIX

Resources

ATI/3DLabs RenderMonkey
http://www.ati.com/developer/sdk/ (continued)
radeonSDK/html/Tools/RenderMonkey.html

Cousinié, Alban et al. OpenMind
http://www.mind2machine.com/gb/openmind/

Cycling ’74, Max/MSP
http://www.cycling74.com/

Didkovsky, Nick, and Burk, Phil JMSL
http://www.algomusic.com/jmsl/

Firelight Technologies, FMOD
http://www.fmod.org/

Interactive XMF Working Group
http://www.iasig.org/wg/ixwg/ixwg.shtml

Laasko, Jani et al. Java ODE
https://odejava.dev.java.net/

Leahy, Mike. Scream
http://audio.egregious.net/scream/

Java Gaming Community
http://www.javagaming.org (Check discussion forums)

McCartney, James et al. SuperCollider3
http://supercollider.sourceforge.net/

OpenGL
http://www.opengl.org

OpenGL Shading Language
http://www.3dshaders.com

Puckette, Miller et al. Pure Data
http://www-crca.ucsd.edu/~msp/software.html

Rychlik-Prince, Caspian et al. LWJGL
http://www.lwjgl.org/

SimBionic AI Engine
http://www.simbionic.com/

Sol System
http://www.solsystem.org/

Sun, Java
http://java.sun.com/

Wardel, Scott. Ms. Pinky
http://www.mspinky.com/

Wright, Matt. Open Sound Control
http://www.cnmat.berkeley.edu/OpenSoundControl/

Yazel, David et al. Xith 3D
http://xith.org/