Mother: Making the Performance of Real-Time Computer Graphics Accessible to Non-programmers

Ilias Bergström
University College London
Institute of Ophthalmology
11-43 Bath Street
London EC1V 9EL
i.bergstrom@ucl.ac.uk

Beau Lotto
University College London
Institute of Ophthalmology
11-43 Bath Street
London EC1V 9EL
lotto@ucl.ac.uk

Abstract
Here we describe a programme that will enable the performance of advanced real-time computer graphics by non-programmers through flexible modules that are easily exchanged and managed.

Keywords
New media art, VJing, Processing, Real-time performance, colour-organ

Introduction

From the colour organs of past centuries, to today's VJing and new media art performances, artists and scientists have shown perpetual interest in the live performance of visual imagery. The limited aesthetic flexibility permitted by existing tools however has meant that artists striving to differentiate their work have most often been the inventors and creators of the tools they use. To address this obstacle, we have developed a set of tools, that make use of and extend the popular Processing [1] open-source multimedia programming environment. Together they will enable artists without the skills of computer programming to significantly influence the content of their visual performance, while also keeping the programming of new graphics algorithms accessible to those seeking greater creative freedom.

Background

The first known machine for the real time performance of colour 'graphics' was Louis-Bertrand Castel's "Clavecin oculaire" (1734) [2]. Many machines have since followed, made to produce either compositions of animated colours, dubbed colour music, or moving compositions of colour and form, a new art-form dubbed "Lumia" by Thomas Wilfred, the developer of the "Clavilux" organ (1922) [2]. In modern times, analogue video synthesizers, laser shows and more recently computer graphics have all been employed for the real-time performance of moving imagery. For instance, at live music concerts and at clubs with music played by a DJ, there are often live graphics performed by a VJ (for Visual Jockey), who mixes pre-recorded video clips together, while altering the playback parameters of the individual clips, as well as processing these clips using real-time video effects. A further advance has been the recent development of computer programming environments, such as Processing and Cycling74's Max/MSP/Jitter combination, that allow the description and performance of real-time procedural graphics by non-expert programmers, and as such their use has begun making its way out of the avant-garde and into VJ sets, and 'new media art' performances.

Existing limitations

In the practice of VJing, the performer has only coarse-grained control over his/her content.. Although what video is played at a particular instant in time can be decided on-line during the performance, as can the playback speed and direction, as well as what video effects are applied, the performer has no significant influence over the content of the video itself. There are variations to described practices, such as using live video sources instead of pre-recorded video, the principle however still remains the same.

While previously mentioned multimedia programming environments enable performers to overcome this limitation by allowing them to control the synthesis of imagery in real-time, these environments have other limitations of their own. For instance, while they are intended to be accessible to non-expert programmers, they require a great deal of time to learn how to use effectively. Furthermore, the created program cannot easily be used by other performers for achieving their individual aesthetic goals without modifying it, which requires significant programming knowledge.

Proposed system

The system we propose here overcomes these limitations, through an integration of the principles of using VJ-software, and of working with mentioned multimedia programming environments.

Although the principle of mixing multiple layers of moving graphics, as in VJing, is retained, here these graphics are not from pre-rendered video clips, but are the output of real-time visual synthesizers ('synths'), running in parallel within the main host application. Each synth is a program that renders a particular visual effect, the control parameters of which are all accessible on-line during a performance, so that the appearance of the visual is animated over time. 'Synths' can be created as Processing 'Sketches' [1], using a Processing library provided that enables the sketch to work within our host application. Users can either create their own synths, use synths from a collection provided with the host, or from other users.

The system therefore comprises of two modules: the main host application and the Processing library. The host application manages the execution of the multiple synths, mixes their individual visual output into the final output, and handles the forwarding of control information to individual synths. The Processing library resides within each synth, and handles its integration with the host. Besides encapsulating the boilerplate functionality for allowing this integration, it also provides an easy mechanism for programming new synths.

Three different categories of synths are envisioned: (i) full-frame synths, covering the whole screen area, thus being most suitable for use as the background layer, (ii) synths partially covering the screen, leaving a transparent background, and (iii) video processing synths, that apply 2d effects to real-time video streams such as the output of other visual synths, as are currently used in VJing software. At the time of writing, the first two categories have been implemented. The third is under development, and will ideally support the incorporation of FreeFrame effects, an already well established standard for video effect plugins [3].

The communication of control data to the host and to individual synths is managed solely through the use of Open Sound Control [4], a protocol for communicating between computers, sound synthesizers and other multimedia devices. Currently there are many systems that allow you to relatively easily define a user interface capable of transmitting OSC control data, both in the form of hardware controllers and of software applications [4]. The choice to completely separate the synth host from any user interface was made because there currently is no established generally applicable user interface paradigm for controlling the performance of real-time computer graphics. Users are instead encouraged to adapt or create their own user interface, depending on what best suits their particular context of use.

Discussion

Here we describe the current state of a system that will facilitate the performance of advanced real-time computer graphics by non-programmers, by encapsulating the programming complexity in flexible modules that are easily exchanged and managed. Performers will be able to pick visual synthesizer modules from an anticipated growing number of freely available ones, provided a community has grown around the use of the system. Creating new visual synthesizer modules will be as easy as creating a Processing 'sketch', which has been proven to be accessible to artists not trained in software engineering. Finally, the host and library will both be released as free open source software, thus making them part of the larger context of such software, which includes Processing. This move will hopefully encourage users to share the visual synthesizer modules they have created, in the same manner as the Processing community already practices with 'sketches'.

References

[1] C. Reas and B. Fry, (2007) Processing: A Programming Handbook for Visual Designers and Artists. Publisher: The MIT Press.

[2] K. Peacock, (1988) Instruments to Perform Color-Music: Two Centuries of Technological Experimentation. Leonardo 21 , pp. 397-406.

[3] FreeFrame, http://freeframe.sourceforge.net/. Last checked June, 17, 2008.

[4] opensoundcontrol.org, http://opensoundcontrol.org/. Last checked June, 17, 2008.