Dynamic Media Arts Programming in Impromptu

Andrew R. Brown

QUT Computational Arts Research Group &
The Australasian CRC for Interaction Design
Kelvin Grove, Australia, 4059
+61 7 3337 7844
a.brown@qut.edu.au

Andrew Sorensen

MOSO Corporation
New Market, Australia, 4005
+61 7 3352 5741
andrew@moso.com.au

ABSTRACT

Dynamic software systems for generative art have particular requirements around rapid development and manipulation of ideas and real-time computational processes. Impromptu is an innovative new programming environment created by Andrew Sorensen, and is designed to facilitate the fluid flow of creative ideas and their expression in software. It has been used for our recent projects, including a variety of Collaborative Music Jamming Systems, because it supports creative processes by leveraging the audio, graphic and video capabilities of the modern operating system within a dynamic programming environment. In our projects collaboration is supported through a synthesis of media types, numerous communication protocols, and facilities for inter-system communications and control. The demonstration will show these environments and how Impromptu supported their development.

Author Keywords: Generative art, collaborative systems, programming environment, media arts, software development, interactivity, multimedia, live coding

ACM Classification Keywords

J.5 Arts and Humanities – Arts, fine and performing, C.3 Special-purpose and application-based systems

INTRODUCTION

Dynamic Media Jamming environments enable users to interact with generative music and visual systems to perform audio-visual works, typically in the style of electronic dance music and VJing. User's become meta-creators by adjusting generative system parameters through a variety of graphical and physical controller interfaces. These systems require a development environment with rich media capabilities, reliable timing and the ability for rapid development and easy modification. The environments we have developed are in the context of interaction design research and thus the ability to iteratively refine the systems in response to usability feedback is particularly important. Impromptu has proven to be an effective tool for this development.

Impromptu is a Mac OS X programming environment for composers, sound artists, VJ's and graphic artists with an

interest in live or interactive programming. Impromptu is a Scheme language environment. Scheme is a member of the Lisp family of languages.

Time plays a major role in the Impromptu environment which allows accurate real-time scheduling of events and code blocks. Impromptu is a dynamic environment designed for the creation and manipulation of running programs in live performance and can also be used for computational arts installations, and developing interactive media experiences.

DYNAMIC ENVIRONMENTS

We define dynamic environments as ones where the system is required to respond flexibly to real-time changes, including changes in the source code, without halting. Contexts where this is useful include live sonic/visual performance, adaptive systems, and live coding. Impromptu is designed to provide a dynamic, real-time, multi-user platform capable of supporting the creation, modification, distribution and evaluation of source code in live performance.

This highly dynamic, real-time environment is ideal for crafting media art works. Impromptu accommodates this with its ability to create and modify code in real-time. Impromptu's Scheme implementation provides a small and elegant language for developing multimedia algorithms 'on-the-fly'. The dynamic nature of Impromptu so rather than *programming* the environment, you feel like you are *playing* the environment.

TIME BASED PROGRAMMING

In real-time media arts systems, especially those involving music and sound synthesis, temporality is critical. One of the primary differences between Impromptu and most other general purpose programming environments is an emphasis on precisely scheduled code execution. Impromptu incorporates a scheduling engine directly into the programming environment allowing tasks to be specified for execution at precise times in the future. Unlike using thread constructs such as `sleep()`, `wait()` etc. that do not provide any strong guarantees about temporal accuracy, Impromptu's scheduling engine is guaranteed to execute its tasks at the requested time (the number of audio samples since Impromptu was started). This temporal guarantee is particularly significant for time critical domains such as music and video. This "strongly timed" feature enables a

Copyright is held by the author/owner(s).

C&C'07, June 13–15, 2007, Washington, DC, USA.

ACM 978-1-59593-712-4/07/0006.

design pattern in Impromptu programming called temporal recursion. By writing a function that schedules itself as its final action a temporally recursive callback loop can be established. This is ideal for repetitive tasks such as music and video loops, cycling graphical processes and evolutionary systems. Impromptu allows any number of temporal recursions to be active at once. You can even create temporal recursions inside temporal recursions each running at its own pace. This timing flexibility presents many opportunities for time-based media activities.

IMPROMPTU'S SYSTEM ARCHITECTURE

Impromptu is an amalgam of six primary components - a real-time scheduler, a Scheme language interpreter, an Integrated Development Environment (IDE), an audio plugin architecture, a vector based graphics engine, and support for input-output protocols. In this section we'll take a brief look at each of these components.

Scheduler

At its most fundamental level, Impromptu's real-time scheduler is a priority queue containing programmed tasks to be executed. This is the basis for all precisely timed activity in the Impromptu system. Scheduling in Impromptu is audio sample based and attempts to guarantee sample accuracy.

AudioUnits

Impromptu uses Apple's CoreAudio engine, and more precisely Apple's AudioUnit plugin architecture for all audio processing. Apple's AudioUnit library provides a plugin architecture allowing individual AudioUnit's to be connected together to form signal processing graphs of arbitrary complexity. The AudioUnit standard is well supported and provides Impromptu with an enormous array of commercial and non-commercial virtual instruments, filters, delays, reverbs, mixers, generators, and so on.

Scheme interpreter

Impromptu uses a modified version of the TinyScheme interpreter, a light-weight Scheme interpreter written in the C programming language. The interpreter runs as a service allowing local or remote connections. This provides a flexible architecture whereby multiple remote users can share a single interpreter or a single user can run multiple local interpreters. Each new interpreter started on a single host shares the same AudioUnit Graph and internal clock but runs within its own Mach kernel thread. A shared clock ensures that multiple local interpreters can be temporally synchronised. Impromptu also provides the ability to set the internal clock, allowing remote interpreters to synchronise across the network. Impromptu adds to the standard Scheme language numerous additional functions that support scheduling, media creation, and interaction with external devices and services.

Integrated development environment

In order to evaluate Scheme expressions 'on-the-fly' Impromptu includes a dynamic code editor that is designed

to assist real-time, dynamic programming. As well as supporting standard features such as colour-syntax highlighting, bracket matching, auto-indenting and code markers, the editor also supports the ability to send code to the Scheme interpreter for evaluation. This is achieved by pressing the evaluate button (or equivalent keyboard shortcut) while the cursor is located within a valid Scheme expression. The Impromptu IDE also supports custom AudioUnit User Interfaces, allowing users to modify AudioUnit parameters in real-time via their User Interface as well as programmatically through Impromptu library functions.

Graphics

The Impromptu IDE also incorporates graphics canvases. These canvases can be used to display bezier paths, font rendering, image manipulation, OpenGL graphics and QuickTime movies. Animators can schedule drawing commands and implement multiple time independent animations using temporal recursion.

Interaction

Impromptu supports standard UI event capture including mouse down, mouse up, mouse drag and QWERTY key up and down. It also supports communications via Open Sound Control, the Musical Instrument Digital Interface (MIDI), and raw inter-process communications between Impromptu systems. By using the Scheme-ObjC bridge it is possible to add support for other hardware interfaces and network protocols as required.

Extension

Impromptu includes a bidirectional, runtime, Objective C bridge that allows Scheme to instantiate and call Objective C objects and for those objects to call back into the Scheme interpreter. This provides it access to OS X library functionality and purpose built frameworks.

CONCLUSION

The development of dynamic Media Art works, such as the collaborative jamming systems created by the researchers, is made easier by development environments that share the priorities of media artists. Impromptu is a dynamic software development environment for Mac OS X, version 10.4 +, that has been optimized for time-based, interactive works. The features and interface of Impromptu have been designed to allow the artist/developer to rapidly create, test and modify their computational ideas. Information about the dynamic media jamming environments mentioned here is available from <http://acid.net.au> and Impromptu is available for free download from <http://impromptu.moso.com.au>.

Acknowledgment

The work reported here has been supported in part by the Australasian CRC for Interaction Design, part of the Cooperative Research Centre Programme of the Australian Government's Department of Education, Science and Training.