Virtual Sound Control Project Description

Jonathan B. A. Thorpe
NXP Software (Current Employer),
johnny.frenchy@gmail.com

January 11, 2013

Contents

1 Introduction

This document describes the Virtual Sound Control Project (VSCP). This project aims to provide novel tools for designing virtual environments allowing the real-time control of sound synthesis. Current development is focused on the sonification of virtual object manipulations. The project is open source (BSD style licence, allowing use in commercial products), currently developed by the author of this document, but future contributions from other developers and cooperations with academia and industry are hoped for.

Firstly, some background to the challenges of compound instrument design will be given in Section 2.1, with specific attention to gestural control and multimodal feedback. The specific scope and motivations for the VSCP will then be described in Section 3, explaining some of the main ideas behind the project and clarifying it's goals. Sections 4 goes into fairly detailed technical considerations, explaining choices of platforms, programming languages and libraries. Finally, a current project status is presented in Section 5, this will be kept up to date as much as time allows.

2 Background

2.1 Natural vs Compound Instrument Design

When using an acoustic instrument the performer is directly interacting with the physical sound production processes. The relationship between the musician's physical input and the sonic output of the instrument is determined by physical laws. The complex and intricate nature of the acoustic processes allows the sophisticated, organic and highly malleable sound achieved by experienced performers. The processing power offered by widely available modern computing hardware allows complex sound synthesis processes to take place in real time and poses the problem of control and interface design. In deed in the case of software based synthesis, the sound generation processes are decoupled from the performer's control input. This de-coupling characterises "compound" instruments.

Interfaces allowing real-time control of virtual sound synthesis methods both benefit and suffer from this decoupling. Although the relationship between performer input and sonic output is no longer constraint by the physical laws that govern the behaviour of systems such as membranes, strings or tubes, the problem of designing a sonic production mechanism allowing the same level of expressivity and control as these physical mechanisms is a very challenging one (Wanderley and Battier (2000); Schnell and Battier (2002)). Musicians should be allowed to explore the new sound worlds offered to them by ever evolving sound synthesis processes in ways that allow the degree of feeling, intricacy and virtuosity witnessed in purely acoustic performances.

This problem has been approached in many ways. The separate of physical interfaces from the sound production mechanism gives complete freedom, not only to mimic the types of interaction witnessed in acoustic interments, but to extend them, and adapt them to the new possibilities offered by present sound synthesis techniques.

2.2 Gestural Sound Control

The ultimate objective of a compound instrument is to allow a high degree of freedom of expression and the effective, intuitive exploration of synthesised soundscapes. A number of interfaces have been developed, which capture control input through devices like buttons, sliders, light beams or touch tables. These have given rise to a number of musical performance modalities, however, they do not allow anywhere near the level of expression offered by acoustic instruments. The simple reason for this is that they capture gestures as a very small number of parameters (position of the hand or finger tips for example).

A complete description of the physical input required for acoustic instruments to generate the rich expression which they are capable of would be highly multi-dimensional and accurate. If compound instruments are to achieve comparable levels of expression, capture devices need to allow a far more detailed picture of the performer's input than that provided by buttons and sliders. Research in a number of areas such as robotic surgery, micro assembly and industrial/military robotics is driving the development of ever better performing motion capture and tactile feedback devices. Relatively affordable hardware allows amongst others, finger flexion and abduction, pinching and grabbing actions, wrist movements and rotation to be captured potentially allowing high-dimensional gestural capture. Devices such as accelerometers and gyroscopes are getting ever smaller and more accurate and are already used in a number of commercial devices to provide affordable gesture tracking.

Although some aspects of acoustic instrument control such as elaborate tactile/force feedback are still difficult to reproduce, the technology used for 3D virtual reality shows great potential for the development of real-time sound synthesis control. As well as allowing detailed and accurate gestural capture, sensor technologies potentially outperform other capture techniques in the context of musical performance. They are not, for example dependent on environmental lighting (like techniques such as image analysis) which during a performance should not be relied upon as complex lighting effects are commonly used to enhance musical performances.

2.3 The Role of Multi-Modal Feedback

Sophisticated multi-modal feedback is crucial for the development of technical virtuosity in acoustic instruments. Experimentation and exploration allow musicians to associate subtle and complex nuances in physical input to their sonic results. Although a quasi infinite number of physical parameters are controlled simultaneously, performers do not think of the actual physics of the sound production mechanism, they rely on automatisms developed through experimentation and practice to relate the acoustic output of their instruments with their physical input. Physical input is then mostly controlled using visual awareness, haptic sensations and proprioception. The importance of immediate multi-modal feedback during instrumental performance and learning cannot be understated.

The performer's knowledge of the synthesis processes taking place should not be required just like a traditional acoustic performer isn't aware of the physical sound production processes, just of the resulting sonic output and the haptic sensations induced by physical interaction. The importance of real-time multi-modal feedback in the case of compound instruments should therefore be just as paramount as in the case of their acoustic counterparts. This feedback is crucial for the performer to feel more involved and engaged with the instrument. The technology is available for visual and haptic feedback to be provided, however they have to be generated as they do not intrinsically result from the physical input.

The generated feedback should to some extend mimic expected natural, physical responses so as to allow intuitive knowledge of natural processes to be built upon. But should also exploit the dematerialisation of the feedback generation in order to produce new types of feedback providing useful information though they may at first seem unusual or alien to the performer.

3 Scope and Motivation

The problem of gestural control of sound synthesis is known to be a challenging one. Garnett and C. (1999) frame the problem at hand with a number of questions relating to the manner with which performers should/could conceptualise the relationship between their physical gestures and the sonic results. For example, they ask if the system should produce the same sound given the same gesture, if a slight change in gesture should result in a slight change in sound. More generally they recommend a relation between sounds and gestures analogous to that which we are habituated to from the physics of the non-synthetic world. They attempt to minimise what they refer to as the "cognitive retraining" problem (which forces performers to retrain their intuitions regarding the sound production mechanisms) by "allowing the designer to associate particular points in the control parameter space with particular sonic results; [...] from this association [...], the system creates a mapping from the space of control parameters to the space of synthesis parameters [which is] consistent, continuous, and coherent." (Garnett and C., 1999). The VSCP is essentially an extension of this concept.

3.1 Interaction in Virtual 3D Space

This section describes the potential interaction within 3D virtual environments for controlling sound real-time sound synthesis.

3.1.1 Hardwired Human Abilities and Cognitive Potential

Humans understand the world in three dimensions. Our natural environment has shaped us so. A huge part of the brain is dedicated to the fine control of the hands and to real world object manipulation allowing incredible levels of potential dexterity given enough practice and experience. Any interface should to some extent allow performers to exploit this great resource to minimise a potential interaction bottleneck. Reducing it to a few parameters (sliders, knobs, keys, light-beams, touch-tables) falls short of capturing the detail and complexity which the human brain is capable of handling, and which is obviously expressed during virtuosic performances on acoustic instruments.

3.1.2 Real-world Experience and Intuitions

As well as an inbuilt understanding of three dimensional environments, humans also have a natural sense of the sonic properties of real-world objects (depending on size, material, shape etc...). We also have an intuition for the sonic effect of impacts (collisions) on the object. For example a larger collision speed is intuitively linked to a louder, more explosive sound. Or a harder or more salient impactor object is intuitively linked to a brighter, more high-frequency-rich sound (hitting a drum skin with a finger nail as opposed to the palm of the hand is a possible illustration of this). Examples are numerous.

The three dimensional representation of the world which is hard wired in the human brain, combined with a sophisticated intuition for the sonic characteristics of objects within it potentially allow hugely powerful interaction paradigms to be developed. Specifically, by emphasising the context sensitive nature of gestures.

3.1.3 Gestures in Context

We perceive are gestures as context sensitive. Swinging an arm will not have the same effect if it is done in thin air or if an object is in the way. A drummer's sticking motion will not have the same effect on a cymbal or on a snare drum, even though the gesture maybe identical. A number of gesture based sound control paradigms have suffered from removing this context, linking sound to gestures alone, not gestures in context. Particularly, the performer's cognitive load increases dramatically if the full breadth of his control potential is encoded purely in his gestures, and not in the way his gestures affect his environment.

It is also important that performers are allowed movements which aren't related to the sound control processes. "Instrumentalists simultaneously execute various types of gestures during performance. Some of them are necessary for the production of sound, others may not be clearly related to sound production, but are nevertheless present in most highly-skilled instrumentalists performances" Schnell and Battier (2002). The feeling that the slightest movement will inevitably affect the sonic output will result in the unwanted effect of feeling trapped or suffocated by the performance environment. The context in which gestures are effective should therefore be restrained spatially. A drummer can, for example, move his stick around in space knowing that sonic effects will only occur when contact is made with elements of his kit.

Approaches based on virtual object manipulations naturally address these issues. For example, sound can be generated when contact is made with virtual objects, with clear parallels to real-world interactions. The following section expand this concept.

3.1.4 Collisions, Sound and Multi-Modal Feedback

Probably the most obvious sonic interaction concept in a virtual environment is that of object collisions. Our natural intuition for colliding objects to produce sound is therefore the first route this project is aiming to explore. As a simplification, objects in a virtual environment can be split in three (non mutually exclusive) categories:

- **Sonic:** Sonic objects are the sound producers, when an object collides with a sonic object, it triggers a chain of (sonic) events, which (as described later) should be entirely configurable.
- **Effector:** An object is an effector if it is used to generate sound by colliding with sonic objects.
- **Controller:** Controller objects are controlled by the performer to create collisions with sonic objects directly (in which case they are also effectors) or indirectly (through other effector objects).

Real-world equivalents of sonic objects would include a drum skin or a guitar string. Pure effector objects would include a bow, a drum stick or a plectrum. Pure controller objects would include the hands of a drummer, or the bowing hand of a violinist. Controller-effectors would include the hands of a tabla player, or the fingers of a finger-picking guitarist. In a 3D virtual environment the controller objects would be, for example, a virtual representation of the performer's hand or other body parts. In simpler interaction paradigms, a controller object could also be a box dragged by a mouse, for example.

When an effector-sonic object collision occurs a multi-modal response should result to enhance interactive immersion and learning/experimenting effectiveness (see Section 2.3). Of course sound should be produced given it is the response of primary importance, but it could be complemented by visual and haptic feedback if the available hardware allows it. Visual feedback could include changes in colour and texture of the sonic object, as well as particle streams or other more elaborate graphical niceties. Even though the generated feedback should emulate real-world phenomena, it can also be extended, giving the sonic objects seemingly magical properties. In any case, the design of the feedback generation process is of paramount importance, and will be expected to be ever-evolving and improving.

3.2 Approach to Performance Environment Design

3.2.1 External Tools for Creating 3D Objects

There are a number of 3D shape editing environments which can be used to define the original shape of virtual objects. This will therefore not be part of the proposed project. Software packages such as 3DS max are a powerful but expensive option, other possibilities are available such as the Open Inventor graphical toolkit and an interpretive version of it developed by Fels and Mase (1998) called InvenTcl. The Blender freeware package is also very powerful and can be scripted using the high level Python programing language. The development of higher level tools for a performer to shape his performance environment and to visually represent the sonic attributes of virtual objects as he wishes will be the next set of objectives for this project.

3.2.2 Mapping Sonic Properties to Visual Cues

In order to reduce the cognitive load imposed on the performer, the environment should give visual cues to the performer regarding the sonic properties of objects. As mentioned in Section 3.1.2, we can, often correctly, infer the sonic character of an object from its visual appearance. For example a large object will be expected to produce more low frequency content than a smaller one; a shiny, metal like object will be expected to produce sharper sounds (in fact we can describe sounds as metal-like, glass-like, wood-like, referring directly to the sound producing material rather than to the sound itself). Mapping visual characteristics to sonic ones therefore seems like an obvious design decision. Specific examples could include:

- Colour, described in RGB space could be used to determine timbre, as a combination of three primitive timbres mapped to the RGB colour components. This would give a continuous three-dimensional colour/timbre mapping.
- Specular reflections (more informally "shininess"), could represent emphasis on the higher-frequency components of the sound.
- Texture/timbre mapping could be developed and although it is not as straightforward as colour mapping, it offers great potential.
- Object size/scale can be linked to a timbral frequency shift.

An object with varying surface properties would then produce different sounds depending on the local properties of the collision location. Note that these concept can and should be extended beyond the pure mimicry of real-world objects.

3.2.3 Mapping Sonic Properties to Collision Parameters

As well as visual cues given by the sonic object themselves, parameters describing the effector objects and the effector-sonic object collisions should also have an effect on the resulting sound. Again, as mentioned in Section 3.1.2, humans have natural expectations regarding this, which should be exploited in order to reduce the cognitive load on the performer. Again, a (very much non-exhaustive, non-definitive) list of possibilities:

- Collision velocity (relative velocity of the colliding objects) can be mapped to sound volume, shorter attack times for the volume or filter cutoff envelopes etc...

- Collision direction vector components can be mapped to timbral properties (for example, hitting an object from underneath will have a different sonic effect to hitting it from above).
- The distance of the collision location to a particular point (in local object space or in absolute world space), can be mapped to pitch.
- Effector object properties should also be variable. For example the shape and texture of the effector could be mapped to different excitation signals for synthesisers based on physical modelling.

3.2.4 Flexibility, Configurability, Extendibility

The main difference between acoustic and compound instruments is that the effect of physical input on the former is completely determined by the designers, without being constrained by the physical rules which govern the prior (see Section 2.1). Although defining original control paradigms for untrained musicians to experiment with is desirable, making the interface immutable and preventing musicians from shaping and adapting the interface and sonic generation processes to suit their needs which will inevitably grow in sophistication, individuality and evolve with experience and practice would deprive compound instruments from their main asset.

Musicians should be involved in the design and evolution of their performance environment, and the VSCP aims to address this by providing tools to this end. In the case of three dimensional interaction, performers should be allowed to shape and personalise the virtual objects they interact with as well as defining their sonic properties, by mapping different descriptors, such as visual cues on the surface of sonic or effector objects, collision descriptors or others (see Sections 3.2.2 and 3.2.3) to sound synthesis parameters (which can be as low or high level as need be). The multi-modal feedback generated by the sonic feedback should also be entirely configurable.

Acoustic instruments have been developed, sophisticated and optimised over hundreds, even thousands of years to achieve the best possible results. The design of compound instruments is still in its infancy and reaching optimised forms will require a lot of experimentation and evolution. Just as experimenting with the acoustic properties of raw materials (such as wood, nylon, metal and other materials) were the basis of acoustic instrument design, experimenting with and developing different real time synthesis control paradigms will be the basis of the development of future compound instrument design. Eventually the process of experimentation and environment shaping should allow performance environments to evolve from archetypes into forms which, although unforeseen by the original system developers, will provide a range of progressively optimised control setups and configurations. This should be expected to be a long process, the digital equivalent of the journey from blocks of wood to violins.

3.2.5 Multiplicity of Interaction Paradigms

Although the descriptions so far have focused mainly on object collisions, the environment should not constrain the performer to this particular type of control technique. Indeed, the notion that "an authoring environment for composed instruments has to integrate multiple models of computation and programming paradigms in correspondence to a variety of different interfaces for expressive musical performance" (Schnell and Battier, 2002) is very much a guiding principle of the VSCP. As an example, the performer should also be able to control the sound characteristic after the initial contact between effector and sonic object. This process essentially extends the concept of after-touch for electronic keyboards, from a simple volume control to the multi-dimensional control of any number of sonic properties. All sounds should be terminated using a a common gesture, complete clenching of the fist for example.

More generally, virtual/augmented environment have the potential to incorporate any type of interaction whether it's augmented instruments, 2D/3D pure virtual object collisions (through rigid-body, and possibly later soft-body physics simulations), continuous control of sound synthesis parameters (using hand position or speed for example), 2D/3D composition of loops, activators (discrete event triggering). All these, and others should, ultimately, be goals for the project. Both discrete control events and continuous flows of control data are required for a powerful interface. Also, mixing completely deterministic control of musical processes with high level control of self evolving sets of musical processes could also be allowed. Users should be able to switch effortlessly between different types of control techniques during the performance.

Virtual objects should also allow a wider range of possible actions covering a number of different semantic categories such as discreet event triggering, performance tool selection or continuous real time interpretation of scores and other predefined sequences.

Additionally, although virtual/augmented environment have great potential, physical systems still offer an unparalleled degree of haptic interaction and should also be used as input devices, exciting virtual sound production mechanisms for example. One of the aims of the VSCP is to allow these physical input devices to be used along with the augmented/virtual environmental elements. Using the voice or facial expressions as input control devices is a particularly attractive possibility as it allows both arms/hands to be devoted to the control of other processes. The voice is particularly interesting as an input control mechanism as it is hugely flexible, controllable and yet cheap to capture.

4 Technical Considerations

4.1 Requirements

Some fundamental requirements for the project have guided the choice of tools which have been chosen as a basis for the development of the VSCP.

- **The tools should be portable.** Although current development is made on OSX, a primary objective of the VSCP is to run on as many platforms as possible, including OSX, iOS, GNU/Linux, Android, Windows, Sony PS, XBox. They should not require any host environment (such as Max/MSP/Jitter) in order to run but should, rather, be able to run natively on each platform.

- **The tools should have permissive licences.** Although currently a prototyping, experimental project, the possibility of it growing to be incorporated into commercial products should be kept in mind. As such open source projects with licenses permissive enough to allow this are preferred. BSD/MIT/Zilb style licences are excellent, Apache style licenses are alright, LGPL style licences are very much discourage (dynamic linking being impossible on iOS, for example), GPL style licenses are entirely out of the question (although GPL projects such as super-collider can be used as peripheral to VSCP, controlled by MIDI/OSC for example).
- **The tools should be fast.** VSCP is a project for real-time interaction. As such, performance is paramount, and libraries designed with this goal in mind are preferred.

4.2 Choice of Programming Languages and Third Party Libraries

4.2.1 C++ as the Main Programming Language

C++ was chosen as the main, cross platform development programming language, as it is a particularly well suited fit to the requirements listed in Section 4.1, in particular portability and performance. Also a number of high-quality C++ libraries, particularly suited to the project and described in the following sections, further cemented C++ as the ideal principal language for the project.

4.2.2 C++ Boost Libraries

One of the most highly regarded and expertly designed C++ library projects in the world (at least according to Herb Sutter and Andrei Alexandrescu, who are pretty much as high an authority as you can get). The Boost C++ libraries are used throughout the project for a wide range of applications, including smart pointers, threads, binding and signals, asserts, loop utilities, timing and date utilities and will probably use more of these excellent libraries as it grows. It is used in place of C++ 11 equivalents, while compilers catch up.

4.2.3 C++ OGRE Library

The OGRE (Object-Oriented Graphics Rendering Engine) C++ library, is used for all things relating to graphics. It is mainly designed for games and real-time simulations, so is well suited to the needs of the project. It offers a wide range of useful abstractions for scene managers, meshes, geometric primitives, colours, textures, cameras, lighting, particle systems and much more (you can find a more detailed feature list here). Also, a number of side projects offer tools for bridging to popular 3D editing tools such as Blender, Maya or 3DS Max.

4.2.4 C++ Bullet Library

The Bullet C++ library is a well established collision detection and rigid/soft body physics simulation engine. It is used extensively in games and other real time systems as well as in CG movie production studios. There is also a wrapper available to facilitate the synchronisation of Bullet physics simulations and graphics rendered using OGRE (see Section 4.2.3), called OgreBullet.

4.2.5 C++ OIS Library

The OIS (Object-Oriented Input System) C++ library offers a cross-platform solution to handle general user input. Keyboards, mice and different types of joysticks are catered for.

4.2.6 C++ STK Library

The STK (Synthesis Tool Kit) C++ library, is a project aiming to provide a framework to accelerate the development of musical applications. It in corporates the RtMidi and RtAudio libraries which provide cross platform MIDI and audio input/output facilities, adding a layer of synthesis primitives (generators, processors, filters, physical modelling etc...).

4.2.7 OSX, iOS, Cocoa and Objective-C++

The main user-interface currently used for development is on OSX. It uses the Cocoa framework, keeping the interface and the core elements as separate as possible to minimise the effort of adapting the interface for other platforms. C++ can be used directly, mixed with the native Objective-C language of the Cocoa framework, in an unholy marriage of syntax where both C++ and Objective-C co-exist without really inter-mixing, named Objective-C++. The same applies to iOS.

4.2.8 GNU/Linux, WX and GTK C++ Libraries

Although not currently in development, possible user-interfaces written using WX or GTK would be possibilities for porting VSCP to Linux.

4.3 Hardware, Sensors and Control

4.3.1 Control Input for Virtual/Augmented Reality

For virtual object manipulation to take place, robust capture devices should allow the movements of the performer to be replicated in the virtual environment with the finest possible detail. A good correspondence between the virtual representation of the performer's movements and the expected movement sensed through proprioception is crucial. Failure to achieve this will result in discomfort, irritation and possible "virtual sickness". Calibration is key to providing a comfortable virtual representation and may be achieved by prompting the user for different default arm, hand or finger positions.

Although a virtual/augmented reality setting would be ideal, and should be a long term objective, the possibility Currently the system uses simple mouse and keyboard input, and plans to add support for MIDI/OSC input controllers are in place for the near future. Further ahead the system should allow more sophisticated input devices such as sensors, virtual reality gloves, and devices capable of transmitting haptic feedback. The bulk of VSCP aims to be abstracted away from hardware consideration. However, it should offer be a flexible and extendable system to interface with specific hardware control input devices. Ways of allowing the control input to be routed across the system, adjusted and calibrated, will be necessary as the system evolves to handle a fully fledged virtual/augmented reality experience, if this does indeed occur.

5 Current Project Status

The project is currently hosted on Github. The repository can be found at:

https://github.com/jbat100/VirtualSoundControl

The main active branch being the develop branch, following the gitflow branching model.

5.1 Core Classes (C++)

Overview of the most important classes and their roles. Proper documentation generate with tools like Doxygen from header comments, will be created when time allows. The content of the root folder in described briefly here.

5.1.1 Application

Classes in this folder provide global application level tools.

- **GlobalApplication:** Used to create and monitor **Environment** (or derived class) objects. Uses a singleton pattern.
- **Environment:** An abstraction for a performance environment. Currently is composed of a **Scene** (or subclass) instance (see Section 5.1.6), and a **CollisionMapper** (or subclass) instance (see Section 5.1.4). Other elements (envelope manager, synthesisers etc) will probably be added later.

5.1.2 Envelope

Provides abstractions for envelopes which can be used for anything. Main current application is for MIDI/OSC envelopes.

- **EnvelopeCoordinate:** Time/Value coordinate in envelope space.
- **EnvelopePoint:** An envelope point (with possible associated control points)
- **Envelope:** Contains a set of points and abstractions for envelope manipulation.

5.1.3 Interface

Provides generic tools for interface related things.

- **Bindings:** Template class allowing actions to be bound to inputs.
- **Bound:** Template class which has bindings associated with it.
- **InterfaceAdapter:** Template class to relay callbacks for keyboard/mouse events from platform specific layer.
- **Responder:** Template responder chain implementation.

5.1.4 Mapping

Provides generic tools for mapping interface input control to actions. Specialist classes allowing collision mapping are also in development.

- **Event** subclasses (**Delay** and **Action**, which is subclassed further) objects are arranged in **EventChain** objects which can be triggered (triggering all the **Action** objects in the chain, with appropriate delays).
- When **Actions** (or subclass) objects are triggered, they produce **Task** objects which are sort of executable blocks with target execution dates (See Section 5.1.8).
- **Mapping** (or subclass) objects provide ways of mapping different **Action** parameters, when they are triggered, so that they provide varying **Task** objects.
- **CollisionMapper:** Listens to Scene **Collisions** and triggers **EventChain** objects associated with each Scene **Element** (see Section 5.1.6).

5.1.5 MIDI

Provides high level tools to discover MIDI devices, open input/output ports, send and receive MIDI messages. Currently only output is implemented.

- **MIDIOutput** is used to send MIDI messages to an output port (uses **MIDIMessageGenerator** internally).
- **MIDIOutputManager** is used to detect MIDI outputs (using **MIDIPortManager** internally). Also used to open output ports and giving access to them (by returning **MIDIOutput** objects).

Future classes for MIDI input will include **MIDIInput**, **MIDIInputManager**, and **MIDIInputListener**. The MIDI folder also contains **Task** subclasses for MIDI specific tasks.

5.1.6 OB

Provides abstractions for virtual reality displays, scenes, elements, collisions, element factories as well as cross-platform user interface (see Section 5.1.3) controllers for these abstractions.

- **Scene** objects represents a virtual reality scene. They contain **Element** objects which are created using **ElementFactory** (or subclass) objects.
- **CollisionDetector** (or subclass) objects are used by scenes to detect collision which are represented by **Collision** objects. Callbacks are sent to **CollisionListener** objects when collision related events occur.
- **Display** objects represent a display of the scene, with an associated camera. This is the last cross-platform link before the OS specific UI elements.
- **SceneController** and **DisplayController** objects listen to user input (they are **Responder** subclasses as described in Section 5.1.3). They use this input to control their associated **Scene** or **Display** object, respectively.

5.1.7 Sound

Attempts at a flexible, multi-channel, real-time sound synthesiser. Not currently in development, switched to sending MIDI/OSC messages to external synthesisers instead. Very motivated to continue development later.

5.1.8 Task

Provides abstractions for runnable background tasks.

- **Task** objects are used to encapsulate behaviour. The behavior is performed by calling "step", after which the task can switch to finished state, or not, in which case step should be called again, until the task is finished. Tasks have a target execution date (when they will start execution).

- **TaskQueue** objects are used to run **Task** objects. Each queue has a single internal thread which it runs the tasks on in a kind of run loop. On each pass through the loop the queue will "step" task execution (once) for each task currently executing (if the task's execution date is not reached then it does nothing). If the task is finished after step, it is removed from the queue. This gives a kind of illusion of parallel execution on a single thread. The time between each iteration is configurable (a queue dedicated to MIDI tasks for example would not not gain much iterating above 400Hz, so it should be as lazy as it can).

5.2 OSX Interface (Objective-C++)

Files implementing OSX/iOS specific aspects are located in the Apple directory. Currently the project is being developed using an OSX interface only. It is early stages and everything is being done to facilitate future ports to other platforms.

References

S. S. Fels and K Mase. Inventcl: A fast prototyping environment for 3d graphics and multimedia applications. In *AMCP '98 : advanced multimedia content processing*, Osaka, November 1998. 3.2.1

G. Garnett and Goudeseune C. Performance factors in control of high-dimensional spaces. San Francisco, CA, 1999. International Computer Music Association. 3

N. Schnell and M. Battier. Introducing composed instruments, technical and musicological implications. In *Proceedings of the 2002 Conference on New Instruments for Musical Expression (NIME-02)*, pages 1–5, Dublin, Ireland,, May 2002. 2.1, 3.1.3, 3.2.5

M. M. Wanderley and M. Battier. *Trends in Gestural Control of Music*. Ircam - Centre Pompidou, 2000. 2.1