Sound and Graphics in CsoundAV

Gabriel Maldonado, Italy

Introduction

Csound is one of the most famous sound-synthesis languages belonging to the Music-N family. This family of languages, that appeared at the early sixties, provided huge generality and synthesis power at the cost of deferred-time rendering, of laborious project planning and designing, and of the lack of interactivity.

First versions of Csound have been developed at MIT and appeared around 1984. Csound was, in a certain way, a true revolution, because it was written using a high-level language (no machine code anymore), so it could be ported to most platforms, even the most heterogeneous ones. At the beginning of the '90s some stuff had been added to use Csound in real-time. At that time the only machines capable to achieve real-time, were the Silicon Graphics and some other expensive high-end UNIX workstations, available only in academic and advanced research contexts. Some years later Intel-based PCs became fast enough to run Csound in real-time, and, nowadays, low-latency AUDIO support, MIDI support and many real-time oriented opcodes[1] have been implemented, to allow the user to control Csound as a sort of musical instrument.

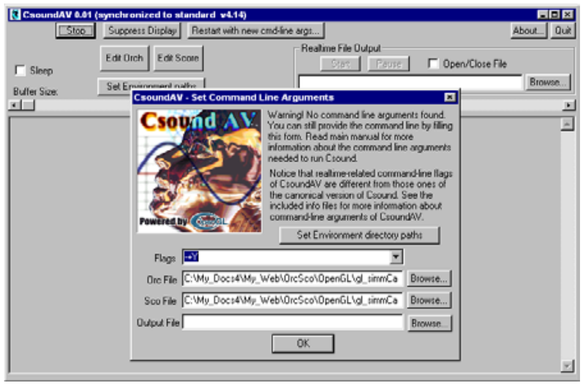
Csound is now one of the most profitable tools to make computer music with, because of its power, its platform and operating-system compatibility, its cost (it is free), its source code, that is freely available to allow users to modify it in order to adapt it to their own requirements and to enhance it, its big circulation and renown, and its plenty of documentation and related resources, such as auxiliary programs, utilities, articles, magazine review, papers, books etc. Being it written in ansi C language, which is universally available, it is very simple to port it to any kind platforms.

The orchestra-score[2] paradigm of Csound is surely somewhat limited, but last extensions and newer opcodes allow it to surpass most constraints. For example, newer stuff related to DirectCsound[3] allow it to generate and play-back numeric scores inside Csound engine[4] itself, at run-time.

CsoundAV

In this paper I will not deepen standard Csound aspects and old DirectCsound features. There are a lot of resources dealing with these topics, most of them are available in the Internet[5]. I will briefly touch only new CsoundAV features, particularly graphics-related stuff.

CsoundAV is the new name given to latest versions of DirectCsound. “DirectCsound” naming will not be used anymore. For anybody already knowing DirectCsound, it should be clear that **all its features are still present** in CsoundAV.



CsoundAV - Set Command Line Arguments

Warning! No command line arguments found. You can still provide the command line by filling this form. Read main manual for more information about the command line arguments needed to run Csound.

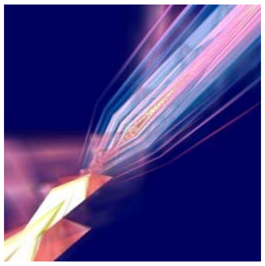
Notice that realtime-related command-line flags of CsoundAV are different from those ones of the canonical version of Csound. See the included info files for more information about command-line arguments of CsoundAV.

Flags: -+Y
Orc File: C:\My_Docs4\My_Web\OrcSco\OpenGL\gl_simmCa
Sco File: C:\My_Docs4\My_Web\OrcSco\OpenGL\gl_simmCa
Output File:

CsoundAV (AV standing for Audio-Video), is a major extension of DirectCsound. CsoundAV supports real-time computer graphics in both 2 and 3 dimensions, basing on OpenGL. It contains a numerous set of opcodes specific to OpenGL and graphic generation control. An additional engine (alongside normal audio engine) is provided to support real-time graphic. This engine works at a new rate (different from Csound k-rate) called **frame-rate**. This rate can be fixed or variable and can be specified and managed by means of special opcodes.

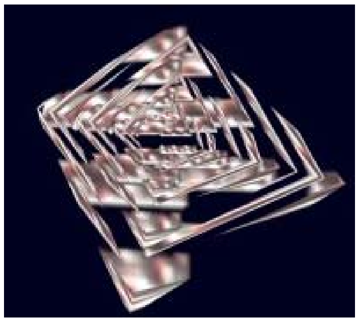
CsoundAV provide the following features (not present in Public Csound nor in previous DirectCsound versions):

- Low-level OpenGL API-wrapping
- middle-level and glu-related opcodes
- High-level OpenGL opcodes
- Flow of control for graphic engine
- Frame-rate math operators
- Frame-rate vector operators
- Frame-rate signal generators
- Video-related
- Pixel-based image processing
- Data output of spectral opcodes
- Recording k-rate signals into tables and playback them
- traverse a sequence of multi-parametrized events
- Send a single trigger signal to different outputs, according to a user-defined scheme

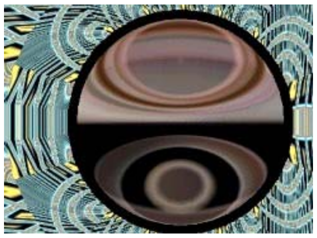


Low-level OpenGL API wrapping allows to call most OpenGL function from within Csound. This allows the huge graphic power of OpenGL to be merged with Csound control signals and its scheduling capabilities. Actually, it is possible to play notes that not only produce sound, but can also generate two or three-dimensional graphics.

Graphic engine setup-related opcodes allow the user to define the insertion positions of OpenGL instruction blocks inside instruction loop chain of graphic engine. This means that the sequential order of instructions processed by graphic engine is under user control. So, even when two Csound notes are activated at different times, user is allowed to decide that some OpenGL instructions will be inserted before than other ones belonging to a concurrent note of a different instrument, and other instructions will be inserted later. This is very important, because OpenGL is a state-machine, i.e. subsequent instruction behavior depends on the state affected by previous instructions. Practically, this allows, among other things, to insert/remove entire animated-graphics scenes having different status settings independently from the fact they are concurrent or subsequent (in the same way of audio notes, that can be played in succession or within chords), taking it into account the precedence of OpenGL instructions belonging to different instruments. User is able to define this precedence. This feature is made possible by means of special opcodes that delimit instruction blocks and allow to set the precedence of blocks themselves inside a single Csound instrument.



This group of opcodes allow the user to decide to change graphic output frames automatically (in this case the user only has to set the frame-rate) or under his/her control (in this case the user has to call each frame drawing explicitly, by a special opcode).



Middle-level graphic opcodes implement API wrapping of most GL Utility Library (GLU) functions.

High-level graphic opcodes implement some GLUT API function wapping and the visualization of text strings made up of three-dimensional fonts.

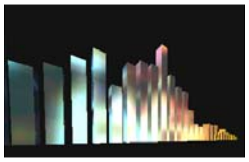
Flow of control opcodes allow to define loops and *if-then-else* control flow of graphic-related instructions.

Frame-rate math operators, *vector operators* and *signal generators* are opcodes that operate at frame-rate but don't affect graphic output directly. They are used to calculate, generate or modify signals that are connected to the inlets of OpenGL-related opcodes, instead.

Video-related opcodes allow to convert AVI file output into time-variant texture-mapping objects. There is also the possibility to capture real-time video via camcorders or other devices and convert it directly into time-variant texture-mapping objects.

Pixel-based image opcodes allow loading pix-map image into OpenGL texture objects and processing pix-map images on pixel basis (for example to modify RGB-alpha gain, filter, convolve, or to apply arithmetic operations to a couple of images). The final result can be converted into a texture-mapping object. Furthermore image data can be used to generate or control audio by scanning pixel data themselves.

Output of *spectral opcodes* (such as "***spectrum***") can now be *copied into a table*, in order to allow to access its data by Csound opcodes that don't handle ***w-type*** data. This allows, among the other things, to use the variant spectrum data of an audio signal to control real-time graphics motion.



CsoundAV contains two opcodes that allow to *record* control-rate signals by storing them into tables and to *playback* them when the user decides to do it. This permits, among the other things, to have a sort of "middle-term-memory" of real-time generation data, that can be used to supply generative music with a coherent compositional structure. Actually *trigger signals* are in all respects control-signal and can be recorded/played-back too. So note events can be recorded and scheduled back too.

Two special opcodes, named ***vphaseseg*** and ***GLvphaseseg*** allow one-dimensional ***HVS*** (Hyper-Vectorial Synthesis) from within Csound, without using any other external program.

Two special opcodes, named allow to load and render *three-dimensional woman models* generated by Prof. Celestino Soddu and to apply some transformation on their elements.

CsoundAV is free and will be soon available at the Internet at the following web site for download:

http://web.tiscalinet.it/G-Maldonado

NOTES

1. The term “opcode” is used to indicate a sort of function that generates a signal or modifies a previously generated signal.

2. Actually Csound recognizes two languages: the *orchestra* language and the *score* language. Normally a Csound source is provided in two separate text files, having extension .orc and .sco . Both languages are quite simple and easy to learn (orchestra is actually a dedicated scripting language and score language is made up almost entirely of numeric fields separated by spaces), even if a sophisticated use of them require a considerable amount of study and experience.

3. DirectCsound is the CsoundAV predecessor. It was a realtime-oriented version of Csound, running under Windows, and all its features are inherited by CsoundAV.

4. Csound is a sort of runtime compiler, its parser parses and compiles the orchestra when Csound is started by converting each instruction in a C-language function call and inserting it into a linked list during the performance. Performance is done by traversing this linked-list of functions, and by inserting/removing groups of function calls (inserting a group of function calls in the linked-list at run-time, is equivalent to play a note). This allows Csound to be very fast and suitable for real-time tasks.

5. Web sites http://www.csound.org and http://www.csounds.com contain a complete listing of Csound related resources.