Composers Desktop Project

Tabula Vigilans Information Page

Description

Tabula Vigilans (i.e., 'Vigilant Table') is an advanced algorithmic composition language written by **Richard Orton** mostly between 1993 and 1998, with assistance regarding YACC from Martin Atkins. There have been some important additions since then coded by John Fitch. As a **score generator**, it can define and replay a huge range of possibilities, from completely conventional melodic and harmonic music to envelope-shaped multi-event textures with random features.

It is script-based, meaning that the user writes a text file which is C-like in nature. The purpose is (usually) the real-time generation of music.

- It supports #include and procedures, making it possible to structure large projects sensibly.
- It will write multiple output files and can produce formated data files, such as breakpoint files or *Csound* score files, thus providing a way to shape data needed by other programs – and a route to the algorithmic handling of sound files.
- Many fairly low level – and therefore flexible – system, musical and mathematical rules can be called within the script, providing enormously powerful programming tools for musical (and other) programming needs. Some seventy or so music-oriented primitive rules can be combined to create rule-networks, supporting recursive as well as normal control-flow features. Constraint rules permit some advanced compositional capabilities, and even fuzzy logic and other artificial intelligence techniques are supported.
- It supports MIDI In and MIDI Out, defining MIDI voices and channels, scheduling of events, etc. and can act as a real-time MIDI performing instrument.
- It handles nested indexing with ease, a crucial requirement for complex musical relationships.
- It comes with a large number of example scripts designed to take the user from the simple to the more complex, and to illustrate strategies to achieve various musical objectives.
- There is a full manual in HTML with a frame-based indexing system for quick access to any part of the manual.

Sample (elementary) script with comments

```
// - 'chords3.tv'
//A Tabula Vigilans tutorial script

//Effect: Chords are built from a series of intervals
//taken from a file; these are then performed either
//as chords or as up-or-downward arpeggios in short
//sequences. The set of intervals selected can be
//varied by using an additional commandline argument
//(from 0 to 14).


table CHORD[4]   //define an array to hold 4 chords
table INTVL[3]   //define an array to hold 3 intervals

start()
{
  midiset 0, 0                          // Piano
  INTVL fill_table "intvls", arg(1)     // 'intvls' is a text data file
  CHORD[0] = 48                         // start position is low C

  for(i = 0; i < 3; i+=1) {             // FOR loop to build up 1st chord
  CHORD[i+1] = CHORD[i] + INTVL[i]      // from the intervals in 'intvl'
  }
  while(1) {
          xx = try(midichord 0, CHORD[0], 64, 0.2, 4, arp) //midichord is a
          if(xx > 0) {                  // T.V. function involving MIDI Out
                  if(CHORD[0] > 76) {   // getting too high, so -=
                  up = 0
                  }
                  if(CHORD[0] < 30) {   // getting too low, so +=
                  up = 1
                  }
                  if(rand() > 0.94) {   // rand() is a T.V. function
                  wait 0.6              // randomly placed pauses
                  }

                  arp = round(rand())   // randomly activated 'arp'(eggios)
                  swp = INTVL[0] // hard-coded swap routine
                  INTVL[0] = INTVL[1]
                  INTVL[1] = INTVL[2]
                  INTVL[2] = swp
                  if(up) {              // if flag 'up' is 1, go up (+=)
                          CHORD[0] += INTVL[0]
                  }
                  else {                // if flag 'up' is 0, go down (-=)
                          CHORD[0] -= INTVL[0]
                  }

                  for(i = 0; i < 3; i+=1) { // builds up more chords
                          CHORD[i+1] = CHORD[i] + INTVL[i]
                  }
          }
  }
}
```

Algorithmic Composition & *Tabula Vigilans*

Algorithmic composition can relate either to tone generation, or to score generation. (***Tabula Vigilans*** deals exclusively with the latter.) In both instances, a complex of instructions is built up in a script and executed together, creating sounds and/or musical events. These can range from defined melodic sequences to arbitrarily complex musical textures. With algorithmic programming, a useful

(and changing) balance point between the definite and the random can be achieved.

Tabula Vigilans is conceived like a musical spreadsheet in that it 'recalculates' the page at machine speed. This means that any variable or combination of variables can be updated at any time. Thus the script is completely dynamic, and any form of time-varying change can be woven into the music as it unfolds in real-time. ***Tabula Vigilans*** is therefore aptly described as a **real-time performance instrument**.

This makes it particularly useful for handling the information that controls 'expressiveness' in music. In MIDI, this means MIDI controllers, alterations in velocity (loudness) etc.

More advanced applications could define higher level musical formal elements, texture types, generic motivic shapes, stylistic and expressive features. For those with some programming experience, it opens up an endless vista of possible projects – all with immediate musical feedback. ***Tabula Vigilans*** provides a way to move beyond tone generation to score generation and the higher level concepts of musical organisation.

Tabula Vigilans can be proving ground for engineering as well as musical ideas. It can be particularly useful for exploring and developing pre-compositional material.

Yet another application is to create *Csound* score files incorporating algorithmic processes. This score file is then used by *Csound* (together with an independently created orchestra file) to produce **audio** output.

Overall, ***Tabula Vigilans*** – as with other software with algorithmic features – offers a way forward into a truly 21st century approach to musical exploration.

Last updated: 9 June 2006