Software for vector synthesis and performance

Douglas Nunn
Cambridge School of Creative Industries
Anglia Ruskin University
douglas.nunn@anglia.ac.uk
Why vector graphics?

• Raster graphics dominate computer graphics, including games/video/animation/art
• But vector graphics offers...
  • 'infinite' resolution (no pixellation)
  • line-based aesthetic (like pen and ink)
  • impermanent display (disappearing ink? CRT phosphor persistence)
• Other factors
  • reusing/repurposing obsolescent hardware
  • reimplementing historic devices (Rutt-Etra, Scanimate)
  • not using the mainstream approach (e.g. Adobe CS, Resolume)
• Hypothesis – vector displays became obsolescent when computers could not fully exploit them. Now they can!
/* here the font abruptly switches from an emulation of the original 1967 Hershey vector font (still used by plotters and engravers) */

@~%

/* to Calibri Light – TrueType fonts use Bézier curves as well as lines */

&@~%
Why use audio tools for vector synthesis?

- Can use the same tools to generate audio and video
- Same demand for complex, timed control of multiple parameters
- Many vector synthesis algorithms/effects correspond directly to common audio processes

<table>
<thead>
<tr>
<th>Vectors</th>
<th>Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draw a circle</td>
<td>Generate cosine and sine waves</td>
</tr>
<tr>
<td>Turn to ellipse</td>
<td>Change gain</td>
</tr>
<tr>
<td>Randomise position</td>
<td>Add random DC offsets</td>
</tr>
<tr>
<td>Repeated linear growth in size</td>
<td>Apply upward sawtooth LFO to gain</td>
</tr>
<tr>
<td>Repeated linear brightness fade</td>
<td>Apply downward sawtooth LFO to third channel</td>
</tr>
<tr>
<td>Show several growing fading ellipses</td>
<td>Multiplex between several such sources</td>
</tr>
</tbody>
</table>
## Display devices

<table>
<thead>
<tr>
<th></th>
<th>analogue o'scope</th>
<th>arcade monitor</th>
<th>Vectrex console</th>
<th>CRT TV</th>
<th>o'scope emulator</th>
<th>stroke-to-raster</th>
<th>laser projector</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cost</strong></td>
<td>$</td>
<td>$$$</td>
<td>$</td>
<td>$</td>
<td>free</td>
<td>$$$</td>
<td>$$$$$</td>
</tr>
<tr>
<td><strong>availability</strong></td>
<td>very common</td>
<td>rare</td>
<td>occasional</td>
<td>very common</td>
<td>free</td>
<td>very rare</td>
<td>easy</td>
</tr>
<tr>
<td><strong>modding</strong></td>
<td>-</td>
<td>possible</td>
<td>easy</td>
<td>hard, more $</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>display size</strong></td>
<td>5” graticule</td>
<td>20”</td>
<td>9”</td>
<td>varies</td>
<td>-</td>
<td>-</td>
<td>huge, coloured</td>
</tr>
<tr>
<td><strong>portability</strong></td>
<td>✓</td>
<td>***</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Vectrex games console

- Monochrome CRT made by Milton Bradley 1982-1984
- Modification (Duff) adds external inputs
- Output captured with HD camera
- “Spot-killer” circuit cuts the beam when the vertical signal has low amplitude or low speed
- This can be defeated by multiplexing an invisible high-frequency signal or the “Holzer-Konopaska” mod
Vector synthesis in hardware

- Often modular analogue synths are used
- Pros
  - Flexible/reconfigurable signal path
  - Lots of physical controls
- Cons
  - Can’t save patches
  - Controls hidden by ‘Kabelsalat’
Software

• **Pros**
  • Effects not possible in hardware
  • Ability to save setup
  • Allows non-real-time rendering

• **Cons**
  • Fewer physical controls
  • More crashes

ReWereHere (Max/MSP)
Hardware or software?

No decision needs to be made!

Both are powerful, and both have pros and cons.
Nothing stops us using hardware AND software!

Derek Holzer's performance setup, Bath Spa, March 2018
Like other audiovisual art, vector graphics uses different degrees of crossmodality

• Visuals accompany audio but are independent or
• Events are both audio and visual, but not directly related or
• Visuals depict audio (e.g. waveform/spectrum display) or
• Visuals are identical to audio (e.g. Jerobeam Fenderson ‘Oscilloscope Music’)
Possible software for performance/synthesis

• Requirements – multichannel, real-time
• Preferences – free, open-source
• Graphical (dataflow) programming
  • **Purr Data** (based on Pure Data extended)
  • Max/MSP
  • VCV Rack – modular synth emulation
  • TouchDesigner – designed for video
• Text-based programming
  • **ChucK** – quick audio prototyping
  • Processing – designed for graphics
  • Faust – low-level DSP
Text generation in Purr Data

• First method - crude results and patch was unwieldy
  • Ringing is clearly seen, and blanking had not been implemented
  • Demonstrated pitch control

• Second method - converted nine fonts from ILDA to WAV using LaserBoy software, then played as samples
  • (has multiplexing artefacts)

• Looks better on Vectrex, with pitch mapped to height
  • (still has multiplexing artefacts)

• One alternative is XYScope for Processing (Ted Davis)
Video vectorisation – *not* real-time

- **Aim** – convert raster video to vectors (using free software)
- **Process**
  - Downsample video to 500*400@16
  - Detect edges in each colour plane, convert to monochrome, threshold
  - Trace edges to DXF vectors (PoTrace)
  - Convert DXF files to WAV (LaserBoy)
  - Remove half of each loop (C)
  - Optimise frames to reduce vertex count (LaserBoy)
  - Extend to 6000 vertices per frame (C) (96000 Hz / 16 fps)
- **Tested with** game footage, music notation and football
- **Results** – principle works, but *lots* of optimisation needed
  - For notation, a better approach might use InScore

*Real-time vectorisation can be done in XYScope!*
Synthesis algorithms

- FM/AM
- Waveshaping
- Sampled vector playback / granular synthesis
- Simulation of chaotic systems
- Rutt-Etra scan processing
  - Rewerehere (Max), Vector synthesis library (Pure Data, but not in Purr Data)
- Audio visualisation
Audio visualisation

• Intended for VJ-style scenarios
• Usually benefits from compression and low-pass and high-pass filters
• Waveform display (like oscilloscope)
• Audio vectorscope (X=L+R, Y=L–R), usually used to show stereo spread
• Spectral display
• Display of “analytic” signal using the “Hilbert transform”
  – The analytic signal is a complex signal with no negative frequencies, e.g. $\cos(\omega t) \rightarrow e^{i\omega t}$
  – The Hilbert transform is non-causal so cannot be used in real time
  – The “fake” Hilbert transform is causal and generates an approximation to $e^{i(\omega t + \theta(\omega))}$
• Indirect methods – e.g. pitch tracking
Performance patch design

• Model on conventional signal flows

• SOURCES – PM/AM/simple Lissajous, 2D/3D shapes, Trochoids (“Spirograph”), Audio vectorscope, Analytic signal, Audio visualisation

• EFFECTS – bitcrusher, square↔circle, tile/mirror, strobe, rotate/translate, dash, blank edges/centre, depth spin, lens distortion, etc.
Production methodology with Purr Data

• Recording of performance
  • Audio from second computer - Eggboy album ‘thirteenpointeight’ and Kuba ‘Animalia’
  • PD records the output vectors and audio
  • Vectrex and PD audio output captured by HD camera (Canon XA10)
  • Synchronisation of recorded video with original audio using Audacity/FFMPEG
  • Can replay vectors in order to record video

• Appraisal
  • Purr Data allows a graphical interface but programming is less straightforward
  • Would benefit from external (i.e. MIDI) controller and/or keyboard controls
  • Audio clicks due to refreshing GUI
ChucK experiments

- Text-based language, closest to C
- Developed by Ge Wang (Princeton)
- Active, but sporadic, development
- MiniAudicle IDE not used (doesn’t support ASIO)
  - ATOM editor template available

Experiments
- Spectral display
- Synthesis ToolKit (“Monsters”)
- Chaotic oscillators
- Joystick control
Synthesis ToolKit (Monsters)

• CCRMA Synthesis ToolKit (Stanford)
  • Audio synthesis library implemented in ChucK (and other languages)
• Patch randomly picks X and Y instruments with random parameters
  • Either different instruments on X and Y, or
  • Same instrument on X and Y, but different parameters, or
  • Same instrument on X and Y, with same parameters, but Y delayed by ¼ cycle
• Range of random parameters adjusted (many are inaudible/invisible)
Chaotic oscillators

• Sample-rate integration is tricky in PD

• Inspired by analogue computers
  • Analog Paradigm (German manufacturer)
  • Analog Ontology (home-built)

• Fourteen chaotic oscillators made in ChucK
Graphical or text-based?

Purr Data
- one of several variants of Pure Data
- very flexible GUI
- unconventional programming paradigm
- good for patching, poor for sequencing
- better for real-time use

ChucK
- lack of GUI (on PC)
- more flexible
- sequencing is easier
- less actively developed, small user base
- better for offline rendering

Both can have audio glitches when the graphics are updated.

Use both? PD/ChucK can communicate via MIDI/OSC.
Other software to consider?

- OsciStudio (talk on Thursday)
- Processing plus XYScope library (talk on Thursday)
- Max/MSP plus ReWereHere patch (talk on Friday)
- Axoloti Patcher (talk on Saturday)

- VCV Rack
- TouchDesigner
- Faust
- Other audio programming environments e.g. Csound, Supercollider
- Other audio tools e.g. Ableton Live
- Non-audio programming environments e.g. Matlab, Octave
VCV Rack

- Open-source virtual modular synth (vcvrack.com)
- Program is free, modules free or paid for
- Powerful but processor-intensive
- Wastes screen space?
TouchDesigner

- Patching environment for video (derivative.ca)
- Free for non-commercial use
- Graphical, but can also use Python
Faust

- Functional programming language for DSP, (see faust.grame.fr)
- Very terse code, which can be:
  - Compiled into C/C++
  - Run in PD, ChucK, SuperCollider etc.
  - Made into an external (Max/MSP, PD, SuperCollider, Csound) or a VST plug-in
- Includes audio libraries, physical models and GUI

```plaintext
// Simple Organ
import("stdfaust.lib");

midigate = button ("gate");              // MIDI keyon-keyoff
midifreq = hslider("freq[unit:Hz]", 440, 20, 20000, 1);  // MIDI keyon key
midigain = hslider("gain", 0.5, 0, 10, 0.01);          // MIDI keyon velocity

process = voice(midigate, midigain, midifreq) * hslider("volume", 0, 0, 1, 0.01);

// Implementation

phasor(f) = f/ma.SR : (+,1.0:fmod) ~ _ ;
osc(f) = phasor(f) * 6.28318530718 : sin;

timbre(freq)= osc(freq) + 0.5*osc(2.0*freq) + 0.25*osc(3.0*freq);

envelop(gate, gain) = gate * gain : smooth(0.9995)
            with { smooth(c) = * (1-c) : + ~ * (c) ; } ;

voice(gate, gain, freq) = envelop(gate, gain) * timbre(freq);
```
<table>
<thead>
<tr>
<th>L</th>
<th>cost</th>
<th>Win</th>
<th>Mac</th>
<th>Linux</th>
<th>designed for</th>
<th>interface</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>audio</td>
<td>graphical</td>
<td>several distributions</td>
</tr>
<tr>
<td>VS library for PD</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>vectors</td>
<td>graphical</td>
<td></td>
</tr>
<tr>
<td>Max/MSP</td>
<td>$100/year</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>audio</td>
<td>graphical</td>
<td></td>
</tr>
<tr>
<td>ReWereHere for Max/MSP</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>vectors</td>
<td>graphical</td>
<td></td>
</tr>
<tr>
<td>Axoloti Patcher</td>
<td>free</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>vectors</td>
<td>graphical</td>
<td>no brightness control, needs Axoloti hardware</td>
</tr>
<tr>
<td>TouchDesigner</td>
<td>free (non-commercial)</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>video</td>
<td>graphical</td>
<td></td>
</tr>
<tr>
<td>ChucK</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>audio</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>Processing</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>graphics</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>XYscope for Processing</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>vectors</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>Faust</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>audio</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>VCV Rack</td>
<td>free (some paid)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>audio</td>
<td>GUI</td>
<td></td>
</tr>
<tr>
<td>Oscistudio</td>
<td>€34</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>vectors</td>
<td>GUI</td>
<td></td>
</tr>
<tr>
<td>LaserBoy</td>
<td>free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>lasers</td>
<td>text</td>
<td>unusual interface</td>
</tr>
</tbody>
</table>
Remaining issues

- Results very dependent on characteristics of display device
  - Bandwidth
  - CRT persistence
  - Spot killer (on Vectrex)
  - Graticule (on oscilloscope)

- Capturing the display raises issues
  - Frame rate
  - Blocking outside light
  - Colour balancing
  - Screen curvature
Software still needs hardware

- The programs are mostly mouse-controlled, but MIDI controllers are very useful
- Audio interface with at least 3 channels (X, Y, brightness), DC coupling, and ideally a high sample rate

- HD camera / SD camera / Smartphone
- Raster display device
Conclusions

- Audio tools are an efficient way to generate vector graphics
- The display device imposes restrictions on the methodologies
- Both PD and ChucK are suitable software platforms, each with clear pros/cons
  - choice depends on the application scenario
  - both can be used simultaneously (e.g. synthesis in ChucK, GUI in PD)
  - both benefit from physical controllers
- Real-time performance is easier in PD
  - But graphical programming is awkward for complex tasks
- Non-real-time rendering is easier in ChucK
  - Lack of GUI is the biggest disadvantage
- Other software deserves further evaluation
References and resources

- Rewerehere – https://www.facebook.com/groups/REWEREHERE
- Video Circuits – https://www.facebook.com/groups/VIDEOCIRCUITS
- DC-coupled audio interfaces – http://www.expert-sleepers.co.uk/siwacompatibility.html
- Oscilloscope emulator – https://github.com/kritzikratzi/Oscilloscope
- Pure Data – https://puredata.info
- ChucK – http://chuck.cs.princeton.edu
- LaserBoy – http://laserboy.org
- PoTrace – http://potrace.sourceforge.net