Developing a Microtonal Clarinet via Wind Controller and 3D Printing

Nick Bailey
Science and Music Research Group, University of Glasgow

The International Clarinet Consortium

Alex South
Scottish Clarinet Quartet

Precedents

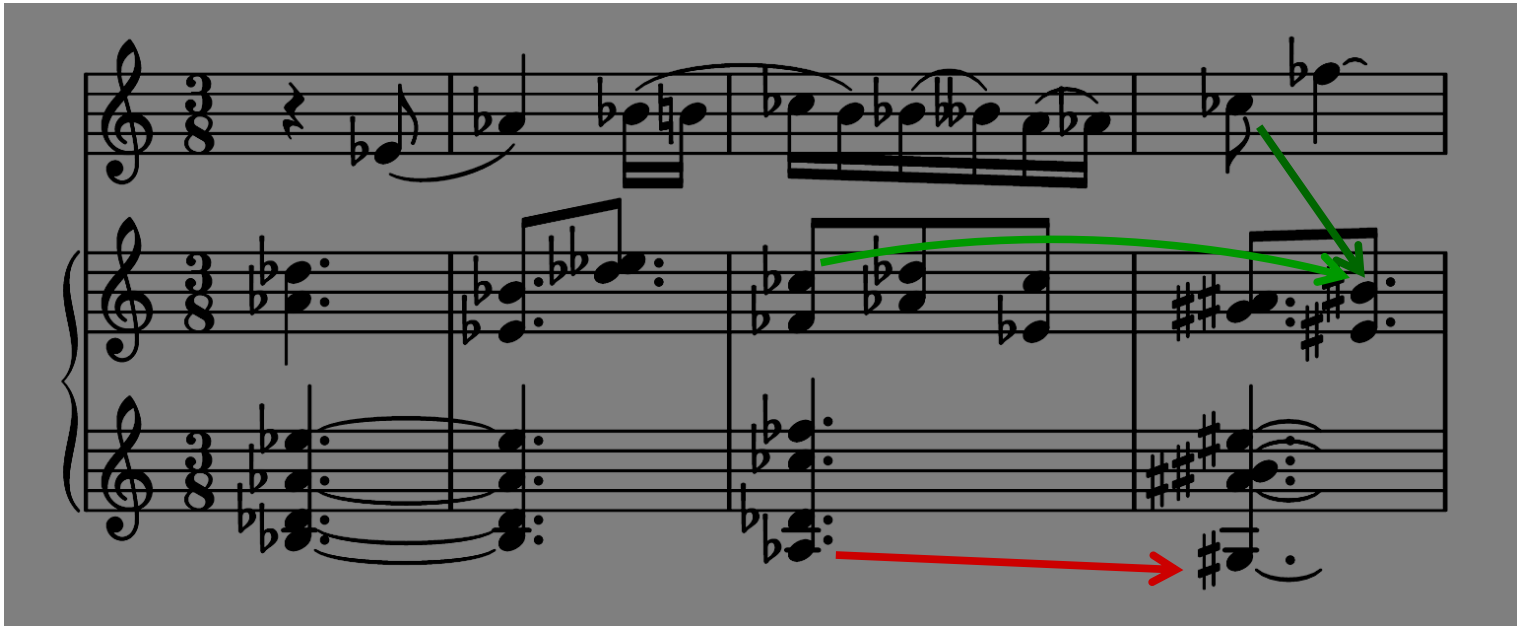
Early harpsichords (before the adoption of well-tempered tuning systems) were sometimes built with “black” keys split front-to back.

This javascript demo compares three different EDO keyboards : a Modern disposition 19-tone keyboard, a standard 12-tone keyboard, and a third 19-tone keyboard with the ancient keyboard layout.

More details at

Why 19-EDO?

Graham Hair:
Vocalise for 19EDO Clarinet/Breath Controller and Digital Ensemble



Neo-Riemannian harmony: chords **share** constant and moving notes across

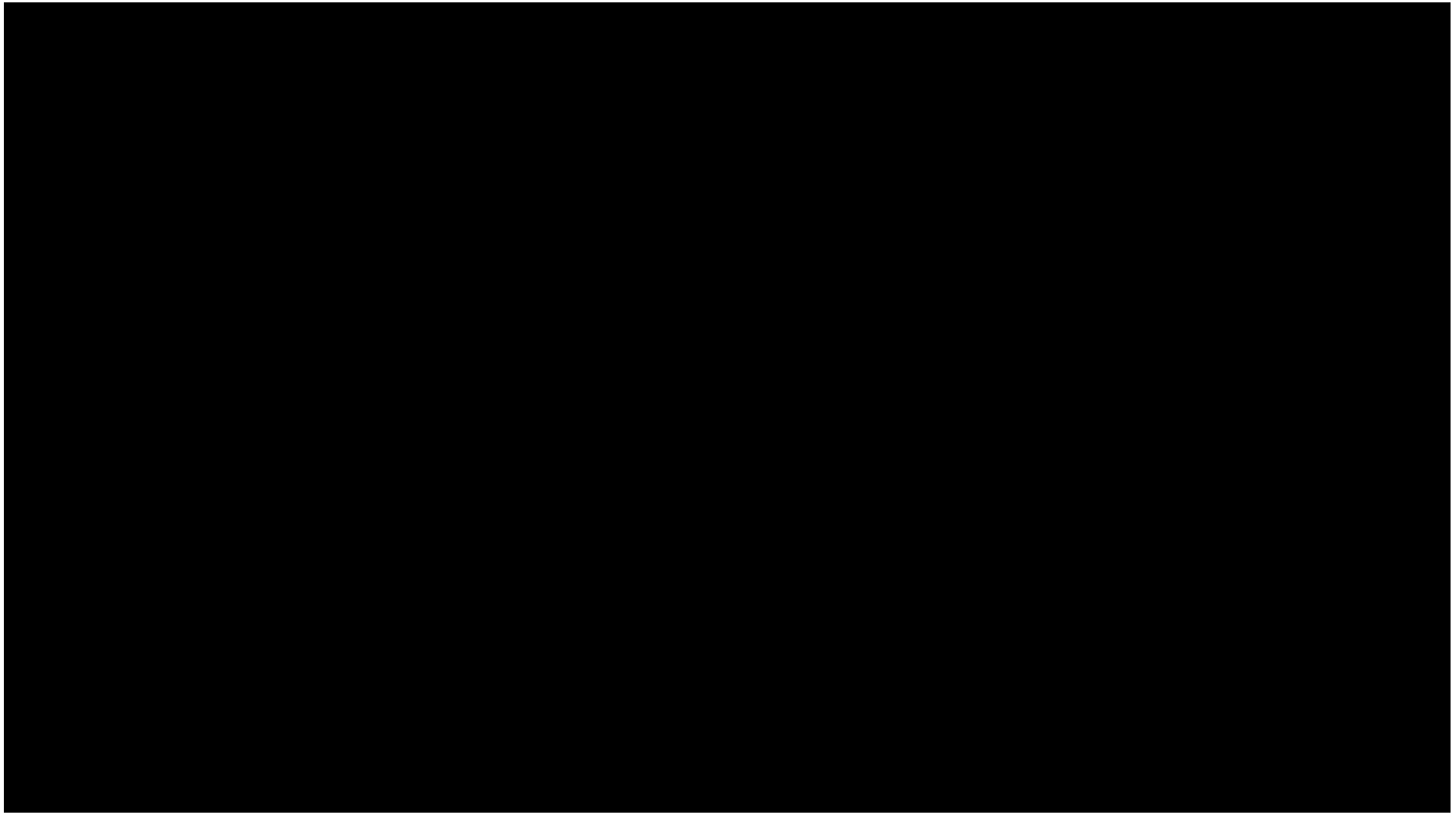
Enharmonically Equivalent in 19-EDO

Hyperchromatic Step in 19-EDO

Specialist Instruments

- Players of “un-fretted” (strings, but also voice, trombone etc) instruments obviously need only modify their technique to produce non-standard intonation.
- Variants of standard instruments have been constructed, for example the 19-tone trumpet with an extra valve to produce a hyperchromatic step.
- The following video is a work originally for two women's voices and organ obbligato with one part performed on the 19-tone trumpet (recorded at Animusic 2014, Braga).

Graham Hair: Turkish Duets.
3. Azrail



Acoustic Instrument with Alternative Fingerings

- Presents a huge challenge to expert performer, who has to play with non-standard fingerings “It was like being 15 years old again” (*Alex South*)
- Attempted by Alex South (SCQ), and Ingrid Pearson (RCM, London; period instrument specialist). Period instruments normally require more exaggerated intonation adjustment.
- The “Ill-tempered clavier”, a scordatura

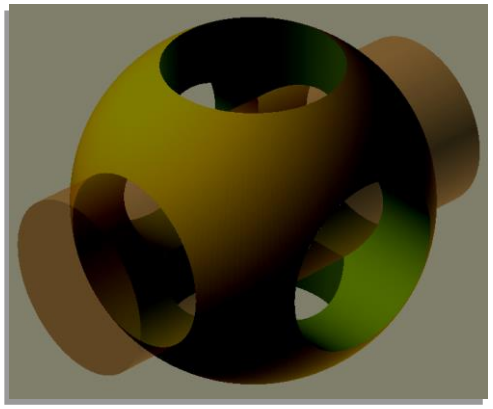
Electroacostic Instrument with Custom Interface

- An alternative approach is to give the performer a familiar interface in some way augmented to permit microtonal inflexion.
- The WX5 breath controller was connected to a patch written in PureData. Additional input was acquired from a two pedals which could raise and lower the standard pitches to achieve the desired microtonal inflexion.
- The synthesis algorithm was that of the STK (Scavone *et al*) which dates back many

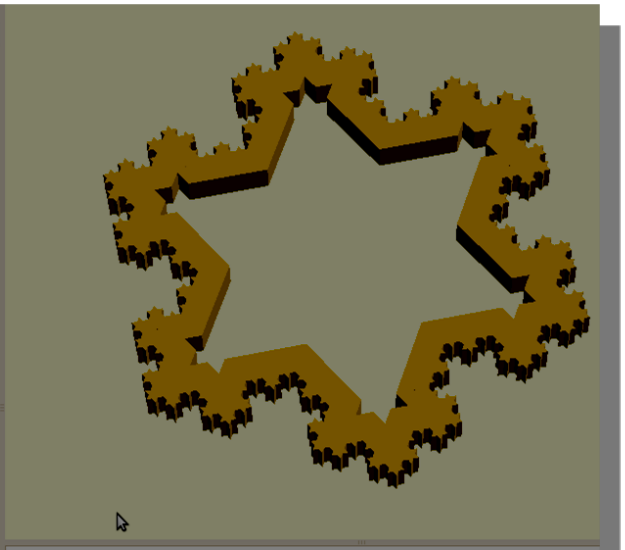
Using the Augmented WX5

Alex South explains some challenges of performing with pedal augmentation.
Studio Recording at Manchester Metropolitian University, UK
with Katrina Nimmo (soprano) and Graham Hair (continuo).

CAD software: OpenSCAD



```
module cubeBeam(p0, theta, beamLength, thickness, height, capAngleA=0, capAngleB=0) {

    // Calculate end point p4
    p4X = p0[0] + beamLength * cos(theta);
    p4Y = p0[1] + beamLength * sin(theta);
    p4 = [p4X, p4Y, 0];


    capOffsetAX = thickness * tan(capAngleA) * cos(theta);
    capOffsetAY = thickness * tan(capAngleA) * sin(theta);
    capOffsetBX = thickness * tan(capAngleB) * cos(theta);
    capOffsetBY = thickness * tan(capAngleB) * sin(theta);


    p1X = p0[0] - thickness * sin(theta) - capOffsetAX;
    p1Y = p0[1] + thickness * cos(theta) - capOffsetAY;
    p1 = [p1X, p1Y, 0];

    p2 = [p1[0], p1[1], height];
    p3 = [p0[0], p0[1], height];


    p5X = p4[0] - thickness * sin(theta) + capOffsetBX;
    p5Y = p4[1] + thickness * cos(theta) + capOffsetBY;
    p5 = [p5X, p5Y, 0];

    p6 = [p5[0], p5[1], height];
    p7 = [p4[0], p4[1], height];


    polyhedron (
        points = [p0, p1, p2, p3, p4, p5, p6, p7],
        triangles = [ [0,1,2],[2,3,0],[4,6,5],[4,7,6],[2,6,3],[6,7,3],[0,5,1],[0,4,5],
                      [0,3,7],[7,4,0],[1,6,2],[5,6,1], ]
    );

}   // END module cubeBeam



// *** PARAMETERS ***
thickness = 2.5;
height = 3;
beamLength = 40;
kochLevels = 4;

// Calculated parameters
segLength = beamLength / (pow(3, kochLevels));
```

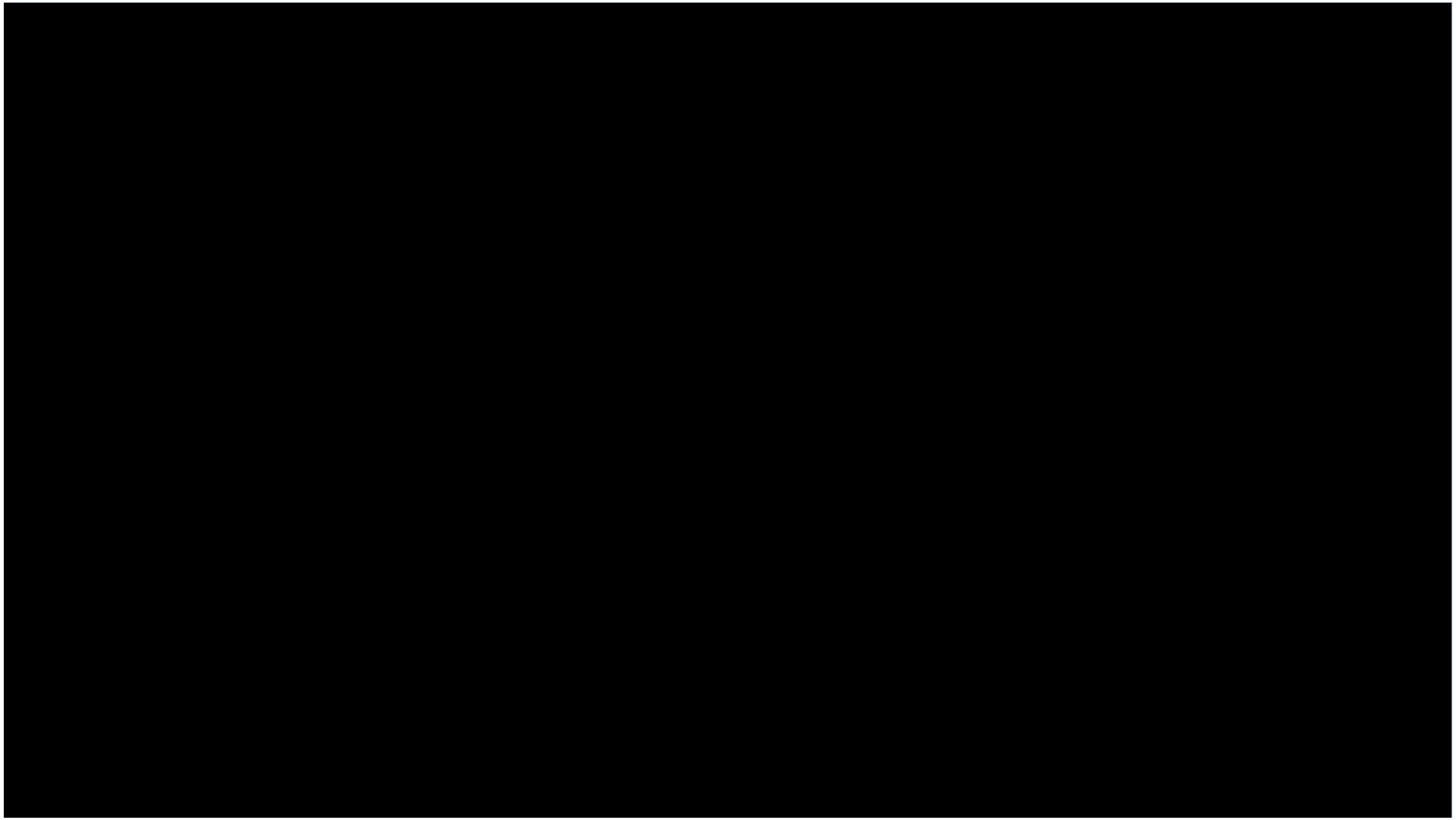


```
Parsing design (AST generation)...
Compiling design (CSG Tree generation)...
Compilation finished.
Compiling design (CSG Products generation)...
Compiling design (CSG Products normalization)...
CSG generation finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

```
Viewport: translate = [ 19.97 3.38 13.33 ], rotate = [ 27.70 0.00 348.30 ], distance = 265.72
```

Using a 3d-markup language such as OpenSCAD makes possible the

Printed Clarinet (top part)



Traveling-wave Acoustical Model

modelled the clarinet as a propagating plane-wave in the 1990s

- A pressure wave travels from the reed to the bell where it is inverted and reflected. The reflected rarefaction interacts with the reed to produce the oscilliatory behaviour.
- Tone holes are modelled as scattering functions
- The model was acoustically satisfactory, but does it produce accurate pitch estimates?

C++ Programing

Original simulator (Robertson & Scavone) refactored in a more object-oriented fashion to permit easier definition of arbitrary instrument geometries

```
LowDWhistle::LowDWhistle()
{
  const double bore {0.011};

  elements = new const Element*[16] {
    new Butterfly,                new Pipe(0.5263, bore),
    new ToneHole(0.005, bore),    new Pipe(0.08141, bore),
    new ToneHole(0.005, bore),    new Pipe(0.07280, bore),
    new ToneHole(0.005, bore),    new Pipe(0.04248, bore),
    new ToneHole(0.003, bore),    new Pipe(0.07464, bore),
    new ToneHole(0.0045, bore),   new Pipe(0.1361, bore),
    new ToneHole(0.005, bore),    new Pipe(0.2147, bore),
    new OpenEnd(bore),
    nullptr
  };

  parse_elements();
}
```

Applying Evolutionary Computation to Instrument Specification

- Choose what to optimise
  - Tone hole placement
  - Tone hole size
  - ...to minimise maximum and average deviation from 19-EDO
- Construct instrument from specification and try all possible fingerings
  - 9 holes: 512 fingerings
  - Ramp breath pressure from 0 to “a lot”
  - Find fundamental frequency of resulting tone

DEAP Programming

In this initial study, evolutionary techniques are used to produce a set of tone hole sizes and positions which best match the 19-EDO scale with an arbitrary fingering

```
def evalClarinet(indv):
  command = ['./hotair']
  for i in range(19):
    scale = 0.02 if (i%2) else 0.5
    command += [ str(indv[i]*scale) ]
  try:
    simulation = subprocess.check_output(command).decode('utf8').split('\n')
    pitch_errors = [ float(l.split('\t')[2]) for l in simulation[:-1] ]
    lowest = float(simulation[0].split('\t')[0])
    abs_pitch_errors = numpy.abs(pitch_errors)
  except:
    lowest = 10000.0
    abs_pitch_errors = [10000.0]*19
  mean_error = numpy.sum(abs_pitch_errors)/19.0
  max_error = numpy.max(abs_pitch_errors)

  if isnan(mean_error): mean_error = 10000.0
  if isnan(max_error):  max_error = 10000.0

  return mean_error, max_error, abs(lowest-110.0)
```

Internal representation

- Parameters are normalised in the range [0, 1.0) to keep DEAP happy.
- A 9-hole instrument has a reed, 9 ToneHoles, 10 Pipes and a Termination.
- The ToneHole diameters are proportions of the bore width.
- The Pipe lengths are multiples on 0.5m.

The BLX-α algorithm

C++ program ends up being invoked like this:

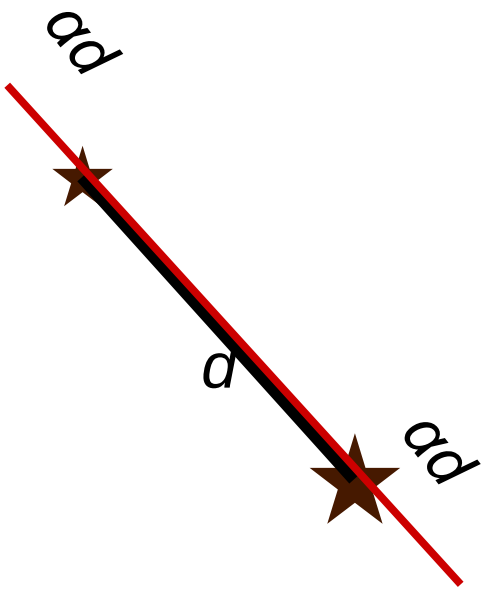
```
./hotair 0.3617473085502436 0.004492622692911909 0.36546847137480726 0.0045551146322185465
0.25133359603790495 0.013490554296424793 0.3853430271667548 0.006161814134620142
0.03459440899334253 0.007133613067089454 0.30866802733869964 0.01750736934150487
0.3468471013558714 0.009253493363501453 0.35033896303336043 0.013991840063922786
0.2815331036362345 0.013620265929286474 0.3939958437763
```

Evolutionary Step:

- Select the two parents
- Their separation is *d*
- Offspring has value
- between them

or up to *αd* outside them

- Mutation: add normally-distrubted random variable to individual



F0	Fingering	Error (cents)
128.5	000000000	0
133.274	010001100	-0.967027
138.226	111011000	20.8373
143.362	010010100	4.07636
148.689	001001100	1.29582
154.213	000000110	-0.148187
159.943	010011100	-3.71662
165.886	110010100	-0.894486
172.049	101100001	8.54025
178.442	100000110	3.473
185.072	010100001	1.19994
191.948	001010001	-0.43322
199.08	101001110	1.91328
206.477	100001011	-1.48208
214.148	110000000	-0.390803
222.105	001101110	3.07595
230.357	011000010	-0.431054
238.916	100101011	-5.19815
247.793	101111011	-0.651494

Did it work?

- Yes!
- Well, mostly.
- Fingerings are so arbitrary that they would have to be applied electromechanically for a real performance.
- 3D-printed keys were not air-tight and wore out very quickly.
- The model does not take overblown tuning into account, which is a major factor in the design of a modern instrument (and

What next?

- Construction of proof-of-concept instrument successful — you can print useful instruments.
- Need more sophisticated model including bore contour to get overblown twelfth in tune.
- Evolutionary Algorithm should take fingering considerations into account.
- ARTool! http://artool.sourceforge.net/
- Keywork on a “final” instument should probably be built by an instrument maker.

Pedro Rubio, Clarinettist

Bassus Ediciones, Madrid