Functional Geometry and the *Traité de Lutherie*

[Functional Pearl]

Harry Mairson

Brandeis University

mairson@brandeis.edu

Abstract

We describe a functional programming approach to the design of outlines of eighteenth-century string instruments. The approach is based on the research described in François Denis's book, *Traité de lutherie*. The programming vernacular for Denis's instructions, which we call *functional geometry*, is meant to reiterate the historically justified language and techniques of this musical instrument design. The programming metaphor is entirely Euclidean, involving straightedge and compass constructions, with few (if any) numbers, and no Cartesian equations or grid. As such, it is also an interesting approach to teaching programming and mathematics without numerical calculation or equational reasoning.

The advantage of this language-based, functional approach to *lutherie* is founded in the abstract characterization of common patterns in instrument design. These patterns include not only the abstraction of common straightedge and compass constructions, but of higher-order conceptualization of the instrument design process. We also discuss the role of arithmetic, geometric, harmonic, and subharmonic proportions, and the use of their rational approximants.

Categories and Subject Descriptors D.3.3 [*Programming Languages*]: Language Constructs and Features—Control structures

General Terms Languages

Keywords Euclidean geometry, ruler and compass constructions, *vesica piscis*, proportion and measurement, Pythagorean approximation, chisels, hand planes, animal glue, maple, spruce, ebony, varnish.

1. Introduction

Programming languages don't exist simply to tell machines what to do. Well recognized as the engineering vernacular of software, they are more importantly the collective mother tongue of algorithmic ideas.

This point of view is a virtual credo among computer scientists. That credo has only recently been reiterated in public emphases on "computational thinking" as a way of understanding complex, constructional processes. In other words, programming languages provide a way for us to describe to each other what we know how to do.

We immediately assume, quite naturally, that modern technologies are the ones where this computational thinking is most appropriate and relevant. But people have been building things for a long time. In what way can computational thinking be an intellectual organizing principle for understanding and describing the past, and making sense of the kinds of expertise that flourished and came to maturity?

To that end, we've designed a programming language ("language" may be overblown—in any case, a set of graphics primitives, and more significant, a programming idiom) for realizing outlines of string instruments. The goal of this approach is to mimic accurately the accepted historical traditions for this design, while automating and facilitating the design and the construction process.

1.1 Historical background

Modern *luthiers* (string instrument makers) inherit a tradition embodied in the famous instruments of even more famous historical colleagues: renowned makers include, among others, Andrea Amati—recognized as the father of the violin—and distinguished members of his family, Andrea Guarneri and his family (especially son Giuseppe, and Giuseppe "del Gesù"), Matteo Gofriller, Dominico Montagnana, and the ubiquitous *primus inter pares*, Antonio Stradivari. It is typical for modern makers to copy and modify the canonical designs of these makers. But where did their designs come from? Surely not from an infinite chain of copyists—a non-well founded induction that violates the well-ordering principle. What are the underlying geometric, computational ideas from which these designs originate?

It is misleading to think that these instruments evolved simply from an amorphous, creative, artistic sense. The crude design of string instruments is a fairly inevitable consequence of ergonomics: a resonating box, with rounded corners to avoid bumping and blocking, and a concave middle 'waist' to allow bow access to the strings. But the further refinement of this basic form came from an interaction both musicians and their performance needs, and with the scientific perspective that was emerging simultaneously. Changing peformance requirements (for example, the evolution of the violoncello from a continuo to a virtuoso, solo instrument) caused changes in design. In addition, these string instruments took form during an age of scientific revolution—the era of Copernicus, Kepler, Galileo, Newton—where the experimental mindset, with its associated scientific insights, likely had an impact on instrument designers and makers (Schleske 2004).

In 2006, a French luthier named François Denis published *Traité de lutherie* (Denis 2006), a comprehensive and seminal treatise which attempted to lay out the manner in which string instrument outlines were constructed. There are few, if any, written works on construction technique written at the time these instruments

ICFP '13, September 25–27, 2013, Boston, MA, USA.

http://dx.doi.org/10.1145/2500365.2500617

emerged on the historical, musical scene: Denis' contribution was a historically inspired and *a priori* reconstruction of what some of those techniques likely were. The most striking thing about Denis's book is that the construction methods are entirely Euclidean, when we are used to thinking in a Cartesian way about most everything. The designs are realized without graph paper, without Vernier calipers, protractors, or just about anything having to do with measurement.[1] Instead, virtually everything is done with an (unmarked) ruler, and a compass, save one fixed dimension. That dimension determines, via entirely proportional constructions, all the other ones.

It's worth saying, without reverting to a commutative diagram, that a great majority of our reasoning about a great deal of subjects is Cartesian. We typically learn Euclidean geometry to introduce the notion of *proof*, but by the time we want to *calculate* anything, we've usually moved on to Cartesian, analytic geometry. In other words, a circle isn't a curve: it's an equation. And the same goes for physics, for economics. Even philosophy—think of the "word problems" solved by translation into propositional logic—the utterly Cartesian construction of *analytic* philosophy, which gave rise as well to computer science.

The big Cartesian idea is this: to solve your problem, turn it into equations. Then solve your equations instead (whence the commutative diagram?), and when you're done, translate your solution back to your problem domain. Instead, the Euclidean solution—and that of Denis for instrument design—is this: it's a geometric problem. So you solve it in the geometric world with geometric constructs.

It deserves mentioning that all string instruments are not alike. Of course, there are tonal variations, as a consequence of wood quality, workmanship, and aspects of design. But it should be emphasized that every maker had a signature—not the label inside the instrument as much as the shape of scrolls, pegboxes, necks, corners, purfling, ff-holes—and instrument outline. Professional appraisers can identify the makers of famous instruments by these virtual fingerprints, just as program committees of programming language conferences could identify the authors of conference submissions even under double-blind conditions.

Almost everything we know about how to do this—still considered as a refined, even mysterious, knowledge—was understood virtually a half-millenium ago.[2] But how, and in what way?

1.2 Some comments on Denis's *Traité*

Learning how to carry out Denis's constructions with pencil and paper leads to two immediate conclusions. One is that the descriptions of the constructions are, in effect, a kind of do-this-then-do-that informal straight-line code. They would benefit by a restatement as a functional program, where notions of hierarchy and abstraction have the potential to make greater sense out of many details. A well-structured description can illuminate the big picture of the forest, and the detail of the trees, with no loss of implementation focus at either end.

A second conclusion is that the learning process itself is facilitated by such a development. Not only does this process require the understanding of the big picture and the little details, the inevitable errors encountered in learning how to do this can be made faster, and can be recovered from faster and easier. A mechanical graphics engine attached to this functional description ensures accuracy, rectilinearity, without the accumulated error of physical drafting methods (though similar issues of numerical analysis may arise and need careful treatment).[3] That speed and accuracy is of no small consequence if the analytic methods are to be used also as an experimental, investigative mechanism.

Briefly, a programming characterization of these outlines would serve the following additional goals. First, we can focus on the putative *design process* (program) itself, in an appropriate vernacular, rather than its output. Second, designs of famous instruments could be made public, where they could be downloaded and modified by modern makers. (Their modifications, of proprietary interest, would most likely not be public.) Third, the ease of producing instrument outlines could be a construction prosthetic, aiding the building of molds, around which rib assemblies are constructed. Finally and most interestingly, these programs might be the foundation of a kind of computational art history, where evolution of instruments could be studied more carefully by the more informed evolution of designs. If nothing else, a result would be an effectively automated version of Denis's book.

We are no longer building catapults or pillars or even film cameras. But we are still making violins—a Renaissance technology that was worked out a half-millenium ago. Making them is surely an art, but in every artistic practice, there is method. Scrolls and archings and outlines and all the details of lutherie are not simply a consequence of artistic temperament. There is a method for everything.

The point of this undertaking is to make some of that method explicit in a contemporary vernacular, while showing the greatest respect for the underlying method itself. In other words, don't change the method as much as find a better way of describing it. In preserving this Euclidean perspective, the goal is to take all of the Cartesian, equational calculation—inevitable in programming—and push it all to the effectively invisible back end, so that the interface used is just the Euclidean part. Building a WYSIWYG graphics engine is a canonical entry point to learning about graphics programming, but our real interest here is as much the *language* as the output. All of a sudden, the seemingly static things you learned in high-school geometry become a dynamic programming language.

A virtue of the language is its record of the construction—the opaqueness of WYSIWYG, resolved by a markup language, is that the latter allows you to *see what you did* to get what you got. The challenge is to find appropriate abstraction primitives. Recursion is noticably absent—it doesn't seem particularly relevant. But that doesn't mean that programming is pointless. Recall the principle underlined in (Abelson and Sussman 1996) that to understand a programming language is to identify its primitive operations, means of combination, and means of abstraction. They are all apparent here.

2. Geometric prelude: point, line, circle, and the pentagon

Rather than describe all the bits and pieces of our primitive graphics language, which is just some fairly straightforward graphics prim-

[1] This should be contrasted with the exegesis in Edward Heron-Allen's famous *Violin Making, as it was and is* (Heron-Allen 1885), which was for many years virtually the only go-to book on instrument making, where the design methods are founded on a rectilinearly-based treatise of Antonio Bagatella, *Regole per la costruzione de violini, viole, violoncelli e violoni* (Bagatella 1782).

[2] Simone Sacconi titled his famous monograph *The 'Secrets' of Stradivari* to emphasize that there were no secrets, but rather refined technique (Sacconi 1979). But the cause and effect reflected in that technique is not really analytically understood—instead, luthiers work within a highly functional and successful "envelope" and venture out, experimentally, in search of further optimization.

[3] "Most of all I would like to advise the maker that the greatest care be taken in using the compass to adapt and calculate the measurements because the slightest movement of the position of the hand is capable of causing an error." (Antonio Bagatella, *Regole per la costruzione dei violini-viole, violoncelli e violoni*, Padova, 1782.)

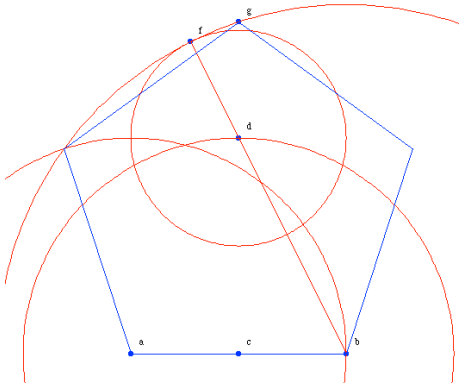

Figure 1. *Constructing a pentagon*

itives on top of Scheme[4], we'll just get on with it. Our demonstration toy problem is how to draw a pentagon with a straightedge and compass.

```
(define (pentagon a s)
  (let ((b (label "b" (xshift a s))))
    (let ((c (label "c"
               (intersect
                 (apply line
                        (intersect (circle a s)
                                   (circle b s)))
                        (line a b)))))
      (let ((d (label "d"
                 (top (intersect
                        (vertical c)
                        (circle c s))))))
        (let ((f (label "f" (top
                   (intersect (line b d)
                              (circle d (distance a c)))))))
          (let ((g (label "g"
                     (top (intersect
                            (vertical c)
                            (circle b (distance b f)))))))
            (let ((p (left (intersect (circle g s)
                                      (circle a s))))
                  (q (right (intersect (circle g s)
                                       (circle b s)))))
              (list (label "a" a) b c d f g p q
                    (makeseg b f)
                    (circle a s) (circle c s)
                    (circle d (distance d f))
                    (circle b (distance b f))
                    (segments (list g p a b q g)) ))))))))
```

Some brief comments on the construction:

`(pentagon` a s`)` Pentagon at point a with base length s.

`(xshift` p d`)` Point p shifted d in the x-direction.

`(line` a b`)` Line through points a, b.

`(horizontal` p`), (vertical` p`)` Horizontal, vertical lines through point p.

`(circle` p r`)` Circle with center at p and radius r.

`(intersect` o_1 o_2`)` Intersection of circles, lines, with possible multiple solutions.[5]

`(top` S`), (bottom` S`), (left` S`), (right` S`)` Choosing among multiple solutions to an intersection.

`(north` C`), (south` C`),` **etc.** Quadrants of a circle.

`(label` ℓ p`)` Mark point p on the plane with the label ℓ, returning value p.

The overall idea is to specify, then draw. Recall that given a unit length, with a straightedge and compass, the constructable point-to-point distances are those points closed over the unit length and $+, -, \times, \div, \sqrt{}$—i.e., "over radicals".[6] (As a consequence, a straightedge and compass serve as a primitive analog computer.) A pentagon is constructible because with a unit edge, its diameter (distance b-f in Figure 1) is the golden mean $\phi = \frac{1}{2}(1 + \sqrt{5})$.

With this example in mind, there's an overwhelming temptation to start programming up every geometric construction you can find—and there are lots of them, of considerable subtlety (Sutton 2009). We mention, for example, Malfatti's problem (Martin 1998): given a triangle, inscribe three circles that are tangent to each other and to the sides of the circle. This problem wasn't solved until the late nineteenth century (see Steiner's synthetic, geometric construction (Steiner)), though draftsman may well have known the solution long before (Denis, personal communication).

3. Musical prelude: the lute of Arnault of Zwolle

Roughly comparable in detail of the pentagon is the lute design of Henri Arnault of Zwolle, from a manuscript dating from 1440 (Harwood 1960; Zwolle 1440). Arnault was a cleric in the court of French king Louis IX, a polymath prototype of Leonardo da Vinci, only who designed musical instruments instead of instruments of war. His manuscript of 1440 describes the construction of a lute. This lute is the "toy problem" of string instrument metrology: what goes for the lute, goes for the violin and violoncello, only more so.

```
(define (lute rad)
  (let ((c (circle origin rad))
        (d (* 2 rad)))
    (let ((p (label "p" (west c)))
          (q (label "q" (north c)))
          (qprime (label "q'" (south c))))
      (let ((r (label "r"
                 (left (intersect (circle (mirror p) d)
                                  (line (mirror p) q))))))
        (let ((s (label "s" (yshift q (distance q r)))))
          (let ((rose (circle (point 0 (/ rad 2))
                              (* (/ 3 10) rad))))
            (list
              (map (lambda (t)
                     (list (label "" (pointfrom origin q t))
                           (label "" (pointfrom origin
                                                (transpose q) t))))
                   (list 0 .2 .4 .6 .8 1.0))
              (circle origin rad) (circle q (distance q r))
              ;draw the rose:
              (makearc (center rose) (north rose) (west rose))
              (makearc (center rose) (west rose) (south rose))
              (makearc origin p qprime)
              (makearc (mirror p) p r)
              (makearc q r s)))))))))
```

Note that `(mirror` p`)` produces a point reflected around the y-axis, so arc drawing is only (explicitly) on the left-hand side of the

[4] This situation is deliberately meant to evoke memory of Peter Henderson's *functional geometry* from the 1980s (Abelson and Sussman 1996; Henderson 2002).

[5] To underline the Cartesian and its departure from the Euclidean: reflect on the intersection of two circles, both with equations having x^2 and y^2 terms with coefficient 1. When you subtract one equation from the other, you're left with a linear term—of what? The line that is the bisector of the *vesica piscis*, the "fish bladder" marking their intersection—see, e.g. (Wikipedia).

[6] See (Stewart 1973), pp. 57-67.

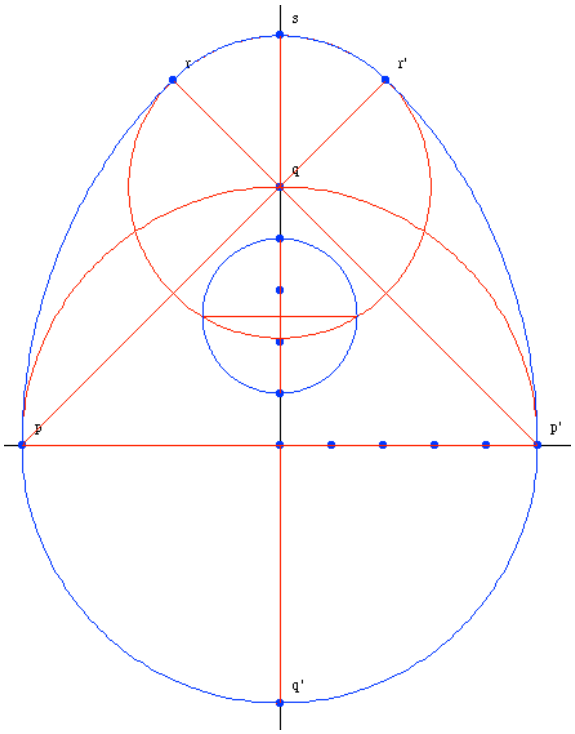

Figure 2. *Lute by Arnault of Zwolle (from a manuscript from 1440), with an approximate soundhole placement*

plane, and mirrored on the right. It's an obvious symmetry that we should take advantage of.

4. Rational approximations

We can make a virtually indistinguishable pentagon by dispensing with the compass almost entirely (or at least, use a "rusty compass" that cannot change dimension), as long as we can draw perpindicular lines. Then, common carpenter's tricks let us code the pentagon with rational, Pythagorean approximants. We use $\frac{14}{9}$ to approximate the height $\sqrt{\phi^2 - \frac{1}{4}}$, $\frac{3}{10}$ for the additional width (beyond the base) $\frac{\phi-1}{2}$, and $\frac{20}{21}$ for the height $\sqrt{1 - \left(\frac{\phi-1}{2}\right)^2}$ of the leftmost and rightmost corners:

```
(define (approximate-pentagon a s)
  (let ((b (label "b" (xshift a s))))
    (let ((t (yshift (midpoint a b) (* s (/ 14 9)))))
      (let ((d (yshift (xshift b (* s (/ 3 10)))
                       (* s (/ 20 21))))
            (c (yshift (xshift a (* s (/ -3 10)))
                       (* s (/ 20 21)))))
        (segments (list a b d t c a))))))
```

What works for pentagons works, by extension, for Arnault's lute construction, and describes in greater detail the underlying geometric symmetries of its design. Figure 4 shows a refined analysis of Arnault's lute construction, which is (as Denis points out) not so much the definition of a real lute as much as it is a prototype, an examplar, of a generic recipe for synthesizing lute designs.

This construction method makes the length of the instrument—a critical dimension for tuning and structural stability—a consequence of the design process, rather than a parameter of it. To design the instrument *ab initio* from its length, we need to think a bit more carefully about the geometry of the underlying construction, in particular its use of harmonic and subharmonic sections.

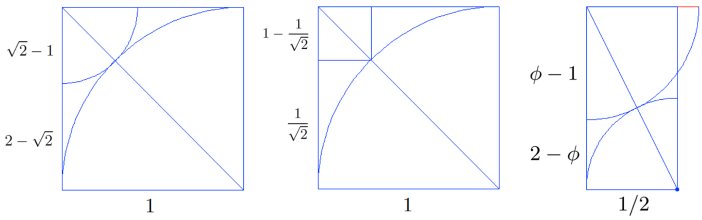

Figure 3. *Harmonic, subharmonic, and geometric sections*

Notice that the length of the lute is $3m + 2s$, where m and s are the radii of circles derived from carefully constructed *geometric sections*. The geometric constructions of these sections are shown in Figures 3 and 4. The highlighted box to the left of the vertical axis of the lute contains a *harmonic section*, and that to the right, a *subharmonic section*. Notice the geometric demonstration that the larger part of a harmonic section equals the smaller part of a subharmonic section that is twice as big.

Now consider the rational approximation of the subharmonic section. Let $s+m = 1$ where s and m are an approximation of this section: then a better approximation is $s' = m$ and $m' = s + 2m$. Solving $\frac{s}{m} = \frac{s'}{m'}$ gives $s = 1 - \frac{1}{\sqrt{2}}$, the smaller part of the section.

Beginning with $s = m = 1$, we get the series of approximants $\frac{1}{3}, \frac{3}{7}, \frac{7}{17}$; luthiers would use initial terms from this series (and those like it) as approximations of its limit. Such constants are mentioned in Arnault's manuscript.[7]

```
(define (approximate-lute s m L)
  (let ((N (+ (* 2 s) (* 3 m))))
    (let ((r (* L (/ (+ m s) N))))
      (let ((q (label "q" (yshift origin (minus r)))))
        (let ((p (label "p" (yshift q L))))
          (let ((cmain (circle (yshift q r) r)))
            (let ((ctop
                   (circle (north cmain)
                           (distance (north cmain) p)))
                  (cright (circle (east cmain) (* r 2))))
 (list
   (apply makeseg    ; bridge
     (intersect cmain
                (horizontal (yshift q (* L (/ s N))))))
   ; sound hole
   (let ((s (yshift p (* (minus L) (/ (+ s (* 3 m)) (* 2 N))))))
     (let ((rs (/ (distance s
                    (left (intersect (horizontal s) cright)))
                  3)))
       (let ((sh (circle s rs)))
         (list sh
               (make-curve q p
                 (list cmain cright ctop)) ))))))))))))
```

(Note that `make-curve` above specifies a curve from `q` to `p`, following the arcs given by circles `cmain`, `cright`, and `ctop`.)

We redraw the lute using this revised code, above, with parameters $s = 3$ and $m = 4$—note $3/7 = .4285\ldots$ is a respectable approximation of $\sqrt{2} - 1 = .4142\ldots$. The length of the lute is now divided into $3m + 2s = 18$ parts, and Arnault's written direc-

[7] Recurrences, familiar to students of discrete mathematics, provide the foundation of these proportional sections. Recall $f(x) = \frac{x+1}{x}$ as defining the geometric section (with initial value $x_0 = \frac{1}{1}$), $g(x) = \frac{x+2}{x+1}$ as defining the harmonic section (also with initial value x_0), etc.

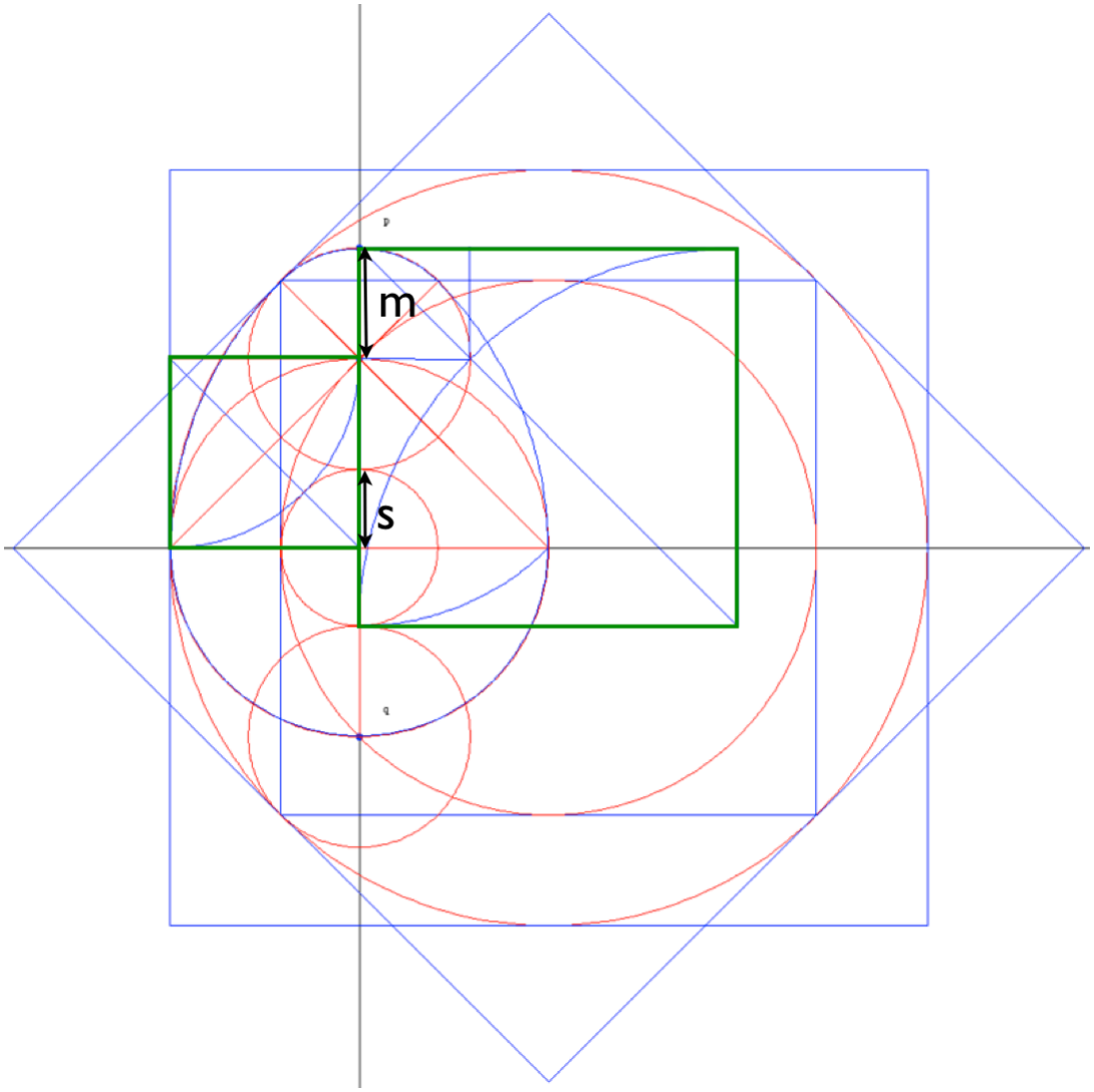

Figure 4. *Geometry of Arnault's lute. The subharmonic section is highlighted on the right, the harmonic section on the left.*

tions that the bridge placement is one-sixth of the instrument length makes rational sense.

Even without this rational approximation, note with $s = \sqrt{2}-1$ and $m = 2-\sqrt{2}$, we have $\frac{s}{2m+3s} = \frac{1}{14}(3\sqrt{2}-2) \approx 0.1602$, realizing Arnault's one-sixth measurement.

Further, the location and size of the soundhole is rationalized through the following argument, a rather striking prestidigitation (see Denis, p. 54 ff.). Its location is said in the manuscript to be halfway from the bridge to the top of the instrument. The length of $3m + 2s$ can be rearranged as $S = m + s$ and $M = 2m + s$. Given the ratio $\frac{s}{m} = \frac{\sqrt{2}-1}{2-\sqrt{2}}$ from the *harmonic* section, we have $\frac{M-S}{S} \approx .5858$ and $\frac{S}{M} \approx .6306$, while the ratio of the *geometric* section is $\frac{2-\phi}{\phi-1} = \phi - 1 \approx .6180$, right between the two.

As a consequence, we can consider the length of the instrument, to sufficient accuracy, as a geometric sequence of overall length $s' + m'$, where the ratio s'/m' can be approximated by Fibonacci numbers of moderate size. Take $s' = 5, m' = 8$ ($m'/s' = 1.625 \approx \phi$). If the bridge is distance s from the bottom of the instrument, then the center of the soundhole is distance $\frac{1}{2}(3m+s)$ from the top of the lute. An exact calculation, following Arnault's specification that the soundhole diameter be one-third of the width, gives a diameter of $(\sqrt{15}-2)/3 \approx .6243$. Using the approximate geometric section (m', s') above gives $s'/m' = .625$. The point of this rational reconstruction is that layout with carefully chosen approximations comes remarkably close to the analytic calculations.

5. Anatomy of an outline: the Amati violin. Geometric problems and their solutions.

With the preamble of the Arnault lute in mind, we can go on to look at the first outline of a violin in the book by Denis: a canonical violin by Andrea Amati. The outline has four essential parts: the framework, which blocks out the regions, and the upper, lower, and middle *bouts*. A key feature of the outline is that it is parameterized by only one fixed dimension: the distance XQ from the top of the outline to the top of the lower bout. All the other dimensions are proportional. The code appears in the Appendix. It needs emphasis that the code is almost a verbatim translation of Denis's construction description.

The do-this-then-do-that of straight line code is not very interesting. But a significant number of low-level details can be abstracted. The curves of all the outlines are made by following sequentially a series of arcs defined by circles, with endpoints, via `(make-curve` p q `(list` C_1 C_2 $\cdots C_k$`))`. There are distances that are partitioned by a series of straightedge and

compass sections constructing, for example, a geometric section `(geometric-section` p q`)`.

Then there are more interesting constructions associated with curious geometric puzzles. For example, given two circles C_1 and C_2 on the plane, and a radius r, find the circle C of radius r that is tangent to both C_1 and C_2. An example of this construction is in the lower bouts of the Amati violin, where the curve is made up of three principal arcs: the middle one is from a circle tangent to others as described here. Solving the equations is a middle-school headache, but the geometric construction is fun.

Similarly, there is a geometric problem in computing reverse curves at the corners of the upper and lower bouts. Given a large circle C that defines the curve at the upper bout, a radius r, and a point p, where is the circle of radius r located that is tangent to the outer side of C, and also tangent to p? Luthiers solved these kinds of problems by construction and without math. With appropriate software, we do the same thing.

```
(define (reverse-curve inner-circle outer pt)
  (let ((outer-radius (- outer (radius inner-circle))))
    (let ((pts (intersect (circle pt (abs outer-radius))
                          (circle (center inner-circle)
                                  (+ outer-radius
                                     (radius inner-circle))))))
      (map (lambda (p) (circle p (abs outer-radius))) pts))))
```

```
(define (inscribe c1 c2 r)
  ; circles tangent to insides of c1, c2 of radius r
  (let ((c1p (circle (center c1) (- (radius c1) r)))
        (c2p (circle (center c2) (- (radius c2) r))))
    (map (lambda (pt) (circle pt r)) (intersect c1p c2p))))
```

Another example comes from the design of a Montagnana violoncello, where a signature characteristic of that maker is found: the principal curve of the upper bout and the reverse curve are not tangent, or mediated by another curve, but instead connected by a straight line. This construction is easy enough by eye, but what is the exact construction? Again, these straightedge and compass constructions have detail, but we can abstract away from them with no loss of detail.

```
(define (tangent small big)
; finds the two lines tangent to (nonintersecting) small,
; big, where the lines are intersecting between them
  (let ((d (line (center small) (center big))))
    (let ((p1 (perpendicular d (center small)))
          (p2 (perpendicular d (center big))))
      (let ((a (top (intersect small p1)))
            (b (bottom (intersect big p2))))
        (let ((i (intersect d (line a b))))
          (let ((j (midpoint i (center big))))
            (let ((pts
                   (intersect big
                              (circle j (distance i j)))))
              (map (lambda (pt) (line i pt)) pts)))))))))
```

6. Abstraction of construction, and of design. Towards a computational art history.

One of the most important tasks which the new science of art could set for itself would be a through analysis of the history of art; to determine the elements, construction and composition in various periods. . . [the analysis] borders on the problems of the "positive" sciences, the second part... touches the problems of philosophy.
Wassily Kandinsky, **Point and Line to Plane** (Kandinsky 1926)[8]

[8] It's worth noting that this book was written in 1928, clearly affected by the logical positivists of the Vienna circle and the analytic philosophy movement before it (see, for example, (Galison 1990) for the working out of this striking but somewhat obvious connection). Analytic philosophy is also a significant intellectual precursor of the development of programming languages.

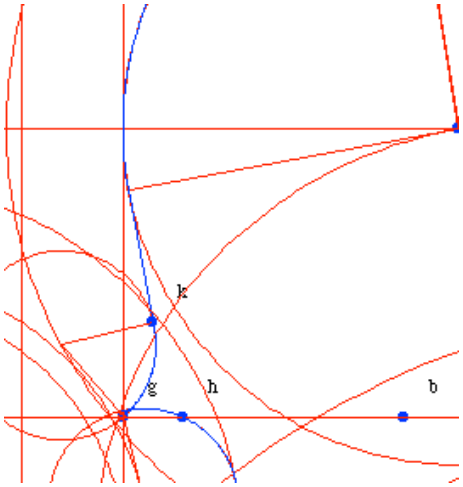

Figure 5. *Detail of Montagnana violoncello, giving upper bout construction using a tangent line between two circles*

The code for the Amati violin describes various computational dependencies. Changing some of the integer ratios in the code changes the design, while maintaining the coherence of the form. At some point, the implied constraints may become unsatisfiable (say, a line that must be to the left and to the right of some critical point, causing an intersection that no longer has a solution), and then the implementation cannot draw the outline. For the most part, this breakdown should be regarded as a feature, not a bug—it's an error message in itself. That the instrument is defined as a set of proportional relations, as opposed to curves that must be set to some fixed set of points, lends a certain plasticity to the specification that probably facilitates the satisfaction of these constraints.

A bit more subtle than strictly computational constraints are design constraints. These don't come from the mechanics of layout, but from dimensions that must be satisfied in a final product. For example, the *body stop* of a violincello, measured from the top of the front plate abutting the neck, to the inner nick at the middle of the ff-hole, is typically 400mm. Few modern instruments diverge widely from that dimension, while some old instruments do. It's not clear to me in what way these kinds of constraints should be built into the automated design process.

A clear advantage of this automated drafting is that we can make hypotheses and check them out faster, make mistakes and recover from them faster, than what might be possible with pencil and paper. One question I am looking at is Denis's thesis that Stradivari marks a turning point where the proportional approach is no longer fully explanatory, and that use of metric measurement is necessary. In order to test this assertion, I've been trying to reverse engineer the outline of the 1707 "Countess of Stanlein" Stradivarius violoncello (see (Reuning 2012)), in order to determine whether it could have an effectively proportional description. I've got a first cut at this outline done, and I expect to give more details on this in the full paper.

Abstraction mechanisms give a higher-order representation to the low-level business of straightedge and compass construction. We've given some examples of this abstraction at the more mechanical level, but I would like to be able to raise this abstraction to a more conceptual level, if we could characterize the kind of computing with shapes that is inherent in instrument design. The

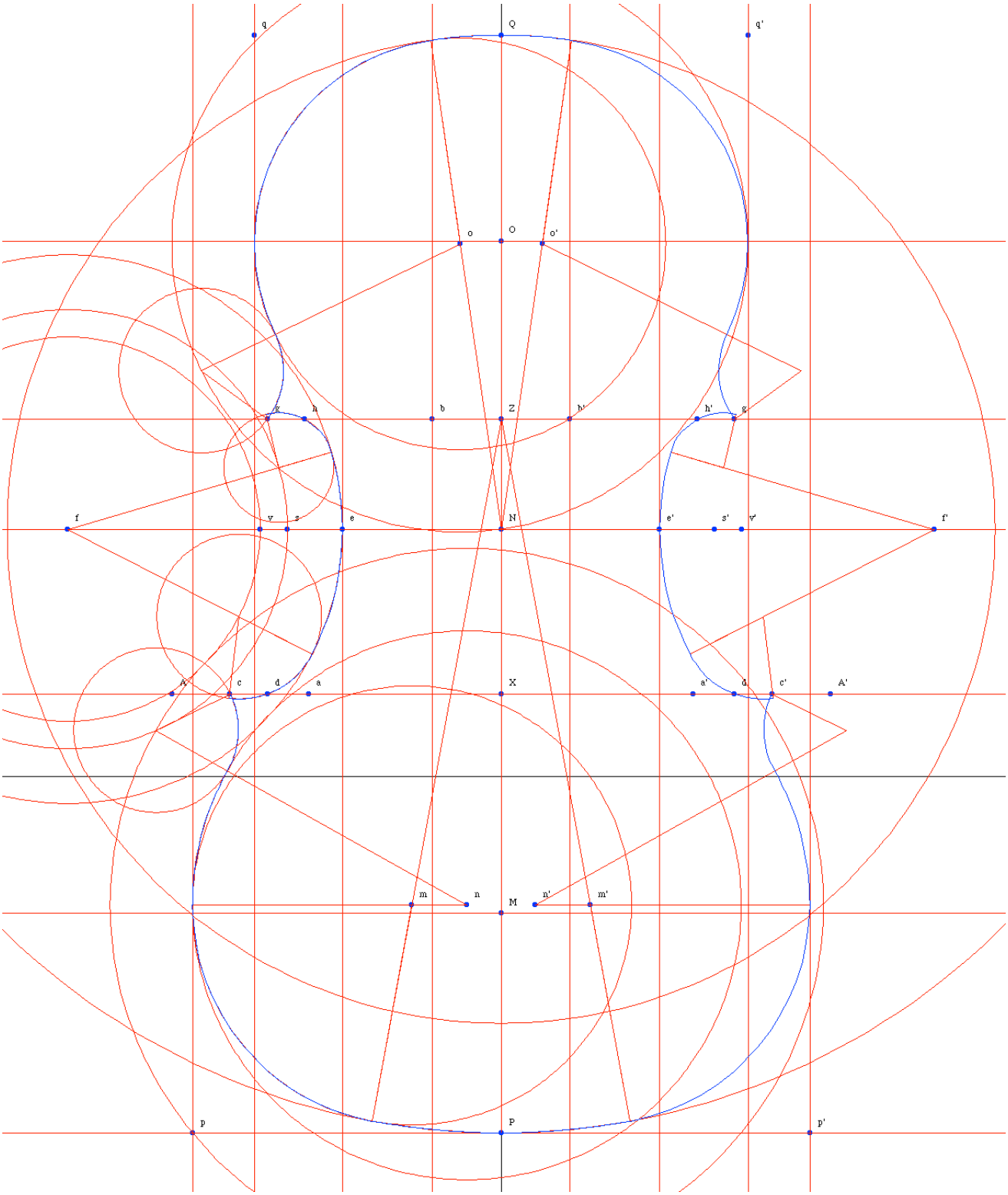

Figure 6. *Violin by Andrea Amati*

Arnault lute gives some hints in this direction, as its outline is a consequence of quadrature, rotation, proportion, and approximation. Bluntly put, if a martini is six parts gin to one part vermouth, what is a Gofriller violoncello? By a familiar analogy: we're used to programming language constructions (say, continuation-passing style) as a basic form, with particular variants (Gödel, Kuroda, Kolmogorov, etc.) derivative from double-negation embeddings; why can't instrument forms be computational variants on a basic theme? What's needed is a kind of grammar of relevant shape and how to compute with it. The thematic homilies of Kandinsky (Kandinsky 1926) given much greater detail in a recent book by George Stiny (Stiny 2008), provide some suggestions of how this work could evolve. In particular, Stiny describes grammars for the evolution of shapes—"how I stopped counting," he says, "and started to see." Good artists know how to see. In this particular example, can their insight, and how it changed, be characterized formally?

7. Historical evidence and the *a priori*

How did luthiers draw outlines? Like many historical and scientific questions, this one has empirical answers, and *a priori* answers. We can look for evidence, as well as analyze from first principles. In his 2010 book on Stradivari, Stewart Pollens writes that he "does not appear to have employed a rigid system of geometry and proportion in designing his violins" and takes a questionable view of such explanations since (and including) the 1782 manuscript of Bagatella (Pollens 2010). The solution of Denis is, on the contrary, heavily *a priori*, with evidence supplied by the 15th-century Arnault manuscript, but not by historical evidence from 18th-century luthiers.

To think that each generation of luthiers has simply copied what came before, perhaps with small modifications, strikes me as the historical equivalent of a non well-founded induction. There has to be a base case. Amati is generally regarded as the first mover in the design of string instruments, as we know them. The solution of Denis is from first principles, effectively contemporaneous with Amati, but also in the spirit of ideas of proportion and symmetry that date from the era of Vitruvius[9], and went on to animate the work of architects, artists, furniture makers, and—as Denis persuasively argues—instrument makers. Nonetheless, it is natural to seek the assurance of empirical confirmation as well. (Such a dissent from Pollens's critique is expressed by Andrew Dipper, the translator of an 18th c. luthier's construction notebook (Dipper 2013), and a cataloguer of items from the Museo Stradivariano in Cremona.)

The situation is analogous to an irreverent remark about the difference between geology and geophysics: geology is the observation of physical phenomena that can't be explained, but geophysics is the explanation of physical phenomena that can't be observed.

8. Conclusions and perspective

A natural and obvious criticism of what I've described here is this: it just using a computer to draw pretty pictures. The pictures are beautiful and of historical and cultural interest, but the programs are not. In response to this point of view, I think that the way we represent what we know is important, and a program clarifies meaning, more than any natural language directions. Moreover, there are inevitably levels of such informal description, from detailed ("do this, then do that") to high-level (literally, "big picture"), and higher-order procedures, well-written, can explain both with no loss of accuracy.

There may be "better" ways of drawing things (for example, with splines), but that doesn't interest me here. What matters is the representation of knowledge that was already known. If research is, literally, the re-searching for things that were already found, then a good justification for it is that in rediscovery, we say it better, or in ways that are more compelling to us. It is a natural progression not only in our understanding of this particular subject, but, more generally, how we represent what we know.

Acknowledgements: In the past half-dozen years, I've become very interested in building string instruments, and in particular, violincellos. I am grateful to Olin Shivers, whose enthusiasm for this project has been a big motivation to me. François Denis has been generous with his time and his advice. Curtis Bryant, a Boston luthier and restorer, has given me a great amount of guidance over several years. Marilyn Wallin taught me a lot about instrument making, varnishing, and setup. George Stiny has been a very helpful guide to relevant literature relating theories of proportion and classical architecture. Thanks also to an inspired reviewer who made many helpful comments. The mistakes I've made and misunderstandings I've had in construction, design, or explanation, are my responsibility and not theirs. Finally, David Van Horn, the POPL workshops co-chair, persuaded me to make a submission to the OBT ("Off the beaten track—New directions in programming language research") workshop; the encouragement I received from him and from workshop participants have motivated this paper.

[9] See (Gordon Smith 2003), especially Book III.

References

Harold Abelson and Gerald J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, Massachusetts, USA, 1996.

Antonio Bagatella. *Regole per la costruzione de violini viole violoncelli e violoni*. Turris, Cremona, 1782.

François Denis. *Traité de lutherie: The violin and the art of measurement*. ALADFI: Association des Luthiers et Archetiers pour le Développement de la Facture Instrumentale, Nice, France, 2006.

Andrew Dipper. *Librem Segreti de Buttegha: The violin and its fabrication, 'a book of workshop secrets'*. Dipper Publications, 2013.

Peter Galison. Aufbau/Bauhaus: Logical positivism and architectural modernism. *Critical Inquiry*, 16(4):709–752, 1990.

Thomas Gordon Smith. *Vitruvius on Architecture*. Monacelli Press, 2003.

Ian Harwood. A fifteenth-century lute design. *Lute Society Journal*, 2, 1960.

Peter Henderson. Functional geometry. *Higher Order Symbol. Comput.*, 15(4):349–365, December 2002. ISSN 1388-3690. . URL `http://dx.doi.org/10.1023/A:1022986521797`.

Edward Heron-Allen. *The Violin, as it was and is*. Ward-Lock, 1885.

Wassily Kandinsky. *Point and Line to Plane*. Dover, New York, 1926.

George Martin. *Geometric Constructions*. Springer, 1998.

Stewart Pollens. *Stradivari*. Cambridge University Press, 2010.

Christopher Reuning. *The "Paganini, Countess of Stanlein" Stradivari Violoncello of 1707*. 2012.

Simone Sacconi. *The 'Secrets' of Stradivari*. Eric Blot (Cremona), 1979.

Martin Schleske. Zeitgeist and violinmaking: Milestones of art and science, a brief journey through time. 2004.

Jakob Steiner. Synthetic solution to a problem of Malfatti. URL `http://en.wikipedia.org/wiki/Malfatti_circles`.

Ian Stewart. *Galois Theory*. Chapman and Hall, 1973.

George Stiny. *Shape: Talking about Seeing and Doing*. MIT Press, 2008.

Andrew Sutton. *Ruler and Compass: practical geometric constructions*. Wooden Books, 2009.

Wikipedia. Vesica Piscis. URL `http://en.wikipedia.org/wiki/Vesica_piscis`.

Henri Arnault of Zwolle. Treatise. *Bibliothèque Nationale, Paris*, ms Lat. 7295, 1440.

Appendix: Violin by Andrea Amati

```
(define Amati
  (let ((xq 400)) ;; should be 208mm in the Amati---this is just a screen fit...

; LAYOUT OF THE AREA on which the curves are drawn...
(let ((X (label "X" (point 0 000)))) ; this could be anywhere---just to center it on the output screen
  (let ((A (label "A" (xshift X (- (/ xq 2)))))
        (Q (label "Q" (yshift X xq))))
    (let ((N (label "N" (pointfrom X Q (/ 1 4)))))
      (let ((q (label "q" (xshift (intersect (horizontal Q) (vertical A))
                                  (/ (distance X N) 2))))
            (vv (xshift A (/ (distance X N) 8)))
            (O (label "O" (yshift Q (- (* (distance X N) (/ 5 4))))))
            (Z (label "Z" (yshift N (* (distance X N) (/ 2 3)))))
            (P (label "P" (yshift X (- (* (distance X N) (/ 8 3)))))))
        (let ((p (label "p" (intersect (horizontal P) (vertical vv))))
              (M (label "M" (pointfrom X P (/ 1 2))))
              (a (label "a" (xshift A (/ (distance X Z) 2)))))
          (let ((b (label "b" (xshift Z (- (/ (distance A a) 2))))))
           (let ((ee (label "e" (xshift (intersect (vertical b) (horizontal N)) (- (* (xdistance b p) (/ 3 8)))))))
             (let ((c (label "c" (xshift (intersect (vertical p) (horizontal X)) (/ (xdistance ee p) 4))))
                   (d (label "d" (xshift (intersect (vertical p) (horizontal X)) (/ (xdistance ee p) 2))))
                   (h (label "h" (xshift (intersect (vertical ee) (horizontal Z)) (- (/ (xdistance ee p) 4)))))
                   (g (label "g" (xshift (intersect (vertical ee) (horizontal Z)) (- (/ (xdistance ee p) 2))))))
                (list X A Q N q O Z P p M a b ee c d h g
                      (horizontal N) (horizontal O) (horizontal Z) (horizontal P) (horizontal X) (horizontal M)
                      (vertical p) (vertical q) (vertical b) (vertical ee)

; THE LOWER BOUTS...
(let ((ZMcircle (circle Z (distance Z M)))
      (ZPcircle (circle Z (distance Z P))))
  (let ((m (label "m" (bottom (intersect ZMcircle
                                         (make-line 1 p) ; line w/slope 1 through p
                                         )))))
    (let ((mcircle (circle m (distance M P))))
      (let ((n (label "n" (xshift m (- (distance X Z) (distance M P))))))
        (let ((ncircle (circle n (+ (distance M P) (distance X Z) (- (distance M P)))))
              (reverse-lower-left
                (lower-circle (reverse-curve (circle n (distance X Z)) (+ (distance X Z) (/ (distance X N) 2)) c))))
                   (list m n (circle n (distance n (center reverse-lower-left)))
                         ZPcircle mcircle ncircle reverse-lower-left
                         (make-curve P c (list ZPcircle mcircle ncircle reverse-lower-left)) ))))))

; THE UPPER BOUTS...
(let ((Ncircle (circle N (distance N Q)))
      (o (label "o" (top (intersect
                          (circle N (distance N O))
                          (make-line -1 q) ; line w/slope -1 through q
                          )))))
   (let ((ocircle (circle o (distance O Q))))
     (let ((reverse-upper-left
           (upper-circle (reverse-curve ocircle
                                        (distance N O)
                                        g))))
              (list o (circle o (distance o (center reverse-upper-left)))
                    Ncircle ocircle reverse-upper-left
                    (make-curve Q g (list Ncircle ocircle reverse-upper-left)) ))))

; THE MIDDLE BOUTS...
(let ((f (label "f" (xshift ee (- (distance X Z)))))
      (v (label "v" (xshift ee (- (/ (distance X N) 2)))))
      (s (label "s" (xshift ee (- (/ (distance N Z) 2))))))
  (let ((ecircle (circle f (distance f ee)))
        (vcircle (circle f (distance f v)))
        (scircle (circle f (distance f s))))
    (let ((reverse-lower-middle
            (upper-circle (reverse-curve ecircle (distance f v) d)))
          (reverse-upper-middle
            (lower-circle (reverse-curve ecircle (distance f s) h))))
              (list f v s (circle f (distance f ee)) reverse-upper-middle reverse-lower-middle
                (make-curve g c (list reverse-upper-middle (circle f (distance f ee)) reverse-lower-middle)) ))))))))))))))
```