De Pen Draait ende Keert.
The pen twists and turns.
Die Federtorsionen und Drehungen.

:kalliculator

Frederik Berlaen
frederik@typemytype.com
www.typemytype.com
TYPE(MEDIA, KABK THE HAGUE, '05'06


```
import interfaceInte
from interfaceInterp
interPenAtorWindow
reload (interfaceInt
from interfaceInterp
interPenAtorWindow

interPenAtorWindow()


##################

import sys
from AppKit import *
from vanilla import
from robofab.interfa
import AskYesNoCance

from PFont import PF
import PFont
reload (PFont)
from PFont import PF

import PFoutPut
reload(PFoutPut)
from PFoutPut import


from PFPen import PF
import PFPen
reload (PFPen)
from PFPen import PF

import postScriptNam
reload(postScriptNam
from postScriptNames
import postScriptNam
theOtherWaypostScrip

class TestCustomNSVi
 def __init__(self):
   self.Gname = None
   self.PFont = None
   self.selectSegmen
   self.pen = None
   self.drawBezier =
   self.drawOval = T

 def viewDidEndLiveR
   self._recalcSize(

 def _recalcSize(sel
   w, h = self.super
visibleRect()[1]
   self.setFrame_(((

   if self.inLiveRes
     self._recalcSize
   ### rescaling of
drawing window
   width, height = s
   fontHeight = self
abs(self.PFont.desce
   marge = 50
```

Thanks to Johan Berlaen for the ‘bezier-maker’,
Erik van Blokland, Just van Rossum and Tal Leming
for robofab, .ufo and vanilla
Elisabeth Demeyere for being in The Hague

The question

Is it possible to generate a pointed pen drawing?

In the first part of the postgraduate Type & Media I got in touch with DrawBot and scripting in the classes of Just van Rossum. DrawBot is an educational tool learning students how to script. I started to play around with the python program language. This was my first attempt to try to generate a pointed pen. After New Year a very early version was ready and Erik van Blokland asked me to present it on the RoboThon scripting conference. This I did, with several enthusiastic comments afterwards. My eagerness for this research only increased by this event.

After this, my final question became more generalized: can a program simulate a pen? Can a program keep calligraphic style elements? I rebuilt the script from the first part of the course with more dept, with a stronger idea that is closer related to the qualities of a real pen. I had a closer look at the broad nib pen, the pointed pen and the relationship between them. This I tried to implement in a program that actually generates contrasts for these two pens.

```
   t = NSAffineTrans
   scaleFactor = (he
fontHeight
   t.scaleXBy_yBy_(s
scaleFactor)
   t.translateXBy_yB
abs(self.PFont.desce

   if self.showMetri
     rwidth = 3
     NSColor._stroke

     NSColor.greenCo

     line = NSBezierP
     line.moveToPoint
     line.lineToPoint
     line.moveToPoint
PFont.xHeight ))
     line.lineToPoint
PFont.xHeight ))

     line.moveToPoint
PFont.descender)-mar
     line.lineToPoint

     line.moveToPoint
Gname].width, -abs(s
descender)-marge  ))
     line.lineToPoint
Gname].width, width*

     line.setLineWidt
     line.stroke()


   pen = PFPen(self.
   pen.selectSegment
selectSegment

   if self.drawOval:
     pen.outputClass
     pen.draw(self.PF
NoPostScriptName = F
   if self.drawBezie
     pen.draw(self.PF
drawInContour = self
NoPostScriptName = F



class setPens(object
a big setPens and a
glyph window

 def __init__(self,
None):

   self.pen = pen
   self.s = s
   self.shapeCheck =

   if self.pen.shape
     self.shapeCheck
```

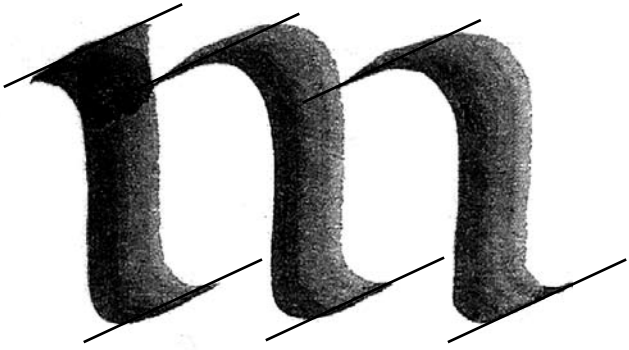
```
form.transform()
ight-marge*2)/

caleFactor,

y_(marge*4,
nder)+marge)

cs:

width = 50

lor().setStroke()

Path.bezierPath()
t_((-marge*4, 0))
t_((width*100, 0))
t_((-marge*4, self.

t_((width*100, self.


t_((0, -abs(self.
ge ))
t_((0, width*100))

t_((self.PFont[self.
elf.PFont.

t_((self.PFont[self.
100 ))

th_(2)



pen)
  = self.



 = CocoaOutput
Font, [self.Gname],
alse)
r:
Font, [self.Gname],
.drawBezier,
alse)



): ### there is
small one in the


s, pen, onTheFly =



 False

 == "oval":
 = True
```

Process of calligraphy

My purpose was to create a tool that behaves as a pen, so I first analyzed the process of writing with a pen.

. . . . . Pen

You start with choosing your pen. On your desk there is a collection of different kind of pens, from a small broad nib to a very wide one, and different kind of pointed pens, one more flexible than the other. From these you choose one. The broad nib is a flat, non flexible pen where the angle in most cases stays the same. For the roman construction the most common angle is around 30°. There are exceptions for the 's' and the diagonals 'k', 'v', 'w', 'x', 'y', 'z' where the pen rotates to 45°. The angle for the cursive construction is around 45°. The capitals have again a 30° basis angle. The same applies to the diagonals: the pen angle is 45° and some capitals like the 'N' or 'M' need a rotation up to 60°. These rotations of the pen influence the proportion and they construct the contrast of a character. The angle of the pen determines the thick and the thin parts. Next to the contrast, the angle of the pen is also responsible for little details, little changes in a shape, because the pen can rotate for a whole contour and also inside that contour. For example to make small serifs, stroke endings, the top part of an 'a' can start around 45° and goes slowly to 30°.



The same principle can be applied for the pointed pen. The rotation of the pen is here even more important because the pen needs to rotate constantly. An other quality of the pointed pen is that you need to give some pressure to draw thick lines. The thin lines are drawn without any pressure. The contrast of a pointed pen is in direct relation with the angle of the pen and with the pressure, which complicates drawing certain characters. Then you need to rotate the pen. In most cases the angle of the pen is 90° and the pressure starts only in the vertical strokes. It is necessary to rotate the pen a lot, because it is only possible to make thick strokes when pressure is given on the pen. This is again only possible when the pen is pointed in the same direction of the stroke you want to draw. The contrast type is called expansion because the thin part grows into the tick part and vice versa.

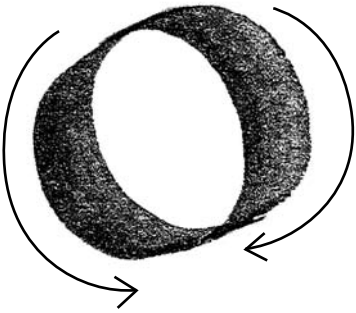
```
   self.s.ovalCheck
10, 70, 50), "oval",
shapeCheck, callback
ovalCheckCallback)
   self.s.rectCheck
90, 10, -10, 50), "r
self.shapeCheck, cal
rectCheckCallback)


   self.s.tWidth = T
20), "width")
   self.s.widthSlide
70, -10, 30), value
w, minValue = 2, max
stopOnTickMarks=Fals
widthSliderCallback,

   self.s.tHeight =
-10, 20), "height")
   self.s.height = E
110, 110, 25), "%s"
callback=self.height
   self.s.heightSlid
130, -10, 30), value
pen.h, minValue = 2,
= 200, stopOnTickMar
callback=self.height
liveFeedback=True)


   self.s.tInterPen
-10, 20), "InterPena
   self.s.tInterPenV
EditText((110, 170,
%self.pen.interPenat
tInterPenValueCallba
   self.s.InterPenSl
200, -10, 30), tickM
= self.pen.interPena
self.pen.minExtrem,
pen.maxExtrem, stopO
callback=self.InterP
liveFeedback=True)
   self.s.tpointed =
-10, 20), "pointed")

   self.s.extrapolat
EditText((10, 260, 7
%self.pen.minExtrem,
extrapolateMinCallba
   self.s.extrapolat
70, 260, -10, 25), "
pen.maxExtrem, callb
extrapolateMaxCallba


   #self.s.ContrastT
-80, 70, 20), "Contr
   #self.s.noContras
70, -80, -10, 20), "
   #self.s.noContras
Slider((10, -60, -10
```

```
= CheckBox((10,
 value= self.
 = self.

= CheckBox((-
ect", value= not
lback = self.


extBox((10, 50, -10,

r = Slider((10,
= self.pen.
Value = 200,
e, callback=self.
 liveFeedback=True)

TextBox((10, 110,

ditText((80,
% self.pen.h,
Callback)
er = Slider((10,
 = self.
 maxValue
ks=False,
SliderCallback,


= TextBox((10, 170,
tion")
alue =
80, 25), "%s"
ion, callback=self.
ck)
ider = Slider((10,
arkCount=9, value
tion, minValue =
maxValue = self.
nTickMarks=False,
enSliderCallback,

 TextBox((-70, 230,


eMin =
0, 25), "%0.2f"
 callback=self.
ck)
eMax = EditText((-
%0.2f" %self.
ack=self.
ck)


ext = TextBox((10,
ast")
tText = TextBox((-
No Contrast")
tSlider =
, 30), value =
```

. . . . . Skeleton

In a drawer of your desk are different sorts of paper. You also choose one of them. In the end, you make a choice in the kind of ink or paint you wish to apply. Then the part starts where you are thinking about what the characters should look like. This is strongly related to the choice of pen, paper and ink. The movement of the hand is the most important part of calligraphy. This movement exists out of strokes. For example the 'o' is made out of two strokes. The first stroke goes from the top and turns left in a bowl to the bottom. The second stroke starts again at the top and turns right to the bottom. The hand must be raised to go to the top again for the second stroke, because the pen does not allow making upstrokes. It causes friction with the paper. The cursive construction is build out of upstrokes. Because there is less friction, upstrokes are possible. The collection of all these strokes construct the skeleton. This imaginary path forms the skeleton of that character. The pen and the angle of that pen define the contrast. If the pen is a pointed pen, the pressure on the pen also determines the contrast and the drawing.
This movement is a skill that can be learned by practice. Penmanship is developed through repetition, reproducing and patience. The quality of it depends on the skill of the writing master. Penmanship is all about beauty, repetition and drill.

I chose the broad nib and pointed pen because they are common used and these pens cover the most of the traditional writing systems.



The reversed process

My idea was to simulate, to generate a pen. The process of the kalliculator is based on the process of calligraphy, but then reversed. First the skeleton must be drawn, which is the movement of the hand, and then you choose the kind of 'pen' and apply it on a 'paper'. The 'pen' became digital and the 'paper' the output, which can be a .pdf or a font. I reversed the process for scripting reasons. There has to be a shape in your head first, in order to apply a pen on it afterwards.

. . . . . The skeleton

In calligraphy the skeleton is the movement of the hand when writing a character. It can also be seen as the middle of a stroke. However, the exact mathematical middle is not equal to the movement of the hand of that stroke. A skeleton is something in between. A pen in a hand turns around and the mathematical middle does not take these rotations into account.

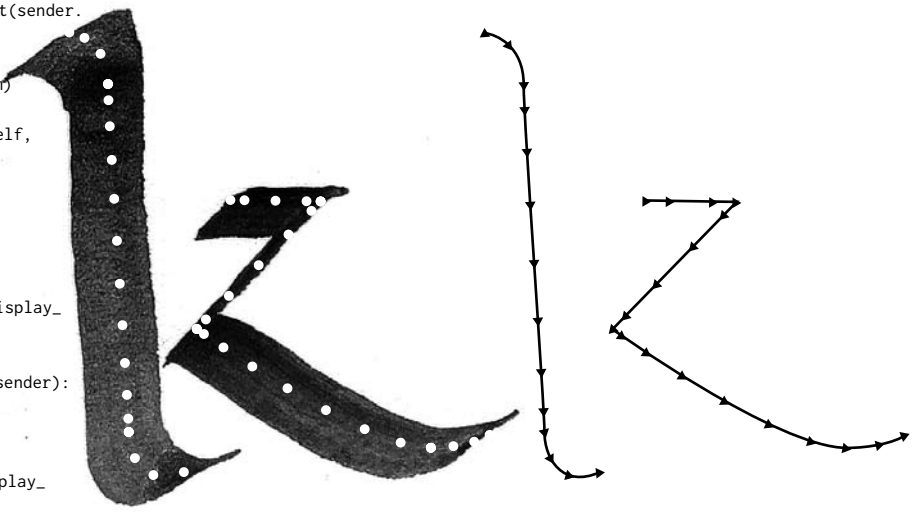
A skeleton is a collection of strokes. The difference with strokes is that a skeleton can be simplified into one line. For example an 'o' is made out of two pen strokes, but the skeleton consists of one line. In the digital movement, the skeleton, you don't lift your hand anymore, only when the stroke starts somewhere else. This is for example the case with a 'd', it exists out of three pen strokes but only two contours. Thinking in strokes has become redundant.

The skeleton of a character, a glyph, is built up out of curves and beziers. One curve is controlled by a starting point, two handles and an ending point. The two handles

```
self.pen.noContrast,
maxValue = 1, stopOn
callback=self.noCont
   self.s.randomtext
-10, 20), "randomnes

 #def noContrastSlid
sender):
 # self.pen.noContra
get())
 # if self.onTheFly:
 #   self.onTheFly.se
(True)
 def heightSliderCal
sender):
   self.pen.h = int(
   self.s.height.set
   if self.onTheFly:
     self.onTheFly.se
(True)
   self.pen.w = int(
   self.s.width.set(
   if self.onTheFly:
     self.onTheFly.se
(True)
 def randomnessSlide
sender):
   self.pen.randomne
get())
   if self.onTheFly:
     self.onTheFly.se
(True)
 def extrapolateMinC
sender):
   if sender.get() !
     self.s.InterPenS
setMinValue(self.pen
 def extrapolateMaxC
sender):
   if sender.get() !
     self.pen.maxExtr
get())
     self.s.InterPenS
setMaxValue(self.pen

 def tInterPenValueC
sender):
   if sender.get() !
     self.s.InterPenS
set(float(sender.get
     self.pen.interPe
float(sender.get())
      self.onTheFly.
(True)

 def rectCheckCallba
   self.pen.shape =
   self.s.ovalCheck.
   if self.onTheFly:
     self.onTheFly.se
(True)

 def ovalCheckCallba
   self.pen.shape =
```

control the curve, where it is bending to. These curves cannot contain any information of a pen inside. I had to find a way to store the skeleton and all the information about the different pens. This is the start of a .pfo, which is a Point Font Object. A .pfo does not have any curve inside but is made out of points which were on a bezier. It is a long list of points. Each point can be compared with a touch of the pen on the paper. During one stroke of the pen, the pen touches the paper a hundred times. At the end it looks fluidly, provided there are enough points. This resolution can be interpreted as the speed of writing. When writing very slowly, which is translated into more points in a .pfo, the precision is higher. The higher the speed of writing, the lower the precision of the drawing. I translated this in a resolution parameter. The end result will have more fluid curves if the resolution is higher.

Each point contains information about the broad nib pen and the pointed pen, necessary to draw the glyph afterwards. It contains the parameters that determine how the pen must be handled, for example the angles of these pens.



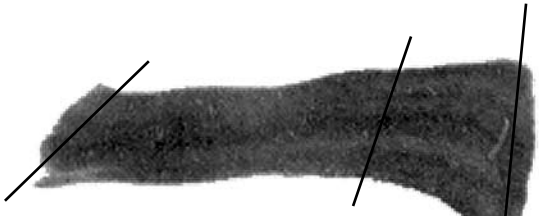
..... The pen

How to automate the pen? There is a contradiction: automating a craft. Can a craft be generic? In the pure sense it cannot, because a craft is an activity involving skill in making things by hand. But still there must be a way to automate a big part of that skill. The most important stage in attempting to generate a craft is to describe what you want to automate. In this case it is the pen, the pointed and broad nib pen.

..... What is a broad nib pen

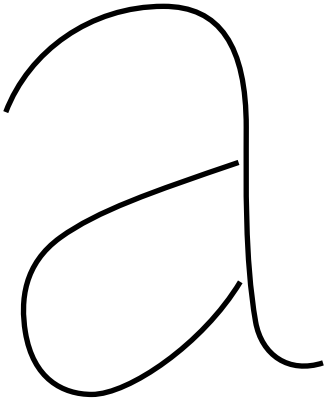
A broad nib pen has a certain width. The nib is not flexible. In most cases the pen is being held in a 30° angle. This angle forms the contrast because the hand follows the skeleton, while the pen is always in the same angle. Differently put; the width of the pen always stays the same. While drawing the shape, your hand turns around following the skeleton, holding the pen in a certain angle and that pen draws a thin line on each point. This is the scripting translation for a broad nib pen.

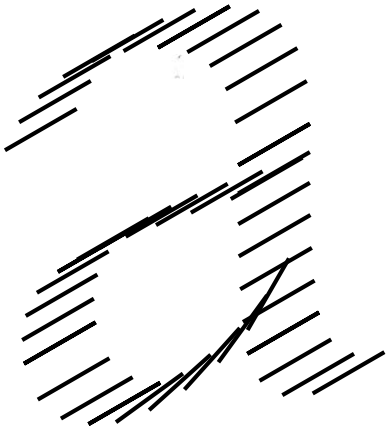
Here it is possible to add a certain thickness to the pen. Now on each point of the skeleton, the digital 'pen' draws a little rectangle or oval. The result of drawing an oval is nicer because it results in a more smoothly formed shape. A rectangle is too square.

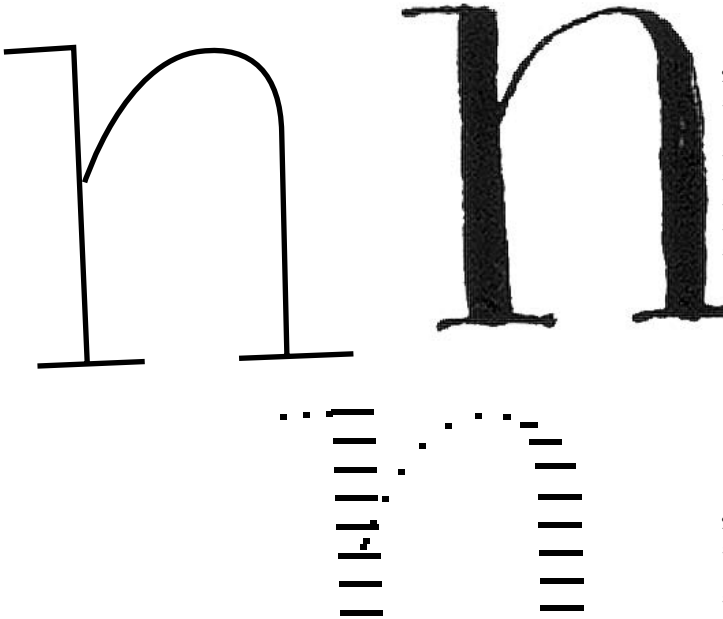


```
   if self.onTheFly:
     self.onTheFly.se
(True)


 def widthCallback(s
   self.pen.w =  flo
   self.s.widthSlide
get()))
   if self.onTheFly:
     self.onTheFly.se
(True)

   self.pen.h =  flo
   self.s.heightSlid
get()))
   if self.onTheFly:
     self.onTheFly.se
(True)

 def InterPenSliderC
sender):
   self.s.tInterPenV
%sender.get())
   self.pen.interPen
float(sender.get())
   if self.onTheFly:
     self.onTheFly.se
(True)
class extraPenTools(
 def __init__(self,
onTheFly, selfSetAng
   self.s = s
   self.pen = pen
   self.PFont = PFon
   self.onTheFly = o
   self.selfSetAngle

   self.s.AWFirst, s
PFont[self.selfSetAn
selfSetAngle.selectS
oothAdjustPenWidth(s
selectSegment[1])


   self.s.adjustWidt
= Slider((10, 20, 15
tickMarkCount=15, va
s.AWFirst, minValue
stopOnTickMarks=True
adjustWidthFirstSlid
liveFeedback=True, s
   self.s.adjustWidt
= Slider((40, 20, 15
tickMarkCount=15, va
s.AWLast, minValue =
stopOnTickMarks=True
adjustWidthLastSlide
liveFeedback=True, s

 def adjustWidthFirs
f, sender):
   if not sender.get
     if not sender.ge
```

For the broad nib pen, two parameters determine the look of the letter. Firstly, the shape of the pen is determinative; the wideness and the thickness, is it an oval or a rectangle. The second parameter is the angle of the pen. The angle can be set for each contour, each segment even for each point on the skeleton. It is important to be able to set a different angle for each point because in some situations the pen rotates a bit inside a segment or a contour. These settings create the opportunity to generate more detailed characters so that the end result is closer to the calligraphic starting idea of my tool.







..... What is a pointed pen

A pointed pen is a flexible pen. The thick parts in a shape are created by pressure on the pen. The two legs of the nib open when giving pressure on the pen and the ink can flow widely. When there is no pressure, the lines are thin. In most cases the thick lines are in the vertical part of a character and the thin parts are horizontal.

```
      return
   self.s.AWFirst =
   if self.selfSetAn
"red":
   else:
     self.s.tAdjustW
%d%% %d%%" %(self.s.
s.AWLast*100))
     self.PFont[self
selfSetAngle.Gname][
selfSetAngle.selectS
setSmoothAdjustPenWi
selfSetAngle.selectS
s.AWFirst, self.s.AW
   self.onTheFly.set
 def adjustWidthLast
, sender):
   if not sender.get
     if not sender.ge
       return
   if self.selfSetAn
"red":
     return
   self.s.tAdjustWid
%d%% %d%%" %(self.s.
s.AWLast*100))
   self.PFont[self.
selfSetAngle.Gname][
selfSetAngle.selectS
setSmoothAdjustPenWi
selfSetAngle.selectS
s.AWFirst, self.s.AW
   self.onTheFly.set


class setAngle(objec
 def __init__(self,
pen):
   self.Gname = Gnam
   self.PFont = PFon
   self.pen = pen

   self.smoothPen =
   self.drawBezier =
   self.drawOval = T
   self.metrics = Fa

   self.broadnibFirs
= self.PFont[self.Gn
selectSegment[0]].ge
selectSegment[1], "b

   self.pointedFirst
= self.PFont[self.Gn
selectSegment[0]].ge
selectSegment[1], "p


   self.w = Window((
minSize = (300,200))

   self.scrollViewNS
TestCustomNSView.all
(((0, 0), (368, 378)
```

```
float(sender.get())
gle.selectSegment ==


idth.set("pen width
AWFirst*100, self.

.
self.
egment[0]].
dth( self.
egment[1], self.
Last)
NeedsDisplay_(True)
SliderCallback(self

():
et() == 0:

gle.selectSegment ==


th.set("pen width
AWFirst*100, self.


self.
egment[0]].
dth( self.
egment[1], self.
Last)
NeedsDisplay_(True)


t):
Gname , PFont,

e
t


False
 False
rue
lse

t, self.broadnibLast
ame][self.
tSmoothAngle(self.
roadnib")

, self.pointedLast
ame][self.
tSmoothAngle(self.
ointed")


600, 400), Gname,


View =
oc().initWithFrame_
))
```

For the pointed pen, four parameters define the digital drawings. The first parameter is again the shape of the pen; is it an oval or a rectangle? How wide can the thickest part be? How thin must the thinnest part be? The second parameter is also the same as for the broad nib pen: the angle. If the contrast only has to be in the vertical part of the character, the angle of the pen is 90°. If you want an inverted contrast, the angle of the pen is 0°. This angle is the same angle of the pen when writing with it. The third parameter is the pressure given on the pen. The pressure determines the thickest part and also the point where the expansion starts to grow from thin into thick. Does the expansion have to start directly or more to the end of the segment or the contour? The fourth parameter is the skeleton angle. This angle determines where the contrast must be. Each point on the skeleton has an angle compared to the total skeleton. The combination of these four parameters defines the drawing of a digital pointed pen.

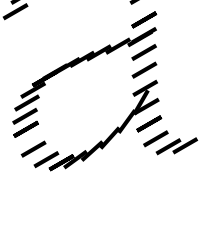
So what actually happens is that the digital pointed pen draws a thin line. The lines are perpendicular on the angle of the skeleton. The starting length of these lines is equal to the smallest value set of the pen. The length can grow to the thickest line set. The growing value is defined by the angle of the pen in relation to the angle of the skeleton and the pressure given upon it. This makes the pointed pen the most complex of the two pens. For me personally, the pointed pen is also in real writing the most difficult one because of all the parameters mentioned above. It is needed to give the right amount of pressure on the pen, in the right angle, to end up with a shape you pictured in your mind. The pointed pen asks a lot of practice and drawing skills.

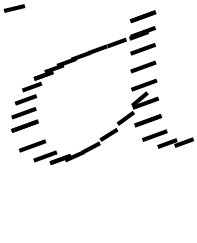
..... What is the relation between these two pens

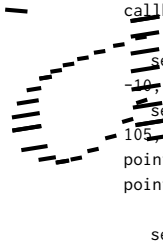
Because these two pens are now translated into parameters and values, it is possible to interpolate, to inter-pen-ate between them. This makes it all very interesting because the result can be 20% broad nib and 80% pointed pen. These new contrasts are not possible to draw by hand because these pens do not exist.

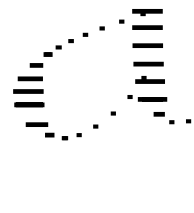
The relationship is based on the parameters of these pens. The first two can interpolate because they share the same information: the shape of the pen (oval, rectangle and their dimensions) and the angle of the pen. The other two parameters of the pointed pen have a decisive influence (100%) when the pointed pen is selected. They have no influence on the broad nib pen.

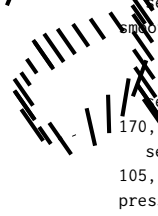
This results in exciting possibilities. When there is a possibility to interpolate, it is also possible to extrapolate. Of course these results are not controllable and definitely not calligraphic. But they are a new sort of contrast, which offers new shapes to experiment with.











```
   self.scrollViewNS
Gname
   self.scrollViewNS
PFont
   self.scrollViewNS
self.selectSegment
   self.scrollViewNS
   self.scrollViewNS
None
   self.scrollViewNS
self.drawBezier

   self.w.View = Scr
-220, -10), self.scr
backgroundColor = NS
autohidesScrollers=T

   self.w.penDrawer
-30, -10, 20), "pen"
penDrawerCallback)
   self.w.extraTools
Button((-200, -30, -
"extraTools", callba
extraToolsDrawerrCal

   self.segmentsList

   for c in self.PFo
     nr = str(c.conto
     for t in range(
       segmentNr = nr
       self.segmentsL
append(segmentNr)

   self.segmentsList
None")
   self.segmentsList
All")
   self.w.tSegment =
-10, 20), "SegmentNr
   self.w.segmentPop
100, 10, -10, 20), s
callback=self.segmen

   self.w.text1 = Te
-10, 20), "Pointed P
   self.w.pointedang
105, 135, -10, 25),
pointedFirst), callb
pointedangleCallback

   self.w.pointedang
55, 135, -10, 25), "
pointedLast), callba
pointedangleLastCall
   self.w.pointedang
smoothPen)

   self.w.tpressure
170, 10, 20), "Set
   self.w.pressure =
105, 165, -10, 25),
pressureFirst, callb
```

```
View.Gname = self.

View.PFont = self.

View.selectSegment =

View.pen = self.pen
View.showMetrics =

View.drawBezier =


ollView((10, 10,
ollViewNSView,
Color.whiteColor(),
rue)

= Button((-100,
, callback = self.

Drawer =
110, 20),
ck = self.
lback)

 = []

nt[Gname]:
ourNumber+1)
c.segmentIndex+1):
r + " " + str(t+1)
list.


.append("Select

.append("Select

 TextBox((-200, 10,
")
Up = PopUpButton((-
elf.segmentsList,
tPopUpCallback)

xtBox((-200, 110,
en")
le = EditText((-
"%i" % round(self.
ack=self.
)

leLast = EditText((-
%i" % round(self.
ck=self.
back)
leLast.show(self.



= TextBox((-200,
Pressure")
 EditText((-
"%0.1f" % self.
ack=self.
```

. . . . . Type for print as pen research

Next to this scripting project, I also researched the difference between these two pens in a printed type version. What I did was designing a broad nib and a pointed pen version for print. I tried to implement my conclusions into my program.

The starting point for drawing this type, used in this paper, was to design a true broad nib and pointed pen version. The only reason of existing for these typefaces is research. They do not solve any technical problem. They are not made to be used in a specific size, but I drew them with a text type in mind. I started with sketching and afterwards digitalized these drawings. The BroadNib Regular and PointedPen Regular have the most complete character set. I added black versions for both styles. The text in this booklet is set in BroadNib Regular, titles are set in the PointedPen Regular.

The first thing that struck me, is that the skeleton of these two types are very different. The general proportions of these two styles are different. When writing with a broad nib pen, there are no upstrokes possible, unlike writing with a pointed pen. This causes that all the connection parts, in a ‘n’, ‘b’, ‘r’, ‘d’, are deeper into the stem with a pointed pen.

In the round characters, the broad nib pen is rounder, and the pointed pen is more square. This is caused by the vertical contrast. If the pointed pen skeleton has the same roundness, it will look extended and have not enough black compared to the broad nib pen.


```
pressureCallback)

   self.w.pressureLa
55, 165, -10, 25), "
pressureLast, callba
pressureLastCallback

   self.w.text2 = Te
-10, 20), "Broadnib
   self.w.tbroadniba
200, 250, -10, 20),
   self.w.broadniban
105, 245, -10, 25),
broadnibFirst, callb
broadnibCallback)

   self.w.broadniban
EditText((-55, 245,
% self.broadnibLast,
broadnibangleLastCal
   self.w.broadniban
smoothPen)

   self.w.bezierPen
-90, -10, 20), "Bezi
bezierPenCallback, v
drawBezier)
   self.w.ovalPen =
-90, 80, 20), "Oval"
ovalPenCallback, val
   self.w.checkSmoot
120, -120, -10, 20),
callback=self.checkS
value=False)

   self.w.metricsOn
-60, -10, 20), "Metr
self.metricsOnCallba
metrics)
   self.w.open()

 def ovalPenCallback
   if sender.get():
     self.drawOval =
   else:
     self.drawOval =
   self.scrollViewNS
setNeedsDisplay_(Tru

 def bezierPenCallba
   if sender.get():
     self.drawBezier
   else:
     self.drawBezier
   self.scrollViewNS
self.drawBezier
   self.scrollViewNS
setNeedsDisplay_(Tru

   if not sender.get
     return
   self.pressureLast
get())
   self.PFont[self.
```

```
st = EditText((-
%0.1f" % self.
ck=self.
)

xtBox((-200, 220,
Pen")
ngle = TextBox((-
"Set Angle")
gle = EditText((-
"%i" % self.
ack=self.


gleLast =
-10, 25), "%i"
 callback=self.
lback)
gleLast.show(self.


= CheckBox((-120,
er", callback=self.
alue=self.

CheckBox((-200,
, callback=self.
ue=self.drawOval)
hPen = CheckBox((-
 "Smooth Pen",
moothPenCallback,


= CheckBox((-120,
ics", callback =
ck, value=self.


(self, sender):

 True

 False
View.
e)

ck(self, sender):

 = True

 = False
View.drawBezier =

View.
e)

():

 = float(sender.
```

Because with the pointed pen the contrast is in general more vertical, it gives the type a more vertical look. The contrast also makes the counters bigger. This gives visual problems when the type is used next to each other. The pointed pen seems a bit bigger, even if they measure the same height of the characters.
A solution to this problem is to lower the x-height in the pointed pen, or to increase the height in the broad nib. This value is put into the program. It scales the type a bit down for the pointed pen, only in the y-direction.

Another contrast issue is the width of the thickest parts in a glyph. Because there is a strong relationship between thick and thin in a pointed pen, there is less black in a glyph of the broad nib pen. The only reason for this is that the width of the pen is always the same in a broad nib pen. Again this is only causing difficulty when they are used next to each other. The problem is that the pointed pen looks too light. It is possible to make the global letter a little darker, but then you are losing contrast.
I wanted to draw with the same 'pen'-dimension as the broad nib and pointed pen. Another way to solve this problem is to raise the contrast by making the thick part of a glyph more thick. I found out that the tick parts of a pointed pen can be up to 10% more thick than the broad nib pen, in order to end up with the same blackness.

Finally, a difference is created when the weight is altered from regular to black. Most of these weight problems can be solved by adapting the skeleton or the angle of the pen, but it is recommended to adjust the skeleton. A black broad nib pen has a different skeleton from the regular weight, because it has a higher contrast. When creating a black based on the skeleton of a regular, the contrast

remains at the correct spots, but the end result does not have the same sharpness. But when the weights are going to an extreme there is at a certain point to much black. In order to solve this problem with a qualitative outcome, the skeleton has to be changed.

At the end the program is given different skeletons, which are interpolations between skeletons. This can even be interpolations between skeletons from a broad nib pen and a pointed pen and their black versions. Imagine then an inter-pen-ation and the search for new contrast is open.

echvwragzuspyboqftidkxljnm

arjqndkpmeuxvsyowglfhbztic

revnwadpmsghtlcfjzqxobkyui

oqlkhsaunxwjimtzgdpvyerfcb

```
Gname][self.selectSe
setSmoothPressureInS
selectSegment[1], se
self.pressureLast)
   self.scrollViewNS
setNeedsDisplay_(Tru

 def broadnibangleLa
sender):
   if not sender.get
     return
   self.broadnibLast
   self.scrollViewNS
setNeedsDisplay_(Tru

 def pointedangleLas
sender):
   if not sender.get
     return
   self.pointedLast
get())
   self.PFont[self.
Gname][self.selectSe
setSmoothAnglePenInS
selectSegment[1], "p
pointedFirst, self.p
   self.scrollViewNS
setNeedsDisplay_(Tru

 def checkSmoothPenC
sender):
    if self.selectSe
      return
    self.smoothPen =
    self.w.pointedar
105, 135, -60, 25))
    self.w.pointedar
self.pointedLast)
    self.w.pointedar
smoothPen)

    self.w.pressure
105, -60, 25))
    self.w.pressureL
self.pressureLast)
    self.w.pressureL
smoothPen)
    self.w.broadniba
105, 245, -60, 25))
    self.w.broadniba
self.broadnibLast)
    self.w.broadniba
smoothPen)

  else:
    self.smoothPen =
    self.w.pointedar
105, 135, -10, 25))
    self.w.pointedar
smoothPen)

    self.w.pressure
165, -10, 25))
```

```
gment[0]].
eg( self.
lf.pressureFirst,

View.
e)

stCallback(self,

():

 = int(sender.get())
View.
e)

tCallback(self,

():

= float(sender.


gment[0]].
eg( self.
ointed", self.
ointedLast)
View.
e)

allback(self,

egment == "red":

= True
ngle.setPosSize((-

ngleLast.set("%i" %

ngleLast.show(self.


.setPosSize((-105,

Last.set("%0.1f" %

Last.show(self.

angle.setPosSize((

angleLast.set("%i" %

angleLast.show(self.


= False
ngle.setPosSize((-

ngleLast.show(self.


.setPosSize((-105,
```

```
    self.w.broadniba
105, 245, -10, 25))
    self.w.broadniba
smoothPen)

 def metricsOnCallba

   if sender.get():
     self.metrics = 
   else:
     self.metrics = 
   self.scrollViewNS
self.metrics
   self.scrollViewNS
setNeedsDisplay_(Tru

 def pressureCallbac
   if not sender.get
     return
   self.pressureFirs
get()
   secondPressure =
   if self.smoothPen
     secondPressure =
     self.PFont[self.
sure(float(sender.g
     else:
     self.PFont[self.
Gname][self.selectSe
setSmoothPressureInS
selectSegment[1], se
secondPressure)
   self.scrollViewNS
setNeedsDisplay_(Tru

 def pointedangleCal
sender):
   if not sender.get
     return
   self.pointedFirst
   secondAngle = sel
     secondAngle = se
   if self.selectSeg
     self.PFont[self.
lePen("pointed", sel
   else:
     self.PFont[self.
Gname][self.selectSe
setSmoothAnglePenInS
selectSegment[1], "p
pointedFirst, second
   self.scrollViewNS
setNeedsDisplay_(Tru

 def broadnibCallbac
   if not sender.get
     return
   secondAngle = sel
   if self.smoothPen
     secondAngle = se

   if self.selectSeg
     self.PFont[self.
lePen("broadnib",sel
```

```
angle.setPosSize((-

angleLast.show(self.


ck(self, sender):


True

False
View.showMetrics =

View.
e)

k(self, sender):
():

t = float(sender.

self.pressureFirst
:
= self.pressureLast
.Gname].setGlobalPre
et()) )

.
gment[0]].
eg( self.
lf.pressureFirst,

View.
e)

lback(self,

():

 = int(sender.get())
f.pointedFirst
elf.pointedLast
ment == "red":
.Gname].setGlobalAng
f.pointedFirst )

.
gment[0]].
eg( self.
ointed", self
Angle)
View.
e)

k(self, sender):
():

f.broadnibFirst
:
elf.broadnibLast

ment == "red":
.Gname].setGlobalAng
f.broadnibFirst )
```

```
   else:
     self.PFont[self.
Gname][self.selectSe
setSmoothAnglePenInS
selectSegment[1], "b
broadnibFirst, secon
   self.scrollViewNS
setNeedsDisplay_(Tru

   if self.segmentsL
get())] == "Select N
     self.scrollViewW
= (1000000, 1000000)

     self.w.pointedar
     self.w.pointedar

     self.w.pressure.
     self.w.pressureL

     self.w.broadniba
     if hasattr(self,
"extraToolsDrawer"):
       self.extraTool
adjustWidthFirstSlid
       self.extraTool
adjustWidthLastSlide

   elif self.segment
get())] == "Select A
     self.w.pointedar
     self.w.pointedar

     self.w.pressure.
     self.w.pressureL
     self.w.broadniba
     self.w.broadniba
show(False)

     self.selectSegme
     self.scrollViewW
= self.selectSegment
     self.w.pointedar

     self.w.pressure.

     self.w.broadniba
       self.extraTool
adjustWidthFirstSlid

   else:
     self.w.pointedar
     self.w.pointedar
smoothPen)
     self.w.pressure.
     self.w.pressureL
smoothPen)
     self.w.pointedar
smoothPen)
     self.w.broadniba
     self.w.broadniba
smoothPen)
     c, s = segment.s
     c = int(c)
```

```
.
gment[0]].
eg( self.
roadnib", self.
dAngle)
View.
e)

ist[int(sender.
one":
NSView.selectSegment


ngle.show(False)
ngleLast.show(False)

.show(False)
Last.show(False)

angle.show(False)
,

sDrawer.
er.show(False)
sDrawer.
r.show(False)

sList[int(sender.
ll":
ngle.show(True)
ngleLast.show(False)

.show(True)
Last.show(False)
angle.show(True)
angleLast.


ent = "red"
NSView.selectSegment

ngle.set("" )

.set("" )

angle.set("")
sDrawer.
er.show(False)


ngle.show(True)
ngleLast.show(self.

.show(True)
Last.show(self.

ngleLast.show(self.

angle.show(True)
angleLast.show(self.

split(" ")
```

```
        s = int(s)
        self.selectSegme
        self.scrollViewN
= self.selectSegment

        self.broadnibFir
broadnibLast = self.
Gname][self.selectSe
getSmoothAngle(self.
"broadnib")
        self.pointedFirs
= self.PFont[self.Gn
selectSegment[0]].ge
selectSegment[1], "p

        self.w.pointedan
self.pointedFirst)

        self.pressureFir
pressureLast = self.
Gname][self.selectSe
getSmoothPressure(se
selectSegment[1])
        self.w.pressure.
pressureFirst)
        self.w.pressureL
self.pressureLast)

        self.w.broadniba
self.broadnibFirst)
        self.w.broadniba
self.broadnibLast)

        if hasattr(self,
"extraToolsDrawer"):
            self.extraTool
adjustWidthFirstSlid
            self.extraTool
AWFirst, self.extraT
AWLast = self.PFont[
selectSegment[0]].ge
th(self.selectSegmen
            self.extraTool
adjustWidthFirstSlid
extraToolsDrawer.AWF
            self.extraTool
adjustWidthLastSlide
extraToolsDrawer.AWL
        self.scrollViewNS
setNeedsDisplay_(Tru

    def extraToolsDrawe
sender):
        if not hasattr(se
"extraToolsDrawer"):
            self.extraToolsD
Drawer((100, 100), s
preferredEdge="botto
            extraPenTools(se
extraToolsDrawer, se
self.scrollViewNSVie
        self.extraToolsDr
    def penDrawerCallba
        if not hasattr(se
```

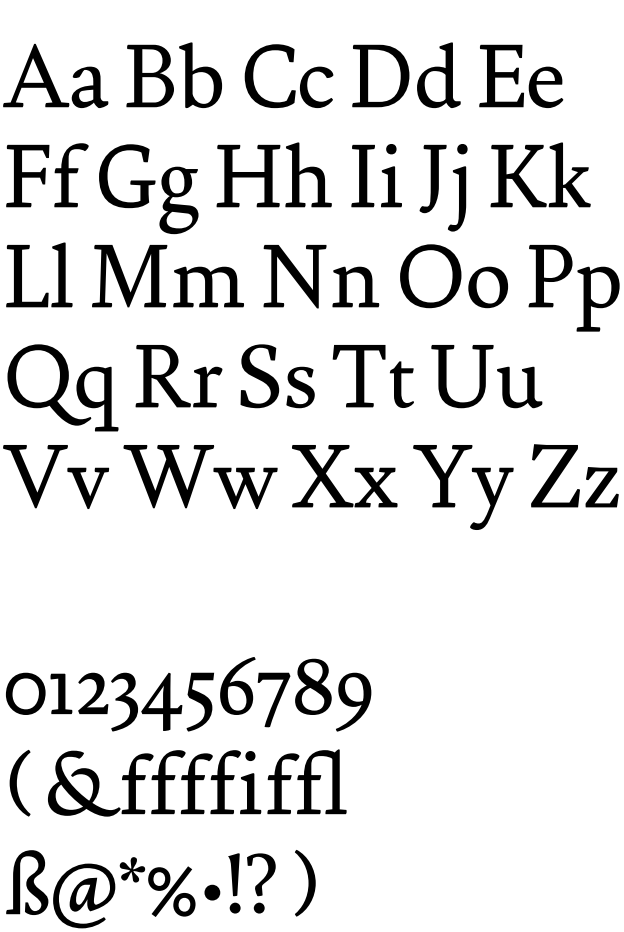
```
ent = (c-1, s-1)
NSView.selectSegment


rst, self.
PFont[self.
gment[0]].
selectSegment[1],

st, self.pointedLast
ame][self.
tSmoothAngle(self.
ointed")

ngle.set("%i" %


rst, self.
PFont[self.
gment[0]].
lf.

.set("%0.1f" % self.

Last.set("%0.1f" %


angle.set("%i" %

angleLast.set("%i" %


,

lsDrawer.
er.show(True)
lsDrawer.
oolsDrawer.
self.Gname][self.
tSmoothAdjustPenWid
t[1])
lsDrawer.
er.set(self.
irst)
lsDrawer.
r.set(self.
ast)
View.
e)

rrCallback(self,


lf,


Drawer =
elf.w,
m")
elf.
lf.pen, self.PFont,
w, self )
awer.toggle()
ck(self, sender):
lf, "Drawer"):
```

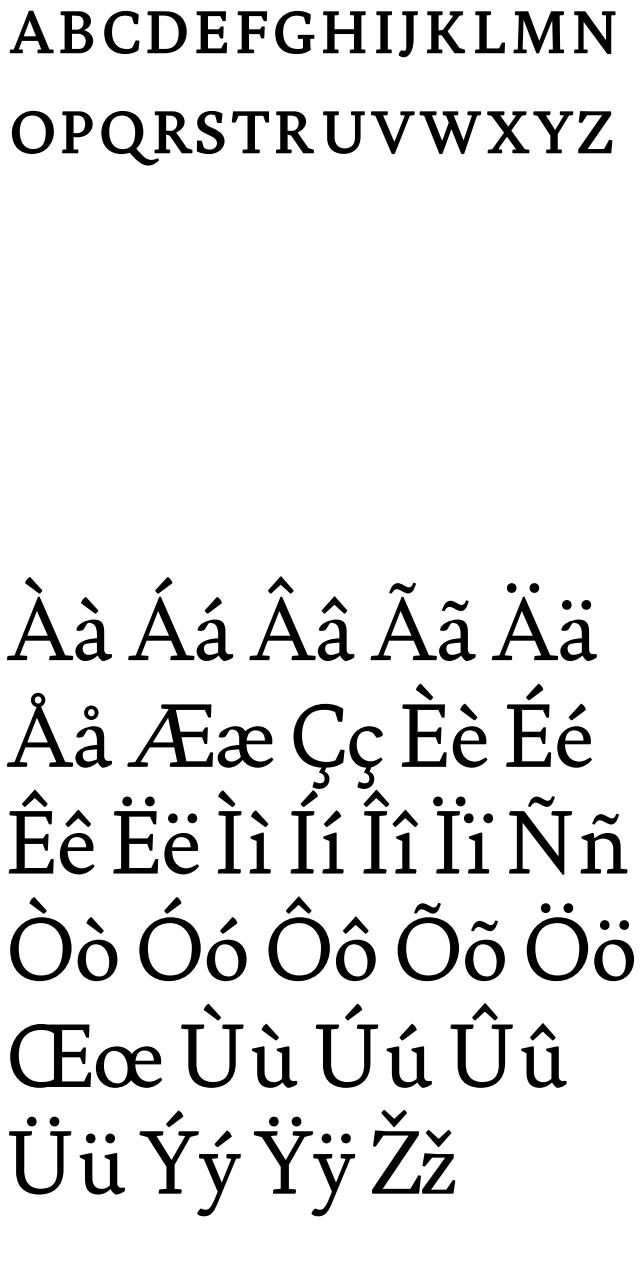

```
      self.Drawer = D
self.w, preferredEdg
      setPens(self.Dra
self.scrollViewNSVie
   self.Drawer.toggl


class interPenAtorWi

 def __init__(self):
   self.PFont = PFon
   self.UFO = None
   self.resolution =

   self.pen = PFPen(
   self.pen.resoluti

   self.w = Window((
"Kalliculator", clos
   self.w.tabs = Tab
-10), titles=["Impor
PenAngle", "Export"]
closeDrawerCallBack)
   self.ImportTab =
   self.PenAngleTab
   self.ExportTab =

   ### ImportTab ###
####################
###########

   self.ImportTab.Im
= Button((10, 20, 15
"Import UFO", callba
ImportUFOButtonCallb
   self.ImportTab.Im
TextBox((180, 22, 30
imported")

   self.ImportTab.Im
= Button((10, 50, 15
"Import PFO", callba
ImportPFOButtonCallb
   self.ImportTab.Im
TextBox((180, 52, 20
imported")
   self.ImportTab.Im
= TextBox((400, 22,
"Resolution")
   self.ImportTab.ma
= Button((400, 50, 1
"Make PFont", callba
makePFontCallback)
   self.ImportTab.re
EditText((490, 20, 5
% self.resolution, c
resolutionCallback)

   ### PreviewTab ##
####################
############
   self.pdfSentence
   #self.PreviewTab.
EditText((10, 10, -1
```


```
rawer((200, 200),
e="right")
awer, onTheFly =
w, pen = self.pen)
e()


ndow(object):


t()

 1

)
on = self.resolution

600, 400),
able = False)
s((10, 10, -10,
t", "Preview", "Set
, callback = self.

self.w.tabs[0]
= self.w.tabs[2]
self.w.tabs[3]

####################
####################


portUFOButton
0, 20),
ck = self.
ack)
portUFOText =
0, 20), "No UFO


portPFOButton
0, 20),
ck = self.
ack)
portPFOText =
0, 20), "No PFO

portResolution
80, 20),

kePFont
50, 20),
ck = self.

solution =
0, 25), "%d"
allback=self.


####################
####################

= ""
textInput =
0, 25), self.
```

```
pdfSentence, callbac
inputTextCallback)

   self.PreviewTab.t
TextEditor((10, 40,
callback=self.textEd
readOnly=False, chec
   self.PreviewTab.p
Button((10, -20, 120
callback = self.prin
sizeStyle="small")
   self.glyphList =
listOfGlyphs()
   self.PenAngleTab.
10, -100, -10), self
ypingSensitivity=Tru
glyphListCallback)
   self.PenAngleTab.
= Button((10, -40, 1
"Export Angle", call
exportAnglesButtonCa
sizeStyle="small")
   self.PenAngleTab.
= Button((10, -70, 1
"Import Angle", call
importAnglesButtonCa
sizeStyle="small")

   ### ExportTab ###
####################
###########


   self.ExportTab.ex
= Button((10, 20, 15
"Export To UFO", cal
exportToUFOCallback)
   self.ExportTab.ex
= Button((10, 50, 15
To UFO bezier", call
exportToUFObezierCal
   self.ExportTab.sh
= Button((400, 20, 1
"Export To PDF", cal
sheetToPDFCallback)
   self.ExportTab.pe
= Button((-120, -30,
20), "pen", callback
penSettingsDrawerCal
   self.w.open()

 def printTextEditor
sender):
   print self.textEd
   for i in self.tex
     if i == "\n":
       print "breakli
 def textEditorCallb

 def closeDrawerCall
   if hasattr(self,
     self.Drawer.clos

 def penSettingsDraw
```

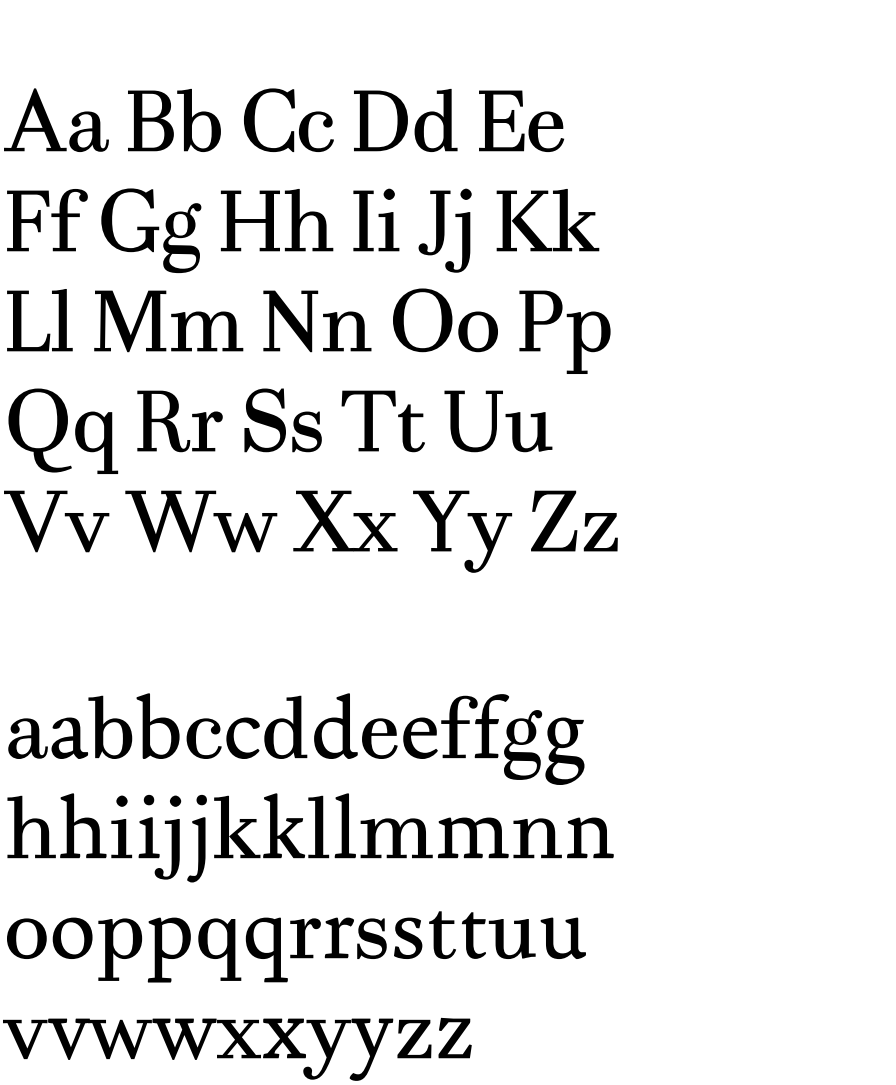
```
k=self.


extEditor =
-10, 200), text="",
itorCallback,
ksSpelling=True)
rintTextEditor =
, 20), "print",
tTextEditorCallback,

self.PFont.

GList = List((150,
.glyphList, enableT
e, callback = self.

exportAnglesButton
20, 20),
back = self.
llback,

importAnglesButton
20, 20),
back = self.
llback,


####################
####################


portToUFO
0, 20),
lback = self.

portToUFObezier
0, 20), "Export
back = self.
lback)
eetToPDF
50, 20),
lback = self.

nSettingsDrawer
 -10,
 = self.
lback)


Callback(self,

itorText
tEditorText:

ne"
ack(self, sender):

Back(self, sender):
"Drawer"):
se()

erCallback(self,
```

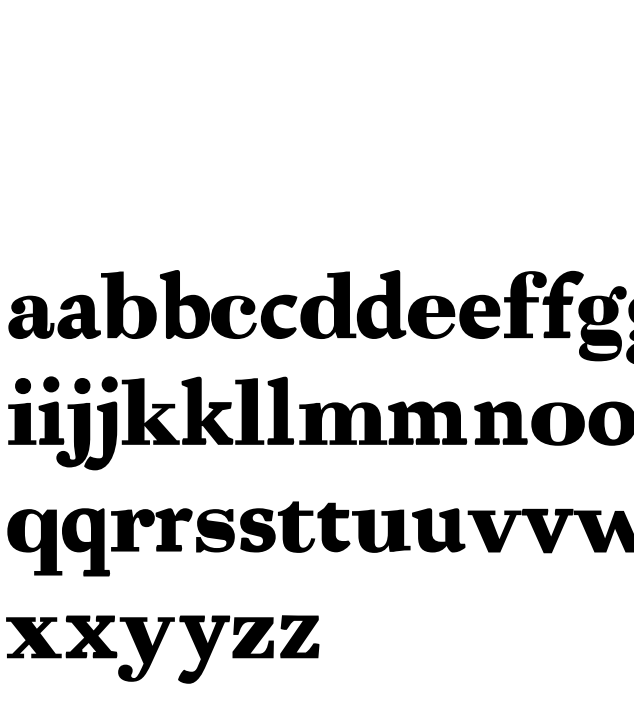





```
sender):
   if not hasattr(se
     self.Drawer = Dr
self.w, preferredEdg
     setPens(self.Dra
pen)

 def importAnglesBut
sender):
   self.PFont.loadAn

 def exportAnglesBut
sender):
   self.PFont.writeA

 def exportToUFObezi
sender):
   self.pen.outputCl
UFOOutputContour
   GlyphList = self.
   #allGlyphs = ""
   # allGlyphs += the
ame[name]
   self.pen.draw(sel
drawInContour = True
= False)

 def exportToUFOCall
   self.pen.outputCl
   GlyphList = self.
   allGlyphs = ""
   for name in Glyph
     allGlyphs += the
ame[name]
   self.pen.draw(sel
 def sheetToPDFCallb
   self.sheet = Shee
w)
   self.sheet.allGly
-100, 70, 32), "All
callback=self.allGly
sizeStyle="small")
   self.sheet.inputT
20, -10, 25), self.p
callback=self.inputT
   self.sheet.button
140, -40, -10, 32),
sheet..", callback=s
sizeStyle="small")
   self.sheet.GOPDFB
-40, 180, 32), "Expo
self.GOPDFButtonPFon
   self.sheet.GOPDFB
= Button((10, -70, 1
"Export Contour", ca
GOPDFButtonContourCa
   self.sheet.open()

 def GOPDFButtonCont
sender):
   self.pen.draw(sel
pdfSentence, drawInC
   self.closeSheet(N
```

```
lf, "Drawer"):
rawer((200, 200),
e="right")
awer, pen = self.


tonCallback(self,

gle()

tonCallback(self,

ngle()

erCallback(self,

ass =

PFont.listOfGlyphs()

eOtherWaypostScriptN

f.PFont, GlyphList,
, NoPostScriptName


back(self, sender):
ass = UFOOutput
PFont.listOfGlyphs()

List:
eOtherWaypostScriptN

f.PFont, allGlyphs)
ack(self, sender):
t((400, 150), self.

phs = Button((10,
Glyphs",
phsCallback,

ext = EditText((10,
dfSentence,
extCallback)
 = Button((-
"Close this
elf.closeSheet,

utton = Button((10,
rt", callback =
tCallback)
uttonContour
80, 32),
llback = self.
llback)


ourCallback(self,

f.PFont, self.
ontour = True)
one)
```






```
 def allGlyphsCallba
   GlyphList = self.
   newList = ""
   for name in Glyph
     newList += theOt
e[name]
   self.sheet.inputT
   self.pdfSentence
   self.pdfSentence

 def GOPDFButtonPFon
sender):
   self.pen.outputCl
   self.pen.draw(sel
pdfSentence)
   self.closeSheet(N

 def closeSheet(self
   self.sheet.close(

   if self.PFont.fam
     self.PFont.write

 def resolutionCallb
   self.resolution =
   self.pen.resoluti

 def makePFontCallba
   if self.UFO:
     self.PFont = PFo
     filename = self.
" "+ self.PFont.styl
     self.ImportTab.I
filename)
     self.glyphList =
listOfGlyphs()
     self.PenAngleTab
glyphList)

 def ImportUFOButton
sender):
   self.PFont = PFon
   self.UFO = self.P
   if not self.UFO:
     return
   self.ImportTab.Im
set(filename)


 def ImportPFOButton
sender):
   if self.PFont:
     if self.PFont.fa
UFO.info.familyName:
       answer = AskYe
made a PFont, import
title="InterPenAtor"
       if answer <= 0
         return
   self.PFont = self
     self.PFont = PFo
     self.glyphList =
listOfGlyphs()
     self.PenAngleTab
```

BRUSSEL De Brusselse politie heeft vrijdagochtend rond twee uur film maker Jan Bucquoy uit het befaamde cafe de dolle mol gezet. Volgens de filmmaker sloeg de politie de boel kort en klein, **volgens de woordvoerder van de politie verliep de uitzetting in aanwezigheid van een deurwaarder in alle rust.De contro) versiele kunstenaar en revolutionair Bucquoy had het beruchte cafe dolle mol in de spoormak) ersstraat eind april gekraakt en heropend uit protest tegen de jarenlange leegstand. op vijf mei besliste de kortgedingrech) ter echter dat bucquoy uit het pand mocht gezet worden dat gebeurde volgens bucquoy bij ver stek en zonder medeweten zijn advocaten hij wachtte sinds**

begin juli op de uitzetting uit het pand cafe dolle mol opende in

de jaren ’60 de deuren en was de-cennia lang de ver-zamelplaats voor

wereldverbeteraars en creatieve dromers sinds oktober staat het leeg tijdens de heropening eind april daagde zondagavond heel wat volk op waaronder enkele bekende figuren zoals johan ver-minnen en de gentse folkzanger

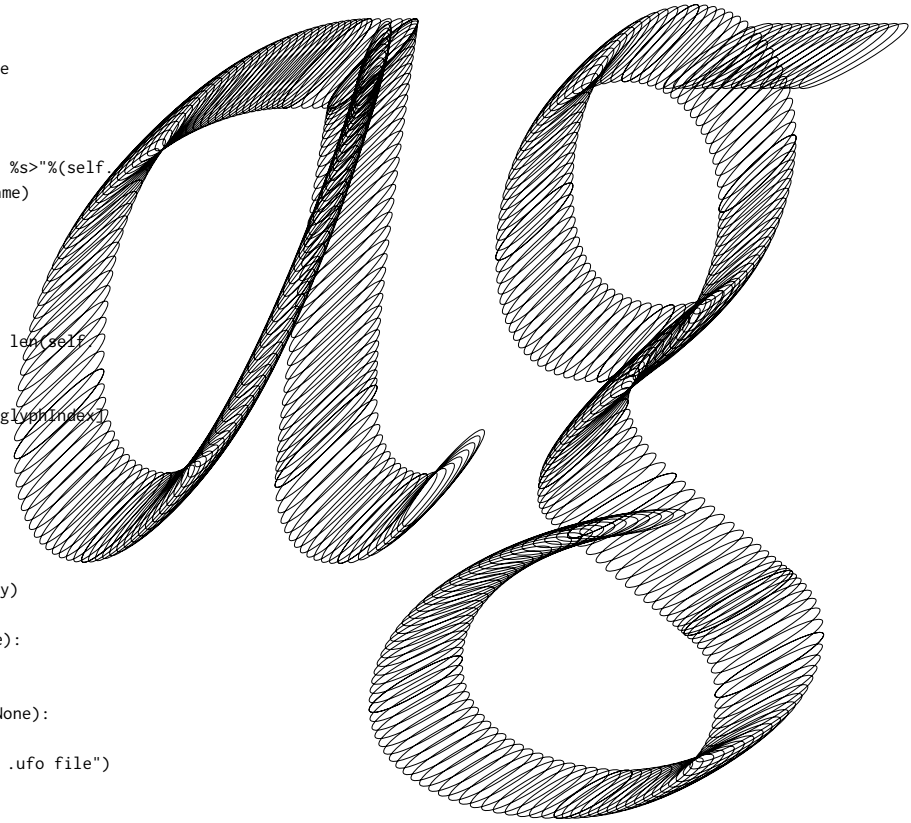
walter de buck.om het cafe te redden. hoopte bucquoy op een finan-ciele tussenkomst van vlaams minister van brussel bert anciaux. de dolle mol behoort tot het cultureel erfgoed van de stad aldus buc-quoy toen. vlaamse steun moet het mogelijk maken het pand aan te kopen en er de nodige investeringen in te doen sinds het leegstaat verkrot het immers alsmaar verder.

```
glyphList)
    self.ImportTab.
set("Nothing Importe
    return

  filename = self.P
" + self.PFont.style
  self.glyphList =
ListOfGlyphs()
  self.PenAngleTab.
glyphList)
  self.ImportTab.Im
set(filename)

 def glyphListCallba
  name = self.glyph
getSelection()[0]]
  setAngle(name, se

##################

import sys
import os.path
from robofab.interfa
import GetFolder, Ge
from robofab.world i
RFont
from cPickle import
from xmlWriter impor
from fontTools.pens.
AbstractPen
from bezierToPoints
bezierToPoints


class PFontPen(Abstr

 def __init__(self,
resolution):
  self.glyph = sGly
  self.resolution =
  self.contour = No
  self.lastPoint =

 def moveTo(self, pt
  # new contour
  self.contour = se
newContour()

 def lineTo(self, pt
  Plist = bezierToP
resolution, self.las
  self.contour.addP
"newSeg")#hier een b
  self.lastPoint =

 def curveTo(self, p
  Plist = bezierToP
resolution, self.las
pt3)
  self.contour.addP
= "newSeg")#hier een
```

de arbitragecommissie van de wereldvoetbal bond fifa stelt zich solidair op tegenover de engelse scheidsrechter graham poll die flaterde donderdag op het wk nadat hij de kroaat josip simunic in het duel **australie kroatie vergat uit te sluiten na een tweede gele kaart maar dit wel deed toen hij de speler een derde keer geel toonde uberhaupt nadat hij had afgefloten. in zijn carriere heeft deze ervaren**

scheidsrechter nooit eerder zo n fout gemaakt, luidt het zaterdag in een communique, de commissie heeft de fout erkend, en stelde bovendien het gebrek aan reactie vast van de andere betrokken **officials. vrijdag deed graham poll zijn verhaal bij de commissie. hij legde toen uit dat hij voor de tweede gele kaart het nummer drie van australie graig moore) had opgeschreven, in**

```
addListTo segment
  self.lastPoint =
 def qCurveTo(self,
  pass

 def closePath(self)
  pass
 def endPath(self):
  pass

 def addComponent(se
transformation):


class PFont(dict):
 def __init__(self):
  self.familyName =
  self.styleName =
  self.xHeight = No
  self.ascender = N
  self.descender =
  self.resolution =
  self.glyphIndex =

 def __repr__(self):
  return "<PointFon
familyName, self.sty

 def __iter__(self):
  return self

 def next(self):
  if self.glyphInde
keys()):
   raise StopIterat
  k = self.keys()[s
  self.glyphIndex +
  return self[k]

 def get(self, key):
  if self.has_key(k
   return self[key]
  self[key] = aGlyp
  return self[key]
 def newGlyph(self,
  return self.get(n

 def loadUFO(self, p
  if not p:
   p = GetFile("Cho
   if not p:
    return
  return OpenFont(p

  self.resolution =
  if not RobofabFon
   return
  self.familyName =
familyName
  self.styleName =
styleName
  self.xHeight = Ro
xHeight
```



..... The output paper

With a pen you write on paper, with these digital pens it is possible to write in any possible digital format; on screen, in a .pdf, in a font.

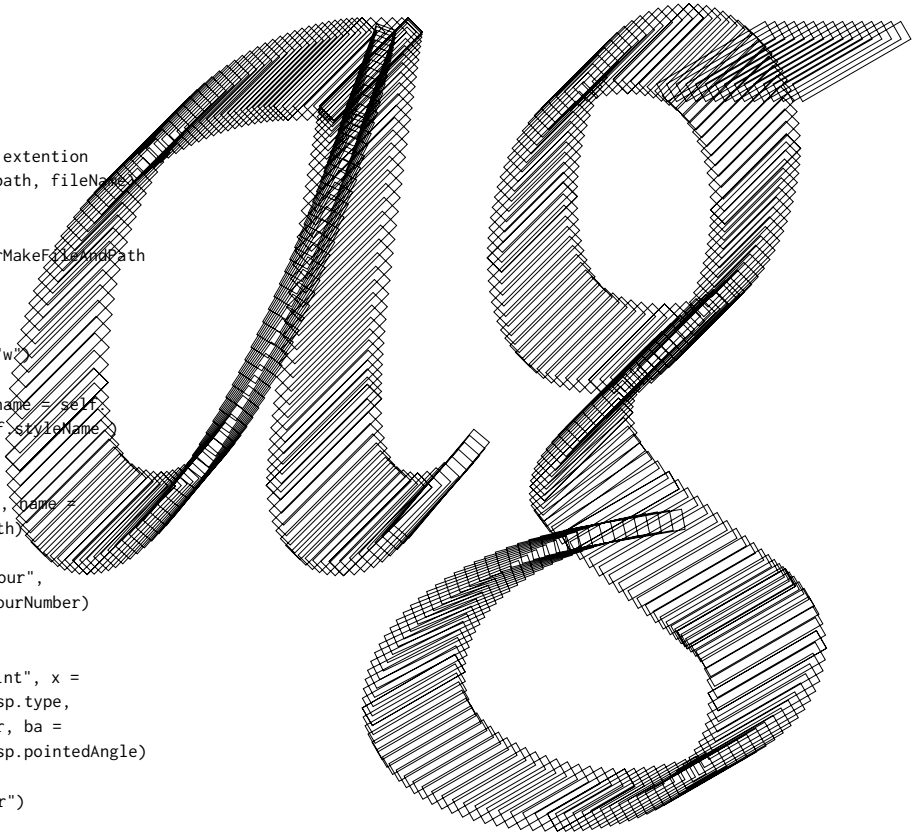
The first action, and the fastest way to do so, is to define on each point of the skeleton the shape of the pen, oval or rectangle in the dimensions set for that pen. This result gives nice shades: a lot of overlapping ovals or rectangles. This gives the glyphs a third dimension; they are not flat any more.

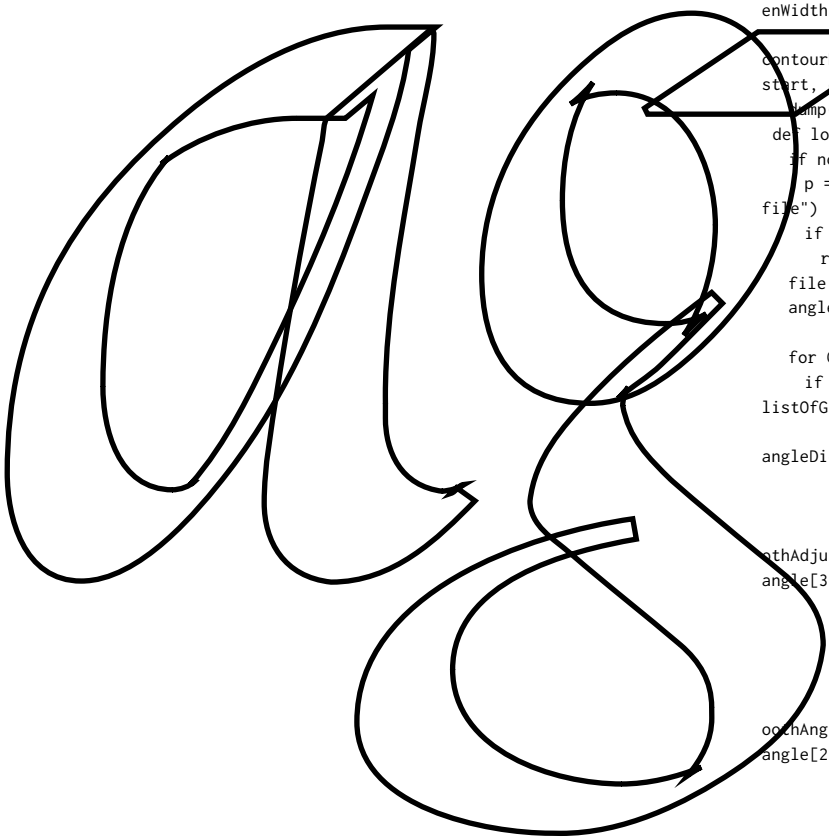
This was actually not the result I was looking for. I wanted to have a clean outline. I came up with the idea to put every extreme of each oval or rectangle in a list. This list is the contour of a glyph. For a rectangle, this is quite easy because there are only four possible points for the extreme. For an oval, this is more complex. The extreme points of an oval depend on the angle of the point where the oval has been drawn on the skeleton. So the skeleton angle determines the extremes of an oval. There are only two points on an oval where the intersection with the skeleton angle has only one value. These two points are the extremes of that oval. This has to be calculated for every oval.

Followed, this list is put into a 'bezier-maker'. This piece of code generates from a list of points nice beziers. The 'bezier-maker' is made especially for type design. It calculates the anchor points, only on the extremes of the curve. This implies that the handles, which control the curve, are not over 90°. The best way to generate beziers out of a list of points is to ensure that the angles of the handles

```
   self.ascender = R
ascender
   self.descender =
descender
   self.glyphSet = G
join(RobofabFont.pat
   for name in self.
    sglyph = aGlyph(
RobofabFont[name].wi
    pen = PointPen(s
    glyph.draw(pen)
    self[name] = sgl


 def getFolderMakeFi
path, extention):
   if not path:
    path = GetFolder
    if not path:
   fn = self.familyN
   if not fn:
    fn = "NoFamilyNa
   sn = self.styleNa
   if not sn:
    sn = "NoStyleNam
   fileName = fn + s
   path = os.path.jo
   return path

   path = self.getFo
(p, ".pfoXML")
   if not path:
    return
   outPut = open(pat
   w = XMLWriter(pat
   w.begintag("PFont
familyName, style =
   w.newline()
   for sg in self:
    w.begintag("PGly
sg.name, width = sg.
    w.newline()
     w.begintag("PC
contourNumber = sc.c
     w.newline()
     for sp in sc:
      w.simpletag(
sp.x, y = sp.y, type
segNr = sp.segmentNu
sp.broadnibAngle, pa
      w.newline()
     w.endtag("PCon
     w.newline()
    w.endtag("PGlyph
    w.newline()
   w.endtag("PFont")
 def writeAngle(self
   path = self.getFo
(p, ".pfoAngle")
   if not path:
    return
   file = open(path,
   angleDict = {}
   for g in self:
```

are fixed. The script then only calculates the length of these handles until the error parameter is reached. This parameter ensures that the distance of a point in that list to the generated bezier is smaller than the error. When this distance is bigger than the error, the script runs again and again until the error parameter is reached. The output exists now out of nice bezier curves for on screen, .pdf and in a font.





```
    angleDict[g.name
    for c in g:
      angleDict[g.na
contourNumber] = []
        start, end =
c.getSmoothAngle(seg
        angleDict[g.
contourNumber].appen
start, end, None))

        start, end =
c.getSmoothAngle(seg
        startPres, e
c.getSmoothPressure(
        angleDict[g.
contourNumber].appen
start, end, startPre

        start, end =
enWidth(seg)
        angleDict[g.
contourNumber].appen
start, end ))
    dump(angleDict, f
  def loadAngle(self,
    if not p:
      p = GetFile("Cho
file")
      if not p:
        return
    file = open(p, "r
    angleDict = load(

    for GlyphName in
      if GlyphName  in
listOfGlyphs():
        for angle in
angleDict[GlyphName]
          if angle[1]
            try:
              self[Gl
othAdjustPenWidth(an
angle[3])
            except:
              pass
          else:
            try:
              self[Gl
oothAnglePenInSeg(an
angle[2], angle[3])
            except:
            if angle[
              try:
                self[G
moothPressureInSeg(a
angle[5])
              except:
                pass

  def writePFont(self
    path = self.getFo
(p, ".pfo")
    if not path:
      return
```

e] = {}

ame][c.

, "broadnib")
name][c.
d((seg, "broadnib",

, "pointed")
ndPres =
seg)
name][c.
d((seg, "pointed",
s, endPres))

c.getSmoothAdjustP

name][c.
d((seg, "width",

ile)
p = None):

oose .pfoAngle

b")
file)

angleDict:
n self.

[c]:
] == "width":

yphName][c].setSmo
gle[0], angle[2],

yphName][c].setSm
gle[0], angle[1],

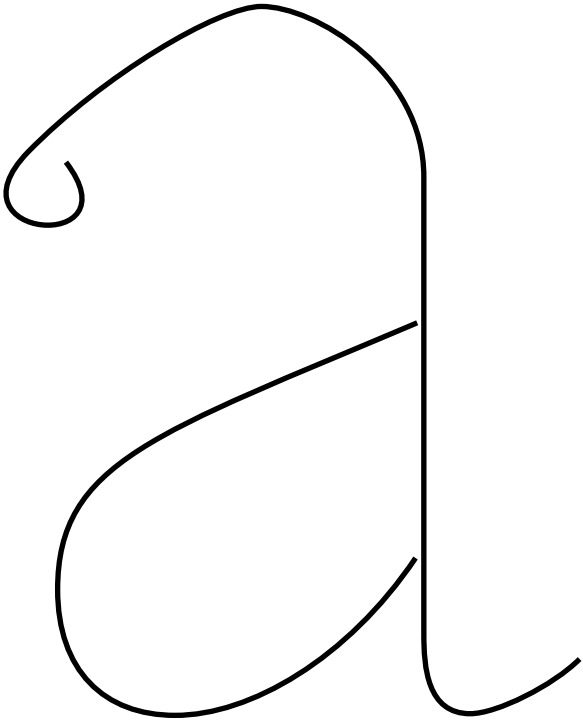
4] != None:

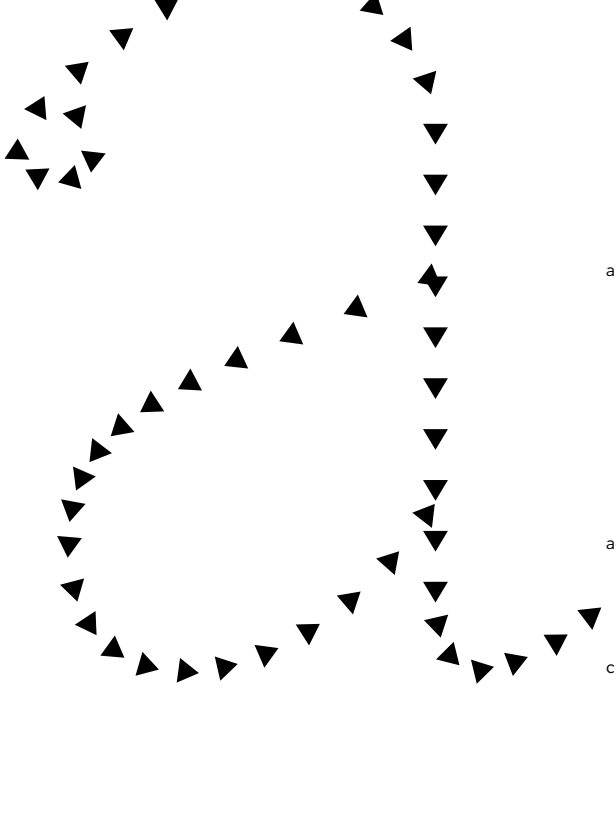
GlyphName][c].setS
ngle[0], angle[4],

, p = None):
lderMakeFileAndPath

Review

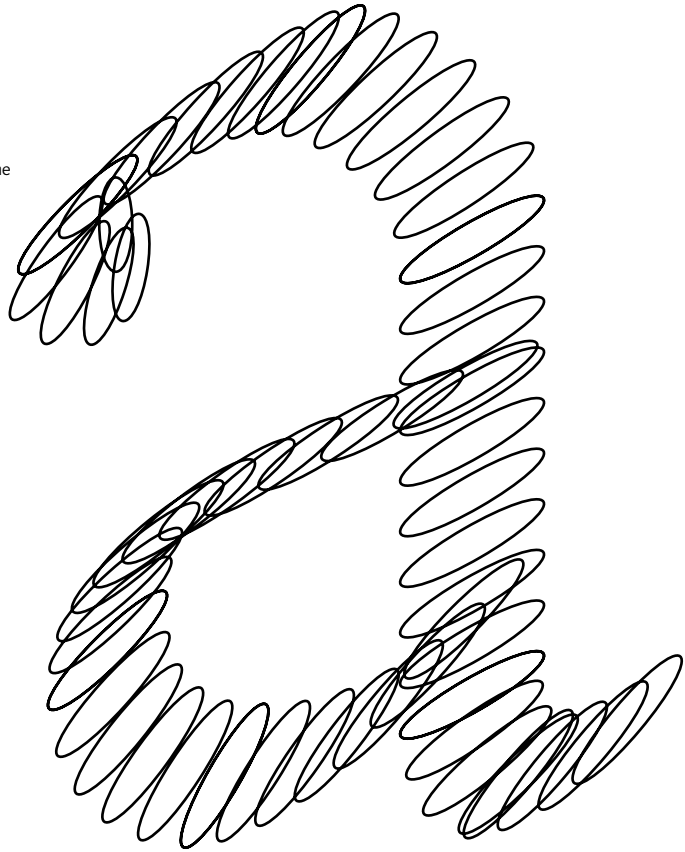
SKELETON — PFO — OVALAATJES — EXTREMES — LINETOS — BEZIERS

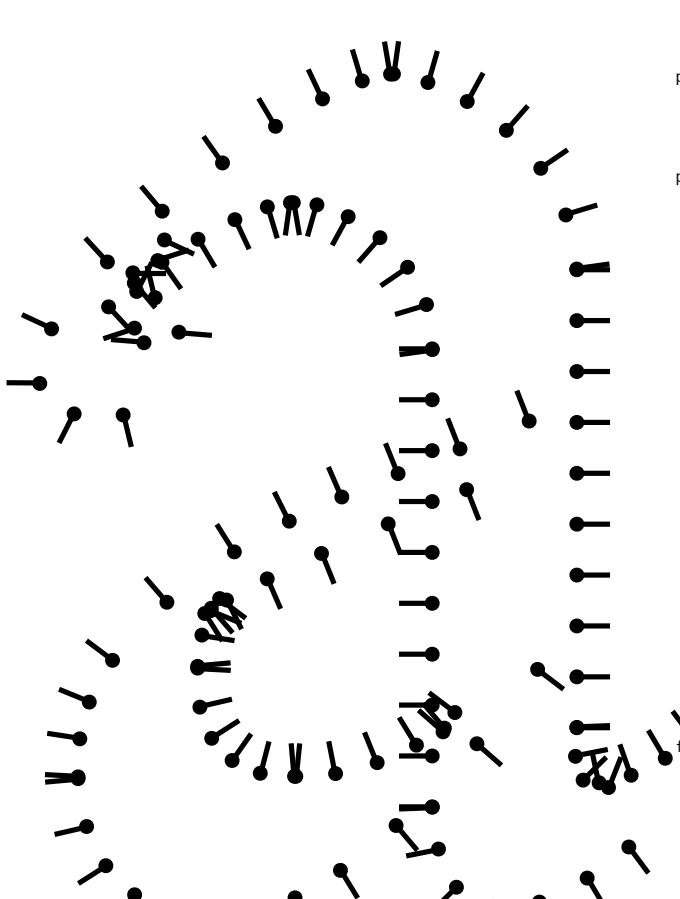




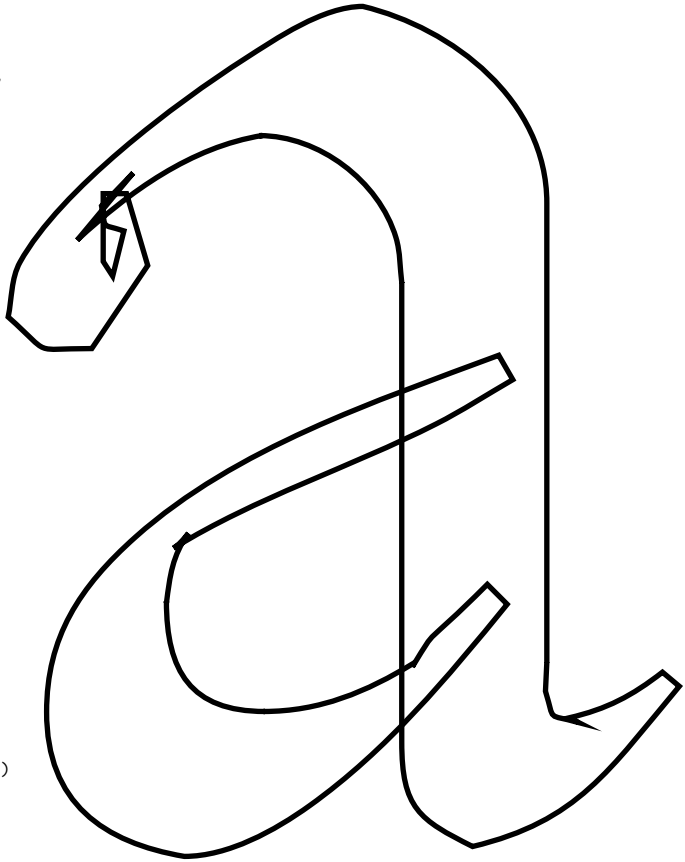
```
  dump(self, file)

 def loadPFont(self,
   if not p:
     p = GetFile("Cho
     if not p:
       return
   file = open(p, "r
   return load(file)

   list = []
   for g in self:
     list.append(g.na
   list.sort()
   return list

class aGlyph(list):
 def __init__(self,
None):
   self.name = name
   self.width = widt
   self.glyphShapes
 def __repr__(self):
   return "<Frederik

 def newContour(self
   s = aContour()
   self.append(s)
   s.contourNumber =
   self.contourIndex
   return self[-1]
 def setGlobalAngleP
angle):
   for c in self:
     for p in c:
       if pen == "bro
         p.broadnibAn
       elif pen == "p
         p.pointedAng
 def setGlobalPressu
   for c in self:
     for p in c:
 def setGlobalAdjust
adjust):
   for c in self:
     for p in c:
       p.adjustPenWid

class aContour(list)
 def __init__(self):
   self.contourNumbe
   self.segmentIndex
   self.pointIndex =
   pl = []
   for p in self:
     pl.append((p.x,
   return pl
 def addPoint(self,
type = None, ba = 30
   s = aPoint(x,  y)
   self.append(s)
   if not type:
     if self.pointInd
       type = "line"
```

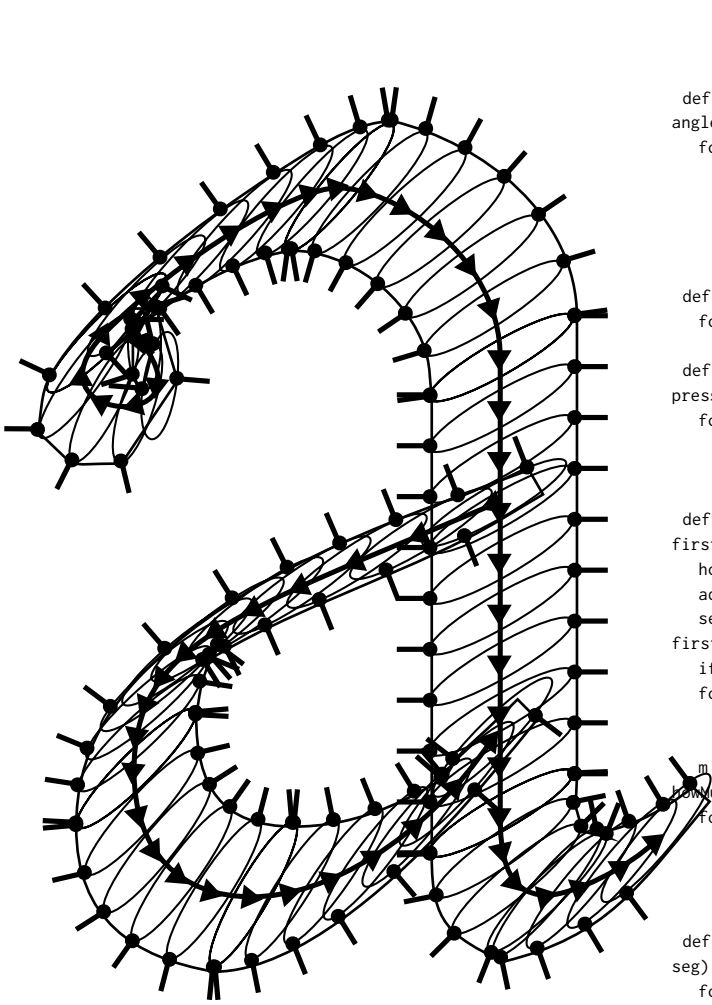
```
 p = None):

oose .pfo file")


b")




ame)



 name, width =


h
= []

Glyph %s>"%self.name

"):


 self.contourIndex
 += 1

en(self, pen,


adnib":
gle = angle
ointed":
le = angle
re(self, pres):


 Width(self,


dth = adjust

:

r = 0
 = -1
 0


 p.y))

 x, y, sg = None,
 , pa = 90):


dex:
```





```
      type = "move"
    if sg == "newSeg"
      self.segmentInde

    s.type = type
    s.segmentNumber =
    s.broadnibAngle =
    s.pointedAngle =
    s.pointIndex = se
    self.pointIndex +

  def addPointList(se
type = None, ba = 30
    for x, y in l:
      if sg == "newSeg
        self.addPoint(
pa)
        sg = None
      else:
        self.addPoint(
pa)

  def getSmoothAngle(
    for p in self:
      if p.segmentNumb
        if pen == "bro
          if not first
            first = p.b
          last = p.bro
        if pen == "poi
          if not first
            first = p.p
          last = p.poi


  def getSmoothPressu
    startPres = None
    for p in self:
      if p.segmentNumb
        if not startPr
          startPres =
        endPres = p.pr
    return float(star
float(endPres)

    howMuchPointsInSe
    addPres = 0.0
    self.setPressureI
    if startPres == e
      return
    for p in self:
      if p.segmentNumb
        howMuchPointsI
    m = (endPres - st
howMuchPointsInSeg
    for p in self:
        p.pressure +=
        addPres += m

  def setSmoothAngleP
pen, firstAngle, las
    howMuchPointsInSe
    addAngle = 0.0
```

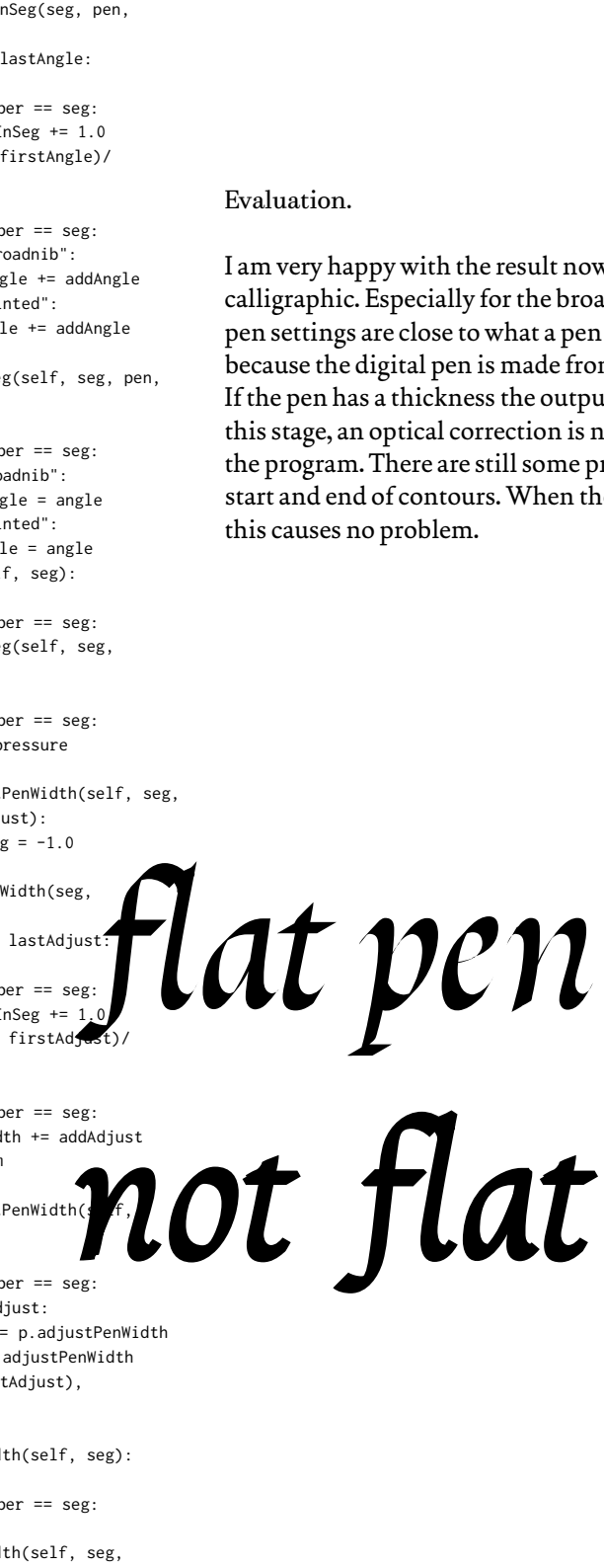


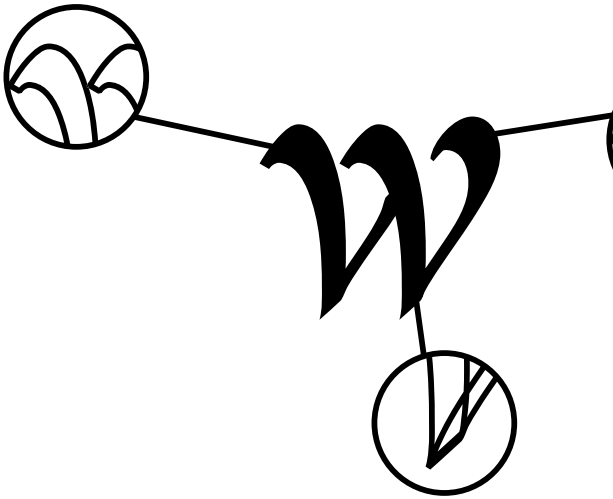


```
   self.setAnglePenI
firstAngle)
   if firstAngle ==
     return
     if p.segmentNumb
      howMuchPointsI
   m = (lastAngle -
howMuchPointsInSeg
   for p in self:
     if p.segmentNumb
       if pen ==  "br
         p.broadnibAn
       if pen == "poi
         p.pointedAng
       addAngle += m
 def setAnglePenInSe
angle):
   for p in self:
     if p.segmentNumb
       if pen == "bro
         p.broadnibAn
       if pen == "poi
         p.pointedAng
 def getPressure(sel
   for p in self:
     if p.segmentNumb
 def setPressureInSe
pressure):
   for p in self:
     if p.segmentNumb
       p.pressure = p

 def setSmoothAdjust
firstAdjust, lastAdj
   howMuchPointsInSe
   addAdjust = 0.0
   self.setAdjustPen
firstAdjust)
   if firstAdjust ==
   for p in self:
     if p.segmentNumb
       howMuchPointsI
   m = (lastAdjust -
howMuchPointsInSeg
   for p in self:
     if p.segmentNumb
       p.adjustPenWid
       addAdjust += m

 def getSmoothAdjust
seg):
   for p in self:
     if p.segmentNumb
       if not startAd
         startAdjust
       endAdjust = p.
   return float(star
float(endAdjust)

 def getAdjustPenWid
   for p in self:
     if p.segmentNumb

 def setAdjustPenWid
```

Evaluation.

I am very happy with the result now. It looks very calligraphic. Especially for the broad nib pen when the pen settings are close to what a pen is: flat. It works good because the digital pen is made from the idea of a real pen. If the pen has a thickness the output is less balanced. At this stage, an optical correction is not yet embedded into the program. There are still some problems to solve at the start and end of contours. When the pen applied is flat, this causes no problem.

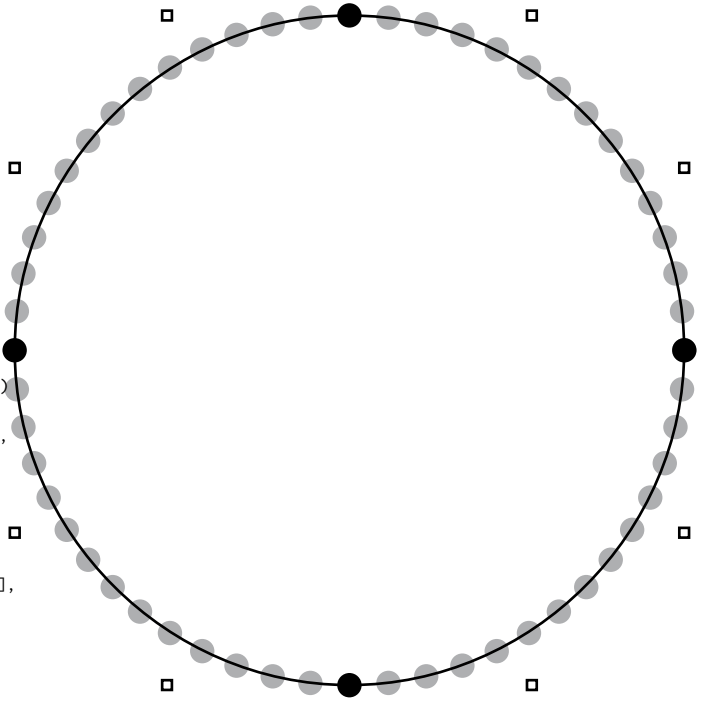




```
adjustWidth):
   for p in self:
     if p.segmentNumb
       p.adjustPenWid

class aPoint(list):
 def __init__(self,
   self.x = x
   self.y = y
   self.segmentNumbe
   self.broadnibAngl
   self.pointedAngle
   self.pointIndex =
   self.pressure = 5
   self.adjustPenWid



##################
from math import sqr
#calculating a bezie
points

#input
#the resolution - th
bezier (for a curve)
#the resolution - st
of a line


def pointOnACurve(x1
cx2, cy2, x2, y2, va
 dx = x1
 bx = (cx2 - cx1) *
 ax = x2 - dx - cx -

 dy = y1
 cy = (cy1 - dy) * 3
 by = (cy2 - cy1) *
 ay = y2 - dy - cy -

 mx = ax*(value)**3
cx*(value) + dx
 my = ay*(value)**3
cy*(value) + dy
 return mx, my

def distance(x, y, x
 return sqrt((x1 - x

def bezierToPoints(r
hp2 = None, p2 = Non

 default = 10.0
 if not hp2: ### if
but a lineTo
   dist = distance(p
hp1[1])
   x1, y1 = p1
   cx1, cy1 = hp1
   cx2, cy2 = hp2
   x2, y2 = p2
   dist = 0
   oldx, oldy = x1,
```

For the pointed pen it is more difficult to set the correct angles, the right pressure, in such a way that the curves generated are fluid. The cause lies in the many parameters that control this pen. The pointed pen does not have this problem when the pen has a thickness like the broad nib. This is due to the qualities of the pointed pen and to the way the program interprets the parameters. The width of the pen is the same as the thickest parts and the thickness of the pen is equal to the thinnest part in a glyph. So when the pen is flat, the thinnest parts are very thin and the contrast is very high.

It is possible to export it to a font file, .pdf and see it on screen. It is very nice that it works to export the glyphs as beziers. The 'bezier-maker' is a very strong script. The input is a list of points and it returns as a list of beziers. I did not imagine I could get so far.



But the power of this tool lies in the skeleton based on calligraphy, and in the combination of the skeleton with the complexity of the digital pen. The skeleton is a flexible proportion definition. This means that the skeleton contains information about the proportion of a glyph. This can be transformed in any direction. Afterwards a pen applied is following these proportions. The pen can have any shape or dimensions. Because the pen is disconnected from the skeleton it can generate pen drawings from any skeleton, irrespective of how the skeleton is turning or bending, the pen will just follow this and calculate the contrast on the right spots.

This program is not generating ready made typefaces, but it is close. It is a tool to generate different contrasts from the same or from several skeletons. It can help type designers because the output has a correct contrast, which is set in the digital pen. It is an intermediate to research what happens when there is a higher, lower or inverted contrast, when the pen is wider or smaller, when the shape is something else.

```
   d = 0.0
   while d <= defaul
calculates the dista
    newx, newy = poi
cx1, cy1, cx2, cy2,
    dist += distance
newy)
    d += 1

 PointsOnCurve = rou
default*(dist*1/defa

 #the amount of poin
relation to the defa
 #default = for a di
points aqual to 100
ammount of points as
 #when the distance
is calculated you ca
PointOnCurve

 #print "aantal punt
PointsOnCurve
 t = 0.0
 while t <= PointsOn
   if not hp2:
    value =  t/Point
    newx = p1[0] + (
value
    newy = p1[1] + (
value
   else:
    value = t/Points
eventueel nog een kw
gevalletje
    newx, newy = poi
cx1, cy1, cx2, cy2,
   listOfPoints.appe


 return listOfPoints


##################


from math import sqr
acos, pi, tan, radia
from random import r
from AppKit import N
from postScriptNames
import postScriptNam
theOtherWaypostScrip
from PFOContour impo

from PFoutPut import


class PFPen:
 def __init__(self,

   self.broadnibAngl
```

```
t: #### this
nce of one bezier
intOnACurve(x1, y1,
x2, y2, d/default)
e(oldx, oldy, newx,


nd(resolution/
ult)) + 1

ts on a curve in
ult.
stance between to
it sets the same
 the resolution
of one bezier
n calculate the


en =",


Curve:

tsOnCurve
(hp1[0] - p1[0])*

(hp1[1] - p1[1])*


sOnCurve###
ardraatisch

intOnACurve(x1, y1,
x2, y2, value)
nd((newx, newy))


t, cos, sin, asin,
ns
andint, random
SBezierPath

e,
tName
rt convertToBezier

 *



anOtherPen = None):

e = None ###inside
```

. . . . . What can be improved

There are some things I did not solve until now. When a contour end or starts, there are still some problems. Because I have to add a point in the contour list at the beginning or ending. It is in some cases difficult to make a choice which points has to be in the list because the pen turns and turns around a curved skeleton.

This unsolved problem causes a bigger problem because the list of points is then not in the right order any more. So I have to write a 'clean-up-point-list' script, which is also removing the overlap. These problems are visible in the return of the 'bezier-maker', the list of beziers.

I have to rebuild the whole program because there are some structural problems. If I want to add a new function to the program I have to change too many aspects to make it work. You can compare it to a small house. On that house I built new floors, terraces, removed rooms and made others rooms bigger. At a certain point it is better to demolish the whole building and to start over again with a bigger house as basis, where there is space foreseen to install later for example an elevator.

. . . . . What is missing

A preview of a string of characters next to each other.

It has to able to work with multiple fonts at the same time and also to view them next to each other.

A skeleton editor to adjust the skeleton inside the program. It must also be able to load several skeletons and interpolate between them.

It should be possible to add tags to a glyph, a contour, a point and make a tag-editor where these tags can be changed. In a font, several glyphs have almost the same rounding, stems, serifs. If they can be tagged,it works faster to adjust the rounding, the stems, the serifs for the whole font.

Because the 'bezier-maker' works very well it must be possible not to start from a skeleton but from a scanned image. It can calculate the skeleton. An other side project with the same 'bezier-maker' is to develop an auto-tracer specially designed for typedesign.

```
    self.pointedAngle
    self.skeletAngle
    self.pressure = N
    self.noContrast =
    self.adjustPenWid
    self.smoothConnec

    self.drawInContou
    self.bezierError
    self.contourLeft
    self.contourRight

    self.resolution =
    self.shape = "ova
    self.interPenatio
    self.randomness =
    self.minExtrem =
    self.maxExtrem =
    self.h = 10.

    self.firstPoint =
    self.lastPoint =

    self.selectSegmen

    self.outputClass

    if anOtherPen:
      self.drawInConto
drawInContour
      self.drawInBezie
drawInBeziers
      self.contourLeft
contourLeft
      self.contourRigh
contourRight
      self.shape = anO
      self.interPenati
interPenation
      self.randomness
randomness
      self.noContrast
noContrast
      self.w = anOther
      self.h = anOther
      self.outputClass
outputClass

  def draw(self, PFon
drawInContour = Fals
= True):
    self.drawInContou
    if self.shape ==
      drawDef = self.
    elif self.shape =
      drawDef = self.

    segNr = -1
    shapeList = []


    for Gname in PFGl

      if NoPostScriptN
```

= None
= None
one
0.0
th = None
tion = False

r = False
= 4.
= []
= []

1.0
l"
n = 0.0
0
0.0
1.0

False
False

t = (10000, 10000)

= None

our = anOtherPen.

ers = anOtherPen.

t = anOtherPen.

nt = anOtherPen.

OtherPen.shape
ion = anOtherPen.

= anOtherPen.

= anOtherPen.

rPen.w
rPen.h
s = anOtherPen.

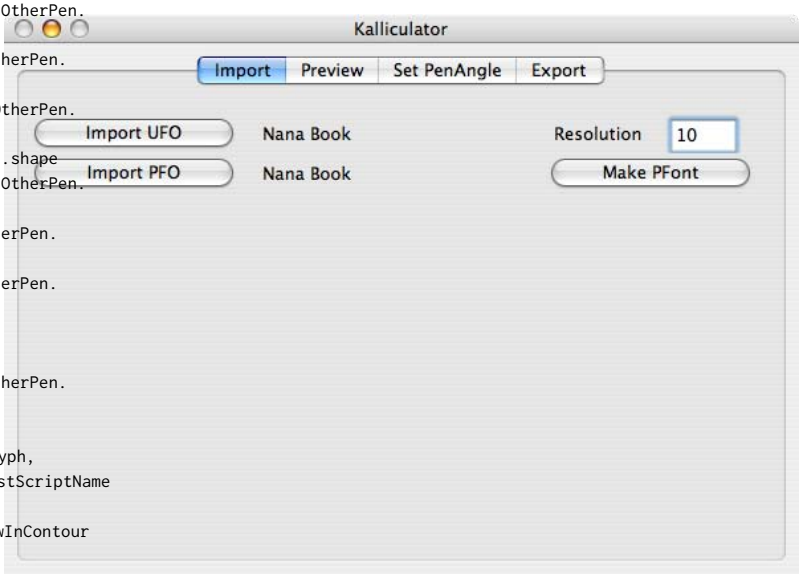
t, PFGlyph,
e, NoPostScriptName

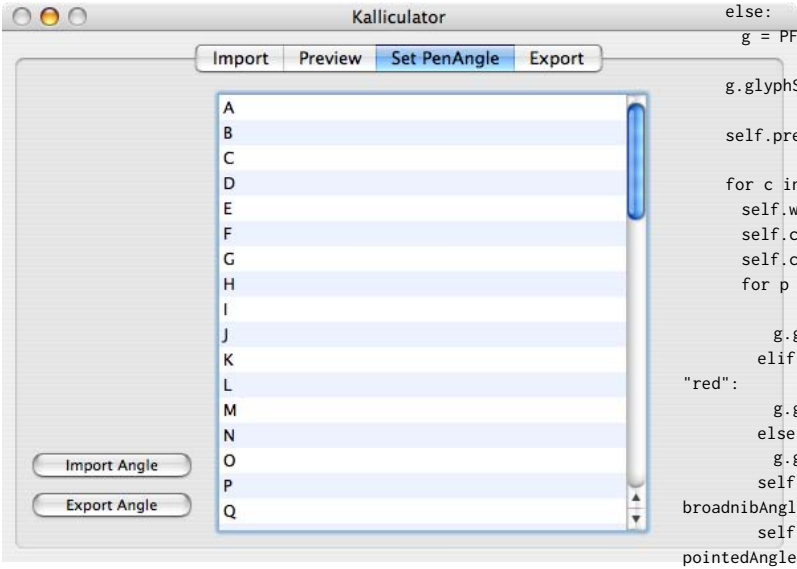
r = drawInContour
"oval":
oval
= "rect":
rect

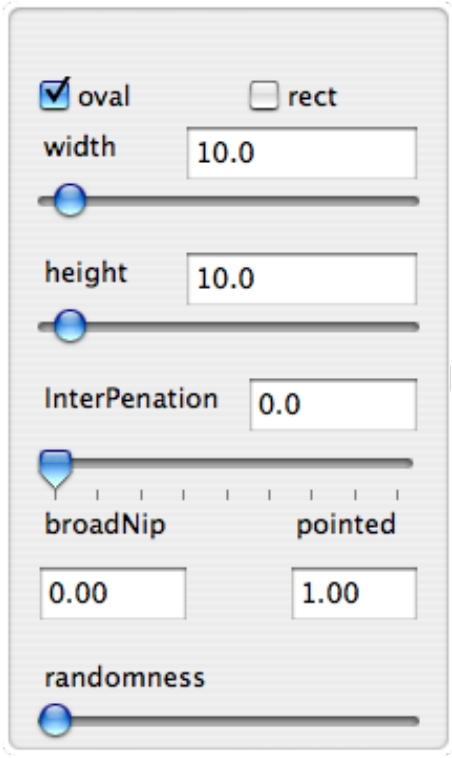
yph:

Name:

Screenshots

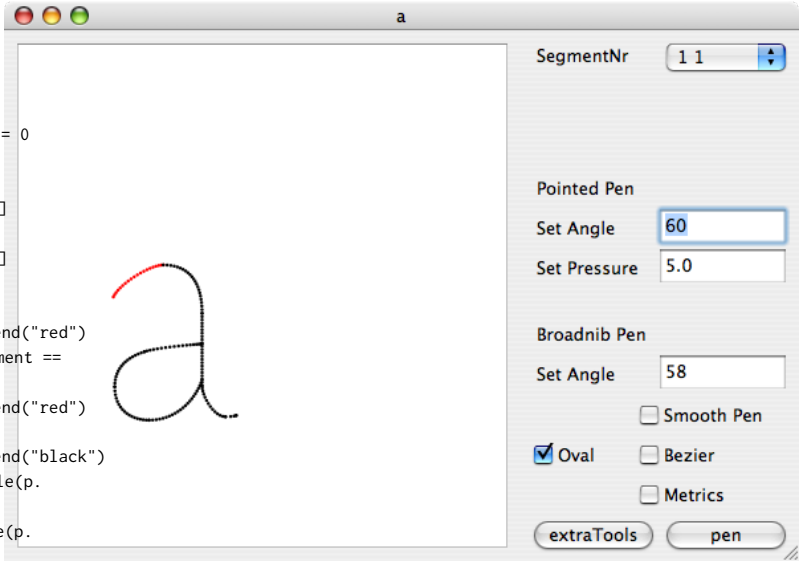




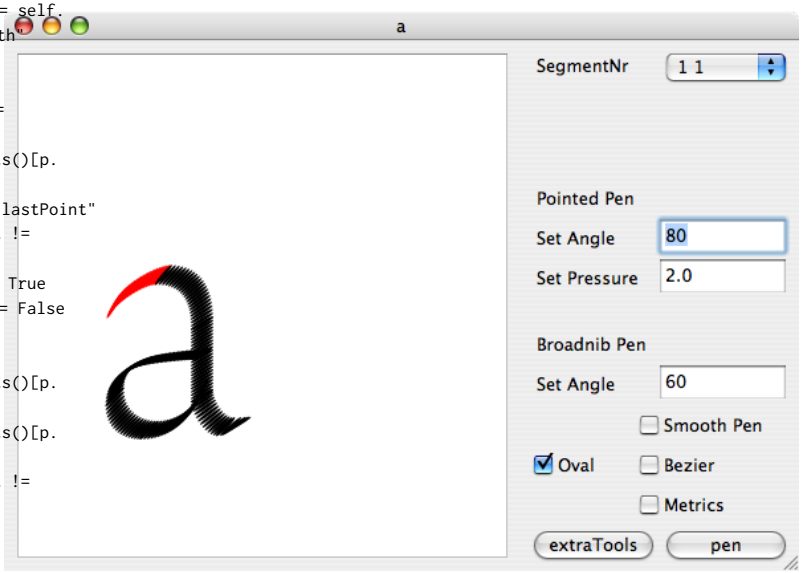



```
      g = PFont[post
##### is nog working
longer than 1 karakt
    else:
      g = PFont[Gnam

    g.glyphShapes =

    self.presSegment

    for c in g:
      self.wholeCont
      self.contourLe
      self.contourRi
      for p in c:

        g.glyphShap
      elif self.se
"red":
        g.glyphShap
      else:
        g.glyphShap
      self.setBroa
broadnibAngle)
      self.setPoin
pointedAngle)
      self.pressur
      self.adjustP
p.adjustPenWidth

        prevp = c.g
pointIndex]
        nextp = c.g
pointIndex+1]
        self.firstP
        self.lastPo
        lastp = c.g
        if prevp ==
          self.firs
lastPoint = "thisIsA


      elif c.point
p.pointIndex+1:
        nextp = c.g
pointIndex]
        self.lastPo
        if self.fir
"thisIsAClosePath":
          self.last
          self.firs

      else:
        prevp = c.g
pointIndex-1]
        nextp = c.g
pointIndex+1]
        if self.fir
"thisIsAClosePath":



      self.setSkel
nextp)
```

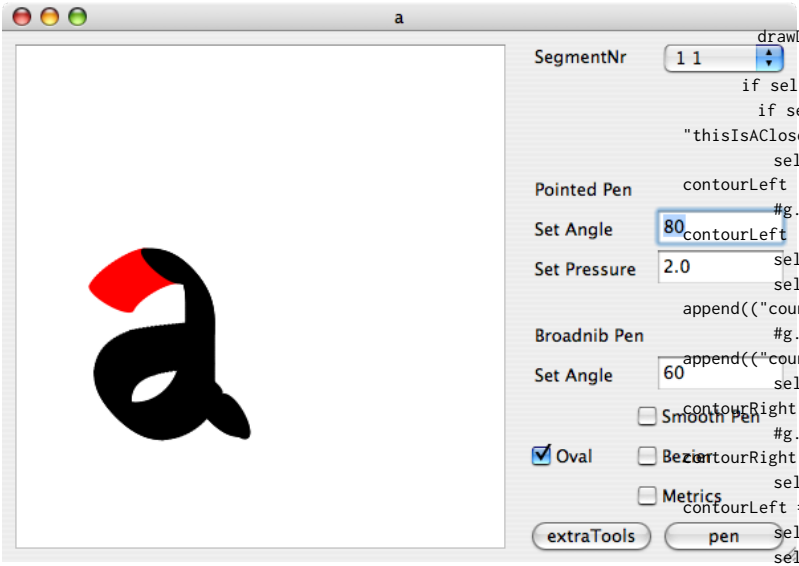
```
cScriptName[Gname]]
 when the name is
er:)

me]

 []

tNumber = 0


tour = []
eft = []
ight = []


pes.append("red")
lectSegment ==

pes.append("red")

pes.append("black")
dnibAngle(p.

tedAngle(p.
```

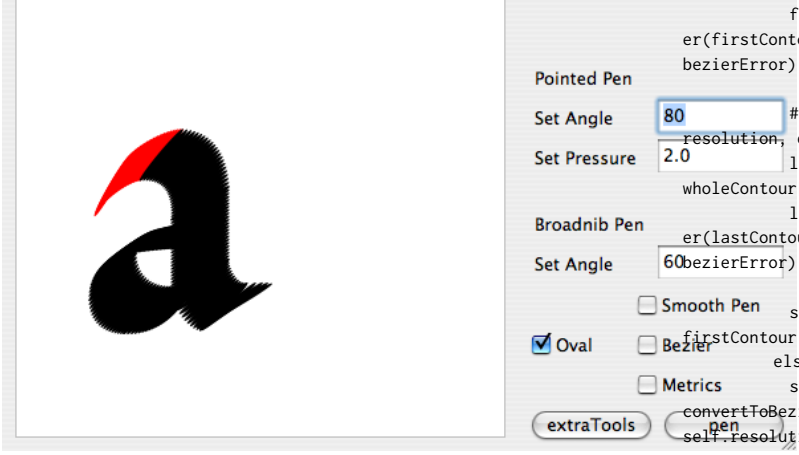


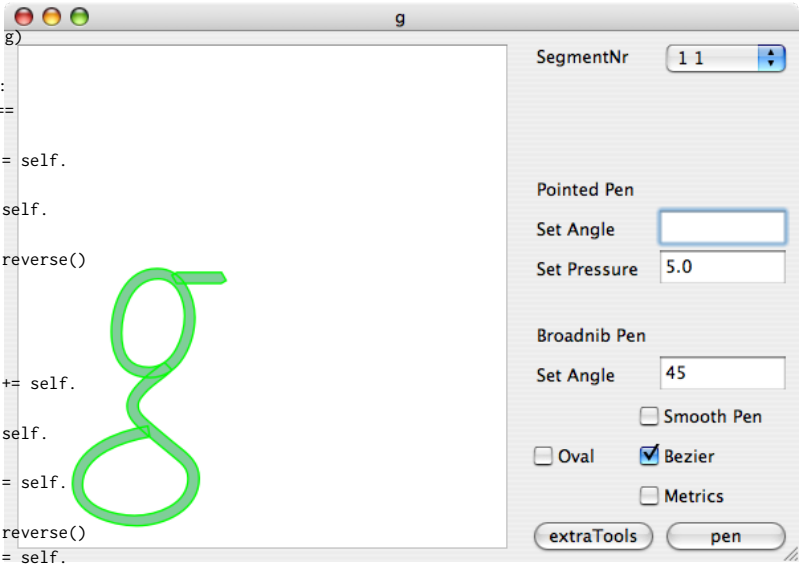

```
e = p.pressure
enWidth =


getPoints()[p.

getPoints()[p.

Point = True
oint = False
getPoints()[-1]
= lastp:
tPoint = self.
ClosePath
```

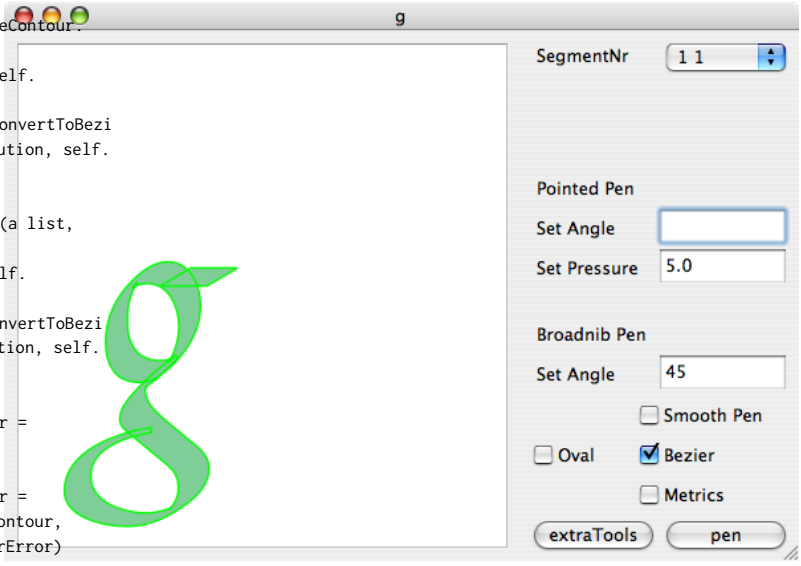


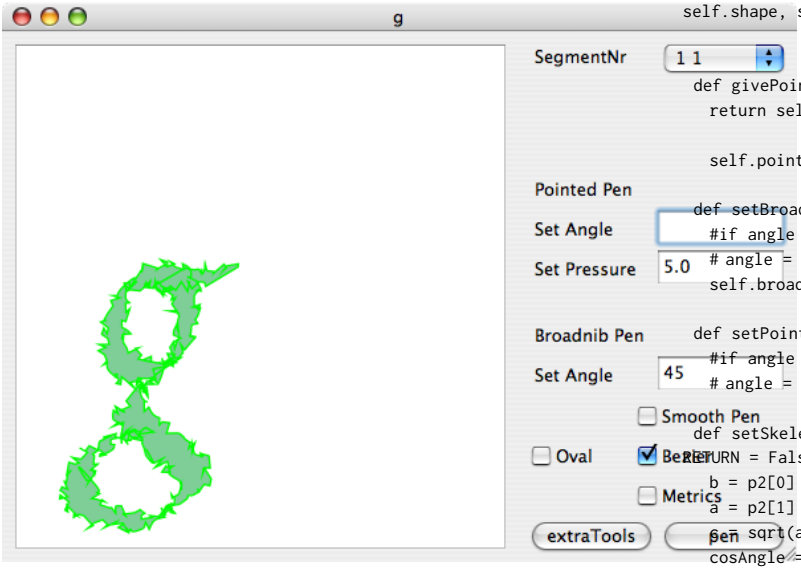

```
Index ==

getPoints()[p.

oint = "lastPoint"
rstPoint !=

Point = True
tPoint = False


getPoints()[p.

getPoints()[p.

rstPoint !=


etAngle(prevp,
```

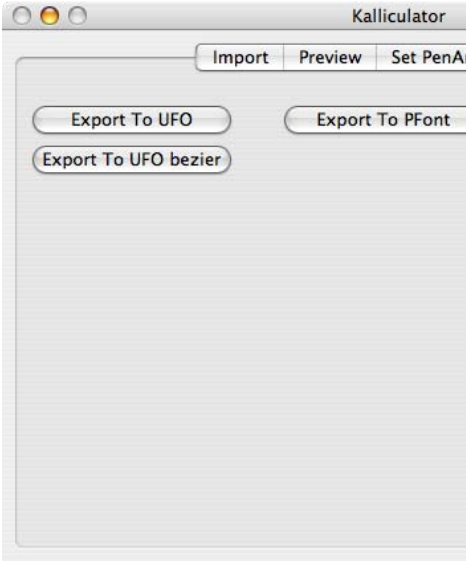

a
SegmentNr
1 1
Pointed Pen
Set Angle
80
Set Pressure
2.0
Broadnib Pen
Set Angle
60
Smooth Pen
Oval
Bezier
Metrics
extraTools
pen


a
SegmentNr
1 1
Pointed Pen
Set Angle
80
Set Pressure
2.0
Broadnib Pen
Set Angle
60
Smooth Pen
Oval
Bezier
Metrics
extraTools
pen


g
SegmentNr
1 1
Pointed Pen
Set Angle
Set Pressure
5.0
Broadnib Pen
Set Angle
45
Smooth Pen
Oval
Bezier
Metrics
extraTools
pen


g
SegmentNr
1 1
Pointed Pen
Set Angle
Set Pressure
5.0
Broadnib Pen
Set Angle
45
Smooth Pen
Oval
Bezier
Metrics
extraTools
pen

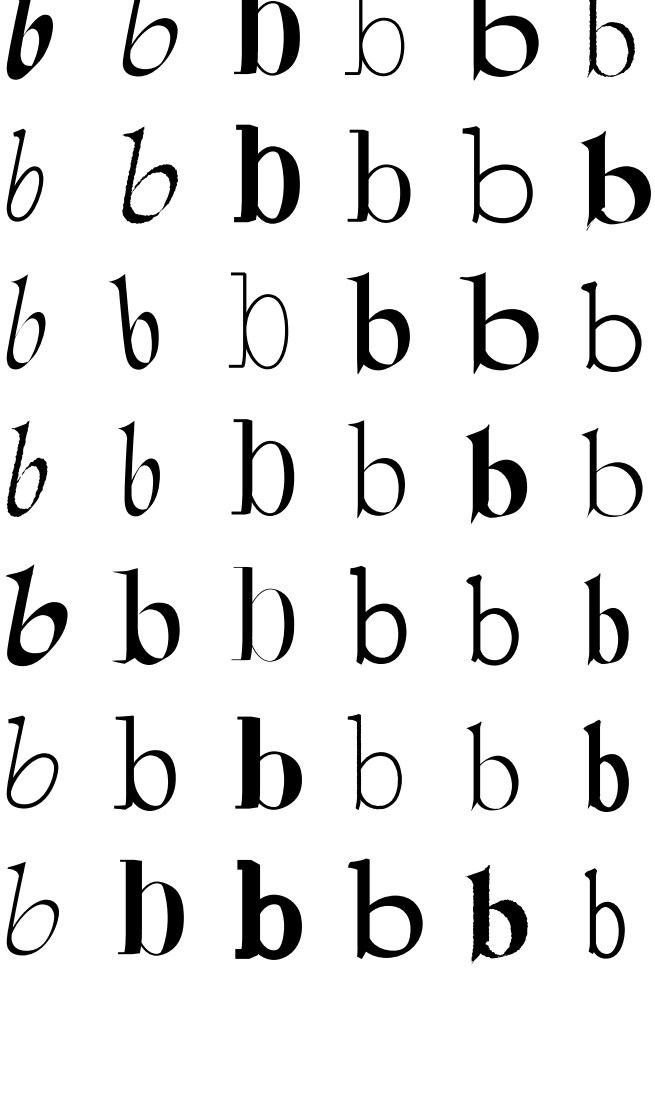





```
width)

  PFDrawInOutPut(se
self.shape, shapeLis

  def givePointList(s
    return self.point

    self.pointList =

  def setBroadnibAngl
    #if angle > 90 or
    # angle = -180 + a
    self.broadnibAngl

  def setPointedAngle
    #if angle > 90 or
    # angle = -180 + a

  def setSkeletAngle(
    RETURN = False):
    b = p2[0] - p1[0]
    a = p2[1] - p1[1]
    c = sqrt(a**2 + b
    cosAngle = b/c
    sinAngle = a/c

    cosAngle = acos(c
    sinAngle = asin(s
    if sinAngle < 0 :
      cosAngle = -cosA
    if RETURN:
      return cosAngle
    self.skeletAngle

 def rotation(self,
xH = "no", yH = "no"

    newX = x0 + (x -
- y0)*sin(a) + (rand
randomness*2)-self.r
    newY = y0 + (x -
y0)*cos(a) + (rand
randomness*2)-self.r
     if xH != "no":
      h1 = self.rota
y , a)
      h2 = self.rota
y , a)
     else:
      h1 = self.rota
yH , a)
      h2 = self.rota
y+yH , a)
     return (newX , n
    else:
     return newX, new

    if -180 > a:
     return a+360
    elif a > 180:
     return a-360
    else: return a
```

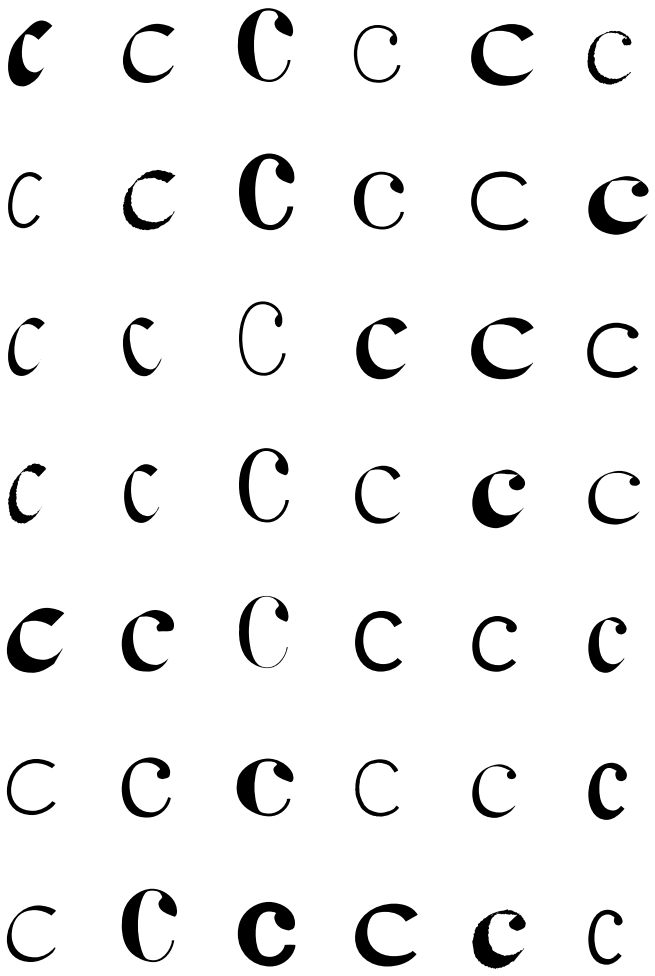
```
lf.outputClass,
t, PFont)


elf):
List

[]

e(self, angle ):
 angle < -90:
angle%180
e = angle

(self, angle ):
 angle < -90:
angle%180

 self, p1, p2,


**2)


osAngle)*180.0/pi
inAngle)*180.0/pi

Angle


= cosAngle

 x0, y0, x, y , a,
):

x0)*cos(a) - (y
int(0, self.
andomness)
x0)*sin(a) + (y
int(0, self.
andomness)

tion(x0, y0, x-xH,

tion(x0, y0, x+xH,


tion(x0, y0, x, y-

tion(x0, y0, x,

newY), h1, h2

wY
```

Specimen

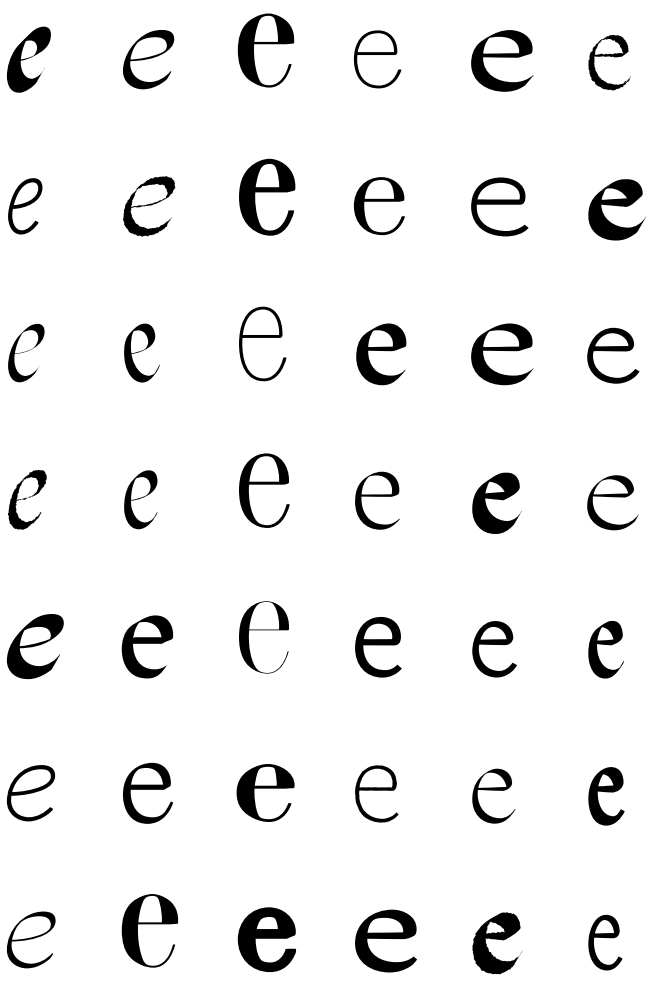


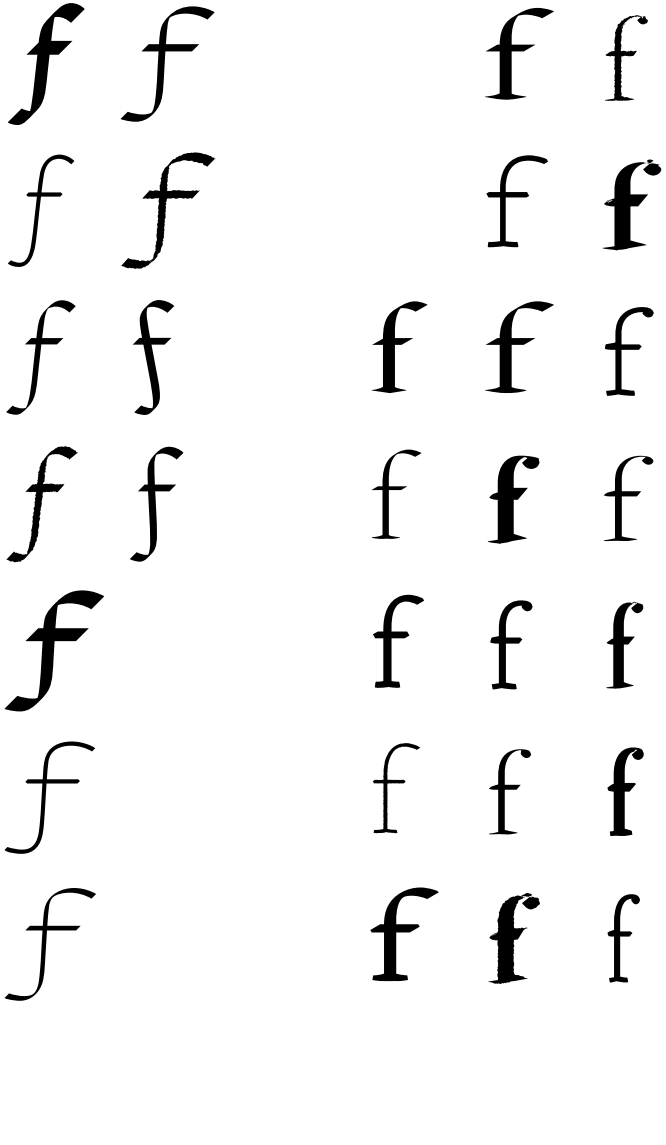
```
 def addPointInAList

   if not p in li:
     li.append(p)
     print "leave me

 def extremOval(self
g): #angle in degree
   ### this is somet
pointToBezier functi

   floatingPointErro
   if int(round(self
10000)) - int(round(
900000:
     floatingPointErr

   if floatingPointE
     newy1 = 0.0
   else:
     m = tan(pi/180.0
skeletAngle - r ))
     c = sqrt(b**2 +
     newx1 = (-a**2*r
     if m > 10000000
1000000000000 and c
       newy1 = 0.0
     else:
       newy1 = m*newx

   newy2 = -newy1

   morePointsx = -ne
   morePointsy = new

   morePointsx2 = ne
   morePointsy2 = -n

   rInRad = r/180.0*
   ## This if statme
inside as inside, sa
the contour
     newx1 *= -1
     newy1 *= -1
     newx2 *= -1
     newy2 *= -1



   if self.firstPoin

     morePointsx += x
     morePointsy += y
     a = self.setSkel
x, morePointsy), (x,

     morePointsx2, mo
self.rotation(0.0, 0
morePointsy2, rInRad
     morePointsx2 +=
     morePointsy2 +=
     a2 = self.setSke
```

```
(self, li, p):



 OUUUT"

, x, y, a, b, r,
s not in rad
hing for the
on in PFDraw

r = False
.skeletAngle*
r*10000)) ==

ror = True

rror:

0 * (self.

 (a**2)*(m**2))
m*c)/c**2
00000 or (m <
> 1000000000000):

x1 + c

wx1
y1

wx1
ewy1

pi
nt is to keep the
me for outside of

t == True:

x
y
letAngle((morePoints
 y), RETURN = True)

orePointsy2 =
.0, morePointsx2,
)
 x
 y
eletAngle((morePoi
```



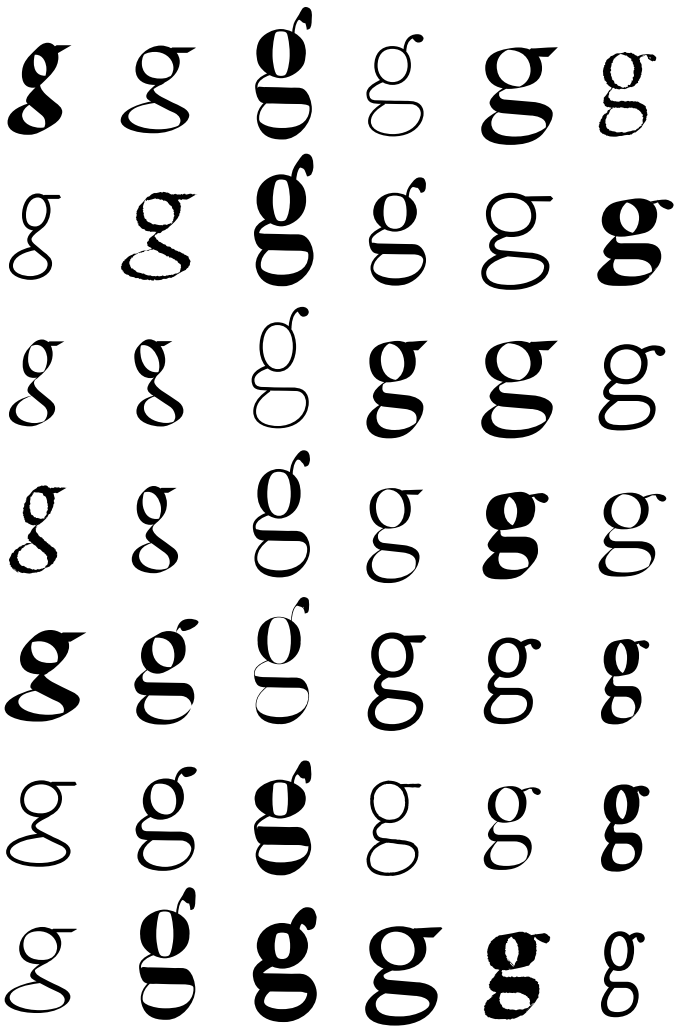
```
ntsx2, morePointsy2)
True)


    a2 = a2 - self.s

    a = self.correct
    a2 = self.correc

    if abs(a) < abs(
      pointToAdd = (
morePointsy)
    else:
      pointToAdd = (
morePointsy2)

    #self.contourLef
pointToAdd))
    self.firstPoint

  lastPointToAdd = [
  if self.lastPoint

    morePointsx, mor
rotation(0.0, 0.0, m
morePointsy, rInRad)
    morePointsx += x
    morePointsy += y
    a = self.setSkel
x, morePointsy), (x,
    morePointsx2, mo
self.rotation(0.0, 0
morePointsy2, rInRad
    morePointsx2 +=
    morePointsy2 +=
    a2 = self.setSke
ntsx2, morePointsy2)
True)

    a = a - self.ske
    a2 = a2 - self.s

    a = self.correct
    a2 = self.correc
    if abs(a) > abs(
      lastPointToAdd
morePointsy)
    else:
      lastPointToAdd
morePointsy2)

    #self.contourRig
append(("lineTo", la



  newx2, newy2 = se
0.0, newx2, newy2, r

  ### extrem points
  newx1 += x
  newy1 += y
```

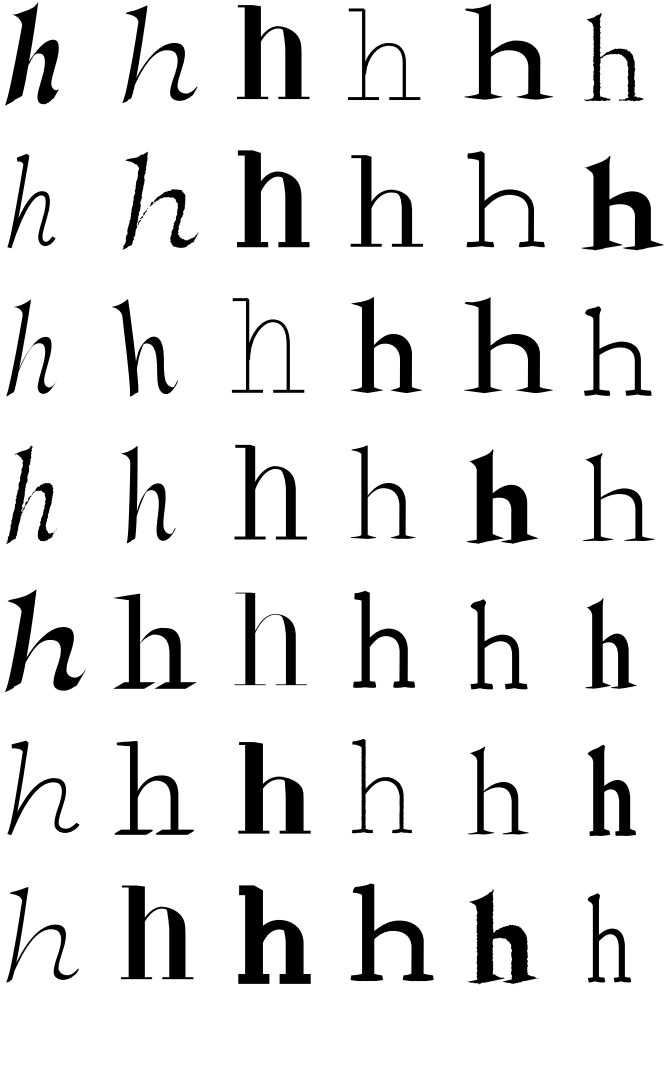




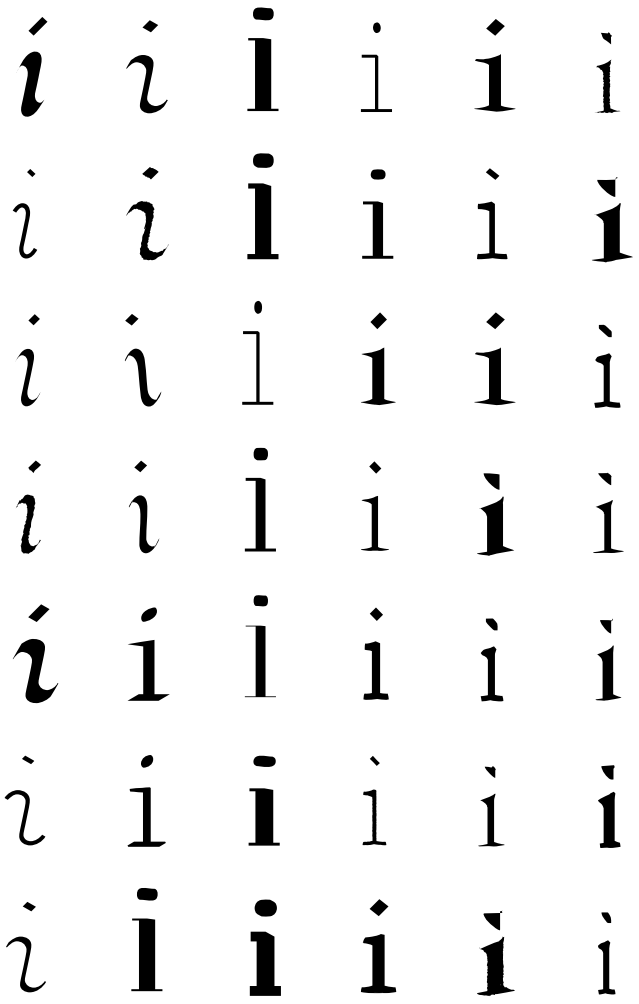
```
    newx2 += x
    newy2 += y



    #if self.smoothCo
    # print self.lastS
lastPointX, lastPoin
    # print self.skele
    # a1 = (lastPointy
in(radians(self.last
(lastPointX - lastPo
lf.lastSkeletAngle))
    # b1 = lastPointy

    # a2 = (newy1 -
newy1+sin(radians(se
/ (newx1 - newx1+cos
skeletAngle)))
    # b2 = newy1 - a2*
    #
    #   extraX = (b1 -
    # else:
    #   extraX = (b1 -
    # extraY = a1* ext
      #self.contourLef
(extraX, extraY)))



    if self.firstPoin
"thisIsAClosePath":
      if len(self.cont
        self.addPointI
contourLeft, ("moveT
newy1)))
      else:
        self.addPointI
contourLeft, ("lineT
newy1)))
        #self.contourL
append(("lineTo", (n

      if self.lastPoin
        self.addPointI
contourRight, ("move
newy2)))
        #self.contourR
append(("moveTo", (n
      else:
        self.addPointI
contourRight, ("line
newy2)))
        #self.contourR
append(("lineTo", (n
      #if self.lastPoi
self.firstPoint == F
      self.addPointIn/
contourRight, ("line
newy2)))
      #self.contourRig
append(("lineTo", (n
      if lastPointToAd
        self.addPointI
contourRight, ("line
```

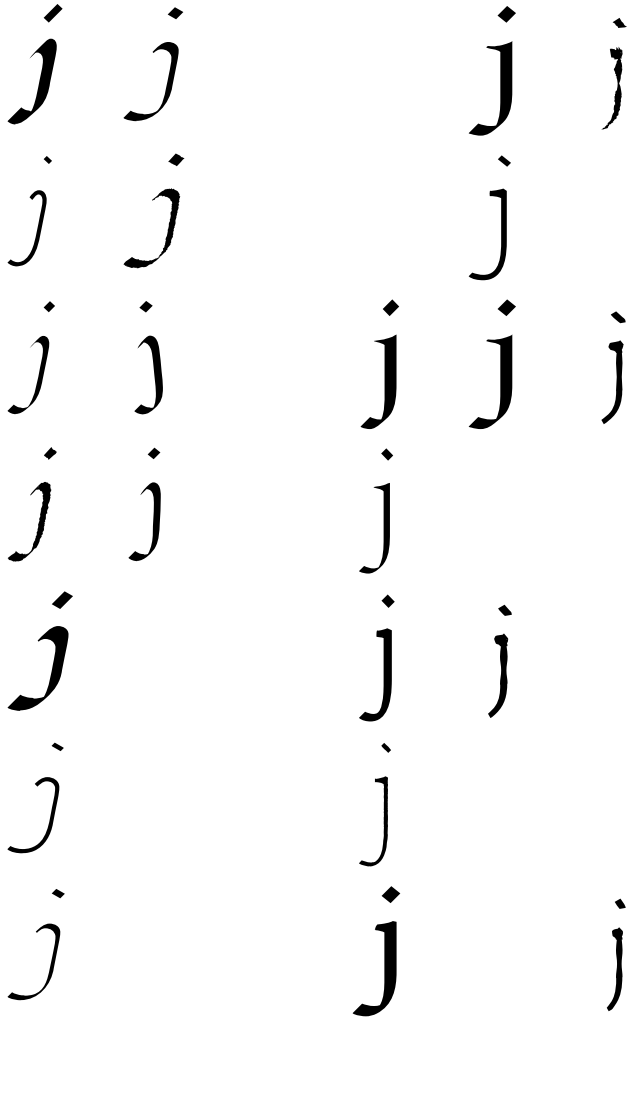
```
nnection:
SkeletAngle,
ty
etAngle
y - lastPointy+s
SkeletAngle))) /
intX+cos(radians(se
)
- a1*lastPointX


lf.skeletAngle)))
(radians(self.

*newx1

- b2)

- b2) / (a2 - a1)
traX + b1
ft.append(("moveTo",


t ==

tourLeft) == 0:
InAList(self.
o", (newx1,


InAList(self.
o", (newx1,

eft.
ewx1, newy1)))

nt == "lastPoint":
InAList(self.
To", (newx2,

Right.
ewx2, newy2)))

InAList(self.
To", (newx2,

Right.
ewx2, newy2)))
int == False and
alse:
AList(self.
To", (newx2,

ght.
ewx2, newy2)))
dd:
InAList(self.
To",
```



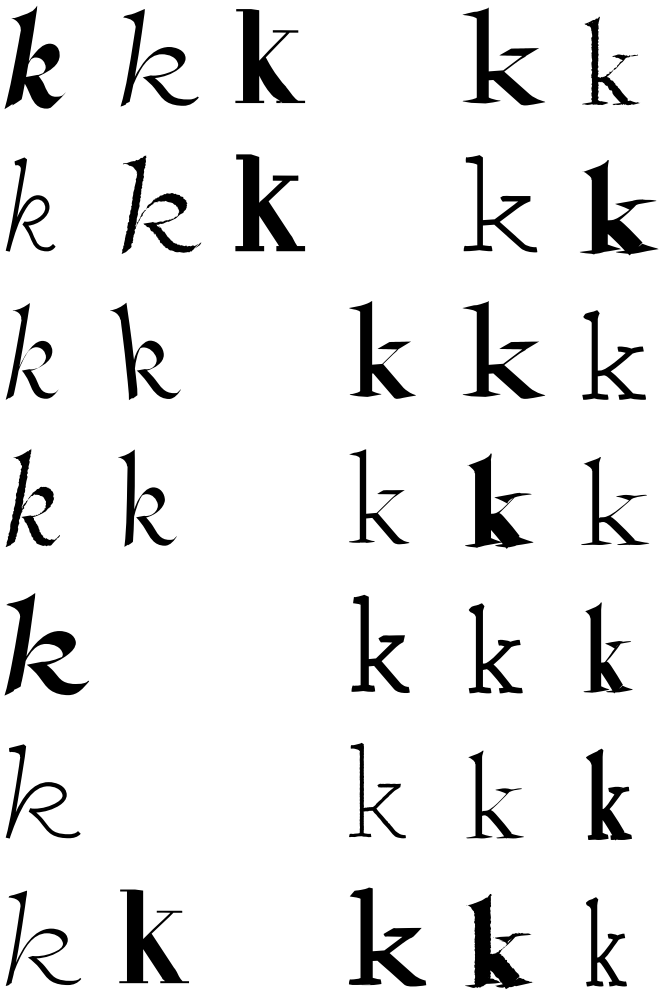


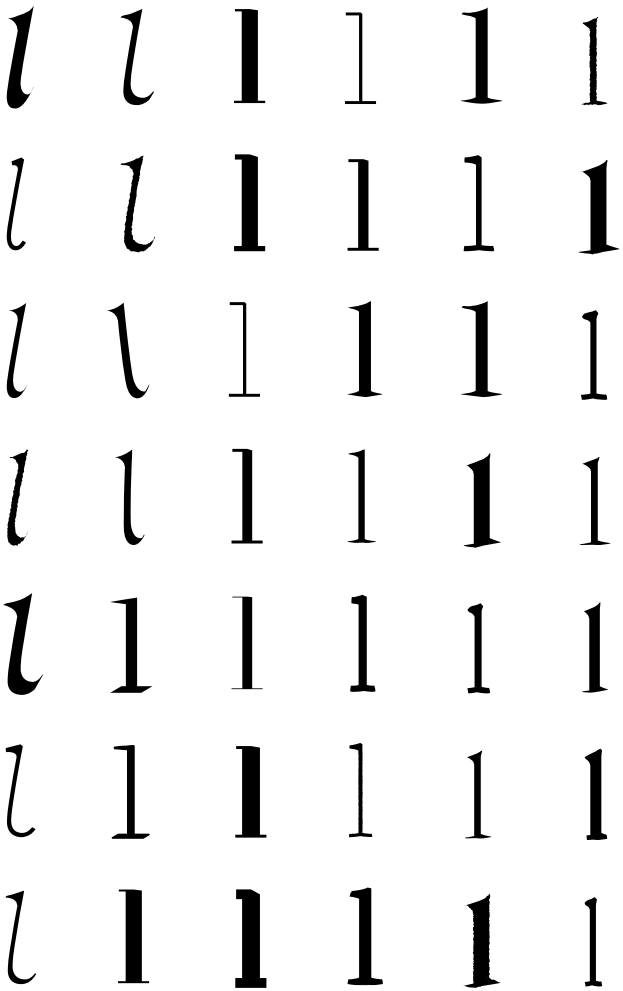
```
lastPointToAdd))
      #self.contourR
append(("lineTo", la

    self.addPointIn/
contourLeft, ("lineT
newy1)))
    #self.contourLef
(newx1, newy1)))

 def oval(self, p, g

   penAngle = self.b
self.interPenation)
pointedAngle-90.0)*(

   pointedAngle = ((
- penAngle)/180.0*pi
   r = penAngle/180.
   f = abs(sin(point
pressure ### for the
pointed pen and maki
the angle is the ske
angle of the pen
   expantion = (self
   x, y = p
   w = abs((self.h +
self.interPenation +
interPenation))/2.
   w = w*self.adjust

   h = self.h/2.

   curve = sqrt(pi)

   if curve == 0:
     curve = 1

   if self.drawInCon
     return
     ## this makes th

   xHandle = w/curve
   yHandle = h/curve

   bot = self.rotati
r, xH = xHandle)

   left = self.rotat
r, yH = yHandle)
   top = self.rotati
r, xH = xHandle)

   pointList = [bot,

   g.glyphShapes.app



 def extremRect(self
newx2, newy2,morePoi
rInRad, x, y):
   morePointsx2 = -m
   morePointsy2 = -m
```





```
   if self.firstPoin

    morePointsx, mor
rotation(0.0, 0.0, m
morePointsy, rInRad)
    morePointsx += x
    morePointsy += y
    a = self.setSkel
x, morePointsy), (x,

    morePointsx2, mo
self.rotation(0.0, 0
morePointsy2, rInRad
    morePointsx2 +=
    morePointsy2 +=


    a = a - self.ske
    a2 = a2 - self.s

    a = self.correct
    a2 = self.correc

    if abs(a) < abs(
     pointToAdd = (
morePointsy)
     pointToAdd = (
morePointsy2)


    self.contourLeft
pointToAdd))
    self.firstPoint

   lastPointToAdd =
   if self.lastPoint

    morePointsx, mor
rotation(0.0, 0.0, m
morePointsy, rInRad)
    morePointsy += y
    a = self.setSkel
x, morePointsy), (x,

    morePointsx2, mo
self.rotation(0.0, 0
morePointsy2, rInRad
    morePointsx2 +=
    morePointsy2 +=
    a2 = self.setSke
ntsx2, morePointsy2)
True)

    a = a - self.ske
    a2 = a2 - self.s
    a = self.correct
    a2 = self.correc

    if abs(a) > abs(
     lastPointToAdd
morePointsy)
    else:
     lastPointToAdd
morePointsy2)
```

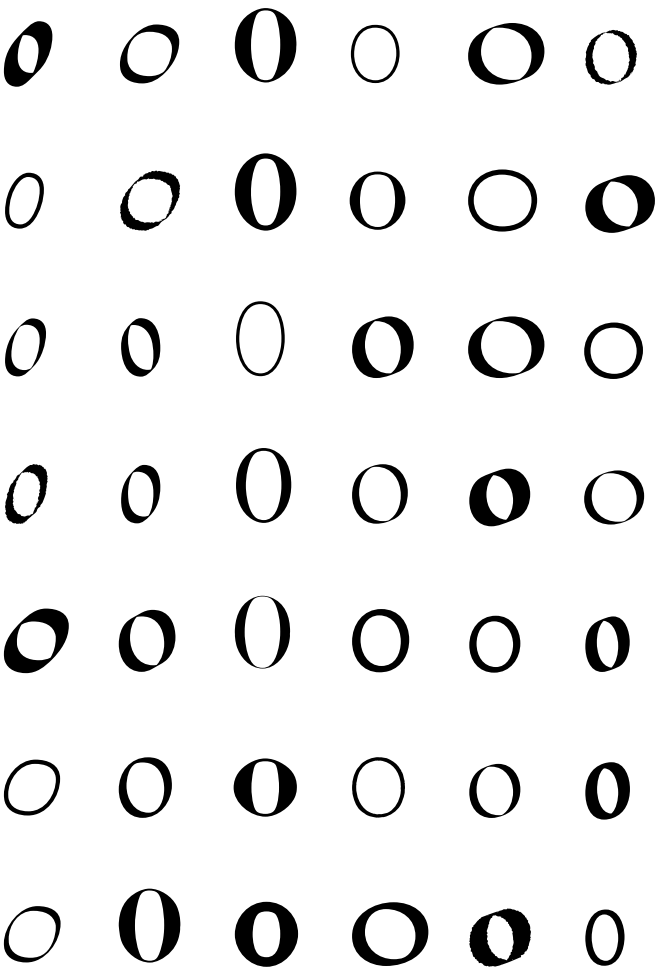




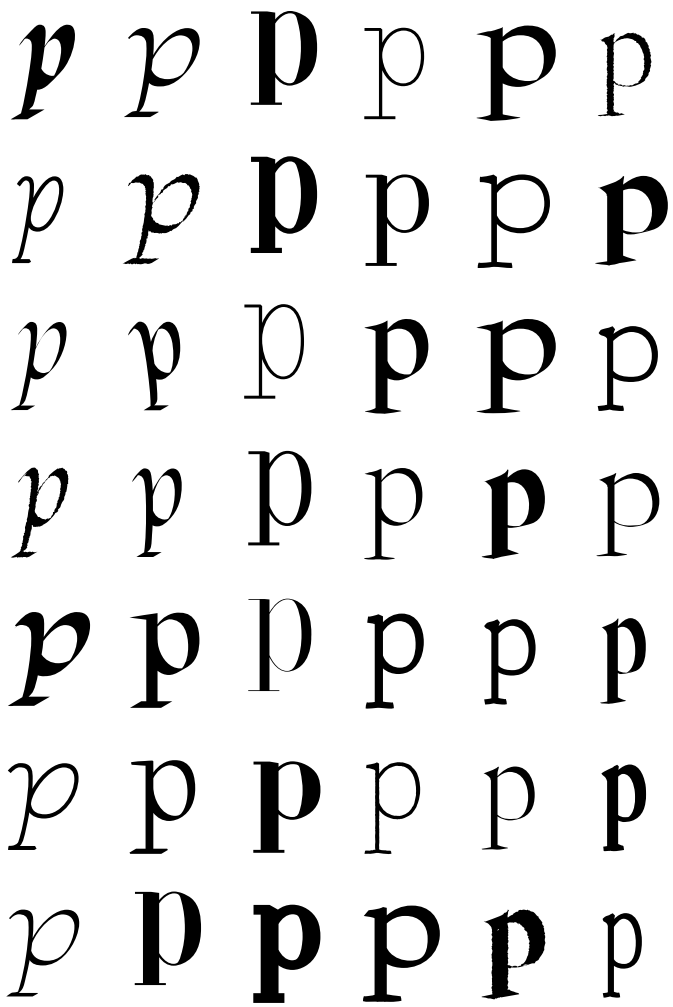
```
    if self.firstPo
"thisIsAClosePath":
      if len(self.co
        #self.contou
append(("moveTo", (n
      else:
        self.addPoin
contourLeft, ("lineT
newy1)))
        #self.contou
append(("lineTo", (n

      if self.lastPo
        self.addPoin
contourRight, ("move
newy2)))
        #self.contou
append(("moveTo", (n
      else:
        self.addPoin
contourRight, ("line
newy2)))
    if self.firstPo
"thisIsAClosePath":
      #if self.lastP
self.firstPoint == F
      self.addPointI
contourRight, ("line
newy2)))
      #self.contourR
append(("lineTo", (n
      if lastPointTo
        self.addPoin
contourRight, ("line
lastPointToAdd))
        #self.contou
append(("lineTo", la

      self.addPointI
contourLeft, ("lineT
newy1)))
      #self.contourL
append(("lineTo", (n
 def rect(self, p, g


   penAngle = self.b
self.interPenation)
pointedAngle-90)*(se

   pointedAngle = ((
- penAngle)/180*pi

   r = penAngle/180.
   f = abs(sin(point
pressure ### for the
pointed pen and maki
the angle is the ske
angle of the pen


   x, y = p
   w = abs((self.h +
```

```
self.interPenation +
interPenation))/2.
   w = w*self.adjust
   h = self.h/2.


   rDegree = r/pi*18

   skr = int(round(s
10000))/10000
   morePointsx = -w
   morePointsy = h


   ### extremen zitt
de pen hoek negatief
   if (skr) > rDegre
rDegree-180:
     if (skr) > (rDeg
(rDegree-90):
       w *= 1
       h *= 1
     else:
       h *= -1
   else:
     if (skr) > (rDeg
(rDegree-90):
       w *= -1
       h *= 1
     else:
       w *= -1
       h *= -1

   BotLeft = self.ro
y - h, r)
   TopLeft = self.ro
y + h, r)
   TopRight = self.r
y + h, r)

   morePointsx = -w
   morePointsy = h

   ### extrem points
   newx1, newy1 = To
   newx2, newy2 = Bo
   if self.drawInCon
     return
   pointList = [BotL
TopLeft, TopRight]
   g.glyphShapes.app



##################

#from postScriptName
postScriptName

#    print "\"%s\" :
(postScriptName[name


postScriptName = {
```

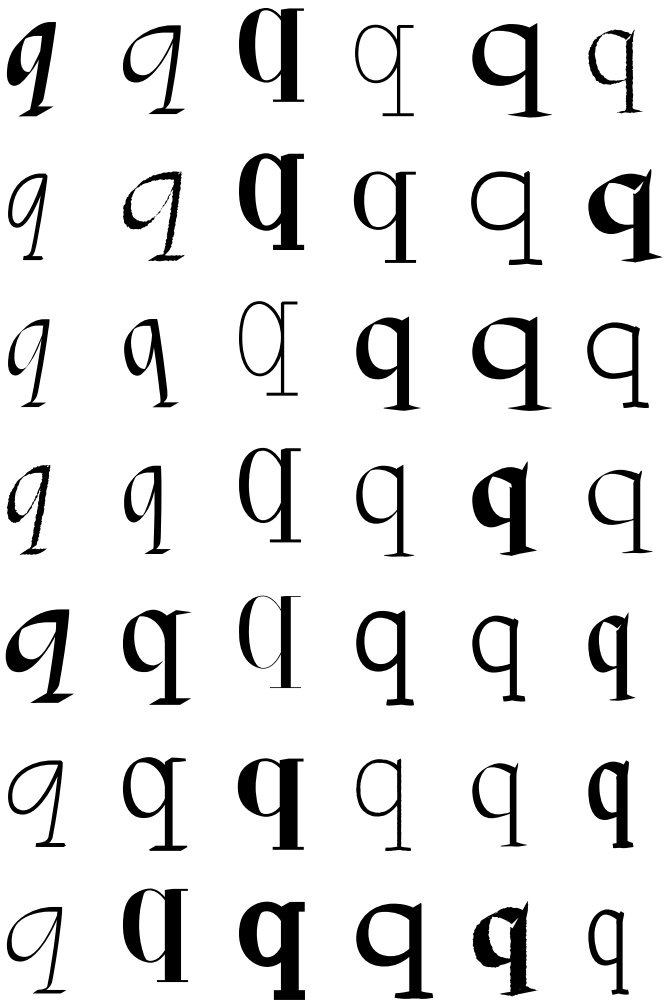
```
 self.w*(1-self.

PenWidth



0.0

elf.skeletAngle*
```



```
en nog niet goe als
 is
e or (skr) <

gree+90) or (skr) <

gree+90) or (skr) <

tation(x, y, x - w,

tation(x, y, x - w,

otation(x, y, x + w,

pRight
tLeft
tour:

eft, BotRight,

end(pointList)



s import


 \"%s\"," %
], name)
```



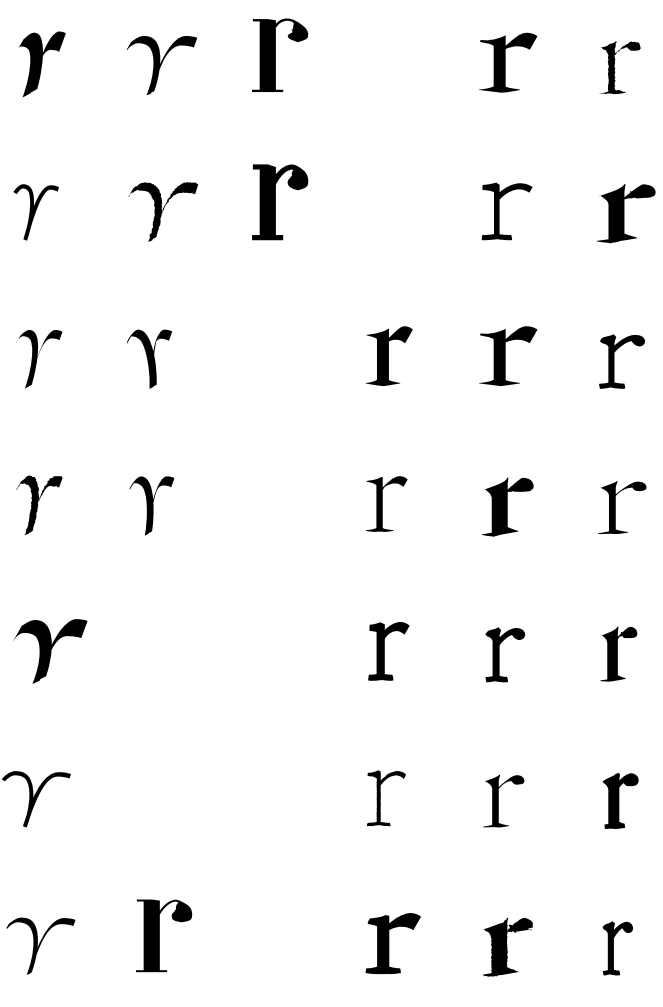
```
" " : "space",

"a" : "a",
"b" : "b",
"c" : "c",
"d" : "d",
"f" : "f",
"g" : "g",
"h" : "h",
"i" : "i",
"j" : "j",
"k" : "k",
"l" : "l",
"m" : "m",
"n" : "n",
"o" : "o",
"q" : "q",
"r" : "r",
"s" : "s",
"t" : "t",
"u" : "u",
"v" : "v",
"w" : "w",
"x" : "x",
"y" : "y",
"z" : "z",
"A" : "A",
"B" : "B",
"C" : "C",
"D" : "D",
"E" : "E",
"F" : "F",
"G" : "G",
"H" : "H",
"I" : "I",
"J" : "J",
"L" : "L",
"M" : "M",
"N" : "N",
"O" : "O",
"P" : "P",
"Q" : "Q",
"R" : "R",
"S" : "S",
"T" : "T",
"U" : "U",
"W" : "W",
"X" : "X",
"Y" : "Y",
"Z" : "Z",

"¬±" : "plusminus",
"√·" : "aring",
"¬™" : "ordfeminine",
"<" : "less",
"√≤" : "ograve",
"4" : "four",
"√ë" : "Ntilde",
"," : "comma",
"√ù" : "Yacute",

"™Ø∞" : "perthousand",
"√ß" : "ccedilla",
"™Ú" : "quotedblright",
```



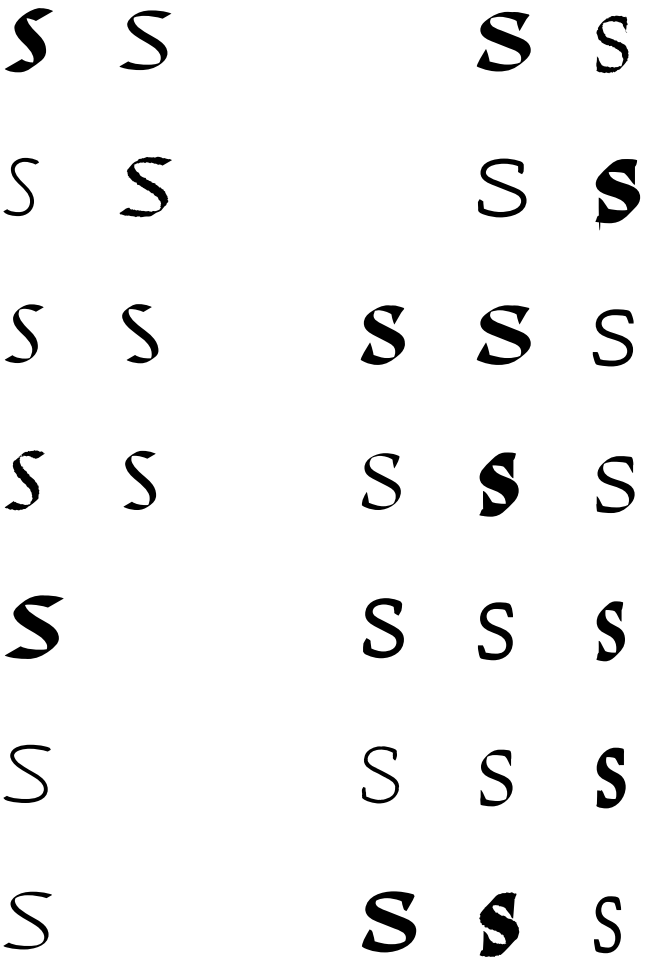
,

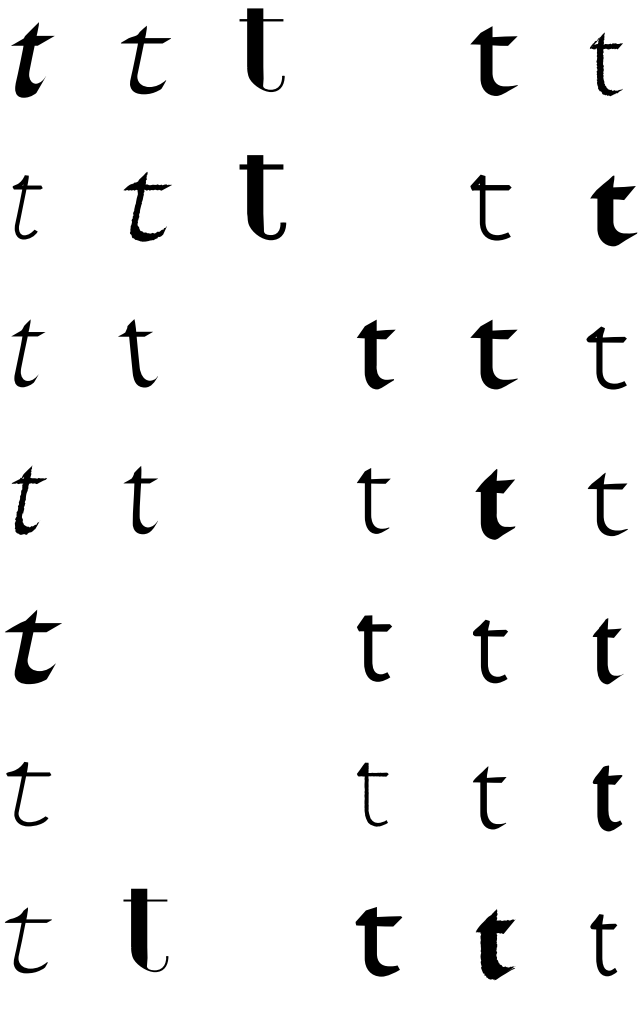
d",

ht",

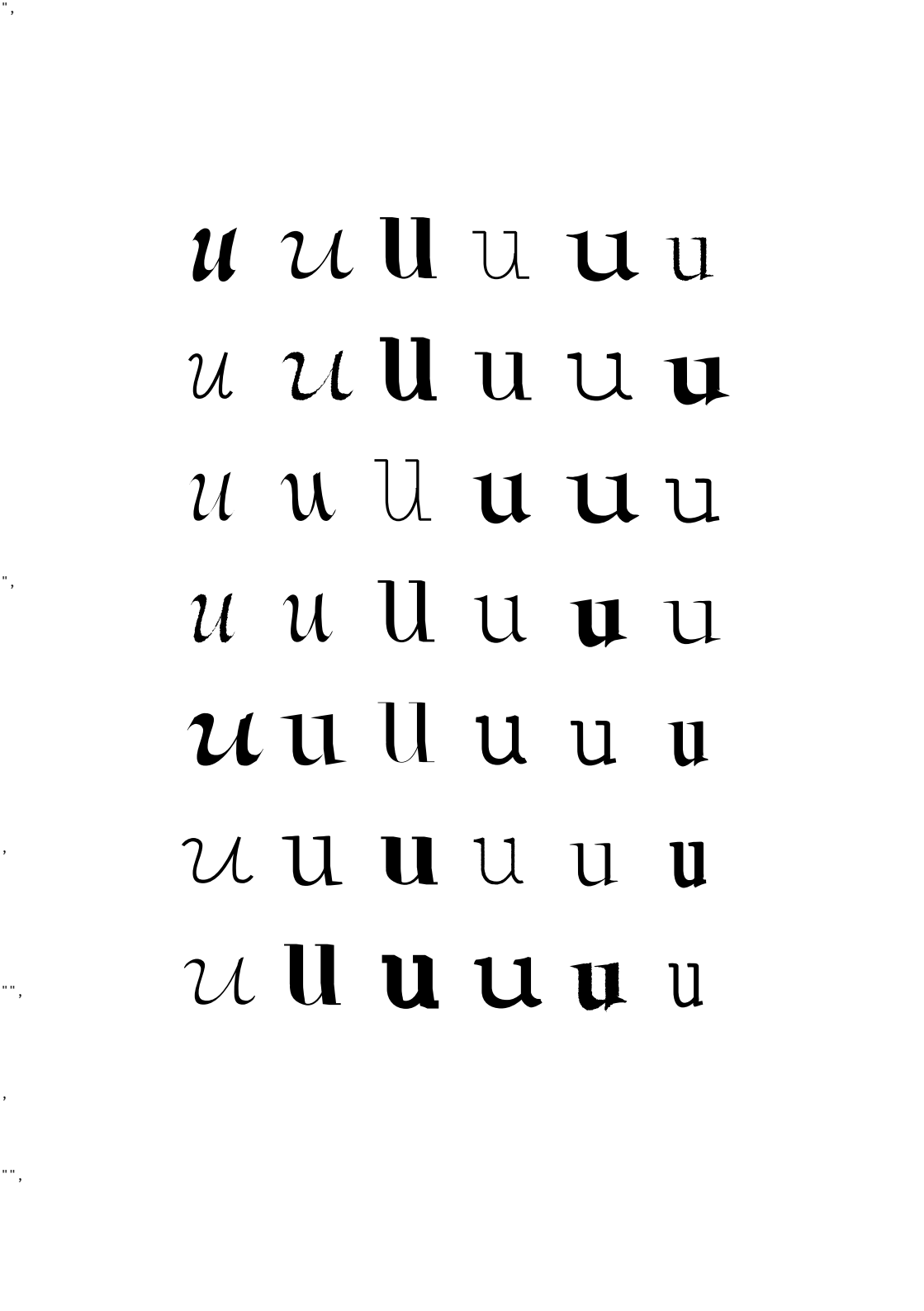


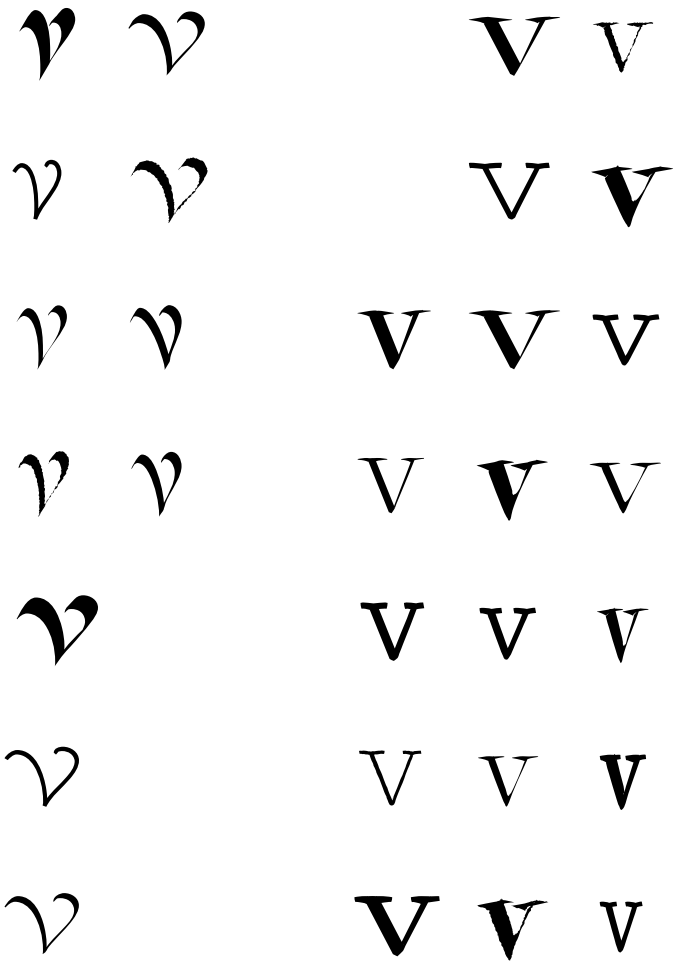
```
"√ê" : "Thorn",
"≈Ω" : "Zcaron",
"¬·" : "yen",
"√µ" : "otilde",
"√ª" : "ucircumflex",
"√Å" : "Aacute",
"™Äö" : "quotesinglb",
"¬≤" : "twosuperior",
"√¥" : "ocircumflex",
"™¢¨" : "Euro",
"√∏" : "oslash",
"√™" : "ecircumflex",

"√∫" : "uacute",
"≈æ" : "zcaron",
"5" : "five",
"_" : "underscore",
"√é" : "Idieresis",
"√ú" : "Udieresis",

"|" : "bar",
"≈í" : "OE",

"√∂" : "odieresis",
"¬°" : "exclamdown",
")" : "parenright",
"™Äì" : "endash",
"√É" : "Atilde",
"√ï" : "Otilde",
"0" : "zero",

"√å" : "Igrave",
"√ß" : "adieresis",
"" : "quotesingle",
"≈™" : "Scaron",
"{" : "braceleft",
"√®" : "egrave",
"√Ñ" : "Adieresis",
"√ü" : "germandbls",
"√ñ" : "Odieresis",

"¬ß" : "section",
"√±" : "ntilde",
"&" : "ampersand",
"/" : "slash",
"√£" : "atilde",
"√Ç" : "Acircumflex",
"™¶" : "ellipsis",
"#" : "numbersign",
"√ò" : "Oslash",
"™Äú" : "quotedbllef",
"¬∞" : "degree",
"√¢" : "acircumflex",
"!" : "exclam",
"√Ö" : "Aring",
"≈ì" : "oe",
"√é" : "Icircumflex",

#"guilsinglleft" : "
#"equal" : "",
#"Edieresis" : "",
#"Ugrave" : "",
#"Agrave" : "",
#"eth" : "",
```





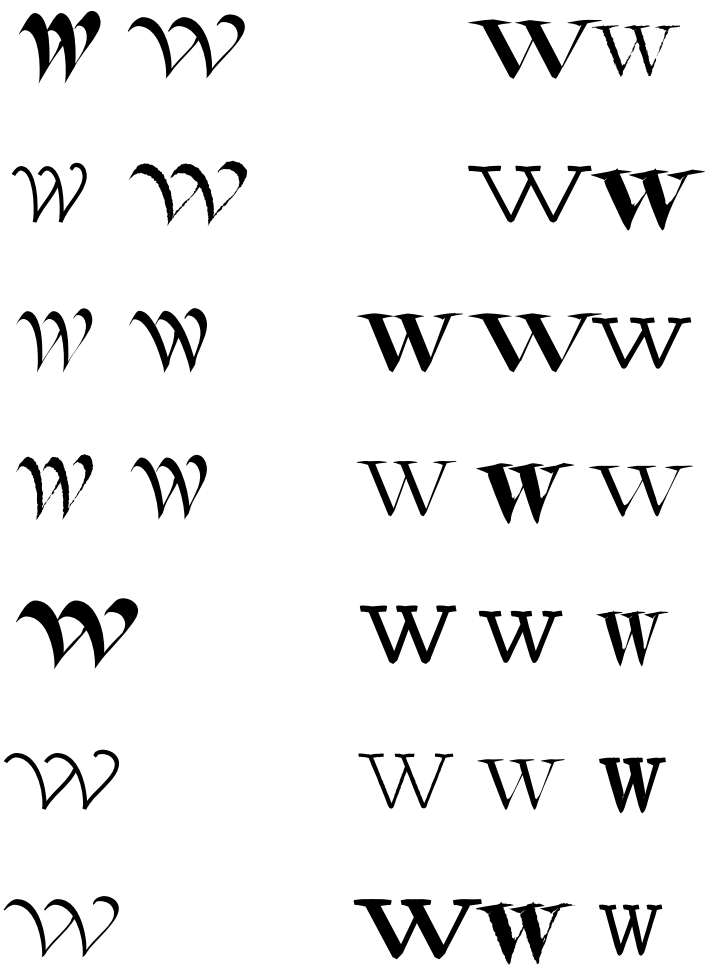

```
#"threesuperior" :
#"udieresis" : "",
#"Egrave" : "",
#"dagger" : "",
#"divide" : "",
#"trademark" : "",
#"hyphen" : "",
#"period" : "",
"1" : "one",
#"asciitilde" : "",
#"colon" : "",
#"eacute" : "",
#"parenleft" : "",
#"Ecircumflex" : "",
#"fl" : "",
#"question" : "",
"2" : "two",
#"cent" : "",
#"lslash" : "",
#"scaron" : "",
#"iacute" : "",
#"registered" : "",
#"Ydieresis" : "",
#"backslash" : "",
#"dieresis" : "",
#"bracketleft" : "",
#"Eacute" : "",
#"asciicircum" : "",
#"guillemotleft" : "
#"Ograve" : "",
#"mu" : "",
#"paragraph" : "",
"9" : "nine",
"3" : "three",
#"Ccedilla" : "",
#"idieresis" : "",
#"minus" : "",
#"braceright" : "",
#"ae" : "",
#"semicolon" : "",
#"brokenbar" : "",
#"quotedblbase" : ""
#"currency" : "",
#"ugrave" : "",
#"Ucircumflex" : "",
#"at" : "",
#"Lslash" : "",
#"edieresis" : "",
#"periodcentered" :
#"daggerdbl" : "",
"6" : "six",
#"percent" : "",
#"yacute" : "",
#"bracketright" : ""
#"sterling" : "",
#"quotedbl" : "",
#"AE" : "",
#"guilsinglright" :
#"aacute" : "",
#"icircumflex" : "",
"8" : "eight",
#"multiply" : "",
#"fi" : "",
#"Eth" : "",
```





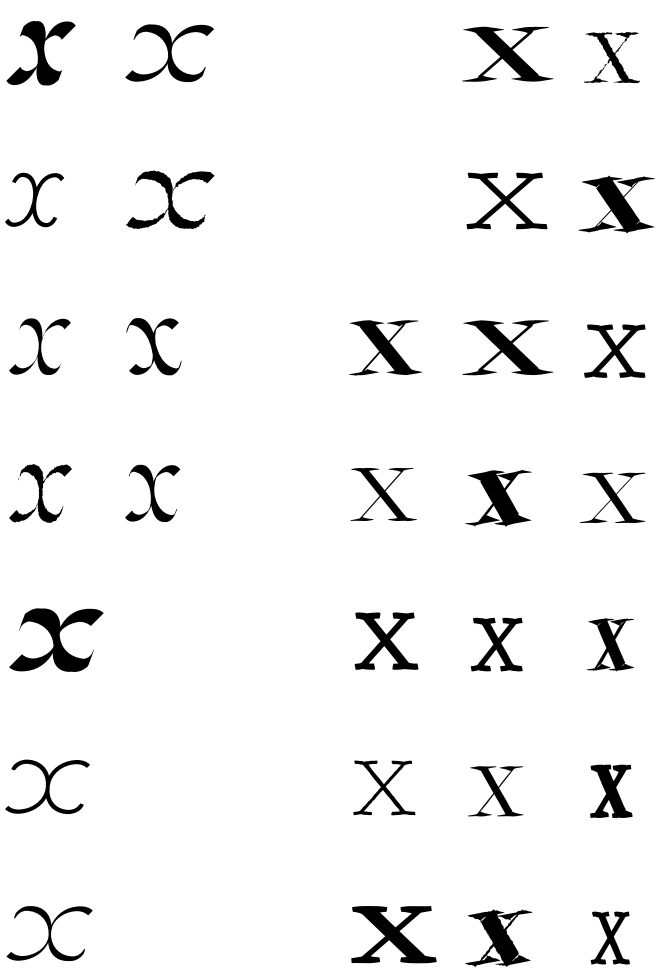
```
#"emdash" : "",
#"grave" : "",
#"Iacute" : "",
#"bullet" : "",
#"thorn" : "",
#"Uacute" : "",
#"oacute" : "",


}

theOtherWaypostScrip
".notdef" : " ",
"space" : " ",
"comma" : ",",
"zero" : "0",
"four" : "4",
"eight" : "8",
"less" : "<",
"Euro" : "™",
"D" : "D",
"H" : "H",
"L" : "L",
"P" : "P",
"section" : "¬ß",
"T" : "T",
"X" : "X",
"d" : "d",
"h" : "h",
"l" : "l",
"p" : "p",
"t" : "t",
"x" : "x",
"bar" : "|",
"quotesingle" : """,
"slash" : "/",
"three" : "3",
"seven" : "7",
"C" : "C",
"G" : "G",
"K" : "K",
"O" : "O",
"S" : "S",
"W" : "W",
"c" : "c",
"z" : "z",
"g" : "g",
"k" : "k",
"o" : "o",
"s" : "s",
"w" : "w",
"braceleft" : "{",
"Icircumflex" : "√é",
"Igrave" : "√å",
"Acircumflex" : "√Ç",
"Aacute" : "√Å",
"Aring" : "√Ö",
"Adieresis" : "√Ñ",
"Oslash" : "√ò",
"germandbls" : "√ü",
"Yacute" : "√ù",
"Udieresis" : "√ú",
"Ntilde" : "√ë",
"Thorn" : "√Þ",
```

tName = {

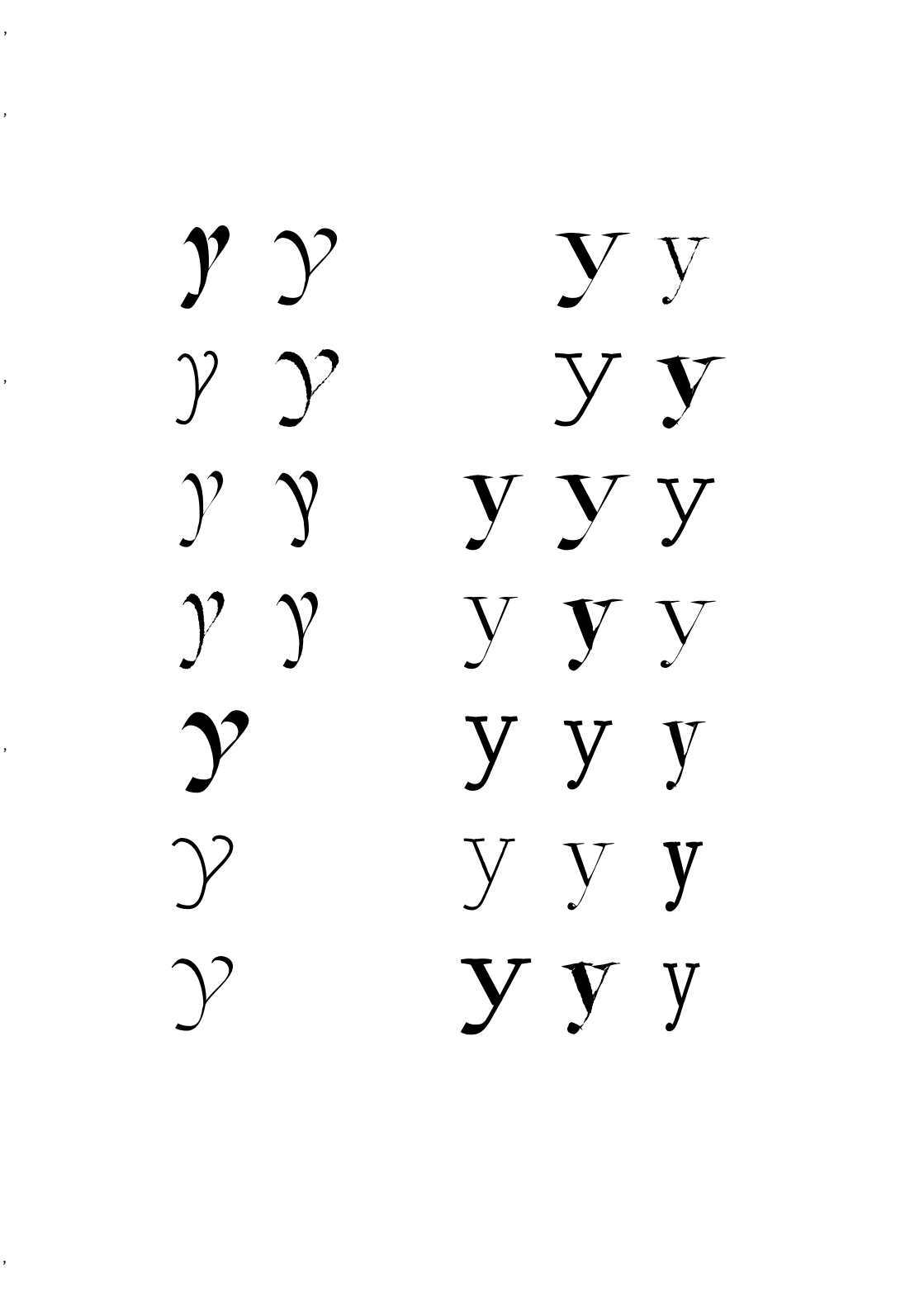


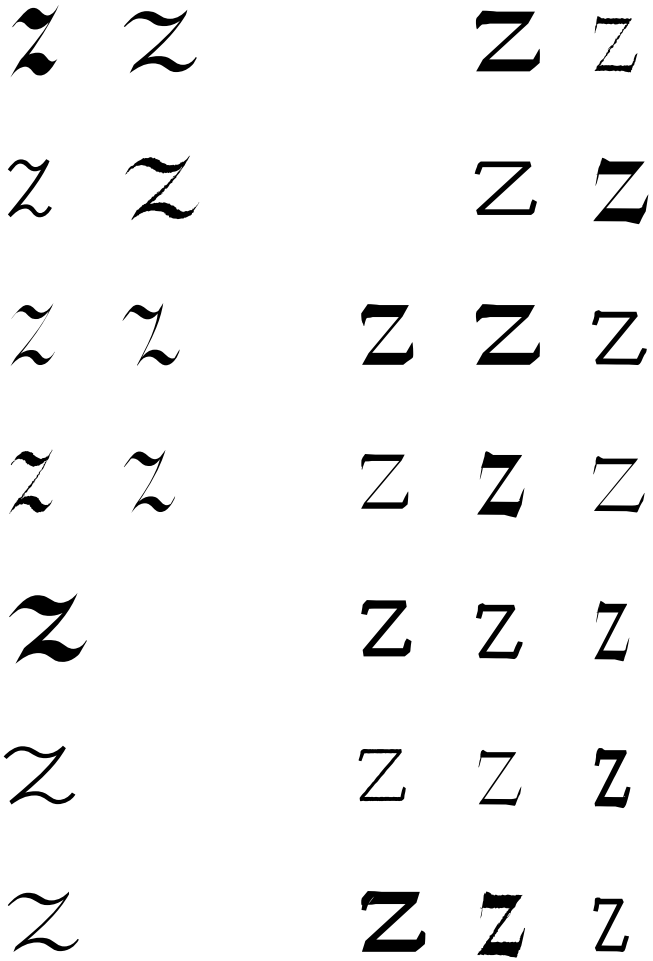
,

,




```
"Otilde" : "√ï",
"ecircumflex" : "√™",
"egrave" : "√®",
"ampersand" : "&",
"atilde" : "√£",
"acircumflex" : "√¢",
"agrave" : "√†",
"ccedilla" : "√ß",
"aring" : "√•",
"adieresis" : "√§",
"uacute" : "√∫",
"oslash" : "√Ø",
"ydieresis" : "√ø",
"six" : "6",
"ograve" : "√≤",
"ntilde" : "√±",
"odieresis" : "√∂",
"otilde" : "√µ",
"ocircumflex" : "√¥",
"B" : "B",
"J" : "J",
"N" : "N",
"R" : "R",
"V" : "V",
"Z" : "Z",
"yen" : "¬•",
"b" : "b",
"exclamdown" : "¬°",
"f" : "f",
"j" : "j",
"ordfeminine" : "ª",
"n" : "n",
"quoteright" : "‘",
"r" : "r",
"degree" : "¬∞",
"plusminus" : "¬±",
"twosuperior" : "¬≤",
"v" : "v",
"greater" : ">",
"oe" : "~ì",
"exclam" : "!",
"parenright" : ")",
"Scaron" : "~¤",
"Zcaron" : "~µ",
"zcaron" : "~æ",
"five" : "5",
"nine" : "9",
"A" : "A",
"perthousand" : "‰",
"E" : "E",
"M" : "M",
"Q" : "Q",
"U" : "U",
"ellipsis" : "¤",
"Y" : "Y",
"a" : "a",
"endash" : "–",
"e" : "e",
"i" : "i",
"quoteleft" : "‘",
"m" : "m",
"quotedblleft" : "’",
"q" : "q",
"one" : "1",
```





```
"u" : "u",
"y" : "y",

"asterisk" : "*",

}

##################

from AppKit import N
NSColor
#from cgDocument.cgD
newDocument, Path
from robofab.interfa
import PutFile
from robofab.world i
from ConvertBezier_P


 def __init__(self,
PFont):
   self.shape = shap
   self.points = poi
   self.selectSegmen
   self.PFont = PFon

   self.ContourPoint
   self.FirstMoveTo

   self.path = None
   #### pdf export s
   #self.pdfDoc = No
   self.PSFile =None
   self.w = 842
   self.h = 595
   self.advanceWidth
   self.scale = .05

   self.GlyphCounter

   self.RoboFont = N
   self.newGlyph = N
   self.UFOPen = Non

   self.prepare()
   self.outDraw()
   self.converToBezi
   self.save()

 def prepare(self):

 def outDraw(self):
   for pt in self.po

     if pt == "red":
       self.selectSeg
       continue
     elif pt == "blac
       self.selectSeg
       continue
       w, advance = s
       #if self.pdfDo
       if self.PSFile
         self.advance
```

de bruine beer die de afgelopen weken rondstruinde in het grensgebied van duitsland en oostenrijk is doodgeschoten. dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekendgemaakt. de beer die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in het kader van een programma om gefokte dieren weer in het wild uit te zetten. de beer stak echter de alpen over en dook een aantal keer op in oostenrijk en duitsland.waar hij zich tegoed deed aan schapen. geiten en konijnen en bijenkorven plunderde.omdat het niet lukte

de gay prideoptocht in parijs mondde dit weekeinde uit in de politieke stellingname voor gelijkheid in het jaar van de presidents en parlementsverkiezingen. de franse organisaties van homos lesbiennes en biseksuelen holebis willen in de verkiezingen van april en mei hun electoraal gewicht laten voelen dat betekent dat het huwelijk en adoptierecht van paren van hetzelfde geslacht een thema in de verkiezingscampagnes wordt. de gay prideparade trok zaterdag in parijs honderdduizenden deelnemers de organisatoren spraken van met hun massale

de antwerpse witte pater robert gaul die in rwanda van rassenscheiding beticht wordt zal er wellicht met een boete vanaf komen.dat meldde kerknet vlaanderen maandag onder aanhaling van de rwandese krant the new times de openbare aanklager in gatsibo heeft een boete van miljoen rwandese frank geist.
in een preek zou gaul uitgehaald hebben naar verenigingen van overlevenden van de genocide in die organisaties helpen tutsi s en benadelen de hutu s en krijgen daarvoor geld van de overheid… zou gaul gezegd hebben.
volgens de regeringsgezinde rwandese krant the new times

het zeeuwse mosselseizoen gaat dit jaar officieel van start op juli. volgens de kranten van sud presse kan het echter tot augustus duren voor de eerste mosselen te krijgen zijn en zou de prijs wel eens dubbel zo hoog kunnen liggen dan vorig jaar.
door het koudere voorjaar is de ontwikkeling van de schelpen laat op gang gekomen en is er een vertraging van zon drie tot vier weken.zegt de commercile directeur van prins en dingemanse in de kranten van sud presse.
de prijs zou volgens sud presse schommelen tussen euro en euro voor twee kilo terwijl dat vorig jaar nog tuss

de bruine beer die de afgelopen weken rondstruinde in het grensgebied van duitsland en oostenrijk is doodgeschoten. dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekendgemaakt. de beer die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in het

de antwerpse witte pater robert gaul die in rwanda van rassenscheiding beticht wordt zal er wellicht met een boete vanaf komen. dat meldde kerknet vlaanderen maandag onder aanhaling van de rwandese krant the new times de openbare aanklager in gatsibo heeft een boete van miljoen rwandese frank geist. in een preek zou gaul uitgehaald hebben naar verenigingen van overlevenden van

de gay prideoptocht in parijs mondde dit weekeinde uit in de politieke stelling. name voor gelijkheid in het jaar van de presidents en parle. mentsverkiezingen. de franse organisaties van homos lesbiennes en biseksuelen holebis willen in de verkiez ingen van april en mei hun electoraal gewicht laten voelen dat betekent dat het huwelijk en adop. tierecht van paren van hetzelfde geslacht

het zeeuwse mosselseizoen gaat dit jaar officieel van start op juli.volgens de kranten van sud presse kan het echter tot augustus duren voor de eerste mosselen te krijgen zijn en zou de prijs wel eens dubbel zo hoog kunnen liggen dan vorig jaar. door het koudere voorjaar is de ontwikkeling van de schelpen laat op gang gekomen en is er een vertraging van zon. drie tot vier weken.

```
float(advance)
        #if self.adv
/ self.scale - 1500:
                if self.adva
 self.scale - 800:
        #self.pdfDo
        self.PSFile
l")
        #self.pdfDo
2000)
        self.PSFile
translate\n")
        self.PSFile
        self.advanc
    else:
        #self.pdfDo
advance))
        self.PSFile
translate\n" %float(
    if self.RoboFo
self.newGlyph != Non
        self.newGlyp
float(advance)
        continue
    elif str(pt).fi
op.
if self.RoboFo
        self.newGlyp
newGlyph(glyphName)
        self.UFOPen
getPen()
    continue
    if self.shape ==
        self.drawOval(
    elif self.shape
        self.drawRect(
def converToBezier(
def save(self):
    pass
def drawOval(self,
    pass
def drawRect(self,
    pass
class CocoaOutput(Ba
def drawOval(self,
    bot = pt[0]
    left = pt[1]
    top = pt[2]
    right = pt[3]
    if self.selectSeg
    else:
        NSColor.blackCol
self.path = NSBez
```

```
anceWidth > self.w

nceWidth = self.w
oc.restoreState()
e.write("gsave\n")
oc.translate((0,
e.write("0 -1700
e.write("gsave\n")
ceWidth = 0.0
oc.translate((float(
e.write("%f 0
advance))
ont != None and
e:
h.width =


nd("GlyphName") ==
ont != None:
h = self.RoboFont.
= self.newGlyph.


= "oval":
(pt)
== "rect":
(pt)
self):


pt):


pt):


seOutput):


pt):


ment:


lor().set()

ierPath.bezierPath()
```

de bruine beer die de afgelopen weken verontruinde in het grensgebied van duitsland en oostenrijk is doodgeschoten dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekendgemaakt. de beer. die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in het

de gay prideoptocht in parijs mondde dit weekeinde uit in de politieke stellingname voor gelijkheid in het jaar van de presidents en parlementsverkiezingen. de franse organisaties van homos lesbiennes en biseksuelen holebis willen in de verkiezingen van april en mei hun electoraal gewicht laten voelen .dat betekent dat het huwelijk en adoptierecht van paren van hetzelfde geslacht een thema in de verkiezingscampagnes wordt. de gay pride parade trok zaterdag in parijs honderdduizenden deelnemerse organisatoren spraken van et hun massale

de antwerpse witte pater robert gaul die in rwanda van rassenscheiding beticht wordt zal er wellicht met een boete vanaf komen dat meldde kerknet vlaanderen maandag onder aanhaling van de rwandese krant the new times de openbare aanklager in gatsibo heeft een boete van miljoen rwandese frank geist in een preek zou gaul uitgehaald hebben naar verenigingen van overlevenden van de genocide in die organisaties helpen tutsi's en benadelen de hutu's en krijgen daarvoor geld van de overheid'" zou gaul gezegd hebben' volgens de regeringsgezinde rwandese krant the new times

bullhead bam handicapping encamp paddled nipping anna chalcidoid acone cup capo cadillac pule a monopodial eloped um quid gum en oh ha pilonidal bounden gluconeogenic pelade cad hen bug pigdom ha bdelloid lampic manoc emendable nincompoophood umm non hi pee acme alima cnidophobia caingin meningioma bebog philhellenic quid ban anigh cede cool lip cull abiological appealing unalienable manducable

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing
unalienable manducable

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing
unalienable manducable

```
        self.path.moveToP
bot[0][1]))
        self.path.curveTo
controlPoint2_((left
left[0][1]), (bot[1]
left[2][0], left[1]
        self.path.curveTo
controlPoint2_((top[
(left[2][0], left[2]
top[1][1]))
        self.path.curveTo
controlPoint2_((righ
right[0][1]), (top[2
right[2][0], right[
        self.path.curveTo
controlPoint2_((bot[
right[1][0], right[
bot[2][1]))
        self.path.fill()

    def drawRect(self,
        BotLeft = pt[0]
        BotRight = pt[1]
        TopLeft = pt[2]
        TopRight = pt[3]

        if self.selectSeg
            NSColor.redColor
        NSColor.blackCol

        self.path = NSBez

        self.path.moveToP
        BotLeft[1]))
        self.path.lineToP
    BotRight[1]))
        self.path.lineToP
    TopRight[1]))
        self.path.lineToP
    TopLeft[1]))
        self.path.lineToP
    BotLeft[1]))

class CocoaOutputCon
    def prepare(self):
        self.path = NSBez
        NSColor._strokewi
        NSColor.greenColo
        NSColor.colorWith
    yellow_black_alpha_(
        fill()

    def drawOval(self,
        instruction, p =

        if instruction ==
            #if not self.Gly
            # self.path = NS
bezierPath()
```

oint_((bot[0][0],

Point_controlPoint1_
[0][0], bot[1][1]),
[1]))
Point_controlPoint1_
0][0], top[0][1]),
[1]), (top[1][0],

Point_controlPoint1_
t[0][0],
][0], top[2][1]),
2][1]))
Point_controlPoint1_
0][0], bot[0][1]),
1][1]), (bot[2][0],

pt):

ment:
r().set()
lor().set()

ierPath.bezierPath()
oint_((BotLeft[0],
oint_((BotRight[0],
oint_((TopRight[0],
oint_((TopLeft[0],
oint_((BotLeft[0],

tour(BaseOutput)

ierPath.bezierPath()
dth = 50
r().setStroke()
DeviceCyan_magenta_
0.8, 0, 1, 0, 0.5)]

pt):
pt

"moveTo":
yphCounter:
sBezierPath.

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing

bullhead bam
handicapping encamp
paddled nipping anna
chalcidoid acone cup
capo cadillac pule a
monopodial eloped
um quid gum en oh
ha pilonidal bounden
gluconeogenic pelade
cad hen bug pigdom
ha bdelloid lampic
manoc emendable
nincompoophood umm
non hi pee acme alima
cnidophobia caingin
meningioma bebog
philhellenic quid ban
anigh cede cool lip cull
abiological appealing

unalienable manducable

bullhead bam handicapping encamp paddled nipping anna chalcidoid acone cup capo cadillac pule a monopodial eloped um quid gum en oh ha pilonidal bounden gluconeogenic pelade cad hen bug pigdom ha bdelloid lampic manoc emendable nincompoophood umm non hi pee acme alima cnidophobia caingin meningioma bebog philhellenic quid ban anigh cede cool lip cull abiological appealing

bullhead bam handicapping encamp paddled nipping anna chalcidoid acone cup capo cadillac pule a monopodial eloped um quid gum en oh ha pilonidal bounden gluconeogenic pelade cad hen bug pigdom ha bdelloid lampic manoc emendable nincompoophood umm non hi pee acme alima cnidophobia caingin meningioma bebog philhellenic quid ban anigh cede cool lip cull abiological appealing

bullhead bam handicapping encamp paddled nipping anna chalcidoid acone cup capo cadillac pule a monopodial eloped um quid gum en oh ha pilonidal bounden gluconeogenic pelade cad hen bug pigdom ha bdelloid lampic manoc emendable nincompoophood umm non hi pee acme alima cnidophobia caingin meningioma bebog philhellenic quid ban anigh cede cool lip cull abiological appealing unalienable manducable

het zeeuwse mosselseizoen gaat dit jaar officieel van start op juli volgens de kranten van sud presse kan het echter tot augustus duren voor de eerste mosselen te krijgen zijn en zou de prijs wel eens dubbel zo hoog kunnen liggen dan vorig jaar. door het koudere voorjaar is de ontwikkeling van de schelpen laat op gang gekomen en is er een vertraging van zon drie tot vier weken. zegt de commercile directeur van prins en dingemanse in de kranten van sud presse. de prijs zou volgens sed

```
            self.path.moveTo
        p[1]))
            self.FirstMoveTo
        elif instruction
        elif instruction
            self.path.curveT
        controlPoint1_contro
        p[5]), (p[0], p[1]),

        elif instruction
        instruction == "coun
            self.path.closeP
            #self.path.lineT
        FirstMoveTo)
            #self.path.setLi
            #self.path.fill(
            #self.path.strok
            #self.GlyphCount
    def drawRect(self,
        self.drawOval(pt)

    def save(self):
        self.path.setLineT
        self.path.fill()
        self.path.stroke(
class CGOutput(BaseO
    def prepare(self):
        #if not self.pdfD
        #self.pdfDoc.sca
    self.scale))
        #self.pdfDoc.tra
    scale, self.h/self.s
    scale))
        #self.pdfDoc.sav
        self.dst = PutFil
    "InterPenator PDF Ou
        FileName = self.d
        self.PSFile = ope
        self.PSFile.writ
    n" %(0, self.h))
        self.PSFile.write
    %(self.scale, self.s
        self.PSFile.write
    n" %(20/self.scale,
    200/self.scale))
        self.PSFile.write

    def save(self):
        #self.pdfDoc.
    setStrokeCMYK((0,0,0
        #self.pdfDoc.setS
        #self.pdfDoc.appl
        #dst = PutFile("S
    "InterPenator PDF Ou
        #if dst is not No
        # self.pdfDoc.save
        # import findertoo
        # findertools.laun
        self.PSFile.write
        self.PSFile.close
        if self.dst is no
            import findertoo
            findertools.laun
        self.PSFile = Non
```

```
oPoint_((p[0],

o = (p[0], p[1])
== "lineTo":
== "curveTo":
ToPoint_
lPoint2_(p[1],
 (p[2], p[3]))

== "closePath":
ter":
Path()
ToPoint_(self.

ineWidth_(5)
()
ke()
ter = False
pt):

Width_(5)
)
utput):

oc:
ale((self.scale,

anslate((20/self.
cale - 100/self.

veState()
e("Save the pdf:"
tput")
st + ".ps"
n(FileName, "w")
e(" %f %f translate\

("%f %f scale\n"
cale))
("%f %f translate\
self.w/self.scale

("1 setlinewidth\n")

,1,1))
trokeWidth(2)
yFillStroke()
ave the pdf:"
tput")
ne:
e(dst)
ols
nch(dst+".pdf")
("showpage\n")
()
t None:
ols
nch(self.dst+".ps")
e
```

de bruine beer die de afgelopen weken rondstruinde in het grensgebied van duitsland en oostenrijk is doodgeschoten dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekendgemaakt de beer die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in het kader van een programma om gefokte dieren weer in het wild uit te zetten de beer stak echter de alpen over en dook een aantal keer op in de antwerpse witte pater robert gaul die in rwanda van rassenscheiding beticht wordt, zal er vrijdag met een boete vanaf komen dat meldde de rwandese krant the new times de openbare aanklager in gatsibo heeft een boete van miljoen rwandese frank geist, in een preek zou gaul uitgehaald hebben naar verenigingen van overlevenden van de genocide in die organisaties helpen tutsis en benadelen de hutus

de gay prideoptocht in par. ijs mondde dit weekeinde uit in de politieke stelling. name voor gelijkheid in het jaar van de presidents en parlementsverkiezingen. de franse organisaties van homos lesbiennes en biseksuelen holebis wil. len in de verkiezingen van april en mei hun electoraal gewicht laten voelen dat betekent dat het huwelijk en adoptierecht van paren van hetzelfde geslacht een thema in de verkiezing. scampagnes wordt. de gay prideparade trok zaterdag in parijs honderd.

het zeeuwse mosselseizoen gaat dit jaar officieel van start op juli volgens de kranten van sud presse kan het echter tot augustus duren voor de eerste mosselen te krijgen zijn en zou de prijs wel eens dubbel zo hoog kunnen liggen dan vorig jaar. door het koudere voorjaar is de ontwikkeling van de schelpen laat op gang gekomen en is er een vertraging van zon drie tot

de bruine beer die de afgelopen weken rondstruinde in het grensgebied van duitsland en oostenrijk is doodgeschoten dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekendgemaakt de beer die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in

de gay prideoptocht in parijs mondde dit weekeinde uit in de politieke stelling: name voor gelijkheid in het jaar van de presidents en parlementsverkiezingen de franse organisaties van homoseksuelen holebis willen in de verkiezingen van april en mei hun gewicht voelen dat betekent dat het huwelijk en adoptierecht van paren van hetzelfde

de antwerpse witte pater robert gaul die in rwanda van rassenscheiding beticht wordt, zal er wellicht met een boete vanaf komen dat meldde kerknet vlaanderen maandag onder aanhaling van de rwandese krant the new times de openbare aanklager in gatsibo heeft een boete van miljoen rwandese frank geist, in een preek zou gaul uitgehaald hebben naar verenigingen van overlevenden

het zeeuwse mosselseizoen gaat dit jaar officieel van start op juli volgens de kranten van sud presse kan het echter tot augustus duren voor de eerste mosselen te krijgen zijn en zou de prijs wel eens dubbel zo hoog kunnen liggen dan vorig jaar. door het koudere voorjaar is de ontwikkeling van de schelpen laat op gang gekomen en is er een vertraging van zon drie tot vier weken. zegt de commercile directeur van prins en dingemanse in de kranten van sud presse

```
def drawOval(self,
  bot = pt[0]
  left = pt[1]
  top = pt[2]
  #p = path()
  #p.moveTo((bot[0]
  #p.curveTo((bot[1
  left[1][0], left[1]
  left[0][1]))
  #p.curveTo((left[
  (top[1][0], top[1][1
  top[0][1]))
  #p.curveTo((top[2
  (right[2][0], right[
  right[2][1]), (right[
  right[1][1]), (bot[2
  (bot[1][0], bot[0][1
  #p.addToContext(s
  self.PSFile.write
  self.PSFile.write
  (bot[0][0], bot[0][1
  self.PSFile.write
  curveto \n" % (bot[1
  left[1][0], left[1][
  left[0][1]))
  self.PSFile.write
  curveto \n" % (left[
  top[1][0], top[1][1
  top[0][1]))
  self.PSFile.write
  curveto \n" % (top[2
  right[2][0], right[2
  right[0][1]))
  self.PSFile.write
  curveto stroke\n"
  right[1][1], bot[2][
  bot[0][0], bot[0][1
  self.PSFile.write

  def drawRect(self,
  BotLeft = pt[0]
  TopLeft = pt[2]
  TopRight = pt[3]
  #p = path()
  #p.moveTo((BotLef
  #p.lineTo((BotRig
  #p.lineTo((TopRig
  #p.lineTo((TopLef
  #p.lineTo((BotLef
  #p.addToContext(s
  self.PSFile.write
  self.PSFile.write
  (BotLeft[0], BotLeft
  self.PSFile.write
  (BotRight[0], BotRig
  self.PSFile.write
  (TopRight[0], TopRig
```

de bruine beer die de af,
gelopen weken rondstruinde
in het grensgebied van
duitsland en oostenrijk is
doodgeschoten dat hebben
de autoriteiten in beieren
maandag zonder nadere
toelichting bekendgemaakt
de beer die door de pers
bruno werd gedoopt werd
enige tijd geleden in de
italiaanse alpen losgelaten
in het kader van een pro,
gramma om gefokte dieren
weer in het wild uit te
zettende beer stak echter
de alpen over en dook een
aantal keer op in oostenrijk
en duitsland, waar hij zich

de gay prideoptocht in
parijs mondde dit week-
einde uit in de politieke
stellingname voor gelijkheid
in het jaar van de presi-
dents en parlementsverkiez-
ingen de franse organisaties
van homos lesbiennes en
biseksuelen holebis wil-
len in de verkiezingen van
april en mei hun elec-
toraal gewicht laten voe-
len dat betekent dat het
huwelijk en adoptierecht
van paren van hetzelfde
geslacht een thema in de
verkiezingscampagnes wordt
de gay pride- parade trok
zaterdag in parijs honderd-

de antwerpse witte pater
robert gaul die in rwanda
van rassenscheiding beticht
wordt zal er wellicht met
een boete vanaf komen dat
meldde kerknet vlaanderen
maandag onder aanhal.
ing van de rwandese krant
the new times de openbare
aanklager in gatsibo heeft
een boete van miljoen
rwandese frank geist.
in een preek zou gaul
uitgehaald hebben naar
verenigingen van overlev.
enden van de genocide in
die organisaties helpen tut.
sis en benadelen de hutus
en krijgen daarvoor geld
van de overheid zou gaul

het zeeuwse mosselseizoen
gaat dit jaar officieel van
start op juli volgens de
kranten van sud presse kan
het echter tot augustus
duren voor de eerste mos,
selen te krijgen zijn en zou
de prijs wel eens dubbel
zo hoog kunnen liggen dan
vorig jaar door het koudere
voorjaar is de ontwikkeling
van de schelpen laat op
gang gekomen en is er een
vertraging van zon drie
tot vier weken, zegt de
commercile directeur van
prins en dingemanse in de
kranten van sud presse,

de prijs zou volgens sud

de bruine beer die de afgelopen weken rondstruinde in het grensgebied van duitsland en oostenrijk is doodgeschoten dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekendgemaakt de beer die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in het kader van een programma om gefokte dieren weer in het wild

de antwerpse witte pater robert gaul die in rwanda van rassensc, heiding beticht wordt, zal er wellicht met een boete vanaf komen dat meldde kerknet vlaan, deren maandag onder aanhaling van de rwan, dese krant the new times de openbare aan, klager in gatsibo heeft een boete van miljoen rwandese frank geist, in een preek zou gaul uitgehaald hebben naar verenigingen van overlevenden van de genocide in die organi, saties helpen tutsis en

de gay prideoptocht in parijs mondde dit weekeinde uit in de politieke stellingname voor gelijkheid in het jaar van de presidents en parlementsverkiez ingen de franse or ganisaties van homos lesbiennes en bisek suelen holebis willen in de verkiezingen van april en mei hun elec toraal gewicht laten voelen dat betekent dat het huwelijk en adop tierecht van paren van hetzelfde geslacht een thema in de verkiez

het zeeuwse mosselseizoen gaat dit jaar officieel van start op juli volgens de kranten van sud presse kan het echter tot augustus duren voor de eerste mosselen te krijgen zijn en zou de prijs wel eens dubbel zo hoog kunnen liggen dan vorig jaar door het koudere voorjaar is de ontwikkeling van de schelpen laat op gang gekomen en is er een vertraging van zon drie tot vier weken zegt de commerciele directeur van prins en dingemanse in de kranten van sud presse de prijs zou volgens sud presse schommelen tussen . euro en euro voor twee kilo. terwijl dat vorig jaar nog tussen

```
            self.PSFile.write
        (TopLeft[0], TopLeft
            self.PSFile.write
        stroke\n" % (BotLeft
            self.PSFile.write
class CGOutputContou
    def drawOval(self,
        instruction = pt[
        p = pt[1]

        #self.PSFile.writ
        if instruction ==
            if not self.Glyp
            self.PSFile.wr
            "
            self.FirstMoveTo
        elif instruction
            self.PSFile.writ
        % (p[0], p[1]))
        elif instruction
            self.PSFile.writ
        %r %r curveto\n" % (
        p[3], p[4], p[5]))
        elif instruction
            self.PSFile.writ
        % self.FirstMoveTo)
            self.PSFile.writ
            self.GlyphCounte
    elif instruction
        self.PSFile.writ
        n")
        self.PSFile.writ
        self.GlyphCounte
        #self.PSFile.writ
    def drawRect(self,
        self.drawOval(pt)
class UFOOutput(Base
    def prepare(self):
        self.RoboFont.inf
PFont.familyName
        self.RoboFont.inf
PFont.styleName
        self.RoboFont.inf
PFont.xHeight
        self.RoboFont.inf
PFont.ascender
        self.RoboFont.inf
PFont.descender
    def save(self):
        self.dst = PutFil
        #interpolator UFO Ou
        if self.dst:
            self.RoboFont.sa
    def drawOval(self,
        bot = pt[0]
        left = pt[1]
        top = pt[2]
        right = pt[3]
```

("%f %f lineto\n" %
[1]))
("%f %f lineto
[0], Both[1]))
("grestore\n")

r(CGOutputPen):

pt):
0]

e("gsave\n")
"moveTo":
phCounter:
rite("gsave newpath\

o = (p[0], p[1])
== "lineTo":
te("%f %f lineto\

== "curveTo":
te("%f %f %f %f
p[0], p[1],

== "counter":
te("%f %f lineto\

te("closepath\n")
er = True
== "closePath":
te("closepath fill\

te("grestore\n")
er = False

e("grestore\n")
pt):

Output):

o.familyName = self.

o.styleName = self.

o.xHeight = self.

o.ascender = self.

o.descender = self.

e("Save as .UFO:",
tput.ufo")

ave(self.dst)
pt):

de bruine beer die de afgelopen weken rondstruinde in het grens gebied van duitsland en oostenrijk is doodgeschoten dat hebben de autoriteiten in beieren maandag zonder nadere toelichting bekend gemaakt

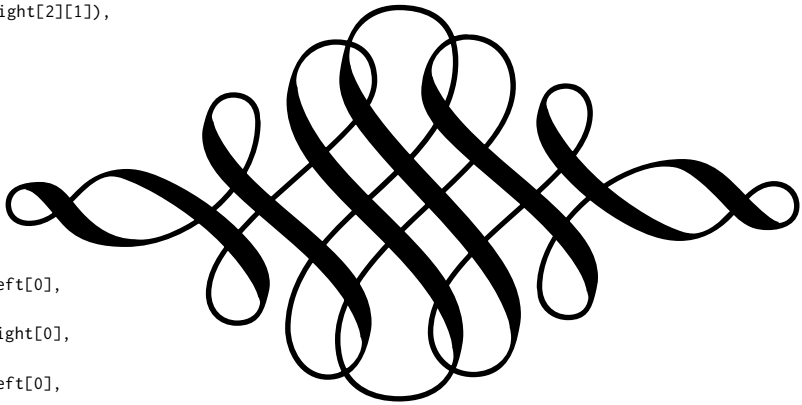
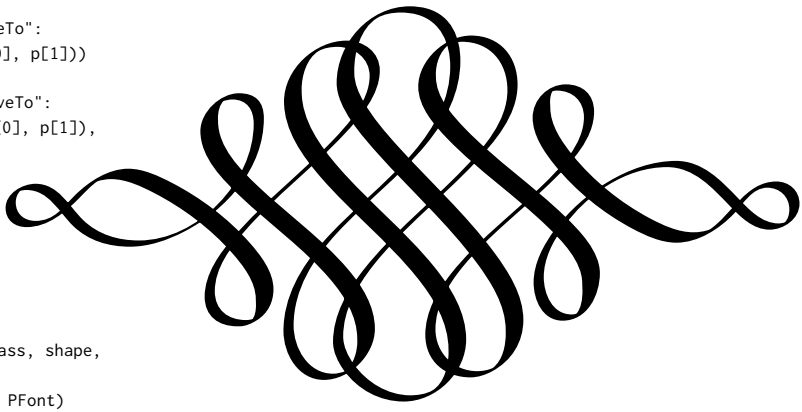
de beer die door de pers bruno werd gedoopt werd enige tijd geleden in de italiaanse alpen losgelaten in het kader van een programma om gefokte dieren weer in het wild uit te zetten

de beer stak echter de alpen over en dook een aantal keer op in oostenrijk en duitsland waar hij zich tegoed deed aan schapen geiten en konijnen en bijenkorven plunderde omdat het niet lukte

de gay prideoptocht in parijs mondde dit weekeinde uit in de politieke stellingname voor gelijkheid in het jaar van de presidents en parlementsverkiezingen de franse organisaties van homos lesbiennes en biseksuelen holebis willen in de verkiezingen van april en mei hun electoraal gewicht laten voelen dat betekent dat het huwelijk en adoptierecht van paren van hetzelfde geslacht een thema in de verkiezingscampagnes wordt.

de gay prideparade trok zaterdag in parijs honderdduizenden deenemers de organisatoren spraken van met hun massale opkomst demonstreerden de franse gays dat zij volgend jaar een belangrijk

```
   self.UFOPen.moveT
bot[0][1]))
   self.UFOPen.curve
bot[1][1]), (left[1]
(left[0][0], left[0]
   self.UFOPen.curve
left[2][1]), (top[1]
(top[0][0], top[0][1
   self.UFOPen.curve
top[2][1]), (right[2
(right[0][0], right[
   self.UFOPen.close

 def drawRect(self,

   BotLeft = pt[0]
   BotRight = pt[1]
   TopLeft = pt[2]
   TopRight = pt[3]

   self.UFOPen.moveT
BotLeft[1]))
   self.UFOPen.lineT
TopRight[1]))
   self.UFOPen.lineT
TopLeft[1]))
   self.UFOPen.lineT
BotLeft[1]))
   self.UFOPen.close

class UFOOutputConto

 def drawOval(self,
   instruction = pt[
   p = pt[1]
   if instruction ==
     self.UFOPen.move
     self.FirstMoveTo

   elif instruction
     self.UFOPen.line

   elif instruction
     self.UFOPen.curv
(p[2], p[3]), (p[4],

     self.UFOPen.clos

 def drawRect(self,
   self.drawOval(pt)


def PFDrawInOutPut(o
points, PFont = None
 outputClass(shape,




###################
```

```
o((bot[0][0],

To((bot[1][0],
[0], left[1][1]),
[1]))
To((left[2][0],
[0], top[1][1]),
]))
To((top[2][0],
][0], right[2][1]),
0][1]))
Path()

pt):



o((BotLeft[0],

o((TopRight[0],

o((TopLeft[0],

o((BotLeft[0],

Path()

ur(UFOOutput):

pt):
0]

"moveTo":
eTo((p[0], p[1]))
o = (p[0], p[1])

== "lineTo":
eTo((p[0], p[1]))

== "curveTo":
veTo((p[0], p[1]),
p[5]))

sePath()

pt):



utputClass, shape,
):
points, PFont)
```