HOW TO DRAW A DUCK IN GROFF WITH GREMLIN AND PIC

dozens

November 20, 2023

CONTENTS

TABLES

FIGURES

1. PIC

There is a really cool project called tikzducks[1] that is basically a bunch of different ducks drawn in LaTeX. They are hecking cute and you should go give them a look. They are drawn using the TikZ[2] package, which is a LaTeX package used for drawing complex pictures, charts, and figures.

Naturally, when I see something done in LaTeX, I am immediately tempted to attempt to recreate it in groff. Unfortunately, groff lacks any kind of sophisticated picture capabilities. It provides only very primitive picture capabilities via the pic language and preprocessor.[3]

For example, here is some pic code that will produce a small chicken-like object made from a spline, and a few circles, arcs, and lines.

```
1  | # head
2  | spline right then up then left then down
3  | # eyes
4  | fillval = 1
5  | circle fill rad 0.05 at 0.2,0.2
6  | circle fill rad 0.05 at 0,0.2
7  | # body
8  | arc from 0.25,0 to 1,0
9  | arc to 0.50,0.30
10 | # legs
11 | line at 1st arc .s - (0.02, 0) down 0.2 left 0.02 then left 0.1
12 | line at 1st arc .s + (0.04, 0) down 0.2 right 0.02 then right 0.1
```

Table 1: A pic program that creates a small duckling

The resulting duck is very small and cute. Also it is basically a baby chicken.

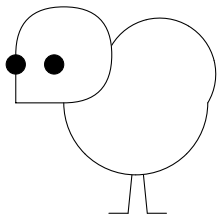


Figure 1: A baby Pic chicken

2. GREMLIN

There is a second, more obscure and archaic preprocessor for the already relatively obscure and archaic groff called grn. It is the gremlin preprocessor, and it will process gremlinfiles. Gremlin caught my eye because it supports bezier curves, which will allow us to draw a much nicer duck.

grn is apparently only available while using the me macro set.[4] So that's something to know. In my experience, the ms macros seem to be the most popular, followed by the mom macro set. I certainly wasn't familiar with 'me'. I had to learn it just to produce this document.

---

[1] https://github.com/samcarter/tikzducks

[2] https://tikz.dev/

[3] https://pikchr.org/home/uv/pic.pdf

[4] https://www.man7.org/linux/man-pages/man7/groff_me.7.html

Here is the output of the example gremlinfile from the grn manpage.

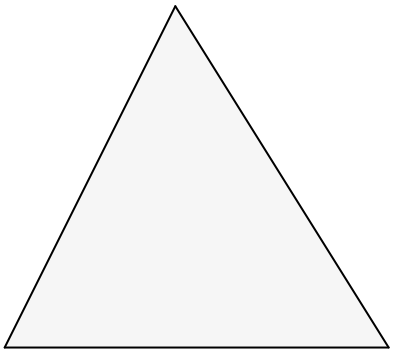


Figure 2: A Gremlin Triangle

Gremlinfiles are super weird. As far as I can tell, GREMLIN is an old graphics editor from 1981.[5] And I guess these gremlinfiles are maybe their save file format? Anyway, somebody added gremlin support to groff via the *grn* preprocessor, and it has just quietly stuck around all this time. There are some Pic tutorials out there. But I can find basically nothing on Gremlin.

A gremlinfile starts with two lines: "sungremlinfile" (or "gremlinfile"), and "1 0 0". (See lines 1 - 2 on Table 2.) sungremlinfile is the SUN workstation version of the file format, whereas "gremlinfile" is the format used for the AED graphics terminal.[6] I chose to go with the SUN version because you can use element names when defining elements (see below), whereas with the AED version, you can only refer to elements by id number in the gremlinfile.

1	sungremlinfile
2	1 0 0
3	POLYGON
4	224 416
5	96 160
6	384 160
7	*
8	3 3
9	0
10	CENTCENT
11	240 128
12	*
13	2 3
14	10 A Triangle
15	-1

Table 2: An Example Gremlinfile. It makes a triangle and a label that says 'Triangle'

[5] A GREMLIN Tutorial for the SUN Workstation. Opperman, Thompson, Chen. Appendix A. https://www2.eecs.berkeley.edu/Pubs/TechRpts/1987/CSD-87-322.pdf

And I didn't feel like looking up element ids in order to use them. The manual literally says of the "1 0 0" line: "The stuff on this line isn't all that important; We suggest using 1 0.00 0.00." So that's neat. I guess we can just ignore what that line means. A gremlinfile always ends with a "-1" on a line. (See line 15 on Table 2.)

Following the opening pair of lines are a series of elements. Element names are followed by a series of coordinates, followed by an asterisk (or a "-1 -1" in the AED gremlinfile; don't know why), a "brush / size" line, and a text label line. See lines 3 - 9, and 10 - 14.

The coordinates mean different things depending on the element. For example, for polygons, they're coordinates of the corners of the polygon. And for arcs and circles, they're coordinates for the center and radius of the circle arc.

The "brush / size" line can also mean different things. For text elements, brush means bold, italic, regular text, etc. For polygons, size means fill pattern. So you really just gotta have the manual out when you're doing stuff I guess.

The text label line is straightforward, but unusual by today's standards. If you're gonna have text, you gotta put how many characters long the string is, and then the string. See line 14. And if you don't have a string, then you have to put 0 to indicate a zero length string. Like on line 9. Isn't that funny!

Anyway, let's make a gremlin duck! I'm just going to copy the LaTeX curves from the answer on stackexchange[7] and tweak them a little bit and see how it looks. (You can see all of the gremlin that creates this duck in Appendix A; It is too long to include here.)

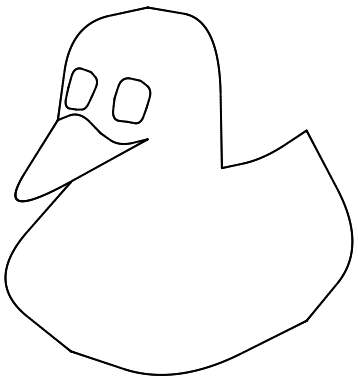


Figure 3: A Gremlin duck

It is kind of ugly, but also weirdly cute! I expected the bezier curves to be more smooth and curvy. I feel like I could have gotten something almost as good just using Pic splines.

3. PIC REDUX

So maybe let's just try this again with pic arcs and splines?

---

[6] It is difficult to find info on the AED graphics terminal thanks to the ubiquity of the Automated External Defibrillator. But check out these dope brochures! http://bitsavers.org/pdf/aed/brochures/

[7] https://tex.stackexchange.com/a/471540

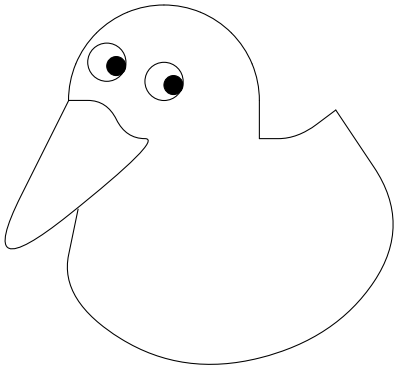


Figure 4: Pic duck 2.0.

You may feel that the lines are not quite right. But they are. I just drew them myself.[8]

This honestly looks much better to me. The lines are sharper and the curves are smoother. Pic is also much nicer to write than gremlin. You can write comments for one. You can write macros, loops, and conditionals. And I find the Logo-esque[9] "up, down, then..." syntax to be much easier to reason about than gremlin's coordinate based syntax.

Here are the pic commands that created this duck.

```
1  .PS 2
2  # BEAK
3  spline left .5 down 1 \
4     then right 1 up .8 \
5     then left .2 \
6     then left .1 up .2 \
7     then to (0,0)
8  # HEAD
9  arc cw rad .5 from 1st spline
10 arc cw rad .5
11 line down .2
12 #BODY
13 spline right .2 then right .2 up .15
14 spline down .6 right .4 \
15    then down .6 left .4 \
16    then left .8 down .2 \
17    then left .65 up .4 \
18    then up .48 right .1
19 # EYE MACRO
20 copy thru {
21    circle rad .1 at $1, $2
22    circle fill rad .05 at $1+.05, $2-.02
23 }
24 .5 .1
25 .2 .2
26 .PE
```

Table 3: Pic Duck 2.0.

[8] Apologies to George Harrison. But it's only a Northern Duck.

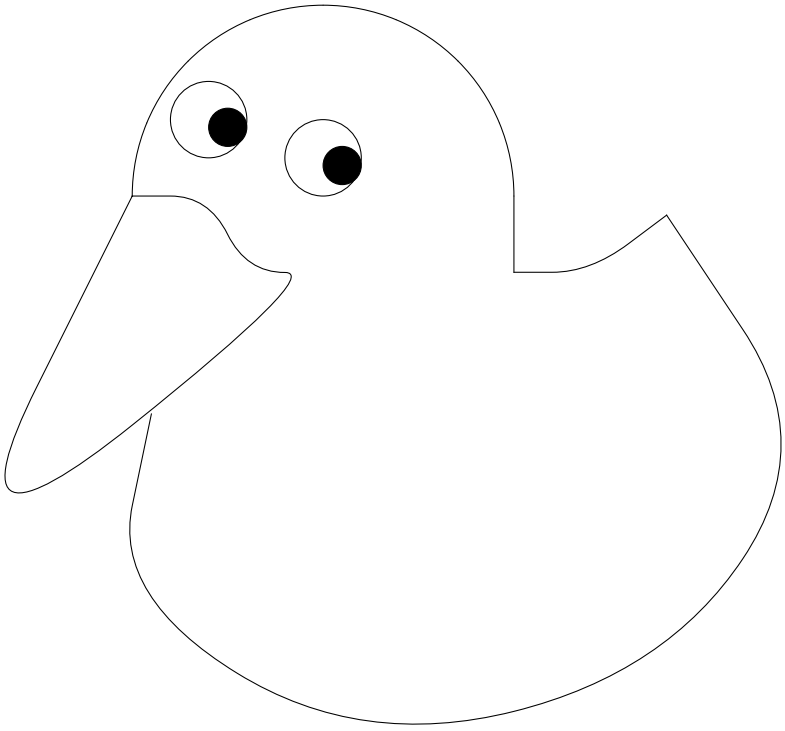
[9] https://en.wikipedia.org/wiki/Logo_(programming_language)

4. CONCLUSION

Gremlin is the only way that I know to get actual bezier curves out of the box using a groff preprocessor. It is possibly the most obscure of groff's preprocessors. And I don't think it looks that good honestly. I think there is little reason to use it other than as a curious novelty or an academic exercise. At least as far as drawing ducks is concerned. The format is kind of arbitrary and inconsistent. And its use of bare coordinates really does make it feel as though it is meant to be autogenerated from a gui graphics program, and not really meant to be written by hand. Especially when compared to Pic's much more ergonomic "left then right then down" syntax and its ability to reference previous elements by name.

Having tried it, I would ultimately recommend just using pic splines if you need pictures with curves. Or just making something in Inkscape. (You can convert the svg to eps with imagemagick, and then embed it in your groff with the .PSPIC macro.)

Okay that's it! Thanks for learning all about drawing ducks in groff with me! Bye!



APPENDIX A: DUCK GREMLINFILE

Here is the entire contents of the gremlinfile that produced the duck. It is not very interesting to read because it is just a solid wall of CURVE BEZIER elements. I don't know how to include comments in the gremlinfile itself, but I have labeled the element sections here for the sake of legibility.

```
1   sungremlinfile        header content
2   1 0 0
3   CURVE BEZIER          Start beak section
4   0.65 2.95
5   -0.1 1.75
6   1.68 2.73
7   *
8   3 3
9   0
10  CURVE BEZIER
11  0.65 2.95
12  0.9 3.05
13  1.14 2.80
14  *
15  3 3
16  0
17  CURVE BEZIER
18  1.14 2.80
19  1.35 2.65
20  1.68 2.73
21  *
22  3 3
23  0                     End Beak Section
24  CURVE BEZIER          Start Head section
25  0.65 2.95
26  0.8 4.05
27  1.68 4.24
28  *
29  3 3
30  0
31  CURVE BEZIER
32  1.68 4.24
33  2.5 4.1
34  2.53 2.4
35  *
36  3 3
37  0                     End Head section
38  CURVE BEZIER          Start body
39  0.8 2.24
40  -0.2 1.1
41  0.8 0.3
42  *
43  3 3
44  0
45  CURVE BEZIER
46  0.8 0.3
47  2 -0.1
```

```
3.5 0.65
*
3 3
0
CURVE BEZIER
3.5 0.65
4.2 1.5
3.5 2.83
*
3 3
0
CURVE BEZIER
3.5 2.83
3 2.5
2.53 2.4
*
3 3
0                       End body
CURVE BEZIER            Right eye
1.3 3.2
1.38 3.45
1.6 3.4
*
3 3
0
CURVE BEZIER
1.6 3.4
1.73 3.3
1.67 3.1
*
3 3
0
CURVE BEZIER
1.67 3.1
1.6 2.9
1.45 2.93
*
3 3
0
CURVE BEZIER
1.45 2.93
1.27 2.95
1.3 3.2
*
3 3
0                       End right eye
CURVE BEZIER            Left eye[10]
0.77 3.32
0.84 3.57
1.03 3.5
*
3 3
0
CURVE BEZIER
1.03 3.5
1.12 3.4
1.04 3.22
*
```

```
106 | 3 3
107 | 0
108 | CURVE BEZIER
109 | 1.04 3.22
110 | 0.97 3.07
111 | 0.87 3.07
112 | *
113 | 3 3
114 | 0
115 | CURVE-BEZIER
116 | 0.87 3.07
117 | 0.72 3.09
118 | 0.77 3.32
119 | *
120 | 3 3
121 | 0                    End left eye
122 | -1                   end gremlinfile
```

---

[10] Rest in peace, Lisa "Left Eye" Lopez