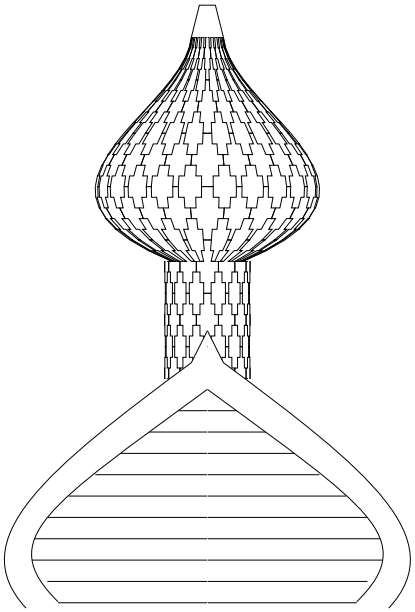
И. В. Романовский

АНАЛИЗ СТАРОЙ ПРОГРАММЫ НА ПОСТСКРИПТЕ



Санкт-Петербург, 2014

Введение

Довольно часто приходится разбираться в старых программах для того, чтобы перевести их на новый язык, модифицировать или использовать старый опыт.

Программы на языке Постскрипт находятся в особом положении: часто готовая программа нарочито сжимается, чтобы занимать меньше места. Рассматриваемая здесь программа как раз такого типа: в ходе написания формат программы сильно менялся. Мне хочется показать, как нужно обращаться с такой программой.

Нужно иметь в виду, что в Постскрипте легко добавлять еще и специальные средства для того, чтобы выделять отдельные части рисунка.

Вначале немного о сюжете. Изображена схема построения "церковной маковки" — небольшой башенки для креста. Эта башенка состоит из основы, которая часто называется **бочкой** или по-английски **cask**, сравнительно небольшого возвышения, конического или цилиндрического, которое называется **барабаном** (**drum**), самой маковки, которая часто называется и **луковицей** (**onion**), и сооружения наверху, обычно это крест. Мы крест изображать не стали, сделали условную шапочку, которую назвали просто **top**.

Эти названия мы будем использовать в идентификаторах.

1. Упрощение программы, выделение стандартной части

Выпишем снова программу, нумеруя строки

```
00  %!
01  /X{exch}def /bp{0 0 m}def /m{moveto}def /l{lineto}def
02  /rl{rlineto}def /rm{rmoveto}def /hrl{0 rl}def /hm{0 m}def
03  /vrl{0 X rl}def /vm{0 X m}def /hrm{0 rm}def /vrm{0 X rm}def
04  /c{curveto}def /CL{closepath}def /CP{currentpoint}def
05  /CNP{clip newpath}def /GS{gsave}def /GR{grestore}def
       /S{stroke}def

06  /onisk{108 sub -41 div dup dup -97 mul 99 add mul
       4 add mul 3 add}def
07  /onichain{3 3 90 {sin dup CP dup ohCH add dup onisk
       6 2 roll dup onisk 4 -1 roll mul X l pop mul X l}for S}def
08  /oniL{GS 0 66 m 42 vrl 32 hrl -42 vrl CL CNP 0 X m
       onichain GR}def
09  /drumchain{5 5 90
      {sin drumR mul CP X pop l dhCH vrl}for S}def
10  /drumL{GS 0 44 m 22 vrl 12 hrl -22 vrl CL CNP 0 X m
      drumchain GR}def
11  /casklines{7 12 17 21 26 30 33 33 30 33
      2 4 38 {bp vrm hrl S} for}def
12  /top{3 108 m -1.5 6 rl -1.5 hrl}def
```

```
13  /Rside{GS /drumR 8 def /dhCH -2 def /ohCH -2 def
14  40 10 100 {drumL} for /dhCH dhCH neg def  10 10 70 {drumL} for
15  70 10 180 {oniL} for  /ohCH ohCH neg def  10 10 100 {oniL} for
16  0 53 m 3 -6 rl 30 27 46 14 34 1 c
17  GS 0 0 l CL 1 setgray fill GR S
18  GS bp 0 42 rl 30 20 40 14 28 2.2 c -2.2 vrl CL
19  CNP bp casklines GR
20  0 42 m 30 20 40 14 28 2.2 c S top S GR}def

21  240 300 translate 2.5 2.5 scale 0.01 setlinewidth
22  Rside -1 1 scale Rside
23  showpage
```

Всего 25 строчек. Нулевая строка — это стандартный "управляющий комментарий", за этой строкой идут 5 строк с экономным написанием самых важных команд, эти стандартные описания вводятся "для непонятности" — для еще большего сокращения текста программы.

```
00  %!
A011  /X{exch}def                 синоним  exch
A012  /bp{0 0 m}def               встать в начало координат
A013  /m{moveto}def               синоним  moveto
A014  /l{lineto}def               синоним  lineto
A021  /rl{rlineto}def             синоним  rlineto
A022  /rm{rmoveto}def             синоним  rmoveto
A023  /hrl{0 rl}def               команда гориз. относит. рисования
A024  /hm{0 m}def                 команда гориз. сдвига
A031  /vrl{0 X rl}def             команда верт.  относит. рисования
A032  /vm{0 X m}def               команда верт.  сдвига
A033  /hrm{0 rm}def               команда гориз. относит. сдвига
A034  /vrm{0 X rm}def             команда верт.  относит. сдвига
A041  /c{curveto}def              построение кривой Безье
A042  /CL{closepath}def           замыкание пути
A043  /CP{currentpoint}def        запись в стек координат текущей точки
A051  /CNP{clip newpath}def       объявление имеющегося контура границей
                                  изображения и начало нового контура
A052  /GS{gsave}def               сохран. графической обстан. в стеке
A053  /GR{grestore}def            возврат из стека сохр. обстан.
A054  /S{stroke}def               прорисовка пути
```

Сейчас я бы добавил к этому списку еще кое-что

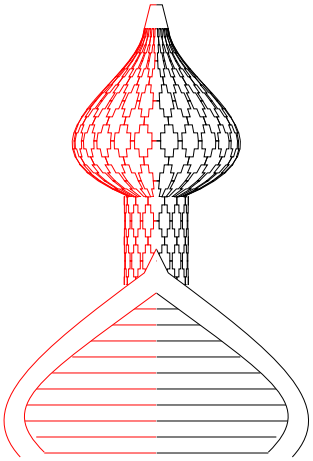
```
B011  /LW{setlinewidth}def        устан. ширины линии setlinewidth
B012  /F{fill}def                 закраш. внутренности контура
B013  /RGB{setrgbcolor}def        установка текущего цвета
B014  /RED{1 0 0 RGB}def          установка красного цвета
B015  /BLUE{0 0 1 RGB}def         установка синего цвета
B016  /GREEN{0 1 0 RGB}def        установка зеленого цвета
B017  /BLACK{0 setgray}def        установка черного цвета
B018  /WHITE{1 setgray}def        установка белого цвета
```

А теперь обратите внимание на строку 23

```
23  Rside -1 1 scale Rside
```

В ней дважды вызывается некая процедура `Rside`, а между этими вызовами расположен текст `-1 1 scale`. Это действие зеркально отражает (относительно начала координат) результат действия `Rside`. Давайте поставим второй вызов `Rside` в скобки `GS ... GR` и установим внутри скобок красный цвет

```
23  Rside GS -1 1 RED scale Rside GR
```

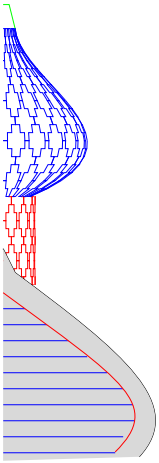


Получим картинку, нарисованную справа. Я сильно увеличил толщину линий, чтобы было виднее.
На картинке видно, что правая и левая половинки рисуются отдельно, и, значит, в дальнейшем мы можем ограничиться тем, как рисуется правая половина.
Поэтому в дальнейшем исследовании программы мы часть строки после `Rside` временно "закомментируем".

2. Разделение главной процедуры

Займемся выделением в правой половине рисунка компонент `cask`, `drum`, `onion` и `top` внутри процедуры `Rside`.

Похоже, что текст `top S` в строке 20 рисует верхушку, текст в строке 14 — барабан, текст в строке 15 — луковицу, а в строках 16–20 — бочку. Неужели бочка настолько сложнее луковицы? Попробуем раскрасить все части рисунка в разные цвета. Получим такое:



top — Ничего интересного. Даже описывать не будем.

onion — Это самая сложная часть.

drum — На самом деле барабан выше. Его фигурный низ закрыт непрозрачной бочкой.

cask — Бочка устоена просто, но необходимость ее закраски требует некоторых усилий. Серый фон сделан временно.

С рисования бочки мы и начнем.

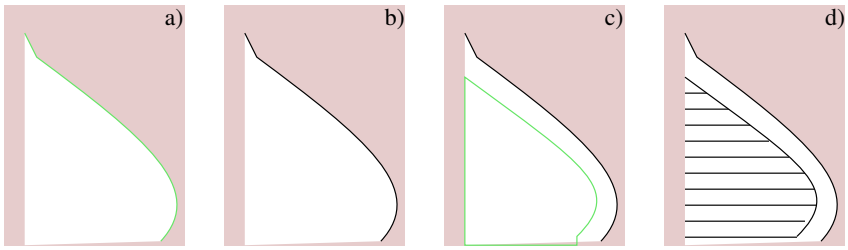
Рассмотрим программный текст предполагаемой бочки

```
16   0 53 m 3 -6 rl 30 27 46 14 34 1 c
```

```
17  GS 0 0 l CL 1 setgray fill GR S
18  GS bp 0 42 rl 30 20 40 14 28 2.2 c -2.2 vrl CL
19  CNP bp casklines GR
20  0 42 m 30 20 40 14 28 2.2 c S
```

(кусочек строки 20 относится к рисованию фрагмента *top* и к завершению описания процедуры *Rside*).

В строке 16 запрограммировано рисование отрезка, идущего из верхней левой точки немного вниз вправо и последующее рисование кривой Безье, ведущей в точку (34,1) (напомним, что `c = curveto` — команда рисования кривой Безье). Дальше создавшаяся графическая обстановка системной командой `GS=gsave` сохраняется в стеке, и имеющийся путь замыкается двумя отрезками — горизонтальным до начала координат и вертикальным до начальной точки. Область внутри получившегося контура закрашивается белым, процесс возвращается к состоянию с прорисованной наружной границей, которая прорисовывается. Это все делается в строке 17. Посмотрите на рисунке последовательные фазы процесса — a) и b). Для наглядности сделан цветной фон и ненарисованные линии тоже подцвечены

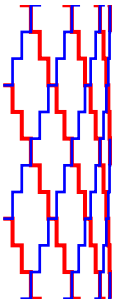


В фазе c) проводится еще одна кривая Безье `30 20 40 14 28 2.2 c`, которая вместе с первой составляет боковую окантовку бочки. С точки зрения программы нам важно, что показанное на рисунке пополнение кривой горизонтальной и вертикальной прямыми (все это показано зеленой линией) образует замкнутый путь, удобный для рисования “досок” облицовки бочки, — если мы выполним команду `clip`, то нам не нужно будет беспокоиться об “обрезании досок” по окантовке, выбранный контур будет ограничивать результат. На фазе d) видно, что доски обрезаны правильно. Пожалуй, вторая снизу линия оказалась коротковата. Нужно будет ее подправить.

Но серьезнее другое. Когда я готовил демонстрационную версию картинки для фазы d), то долго не смог справиться с путями, содержащими упомянутую вторую кривую Безье — назовем ее `cfacing2`. Наконец, вспомнилось такое эвристическое правило: если какое-то место программы не удается, нужно писать его проще. В данном случае можно описать процедуру `cfacing2`, которая будет вызываться дважды, и не нужно будет держать ее в стеке, проще два раза вызвать.

3. Рисование барабана

Нарисуем барабан покрупнее.



Если покрасить половину линий в красный цвет, а другую половину в синий, то мы увидим, что сетка состоит из лесенок — опускающихся направо красных и поднимающиеся направо синих. Каждая такая лесенка, если все ступеньки у нее одинаковые, программируется очень просто. На самом деле ширина этих ступенек должна уменьшаться, грубо говоря, "по синусу". При чем здесь синус? Дело в том, что поперечные сечения цилиндра — это окружности, и равные ступеньки лесенки — это равные дуги окружностей, имеющие равную угловую меру. Проекции краев этих ступенек вычисляются с помощью функции sin.

Теперь уже можно смотреть фрагменты программы, где фигурирует идентификатор `drum`. Таких строчек четыре (я их разобью для удобства на части)

```
09  /drumchain{5 5 90
      {sin drumR mul CP X pop l dhCH vrl} for S}def
10  /drumL{GS 0 44 m 22 vrl 12 hrl -22 vrl CL CNP 0 X m
           drumchain GR}def
13  /Rside{GS
/drumR 8 def
    /dhCH -2 def
%   /ohCH -2 def          --- константа, используемая в onion
14  40 10 100 {drumL} for    --- цикл рисования цепочек
                                 в одном направлении
    /dhCH dhCH neg def       --- смена направления
    10 10 70 {drumL} for     --- цикл рисования цепочек
                                 в другом направлении
```

Таких строчек тоже четыре. Строки 13 и 14 — это начало описания самой процедуры `Rside`, ее значение нам известно. В строке 14 написаны два цикла, рисующих цепочки в разном направлении, мы еще не знаем, какое в каком. Нужно срочно разбираться. Описанная в строчке 10 процедура `drumL` готовит почву для вызова процедуры `drumchain`. Сама процедура `drumL` получает в стеке некоторое значение `y`, последовательно пробегающее в первом цикле значения от 40 до 100, а во втором цикле от 10 до 70, в обоих случаях с шагом 10.

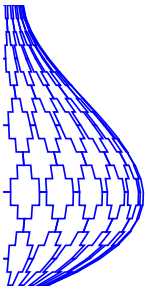
Вся работа в `drumL` происходит в скобках `gsave ... grestore`, сохраняющих графическую обстановку. Внутри же этих скобок строится прямоугольник `0:12 x 44:66`, который используется для ограничения изображения (необходимая для этого команда `clip` спрятана в команде `CNP`), после чего начальным состоянием делается точка `(0,y)`, где `y` — значение в стеке.

Команда `drumchain` исполняет цикл со счетчиком, меняющимся от 5 до 90 с шагом 5, в котором строится какой-то путь, а потом этот путь рисуется командой `S=stroke`. Значения счетчика цикла — это, безусловно, углы, которые в Постскрипте измеряются "по-школьному", в градусах. Командой `sin` вычисляется синус этого угла, и он умножается на радиус барабана `drumR`. Получившееся значение будет первой координатой очередной вер-

шины лесенки. А вторая координата получается из текущей точки последовательностью команд `CP X pop` (или `currentpoint exch pop`). В эту точку проводится отрезок, а затем строится вертикальный отрезок размера `dhCH` (значит, в первом использовании процедуры `Rside` лесенка строилась вниз, а во втором — вверх). Как уже говорилось, `clip`-контур вырезает из лесенки нужный фрагмент.

4. Рисование луковки

Как и раньше, начнем с более крупного рисунка.



Тут уже ничего перекрашивать, вероятно, не требуется. Здесь та же, что в барабане система двух лесенок, возрастающей и убывающей, только кроме перевычисления координаты x понадобится и перевычисление y. В принципе, нужно было бы более сложно рисовать и сами вертикальные линии, но мы попробуем этим пренебречь.

Выпишем из программы то, что относится к рисованию луковицы. Это одна строчка и еще маленький кусочек из `Rside`, а также три отдельные процедуры.

```
06  /onisk{108 sub -41 div dup dup -97 mul 99 add mul
       4 add mul 3 add}def
07  /onichain{3 3 90 {sin dup CP dup ohCH add dup onisk
       6 2 roll dup onisk 4 -1 roll mul X l pop mul X l}for S}def
08  /oniL{GS 0 66 m 42 vrl 32 hrl -42 vrl CL CNP 0 X m
       onichain GR}def
13  ... /ohCH -2 def
15  70 10 180 {oniL} for
    /ohCH ohCH neg def
    10 10 100 {oniL} for
```

В принципе все понятно: действия в строке 15 точно повторяют конструкцию из строки 14 для барабана: это два цикла для рисования нисходящих и восходящих цепочек с выполняемой между ними сменой знака у константы `ohCH`. В строке 08 выписана процедура `oniL`. Дальше, понятным образом, в строке 07 — процедура рисования лесенки `onichain`.

Наконец, в строке 06 находится функция `onisk`, вычисляющая радиус фигуры как функцию от высоты точки. Проще всего понять, как работает эта процедура, если сначала написать в Постскрипте вычисление значения полинома третьей степени с фиксированными коэффициентами. Рассмотрим полином $f(y) = a + by + cy^2 + dy^3$. Вычисление происходит так: первоначально в стеке лежит `y`

```
              %  y  вычисление значения полинома по схеме Горнера
  dup dup     %  y  y  y   сразу запасаем еще два экземпляра y
  d mul       %  y  y  dy
  c add mul   %  y  y*(c+dy)
  b add mul   %  y*(b+y*(c+dy))
  a add       %  f=a+y*(b+y*(c+dy))
```

Мы видим здесь буквальное повторение этой схемы в конце процедуры `onisk` со значениями $a = 3$, $b = 4$, $c = 99$ и $d = 97$. Осталось понять начало процедуры, но это просто линейное преобразование y. Как я выбирал эти коэффициенты, я уже вспомнить не могу. Хотя форма луковки, по-моему, красивая.

5. Сборка рисунка и программы

При сборке рисунка важно только, чтобы барабан рисовался раньше чем бочка, так как она должна закрыть нижний ровный срез барабана, остальное безразлично.

При сборке программы нужно вначале написать в сжатом виде сокращения служебных слов и важнейших конструкций. Можно даже заготовить специальный файл, в котором хранятся такие определения, — это очень удобно, если у вас идет какая-то серия рисунков с однотипными командами (я планирую написать про это на другом примере).

В конце программы нужно написать установку начальной точки и некоторых важнейших параметров: ширину линии, способ стыковки отрезков при рисовании кусочно-линейных путей, тип используемого пунктира.

В середине нужно написать используемые параметры (с комментарием) и процедуры.

В нашем случае нужно сначала написать главную процедуру `Rside` в таком виде

```
/Rside{drum cask onion top}def
```

затем описания этих процедур и их составных частей. Конечно же, описания кривых Безье следует помещать в процедуры и комментировать.

Нужно, чтобы получилось так

```
%!
/X{exch}def /bp{0 0 m}def /m{moveto}def /l{lineto}def
/rl{rlineto}def /rm{rmoveto}def /hrl{0 rl}def /hm{0 m}def
/vrl{0 X rl}def /vm{0 X m}def /hrm{0 rm}def /vrm{0 X rm}def
/c{curveto}def /CL{closepath}def /GS{gsave}def /GR{grestore}def
/S{stroke}def /LW{setlinewidth}def /F{fill}def
/CP{currentpoint}def /CNP{clip newpath}def
/RGB{setrgbcolor}def /RED{1 0 0 RGB}def /BLUE{0 0 1 RGB}def
/GREEN{0 1 0 RGB}def /BLACK{0 setgray}def /WHITE{1 setgray}def

/Rside{GS drum cask onion top GR}def
/top{3 108 m -1.5 6 rl -1.5 hrl}def
/onion{ /ohCH -2 def       70 10 180 {oniL} for
/ohCH ohCH neg def 10 10 100 {oniL} for }def
/drum{ /drumR 8 def /dhCH -2 def 14  40 10 100 {drumL} for
   /dhCH dhCH neg def            10 10  70 {drumL} for}def
/cask{ 0 53 m caskcurv1 GS 0 0 l CL WHITE F GR S
   GS bp 0 42 rl caskcurv2 -2.2 vrl CL CNP bp casklines GR
   0 42 m caskcurv2 S}def
/caskcurv1{3 -6 rl 30 27 46 14 34 1 c}def
```

```
/caskcurv2{30 20 40 14 28 2.2 c}def
/casklines{7 12 17 21 26 30 33 33 31 33
      2 4 38 {bp vrm hrl S} for}def
/drumL{GS 0 44 m 22 vrl 12 hrl -22 vrl CL CNP
   0 X m drumchain GR}def
/drumchain{5 5 90
{sin drumR mul CP X pop l dhCH vrl}for S}def
/oniL{GS 0 66 m 42 vrl 32 hrl -42 vrl CL
CNP 0 X m onichain GR}def
/onichain{3 3 90 {sin dup CP dup ohCH add dup onisk 6 2 roll
dup onisk 4 -1 roll mul X l pop mul X l}for S}def
/onisk{108 sub -41 div dup dup
   -97 mul 99 add mul 4 add mul 3 add}def

240 300 translate 2.5 2.5 scale 0.01 LW
Rside -1 1 scale Rside
showpage
```

При проверке этого текста обнаружилась всего одна ошибка. Невнимательность — забыл в слове `top` вставить команду рисования. Вставьте сами!