Project phase 4 — PostScript interpreter (rounding out the language) — assigned Friday 7 October, due Tuesday 18 November

This is a continuation of phase 3.

In the tradition of 351 project assignments, this is only a first draft and is subject to change (to add further clarifications).

4.1 Task

Write the remaining Java classes and interfaces to implement the functions of a PostScript interpreter, so that it can execute (almost) all non-graphics programs.

4.2 Task in detail

4.2.1 PostScript operators that must be implemented (in addition to those implemented in Phase 3)

We add a second batch of 51 operators to the language of Phase 3:

Arithmetic and Math Operators

- neg
- ceiling
- floor
- round
- truncate
- sqrt
- atan
- cos
- sin
- exp
- ln
- log
- rand
- srand
- rrand

Array Operators

- getinterval
- putinterval
- aload
- astore

Dictionary Operators

- dictstack
- maxlength
- load
- store
- undef
- known
- where

String Operators Remember to add string functionality to operators, such as length, that you have already implemented to handle arrays and dictionaries.

- string
- anchorsearch
- search
- token

Relational, Boolean, and Bitwise Operators

- and
- not
- or
- xor
- bitshift

Control Operators

- quit

Type, Attribute, and Conversion Operators

- type
- cvlit
- cvx
- xcheck
- cvi
- cvn
- cvr
- cvrs
- cvs

File Operators

- file
- closefile
- read
- write
- print

Miscellaneous Operators

- bind

Virtual Memory Operators (no new ones)

4.2.2 Other PostScript names that must be implemented

(no new ones)

4.2.3 Features to pay attention to

The following types (from the master list on p. 33 of PLRM2) must be implemented:

- boolean
- integer
- mark
- name
- null
- operator
- real
- save
- array
- dictionary
- file
- string

4.2.4 Clarifications of PLRM2

- none

4.2.5 Deviations from PLRM2

- Strings and other composite objects must be treated equally by save and restore. This is cleaner than what PLRM2 says (PLRM2, Section 3.7, p.60 (“except strings”)), and it was the original intention in the design of the PostScript language (see the Green Book); the current difference in behavior is probably a concession to implementation efficiency in Adobe’s early interpreters. As far as I can tell, none of the PostScript programs we have in our test suite relies on the PLRM2-specified behavior.

4.2.6 Features in PLRM2 that need not be implemented

- all operators not listed in 4.2.1 above or in Phase 3
- the following types (from the master list on p. 33 of PLRM2) need not be implemented:
  - fontID
  - condition

  - gstate
  - lock
  - packedarray

- graphics (to be tackled in the next phase): only reserve placeholders in the state of the interpreter for the current graphics state and for the graphics state stack
- any features having to do with *Display PostScript*
- features having to do with print servers, print job control, etc.

How to turn in

Turn in your code by running

~barrick/handin your-file

on a regular UNM CS machine.

You should use whatever filename is appropriate in place of your-file. You can put multiple files on the command line, or even directories. Directories will have their entire contents handed in, so please be sure to clean out any cruft.