G53DOC
Useful PostScript Commands

Steven R. Bagley

Introduction

This document contains a brief description of a selection PostScript operators that you might find useful when implementing the programs described in the first set of lab exercises. Further details of the operators can be found in the *PostScript Language Reference Manual* which appears to be available at: `http://www.adobe.com/products/postscript/pdfs/PLRM.pdf`

Operand Stack Manipulation Operators

any	**pop**	–	Discard top element
any_1 any_2	**exch**	any_2 any_1	Exchange top two elements
any	**dup**	*any any*	Duplicate top element
any_{n-1} … any_0 *n j*	**roll**	$any_{(j-1) \bmod n}$ … any_0 any_{n-1} … $any_{j \bmod n}$	Roll **n** elements up **j** times
⊥ any_1 … any_n	**clear**	⊥	Discard all elements

Arithmetic and Math Operators

num_1 num_2	**add**	*sum*	Return num_1 plus num_2
num_1 num_2	**div**	*quotient*	Return num_1 divided by num_2
int_1 int_2	**idiv**	*quotient*	Return int_1 divided by int_2
int_1 int_2	**mod**	*remainder*	Return remainder after dividing int_1 by int_2
num_1 num_2	**mul**	*product*	Return num_1 times num_2
num_1 num_2	**sub**	*difference*	Return num_1 minus num_2
num_1	**neg**	num_2	Return the negative of num_1
num	**sqrt**	*real*	Return the square root of *num*
num den	**atan**	*angle*	Return the arctangent of *num* over *den*
angle	**sin**	*real*	Return sine of *angle*
angle	**cos**	*real*	Return cosine of *angle*

Note that *angles* are specified in degrees.

Array operators

–	**[**	*mark*	Start array construction
mark obj_0 … obj_{n-1}	**]**	*array*	End array construction

Dictionary Operators

int	**dict**	*dict*	Create dictionary with capacity for *int* elements
dict	**begin**	–	Push dict on the dicitionary stack
–	**end**	–	Pop current dictionary off dictionary stack
key value	**def**	–	Associate *key* and *value* in the current dictionary

String operators

int	**string**	*string*	Create string of length *int*

General composite operators

array *dict* *string*	**length**	*int*	Return number of elements in …
array index *dict key* *string index*	**get**	*any* *any* *int*	Returns the specified value from the composite
array index any *dict key any* *string index any*	**put**	–	Replace a signle element of the value of the first operand
array proc *dict proc* *string proc*	**forall**	–	Execute *proc* for each element of …

Relational, Boolean and Bitwise Operators

any_1 any_2	**eq**	*bool*	Test equal
	ne		Test not equal
num_1|str_1 num_2|str_2	**ge**	*bool*	Test greater than or equal
	gt		Test greater than
	le		Test less than or equal
	lt		Test less than
$bool_1$|int_1	**not**	$bool_2$|int_2	Perform logical|bitwise not
$bool_1$|int_1 $bool_2$|int_2	**and**	$bool_3$|int_3	Perform logical|bitwise and
	or		Perform logical|bitwise or
	xor		Perform logical|bitwise xor
–	**true**	*true*	Return boolean value *true*
–	**false**	*false*	Return boolean value *false*
int_1 *shift*	**bitshift**	int_2	Perform bitwise shift of int_1

Control Operators

bool proc	**if**	–	Execute *proc* if *bool* is true
bool $proc_1$ $proc_2$	**ifelse**	–	Execute $proc_1$ if *bool* is true, $proc_2$ if false
initial increment limit proc	**for**	–	Execute *proc* with values from *initial* by steps of *increment* to *limit*

Miscellaneous Operators

–	**null**	*null*	Push null on the stack

Device-independent Graphics State operators

–	**gsave**	–	Push graphics state
–	**grestore**	–	Pop and restore graphics state
num	**setlinewidth**	–	Set line width
num	**setgray**	–	Set colour space to greyscale and color to specified grey value (0 = black, 1 = white)
red green blue	**setrgbcolor**	–	Set colour space to RGB and colour to specified value
t_x t_y	**translate**	–	Translate user space by (t_x , t_y)
s_x s_y	**scale**	–	Scale user space by (s_x , s_y)
angle	**rotate**	–	Rotate user space by *angle* degrees

Path Construction Operators

–	**newpath**	–	Initialize current path to be empty
–	**currentpoint**	*x y*	Return current point co-ordinates
x y	**moveto**	–	Set current point to (*x*,*y*)
dx dy	**rmoveto**	–	Perform relative **moveto**
x y	**lineto**	–	Append straight line to (*x*,*y*)
dx dy	**rlineto**	–	Perform relative **lineto**
$x\ y\ r\ angle_1\ angle_2$	**arc**	–	Append counter-clockwise arc
$x\ y\ r\ angle_1\ angle_2$	**arcn**	–	Append clockwise arc
$x_1\ y_1\ x_2\ y_2\ x_3\ y_3$	**curveto**	–	Append Bezier cubic section
$dx_1\ dy_1\ dx_2\ dy_2\ dx_3\ dy_3$	**rcurveto**	–	Perform relative **curveto**
–	**closepath**	–	Connect sub-path back to its starting point
–	**flattenpath**	–	Convert curves to a series of straight lines
string bool	**charpath**	–	Append glyph outline to path
–	**clip**	–	Clip using the non-zero winding number rule
–	**eoclip**	–	Clip using the even-odd rule

Path Painting Operators

–	**stroke**	–	Draw line along current path
–	**fill**	–	Fill current path with current colour
–	**eofill**	–	Fill current path using even-odd rule

Font and Glyph Operators

key	**findfont**	*font*	Return **Font** resource instance identified by *key*
font scale	**scalefont**	*font'*	Scale *font* by *scale* to produce *font'*
font	**setfont**	–	Set font in graphics state
key scale	**selectfont**	–	Set font given name and scale (equivalent to *font* **findfont** *scale* **scalefont setfont**)
string	**show**	–	Paint glyphs for *string* in current font
string	**stringwidth**	$w_x\ w_y$	Return width of glyphs for *string* in currnet font

Device Operators

–	**showpage**	–	Transmit and reset current page