Network Working Group
Request for Comment No. 199
NIC No. 7151
Category: D.6
Updates: None
Obsoletes: None

T. Williams
SDC
15 July 1971

SUGGESTIONS FOR A NETWORK DATA-TABLET GRAPHICS PROTOCOL

The work reported herein was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC15-67-C-0149, ARPA Order No. 1327, Amendment No. 3, Program Code No. 1D30, and 1P10.

SERIES BASE NO VOL REPEAT
TM-4764 /000/01

AUTHOR T. G. Williams
T. G. Williams
TECHNICAL
RELEASE M. I. Bernstein
M. I. Bernstein
for
C. Weissman

DATE 7/15/71 PAGE 1 of 1

TECH MEMO *a working paper*

System Development Corporation/2500 Colorado Ave./Santa Monica, California 90406

Suggestions for a Network Data-Tablet Graphics Protocol

INTRODUCTION

The purpose of this document is to add SDC's comments to the discussion of a protocol for network graphics within the ARPA Network community. In general, we are concerned with the development of the graphics protocol in two areas: non-interactive graphics and data-tablet graphics, as opposed to fully interactive graphics. By non-interactive graphics we mean situations in which there is little or no requirement for interaction with displays. Such displays are used, for instance, in data retrieval systems using graphics to display retrieved information in the form of charts, X-Y graphs, histograms, scatter plots, tabular displays, etc. In these systems, each interaction with the system produces an entirely new display. The displays themselves have little, if any, structure. There is no necessity to interact with the picture itself other than, perhaps, by the use of light buttons. It is important that non-interactive graphics be simple to implement and use on the network. Therefore, we suggest that the graphics protocol design be based upon non-interactive graphics systems, and that capabilities needed for interactive graphics be added as a super-set. This will ensure that the protocol complexities associated with interactive graphics do not impose problems for the user of non-interactive graphics, as they would if a non-interactive subset were developed from a protocol based initially on interactive graphics. The section of Request for Comment

---

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

Relative to the amount of picture information contained in the data stream, in general, the data-tablet input is far more voluminous than a similar computer-generated image. Additionally, the data-tablet input stream contains temporal information that, in certain cases, is vital to the proper processing of the input. Therefore, ways must be found to implement a data-tablet graphics protocol that is flexible enough to accommodate a broad spectrum of data volume and that is compatible with the protocol for non-interactive display images.

[redacted]

Data-tablet input can consist of anything from a single point (as would occur when something was being pointed at) to literally thousands of bytes representing a hand-drawn rendering of a picture or a line of text. In many instances, the raw data-tablet input is preprocessed before it is passed to the principal processing program. This preprocessing can consist of such things as a variety of smoothing algorithms, filtering for thinning and/or redundancy removal, detection of certain operator actions such as uniquely marking each occurrence of placing the pen on the writing surface and raising it, and possibly other, more exotic processes such as corner detection, fitting straight-line segments, and the like. Most of these latter processes will not be considered for inclusion in the protocol, since they are usually unique to a particular investigator and his research.

Therefore, a data-tablet graphic protocol should permit the sender to specify, and the receiver to discriminate among, at least four types of data-tablet input:

- Unpreprocessed (Raw) Asynchronous Data Input Format:

Byte 0: Data tablet input op code

Byte 1: Type, 1 = raw asynchronous

Byte 2: Flags

Byte 3: Scale of Δ's

Byte 4-5: Number of points

Byte 6-7: 1st X-coordinate

Byte 8-9: 1st Y-coordinate

Byte 10: ΔX_1

Byte 11: ΔY_1

.

.

.

Byte 2n+10: ΔX_n

Byte 2n+11: ΔY_n

8	8	8	8	16	16	16	8	8		8	8
Op	1	Flags	Scale	No. points	X_0	Y_0	ΔX_1	ΔX_2	..	ΔX_n	ΔY_n
0	1	2	3	4 5	6 7	8 9	10	11		2n+10	2n+11

• Unpreprocessed (Raw) Synchronous Data Input Format:

Byte 0:	Data tablet input op code
Byte 1:	Type, 2 = raw synchronous
Byte 2:	Flags
Byte 3:	Scale of Δ's
Byte 4:	Sampling rate to the nearest 100 μsec
Byte 5-6:	Number of points
Byte 7-8:	1st X-coordinate
Byte 9-10:	1st Y-coordinate
Byte 11:	ΔX_1 (sign magnitude code)
Byte 12:	ΔY_1
.	
.	
.	
Byte 2n+11:	ΔX_n
Byte 2n+12:	ΔY_n

8	8	8	8	8	16	16	16	8	8		8	8
Op	2	Flags	Scale	Rate	No. points	X_0	Y_0	ΔX_1	ΔY_1	..	ΔX_n	ΔY_n
0	1	2	3	4	5 6	7 8	9 10	11	12		2n+11	2n+12

PREPROCESSED-DATA INPUT FORMAT

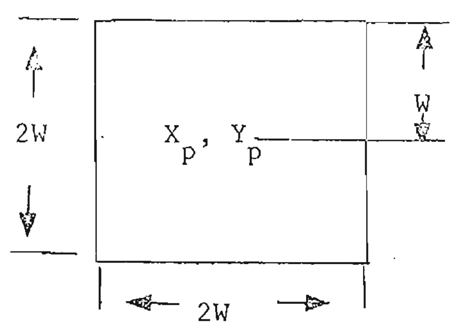
There are a variety of processes that can be applied to raw tablet data before it is transmitted to the requesting program. For instance, when the tablet input is "noisy" or jittery, various smoothing algorithms may be applied. The most common of these is some form of weighted, clumped or moving average. At SDC, we have settled on an 8-point moving average when smoothing is desirable. Another fairly common form of preprocessing is "thinning" or filtering to remove unnecessary or redundant data points. Depending on the end use of the data, filter "windows" can have a variety of geometries, including square, rectangular, diamond, and circular, and the filter may be single or double windowed. SDC currently uses a single square window filter on all tablet input. The window size is a variable and may be set to zero, thus, eliminating the filter.

Logically, the filter may be described as:

Take (X,Y) if $|X_p - X| \geq w$ or $|Y_p - Y| \geq w$ is true

where: (X,Y) is the current data point, (X_p, Y_p) is the previously accepted data point that either passed the filter or was the first point of the stroke, and w is the window size.

Pictorially, this can be represented as:



Any point inside the square will be rejected, any point on the boundary or beyond is accepted and becomes (X_p, Y_p).

In addition to smoothing and filtering, we have found it necessary that our character recognition algorithms be able to estimate the velocity along the path of the stroke. Therefore in addition to saving the X, Y coordinates that pass the filter (smoothing, if done, precedes filtering and is done on the raw data points), we count and store the number of rejected points between the saved ones. Since the data-tablet input is synchronous, the count times the sampling rate divided into the distance between adjacent points is a sufficient approximation for our purposes. Our character-generator also requires the rectangle surrounding a stroke (defined by the minimum and maximum values of X and Y in the stroke); this information is very easy to generate during preprocessing.

Assuming that other Network nodes wanted to use SDC's tablet graphic software--the character recognizer in particular--we would have to know what, if any, preprocessing was done to the input data before it was sent. Our suggested format for this form of tablet data, then, is:

Byte 0:	Data tablet op code
Byte 1:	Type, 3 = preprocessed
Byte 2:	Flags
Byte 3:	Scale of Δ's
Byte 4:	Sampling rate if synchronous (as indicated by flag)
Byte 5:	Window Size
Byte 6-7:	Number of Points
Byte 8-9:	1st X-coordinate
Byte 10-11:	1st Y-coordinate
Byte 12-13:	Minimum value of X in the stroke
Byte 14-15:	Minimum value of Y in the stroke
Byte 16-17:	Maximum value of X in the stroke
Byte 18-19:	Maximum value of Y in the stroke

Two forms follow from here, depending upon another flag:

Counts included		and	Counts deleted	
Byte 20	ΔX_1		Byte 20:	ΔX_1
Byte 21:	ΔY_1		Byte 21	ΔY_1
Byte 22:	rejected point $count_1$			
Byte 3n+20	ΔX_n		Byte 2n+20:	ΔX_n
Byte 3n+21:	ΔY_n		Byte 2n+21:	ΔY_n
Byte 3n+22:	RPC_n			

8	8	8	8	8	8	16	16	16	16	16	16	16
	3	Flags	Scale	Rate	Window Size	# points	X_0	Y_0	X_{min}	Y_{min}	X_{max}	Y_{max}
0	1	2	3	4	5	6 7	8 9	10 11	12 13	14 15	16 17	18 19

8	8	8		8	8	8
ΔX_1	ΔY_1	RCP_1		ΔX_n	ΔY_n	RCP_n
20	21	22		20	21	22

Counts included

8	8	8	8	8	8	16	16	16	16	16	16	16
Op	3	Flags	Scale	Rate	Window Size	# points	X_0	Y_0	X_{min}	Y_{min}	X_{max}	Y_{max}
0	1	2	3	4	5	6 7	8 9	10 11	12 13	14 15	16 17	18 19

8	8	8		8	8
ΔX_1	ΔY_1	ΔX_2	...	ΔX_n	ΔY_n
20	21	22		2n+20	2n+21

Counts deleted

The flags in this format not only indicate whether or not the data is synchronous and whether the counts are present, but may also be used to indicate whether or not the data was smoothed and the type of filtering.

CHARACTER SETS AND CHARACTER GENERATION

Our work in character recognition impacts the proposed protocols in one other area, that of character sets and character generation. Figure 1 shows the displayable characters presently available. We have planned extensions that will bring the set to 192 characters. The availability and use of our and others' extended character sets must be provided for in the protocol.

The character-set problem, though, is the easy one. We have found that when dealing with hand-printed input, the computer-generated output must be flexible enough to retain the geometry of the user's input--at least temporarily. This requires that we be able to generate characters in a large variety of sizes, with variable aspect ratios (independently specified sizes for X and Y). Since this is not an available hardware function, all of our characters are program generated. We currently specify character size and ratios in terms of X and Y multipliers applied to a character prototype. The character prototype is constructed on a 5" x 7" grid (extended, if necessary, to handle the long tails on p's, q's, etc.), where the grid-line spacing is 2^{-10} times the screen size. The important point is that network transmissions must be capable of specifying those types of characteristics when needed.

We propose, then, that a message format that specifies:

- Character code
- Character position
- Character height and width

As an aside, we would prefer that the character origin be the left-hand baseline point rather than the center--primarily because the center is ill-defined unless the character space is specified to include vertical extensions in both directions but also because it is difficult to take advantage of variable spacing to justify characters that are of unequal width (an aesthetic consideration of relevance in some displays).

SDC EXTENDED CHARACTER SET

Least Significant Digit (Hex)

Most significant digit (Hex)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		α	β	γ	δ	ε	ζ	η	λ	μ	υ	[illegible]	ρ	σ	τ	↖
1	↗	→	↘	+	Σ	Γ	Δ	Θ	Ω	γ	Π	{	√	}	∫	¬
2	b*	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
6	∞	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	↙	≤	≠	≥	EOM†
8	ψ	ẏ	δ	λ	⌈	⌉	~	∧	□							
9	κ	ι	ϑ	φ	ξ	ς	∂	∇	Λ							
A																
B																
C																
D																
E																
F																

Reserved for control characters (rows C–F)

*b blank

†EOM End of message

Figure 1