Character generation from resistive storage of time derivatives

by MICHAEL L. DERTOUZOS

Massachusetts Institute of Technology
Cambridge, Massachusetts

INTRODUCTION

Recent advances in man-machine communication have stimulated increased interest in techniques and special circuits that generate characters, for graphical and alphanumeric Cathode-Ray-Tube (CRT) display terminals, at the display site. The primary advantage in employing such local character generation is compression of the data that is required to store and communicate a character from the computer to the display—a single binary word of length n is all that is required to instruct the character generator to display one of 2^n possible characters. The primary disadvantage of local character generation is display cost, for it is generally considerably less expensive to generate characters from a longer sequence of more elementary commands—for example commands that cause the CRT beam to move right, left, up or down by a minimum resolvable increment. Besides these conflicting costs of data storage and transmission versus local-display generation, several other less tangible criteria such as character stability and fidelity (aesthetics), are instrumental in the design and evaluation of a local character-generation approach.

This paper discusses a character-generation technique which requires, for each character, the storage in a resistive memory of the time derivative functions for the horizontal and vertical CRT deflection signals. The first section of the paper describes specific geometrical primitive segments that can compose a large class of characters and symbols; the choice of such primitives is important, since it affects directly the quality of the displayed characters and the display cost. Also given in this section is a complete list of primitive sequences for the 94-character ASC-II set. The second section of the paper describes a character-generation system that stores the above primitives in a resistor matrix, and uses them to compose desired characters on a CRT display. In the third section, this approach is evaluated and compared to more conventional methods of dot intensification, in terms of cost, speed, and fidelity.

Character primitives

Characters and symbols, generated on CRT displays, are made up of certain elementary graphical segments. *Character primitives* over a character set will be called those segments which are (i) atomic or indivisible to smaller segments, and (ii) sufficient in number and quality to compose within acceptable accuracy every character in that set. At one extreme, the points of a uniformly spaced grid are adequate character primitives (Figure 1a); however, as the number of these points is reduced (Figures 1b and c), it becomes progressively more difficult to recognize the displayed characters. At the other extreme, the set of all characters may be considered itself as a set of character primitives. This set, however, is not very useful, for while it is generally easy to construct a system capable of implementing the primitives of Figure 1, it is considerably more difficult and expensive to implement the primitives at the other extreme. Conversely, it takes only seven bits to specify one of the 94 characters of the ASC-II set, while it takes 49 bits to specify every one of the possible subset of

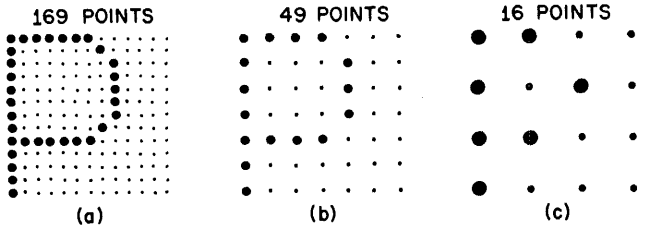

Figure 1—Points as character primitives

dots of Figure 1b. These simple observations on the above two extremes are characteristic of the problems of character generation and of the objectives in the design of an effective character generator—that is the desirability for a small number of primitives which can be economically implemented.

The primitives used in the character generation technique of this paper are continuous strokes which are either (i) straight lines or (ii) so-called "cusps". A straight-line primitive is specified relative to a point P by increments Δ_x, Δ_y which are real numbers; in our notation each such primitive is denoted, when visible, by (Δ_x, Δ_y) or, when invisible by an underscore $\underline{(\Delta_x, \Delta_y)}$. Figure 2a shows two such primitives. The equation of primitive (Δ_x, Δ_y) is relative to a coordinate center at point P as follows:

$$\frac{y}{\Delta_y} = \frac{x}{\Delta_x} \text{ for } 0 \leq \frac{y}{\Delta_y} \leq 1,\ 0 \leq \frac{x}{\Delta_x} \leq 1 \qquad (1)$$

where x and y are the horizontal and vertical coordinates of every point on that primitive.

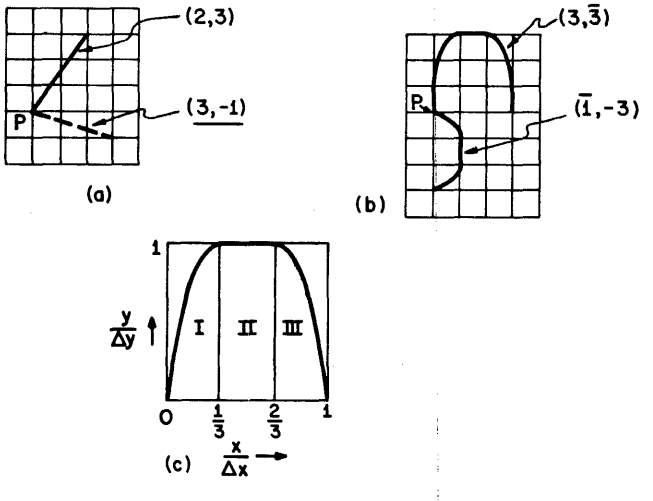

Figure 2—Straight/cusp primitives

The cusp primitive, on the other hand, is specified relative to a point P by increments Δ_x, Δ_y, which are real; moreover, one of these increments is overscored, and is called the *cusp increments*; that is either $(\bar{\Delta}_x, \Delta_y)$ or $(\Delta_x, \bar{\Delta}_y)$ are valid cusp primitive notations. Geometrically, a cusp primitive is, as shown in Figure 2b, contained in a rectangle of dimensions Δ_x, Δ_y; the curved segment corresponding to the overscored increment is obtained by dividing the other increment into three equal parts, fitting a straight line in the middle section and a parabola in each of the other two sections so that the parabolas are tangent to the above straight line. More precisely, the cusp, $(\Delta_x, \bar{\Delta}_y)$, shown normalized in Figure 2c, is given, relative to a coordinate center at point P, by

In Region I $(0 \leq \frac{x}{\Delta_x} < \frac{1}{3})$;

$$\frac{y}{\Delta_y} = 1 - \left(1 - 3\frac{x}{\Delta_x}\right)^2 \qquad (a)$$

In Region II $(\frac{1}{3} \leq \frac{x}{\Delta_x} < \frac{2}{3})$;

$$\frac{y}{\Delta_y} = 1 \qquad (b)\ (2)$$

In Region III $(\frac{2}{3} \leq \frac{x}{\Delta_x} \leq 1)$;

$$\frac{y}{\Delta_y} = 1 - \left(3\frac{x}{\Delta_x} - 2\right)^2 \qquad (c)$$

The cusp $(\bar{\Delta}_x, \Delta_y)$ is obtained from Equations (2) by interchanging literal x with literal y everywhere in these equations. A cusp is always visible. These apparently mysterious primitives are justified on two counts: (i) ability to represent a large class of characters and symbols with a small number of primitives, as discussed immediately below and (ii) ease of implementation, as discussed in the following section.

A character or symbol is composed from a sequence of these two types of primitives; here the first primitive is specified relative to the lower left corner of the character field, and each subsequent primitive is specified relative to the terminating point of the preceding primitive. For example, capital letter A is formed in Figure 3a by the primitive sequence

$$S_A = (.45, 1.2)(.45, -1.2)\underline{(-.788, .3)}(.676, 0)$$

Observe that the first segment is a visible straight primitive which starts at the lower left corner and

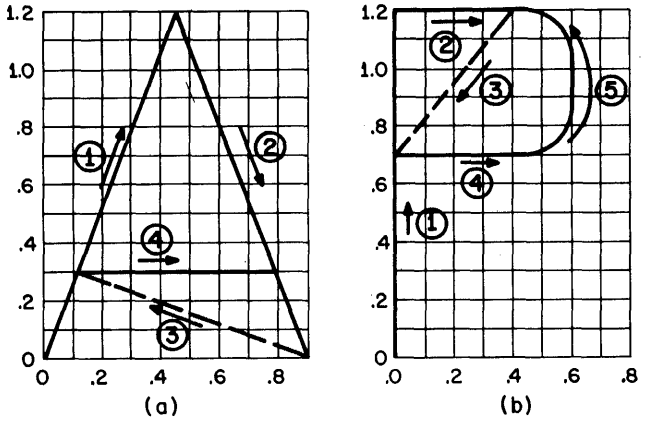

Figure 3—Character composition by straight/cusp primitives

terminates at the point [.45, 1.2]. The second segment is again a visible straight primitive, which starts as point [.45, 1.2] and terminates .45 units to the right and 1.2 units below that point. Observe further that the third segment is invisible, and that the direction and order in the sequence of each primitive is shown adjacent to each segment in Figure 3a. Capital letter P of Figure 3b is formed by the primitive sequence

$$S_P = (0, 1.2)\ (.4, 0)\ (-.4, -.5)\ (.4, 0)(\bar{.2}, 5)$$

Here, the first four primitives are straight with the third primitive invisible. The fifth primitive however is a cusp which starts at the point [.4, .7] and ends at the point [.4, 1.2].

Figure 4 shows the primitive sequences corresponding to all 94 alphanumeric characters and symbols of the ASC-II code. This Table is arranged exactly as the table of the ASC-II code for reference purposes. Some statistics of interest here are as follows:

1. The average number of primitive segments per character is 4.43.
2. The maximum number of primitives per character is eight.
3. The total number of different magnitudes for the primitive increments is 13.
4. No character uses more than two cusp primitives; these primitives occur (intentionally) either at the fifth, at the seventh, or at both the fifth and seventh segments of that character's primitive sequence.*

Of the above observations, 1, 2, and 3 indicate that a relatively small number of primitives can form a relatively large class of symbols. The fourth as well

* or they can be made to occur at these segment positions by introducing primitives (0, 0) anywhere in the sequence.

	0 (.6,.9)(-.6,.3)(0,-.5)(.6,-.4)(0,.5)	@ (.5,.3)(.3,.6)(0,-.6)(-.3,.6)(-.3,-.6)(-.1,-.3)(-.2,-.1)(.2,-.3)	P (0,1.2)(.4,0)(-.4,-.5)(.4,0)(.3,.5)	(.5,1.0)(-.2,.2)	p (0,-.3)(0,1.2)(0,-.6)(.6,-.3)(0,.3)(-.6,.3)
! (.4,1.2)(0,-1.0)(0,-.1)(0,-.1)	1 (.2,1.0)(.2,.2)(0,-1.2)	A (.45,1.2)(.45,-1.2)(-.788,.3)(.676,0)	Q (.6,0)(-.2,.2)(.2,.7)(-.6,.3)(0,-.5)(.6,-.4)(0,.5)	a (.6,0)(0,.9)(0,-.6)(-.6,-.3)(0,.3)(.6,.3)	q (.6,-.3)(0,1.2)(0,-.6)(-.6,-.3)(0,.3)(.6,.3)
" (.3,1.0)(0,.2)(.2,-.2)(0,.2)	2 (0,.9)(.7,.3)(-.7,-.9)(.6,0)	B (0,1.2)(.4,0)(-.4,-.5)(.4,0)(.3,.5)(0,-.5)(-.4,-.7)(-.4,0)	R (0,1.2)(.4,0)(-.4,-.5)(.4,0)(.3,.5)(0,-.5)(.4,-.7)	b (0,1.2)(0,-.6)(.6,.3)(0,-.3)(-.6,-.3)	r (.1,0)(0,.9)(0,-.2)(.7,.2)
# (.5,0)(.2,1.2)(.3,0)(-.2,-1.2)(.25,.4)(-.7,0)(.1,.4)(.7,0)	3 (.1,1.2)(1,0)(.5,-.5)(.5,-.7)(-.2,0)	C (.6,.9)(-.6,.3)(0,-.5)(.6,-.4)	S (.6,.9)(-.6,.3)(.6,-.5)(-.6,-.4)	c (.6,.3)(-.6,-.3)(0,.3)(.6,.3)	s (0,.3)(.6,-.3)(-.6,.3)(.6,.3)
$ (.4,.4)(0,-1.6)(.4,1.1)(-.6,.3)(.6,.5)(-.6,-.4)	4 (.7,0)(0,1.2)(-.7,-.9)(.6,0)	D (0,1.2)(.4,0)(.4,-1.2)(-.4,0)	T (.4,0)(0,1.2)(-.4,0)(.6,0)	d (.6,0)(0,1.2)(0,-.9)(-.6,-.3)(0,.3)(.6,.3)	t (.6,0)(-1,0)(-1,1)(0,1.1)(-.2,-.3)(.4,0)
% (.1,0)(.6,1.2)(-.6,0)(0,-.2)(.6,-.6)(0,-.2)	5 (.6,.7)(.1,.5)(.6,0)	E (0,1.2)(.7,0)(-.7,-.5)(.6,0)(-.6,-.7)(.6,0)	U (0,1.2)(0,-.6)(.6,-.4)(0,.6)	e (0,.5)(.6,0)(0,.1)(-.6,.3)(0,-.3)(.6,-.3)	u (.6,0)(0,.9)(0,-.6)(-.6,-.3)(0,.6)
& (.6,0)(-.6,.1)(.3,.2)(-.5,-.6)(.6,-.4)	6 (.6,1.2)(-.2,0)(-.4,-1.2)(.4,.7)(-.4,-.2)	F (0,1.2)(.7,0)(-.7,-.5)(.6,0)	V (0,1.2)(.4,-1.2)(.4,1.2)	f (.3,0)(0,1.0)(.2,-.1)(-.3,0)(.1,.1)(.4,.2)	v (0,.9)(.4,-.9)(.4,.9)
' (.4,1.2)(0,-0.2)	7 (0,1.2)(.6,0)(-.6,-1.2)	G (.6,.9)(-.6,.3)(0,-.5)(.6,-.4)(-.3,0)	W (0,1.2)(.2,-1.2)(.25,.6)(.25,-.6)(.2,1.2)	g (.1,-.3)(.5,0)(.2,.2)(0,.7)(-.6,.3)(0,-.3)(.6,-.3)	w (0,.9)(.2,-.9)(.2,.9)(.2,-.9)(.2,.9)
((.6,0)(-.2,1.2)	8 (.6,.9)(-.6,.3)(.6,-.5)(-.6,-.4)(.6,.5)	H (0,1.2)(0,-.5)(.6,0)(0,.5)(0,-1.2)	X (0,1.2)(.6,-1.2)(-.6,0)(.6,1.2)	h (0,1.2)(0,-.6)(.6,.3)(0,-.6)	x (.7,.9)(-.6,.0)(.7,-.9)
) (.2,0)(.2,1.2)	9 (.2,0)(.2,0)(.4,1.2)(-.4,-.7)(.4,.2)	I (.2,0)(.4,0)(-.2,0)(0,1.2)(-.2,0)(-.4,0)	Y (0,1.2)(.4,-.6)(.4,.6)(-.4,-.6)(0,-.6)	i (.4,0)(0,.9)(0,.3)(0,0)	y (.1,-.3)(1,0)(.1,.1)(.5,1.1)(-.6,0)(.4,-.9)
* (.2,.7)(.4,-.4)(.1,-.2)(-.6,0)(.1,.2)(.4,-.4)	: (.4,.6)(0,0)(0,-.6)(0,0)	J (.3,1.2)(.4,0)(-.2,0)(0,-1.0)(-.4,-.2)	Z (0,1.2)(.6,0)(-.6,-1.2)(.6,0)	j (.6,1.2)(0,0)(0,-.9)(0,-.9)(-.6,-.3)	z (.7,.9)(-.6,0)(.1,-.9)(.6,0)
+ (0,.6)(.6,0)(-.4,.4)(0,-.8)	; (.4,.6)(0,0)(0,-.6)(-1,-.2)	K (0,1.2)(0,-.7)(.7,.7)(-.5,-.9)(.6,-.7)	[(.7,1.2)(-.4,0)(0,-1.2)(.4,0)	k (0,1.2)(0,-.6)(.6,.3)(-.6,-.4)(.6,-.5)	{ (.6,-.3)(-.2,.2)(0,.5)(-1,.1)(1,.1)(0,.5)(.2,.2)
, (.4,0)(-.1,-.2)	< (.7,0)(-.6,.6)(.6,.6)	L (0,1.2)(0,-1.2)(.6,0)	\ (0,1.2)(.6-1.2)	l (.4,0)(0,1.2)	| (.4,-.3)(0,1.2)
- (0,.6)(.6,0)	= (0,.3)(.6,0)(0,-.4)(-.6,0)	M (0,1.2)(.4,-.8)(.4,.8)(0,-1.2)	] (.1,0)(.4,0)(0,1.2)(-.4,0)	m (0,.9)(.4,-.3)(0,-.6)(0,.6)(-.4,.3)(.4,0)(.4,.3)	} (.2,-.3)(.2,.2)(0,.5)(.1,.1)(-.1,.1)(0,.5)(-.2,.2)
. (.4,0)(0,0)	> (.1,0)(.6,.6)(-.6,.6)	N (0,1.2)(.6,-1.2)(0,1.2)	^ (0,1.4)(.4,.3)(.4,-.3)	n (0,.9)(0,-.3)(.6,.3)(0,-.6)	~ (0,.2)(.6,0)(-.2,0)
/ (.6,1.2)	? (.4,0)(0,.1)(0,.1)(0,.2)(.3,.5)(-.6,.3)	O (.6,.9)(-.6,.3)(0,-.5)(.6,-.4)(0,.5)	_ (0,-.2)(.6,0)	o (0,.3)(0,.3)(.6,.3)(0,-.3)(-.6,-.3)	

Figure 4—Straight/cusp primitive sequences for 94-character ASC-II set

as the other observations above will be used in the following section in connection with the implementation of this character generation technique.

The character generator

A local character generator for a CRT display is generally a system (Figure 5) with input a seven-bit word, denoting a character, and output two deflection and one beam-intensification waveforms (functions of time), which when applied to the CRT deflection and beam controls, respectively, display that character relative to beam position, x_p and y_p. Character and line spacing is usually accomplished by a control unit external to the generator, which varies x_p and y_p upon completion of each character and line, respectively. If the CRT display module is of the refresh type, then the codes of characters to be displayed are stored in a local storage medium, usually a delay line, and are presented periodically, usually every 1/30 to 1/40 sec to the character generator. If the CRT display module is of the storage type, then the character generator generates the waveforms x, y and b only once for each character to be displayed, and the corresponding character is stored on the screen of the CRT.

Any given character primitive $y = f(x)$ can be generated by such a system in an infinite number of ways, since for every one of many possible choices for a horizontal deflection waveform x(t), where t is time, there is always a vertical deflection waveform $y(t) = f(x(t))$ which when applied simultaneously with x(t), causes the CRT beam to trace the primitive $y = f(x)$. Two particular types of waveforms, s(t) and c(t) were chosen to implement the primitives of the preceding section; they are shown in Figure 6a, and their time derivatives in Figure 6b.

A straight-line primitive about any point is generated by applying waveform s(t), appropriately scaled, to both the horizontal and vertical axes. Thus, setting

$$x(t) = \Delta x\, s(t) + x_1 \qquad \text{(a)}$$

$$y(t) = \Delta y\, s(t) + y_1 \qquad \text{(b)} \qquad (3)$$

where Δx and Δy are real numbers, results in a straight line primitive from $[x_1, y_1]$ to $[x_1 + \Delta x, y_1 + \Delta y]$ given by

$$\frac{x - x_1}{\Delta x} = \frac{y - y_1}{\Delta y} \qquad (4)$$

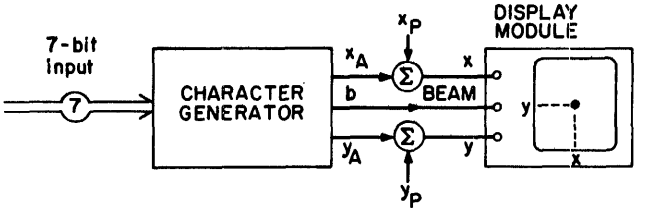

Figure 5—Local character generator

and shown in Figure 6c. This is the desired primitive of Equation (1).

A cusp primitive about any point, is generated by applying waveform s(t) to one axis and waveform c(t) to the other, after each waveform has been appropriately scaled. Figure 6d shows the resulting segment when s(t) is applied to the horizontal axis, and c(t) to the vertical axis, and Figure 6e shows a segment obtained with different scaling and interchange of the two waveforms. More generally, setting

$$x(t) = \Delta x s(t) + x_1 \qquad \text{(a)}$$

$$y(t) = \Delta y c(t) + y_1 \qquad \text{(b)} \qquad (5)$$

where Δx and Δy are real numbers, yields a cusp primitive, about-point $[x_1, y_1]$ described as follows:

for

$$0 \le \frac{x - x_1}{\Delta x} < \frac{1}{3},$$

$$y = y_1 + \Delta y \left[1 - \left(1 - 3\,\frac{x - x_1}{\Delta x}\right)^2\right] \qquad \text{(a)}$$

for

$$\frac{1}{3} \le \frac{x - x_1}{\Delta x} < \frac{2}{3}, \qquad y = y_1 + \Delta y \qquad \text{(b)} \qquad (6)$$

for

$$\frac{2}{3} \le \frac{x - x_1}{\Delta x} \le 1,$$

$$y = y_1 + \Delta y \left[1 - \left(3\,\frac{x - x_1}{\Delta x} - 2\right)^2\right] \qquad \text{(c)}$$

Equation (6) is identical in form to the desired cusp primitive, given by Equation (2). Since Equations (4) and (6) implement exactly all the primitives of the previous section, about any point (x_1, y_1), it remains only to provide means for forming a string of

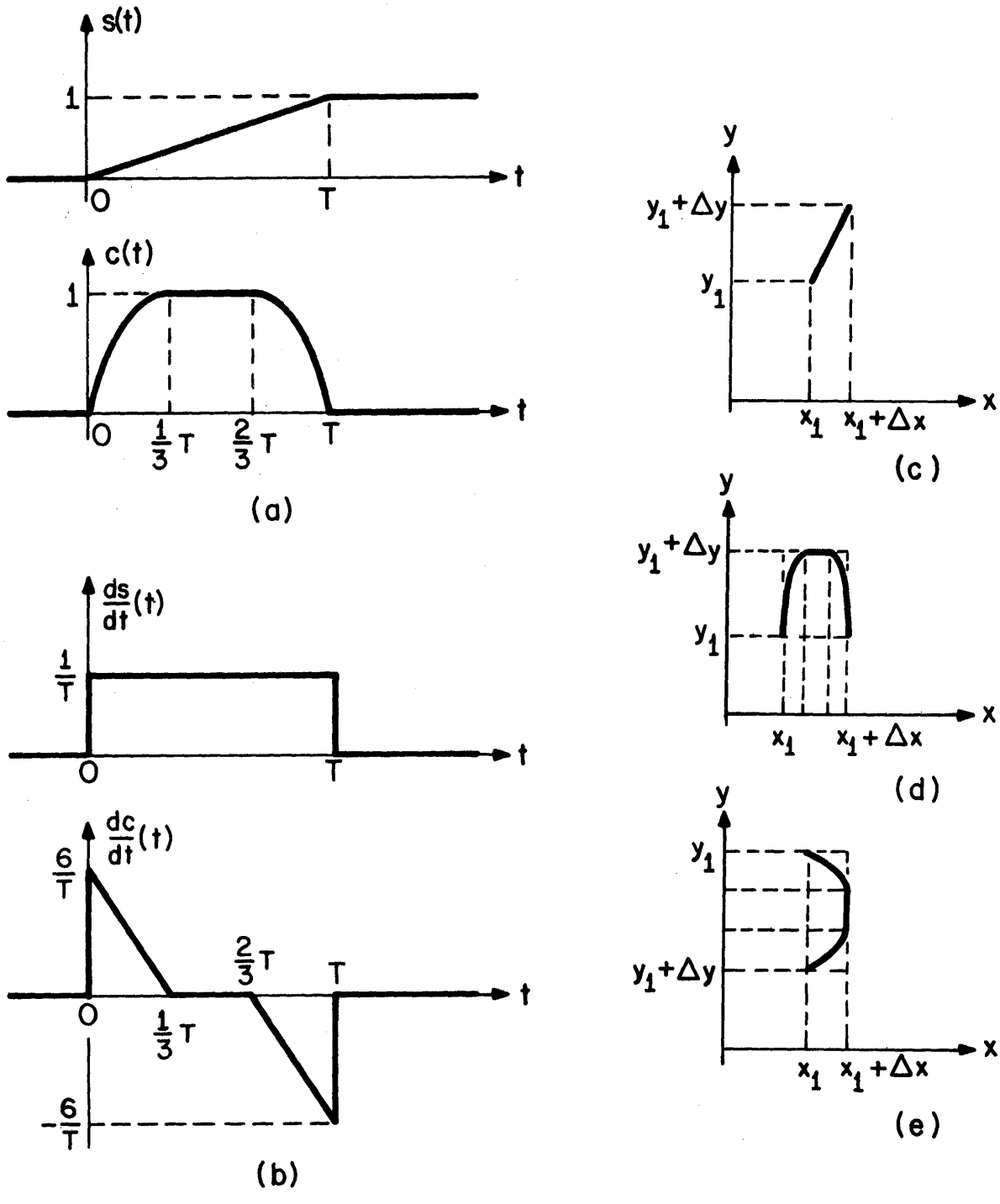

Figure 6—Waveforms for straight/cusp primitives

primitives, so that all the characters of Figure 4 may be implemented.

The formation of strings of primitives, that is of characters and symbols, is accomplished by concatenating the derivative waveforms of Figure 6b, for each primitive segment, after they have been scaled by Δx and Δy. Such waveforms, denoted by $(1/T)dx/dt$ and $(1/T)dy/dt$, (T constant) are shown for letter P

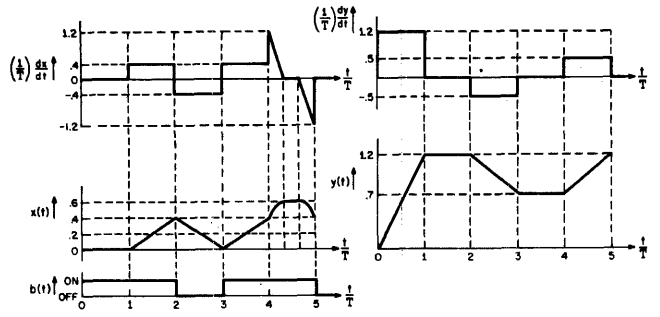

Figure 7—Composition of CRT deflection and beam waveforms

on the top half of Figure 7; subsequent integration in time of these waveforms yields the deflection waveforms x(t) and y(t), shown on the lower half of Figure 7. Also shown in Figure 7 is the beam waveform b(t) which turns the beam off in the third time segment $2T \leq t \leq 3T$. The character resulting from simultaneous application of these x(t) and y(t) waveforms on the CRT is the letter P of Figure 3b, specified by the primitive string:

$$S_P = (0, 1.2)(.4, 0)(-.4, -.5)(.4, 0)(.\bar{2}, .5)$$

Observe that these five primitives correspond to and are ordered as the five time segments of Figure 7.

One way of implementing this character-generation approach is shown in Figure 8. Here, sixteen lines carry eight rectangular constant-amplitude voltage

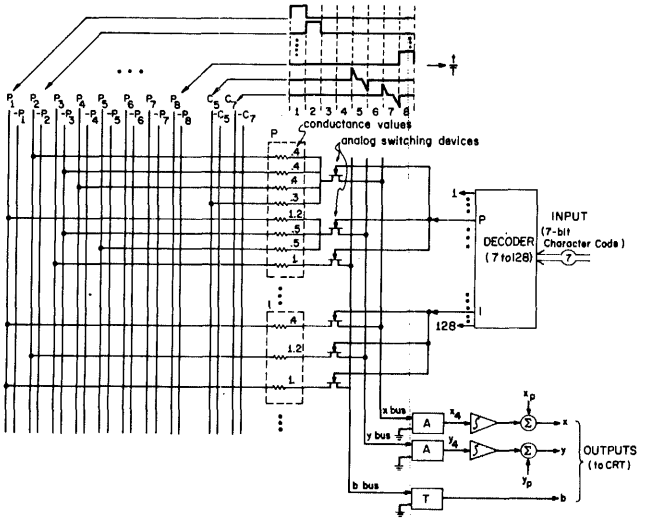

Figure 8—Character generator implementation

pulses P_i, and their negatives and four lines carry two cusp-derivative pulses C_i and their negatives. Waveforms and relative timing of these pulses are shown on the top center of Figure 8. Operation of the system is as follows: a character to be displayed is specified to the decoder shown on the right side of Figure 8, by, a seven-bit binary word. This word is "decoded", so that one of the 128 output lines of the decoder, say the line marked P, becomes energized. That line, turns "on" the three analog switching devices to which it is connected, and starts the timing sequences of the P_i and C_i pulses. The dx/dt, dy/dt and b waveforms for the selected character are formed by resistive mixing of the above pulses in three groups, respectively. For the case under discussion, letter P is "stored" in the values and manner of interconnection of eight resistors shown enclosed by dashed lines. Here, the top four resistors mix pulses P_2, $-P_3$ and P_4, all equally weighted by a conductance of .4 units; the fourth resistor in that group weighs waveform C_5 by .2. As a consequence of this mixing, the resulting current in the so-called xbus is the weighted sum of all these waveforms and is identical to the dx/dt waveform of Figure 7. The next group of three resistors having conductances 1.2, .5 and .5 respectively forms, in a similar manner a current in the ybus which is the dy/dt waveform of Figure 7. Finally, the complement of the beam waveform of Figure 7 is formed by the last group consisting of one resistor of unit conductance, as a current in the b bus. The dx/dt and dy/dt currents are subsequently amplified by low-input-impedance amplifiers A and integrated in time to yield the $x_A(t)$ and $y_A(t)$ waveforms of Figure 8. These are identical to the desired x(t) and y(t) waveforms of Figure 7. These waveforms are, in turn, summed with the constants x_P, y_P and the beam waveform is inverted resulting in a display of character P about point $[x_P, y_P]$. At the end of this sequence, the integrators are reset to zero output and the analog switching devices are turned off, thereby making the character generator ready for display of the next requested symbol. Also shown in Figure 8 is the resistor "memory" for character 1; the reader may verify that when this character is selected, the system does indeed generate the primitive sequence for that character, shown in Figure 4. Observe finally that the system of Figure 8 has two rather than eight cusp lines, C_i, which are active at timing positions five and seven. The reason for this choice is one of economics, since as we discussed in connection with Figure 4 it has been established over a large class of characters and symbols that these pulses at such relative positions are quite adequate.

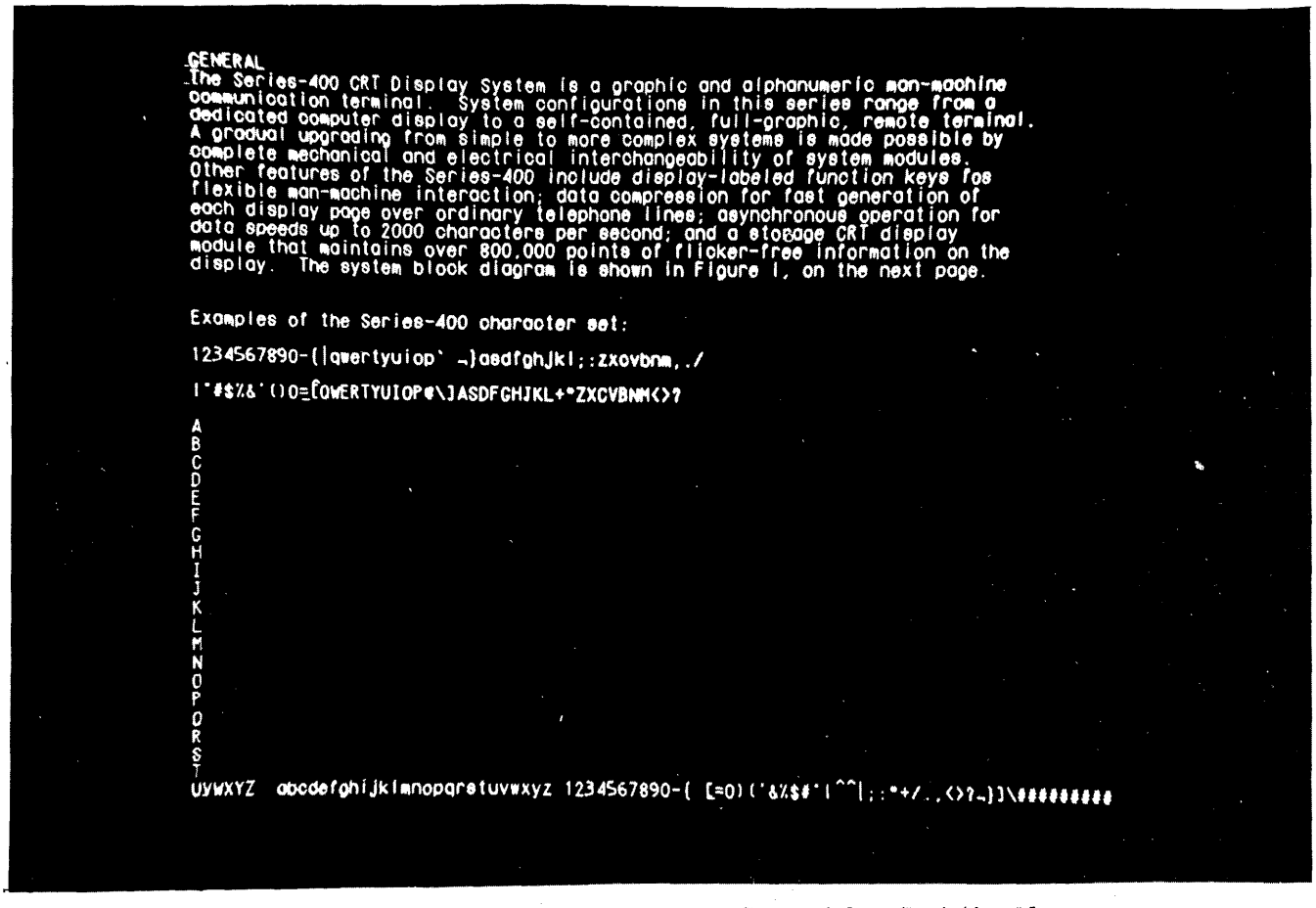

Figure 9—Implemented characters (Courtesy of Computek Inc., Cambridge, Mass.)

A photograph of characters and symbols generated by such a system is shown in Figure 9.

An alternative realization of the above character-generation technique would be to store for each character k bits in a digital read-only memory. These bits would, in turn, control a common, over all characters, resistive mixing network, by varying in discrete steps the conductances of this network. Such an implementation, however, requires approximately k = 90 bits of storage per character and is considerably less economical than the system of Figure 8.

COMPARISONS AND CONCLUSIONS

Ultimately, the merits and disadvantages of a character generator rest on economic and aesthetic criteria. The former are very strongly dependent on technology and are subject to rapid change, while the latter are, beyond a certain point, quite subjective. Nevertheless, certain conclusions can be drawn.

First, the use of stroke primitives such as straight lines and cusps results in more economical character storage than the use of points; and the relative advantage of such storage increases, over a certain range, with finer resolution. Consider for example that every character is formed on a grid of n^2 points. A straightforward point-intensification or incremental-stroke scheme on such a "dot-matrix" would require the

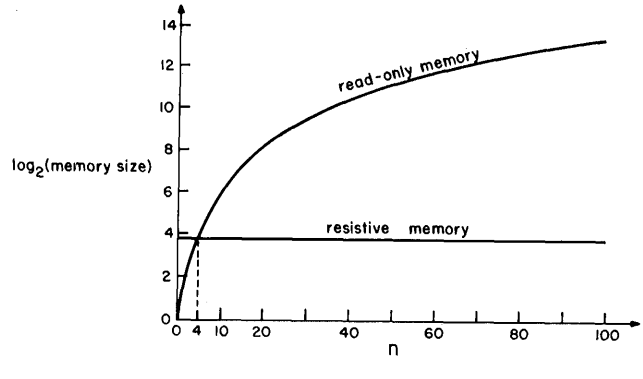

Figure 10—Memory growth versus resolution

storage of n^2 bits per character, indicating the points "which must be intensified—the corresponding memory growth curve, giving the number of diode components per character, is shown in Figure 10 and is labeled read-only memory". From Figure 4, however, we know that the average number of segments per character, over the 94 character ASC-II set, is 4.43 for the approach of this paper. Each segment, in turn requires two resistors, for x and y. We also know from Figure 4 that there will be of the order of 1.4 resistors per character for the beam. Hence, the average number of resistors per character is constant or

$$2(4.4) + 1.4 \cong 10$$

In addition, each character requires three analog switching devices (FETS) and their driver, or the order of five components. Thus, the total number of components is 15 per character, remaining constant within the limit of analog resolution, or $n \leq 100$, as shown by the graph labeled "resistive memory" in Figure 10. With present technology, it is more economical to construct the character generator out of discrete components; the resulting cost, is for an acceptable resolution $n^2 = 240$, lower than that of a read-only memory of ¼ the resolution. With forthcoming technology, the above ten resistors and five active components, should cost each about as much as a diode, hence an even better cost advantage can be expected. Observe however, as was indicated above, that resolution cannot exceed that of analog circuitry, since the storage and generation of characters is analog in nature. On the other hand, the CRT is an analog device, on which resolutions higher than analog cannot be effectively used. The above savings in character storage, result in lower generator cost, and reduced generator size.

Second, the speed of character generation of such a stroke technique is of the same order as that of dot-intensified character generation, since the current through resistors will generally change over its minimum resolvable increment as rapidly as, or faster, than the full current swing through a diode.

Third, the mixing of *time-derivative* waveforms, and the subsequent integration of these waveforms provides good character appearance through suppression of spurious noise and continuity of the integrated waveforms.

Finally, the fidelity of continuous-stroke characters with the above primitives is considerably higher than that of dot-intensified, or incremental-vector characters of comparable resolution. Such a comparison can be made visually by the reader on Figure 11 for a resolution of n = 4, or a grid of 16 points.

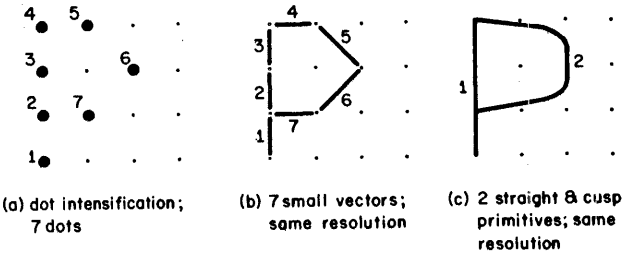

Figure 11—Comparison of straight/cusp and point intensified characters of same resolution

The approach discussed in this paper can be further extended to a more complete hardware "grammatical" structure, through a straightforward extension. That is, characters can be constructed from primitives and other simpler constructs which are themselves composed of primitives and/or other constructs of the same class. For example, as seen from Figure 4, the primitive sequence, S_8, for numeral 8 contains the primitive sequence S_S for capital S. That is, $S_8 = S_S$ (.8, .5) It is not yet clear whether such a hardware structure will result in even lower cost, without sacrifice of performance.

Finally, we would like to close with the philosophical observation that the use of sizable straight-line and cusp primitives is well suited to character generation, since characters and symbols were generated, on the first place, through such strokes, by pen or stick, on paper or sand, rather than by dots or by infinitesimal straight-line segments.

ACKNOWLEDGMENTS

I wish to thank Dr. H. L. Graham of Computek Inc., for his contributions to the design and construction of a prototype generator; Mr. Ben Simpson of Computek Inc., for the design and development of a practical generator, and Mr. L. Dounias of Ashely Meyer & Associates for his aesthetic design of the character set. This work was performed on a consulting basis for Computek Inc., 143 Albany Street, Cambridge, Massachusetts.