Table Paint

Alvy Ray Smith
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91103

28 October 1979

Published as Tutorial Notes at SIGGRAPH '80 and SIGGRAPH '81. This document was reentered by Alvy Ray Smith in Microsoft Word on Apr 21, 1999. Spelling and punctuation are generally preserved, but trivially minor spelling errors are corrected. Otherwise additions or changes made to the original are noted inside square brackets or in footnotes.

Abstract

Table look-up procedures are particularly useful at computer graphics installations because of the large chunks of cheap random access memory (framebuffers) which tend to be components of these facilities. We explain here how one additional framebuffer and one analog control device in addition to the tablet create a powerful and flexible extension to any typical framebuffer painting program. In particular, this readily available extra equipment gives the artist brushes which change size, shape, orientation, and/or color in realtime and smoothly. Examples are rotating brushes, animated brushes, airbrushes with changing spread and density, oriented brushes which track the direction of motion, and many more. That is, the artist is given a third dimension of stylus control which may be as graceful as the spatial two. He may define this new dimension from a menu of selections. The extra framebuffer is used as table memory. The extra device (e.g., joystick, trackball, pot, footpedal, microphone, strain guage, keyboard) is used as a controller for the added dimension. Realtime is obtained by using the output of the controller for table look-up.

KEY WORDS AND PHRASES: paint program, realtime, table look-up, general purpose graphics

CR CATEGORY: 8.2, 3.41

Introduction

What I shall call "ordinary" paint programs have been implemented in many places—Xerox, Utah, NYIT, MIT, Cornell, to name some of the older sites. Each of these is implemented on what I shall call a "general purpose graphics machine" which consists of a cpu, a framebuffer, a tablet, a color monitor, and a menu monitor. Actual systems vary considerably as to peripherals attached to the cpu, the number and bit-depth of framebuffers, the tablet resolution, the number and type of auxiliary control devices (knobs, switches, joysticks, etc.), the number and type of color monitor (RGB or NTSC or even black-and-white video monitors, non-standard resolution monitors), and the number and type of menu

displays (full calligraphic or simple text). The menu and color monitors can be combined into a single display but they are kept separate in the basic component listing as a conceptual convenience. Similarly, the framebuffer could be considered part of the cpu's main memory but is listed separately for convenience.

Although several control devices other than a tablet (with its stylus) could be used for painting, I believe the tablet to be the most versatile and natural choice for the basic listing. The table paint generalization of ordinary paint programs described below requires another control device and it is probably unnatural that it also be a tablet.

The general purpose graphics machine does not include special purpose graphics hardware such as a 4x4 matrix multiplier, realtime convolution, or realtime rotation boxes. The term “general purpose” is intended to convey the notion that a picture in the framebuffer is computed with a serial cpu, that the 2-dimensionality is only an interpretation of the contents of a piece of ordinary digital memory.

The assumption here is that there will be many general purpose graphics machines in existence soon and that users of these machines will desire capabilities of special purpose hardware (e.g., realtime rotation boxes) but won’t be able to afford them or won’t be able to buy them because they don’t exist as products. The argument of this paper is that additional components, but not different components, can extend the power of a general purpose graphics machine to include what apparently requires special purpose hardware. A case in point is a paint brush that rotates in realtime. This and many other realtime actions become possible with an additional framebuffer and control device, but nothing fancier.

Ordinary Painting

Details of painting programs appear elsewhere [1, 2][1]. Only a single detail is elaborated here so that the generalization to table paint can occur easily.

Painting a color, say red, into a framebuffer is the simple process of writing red into each pixel which falls under a 1 in a rectangular bitmask. A pixel under a 0 is not changed. The bitmask is called a brush and is translated over a framebuffer by a user who moves a stylus over a tablet surface.

In practice, the brush is usually more than a bitmask although it may be used as one. For example, a brush is often a rectangular portion of the framebuffer which has been saved at some earlier time. If the framebuffer is 8 bits deep then so is the brush. It may be treated as a bitmask by a mapping such as this: 0 maps to 0, 1-255 map to 1.

A variation on simple painting described above is “rubberstamping” where the little picture, which is the brush, is copied verbatim into the position in the framebuffer indicated from the tablet. One or more values in the brush may be thought of as transparent, which means simply that the corresponding pixels in the framebuffer are left unchanged.

Painting must occur in realtime to be interesting–i.e., a stroke should appear on the color monitor simultaneously (apparently) with the user’s hand motion on

[1] “[Alvy, Negroponte]” in the original.

the tablet. This means that the brush must be in local memory for fast access. Disk file access is untenable. An alternative location would be another framebuffer. This is what is proposed below in the table paint generalization.

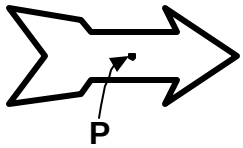
The Table Paint Schemata

An auxiliary control device is used to index, in realtime, into an auxiliary framebuffer which holds a table of brushes.

Rotating Brush Example

I will use the rotating brush as an example, then indicate the many other brush dynamics the table paint schemata encompasses.

Consider the following scenario: An artist user is painting in a framebuffer by moving a stylus with his right hand over a tablet. With his left hand, he grips a knob which he can rotate either way. Suppose his brush has this arrow shape:



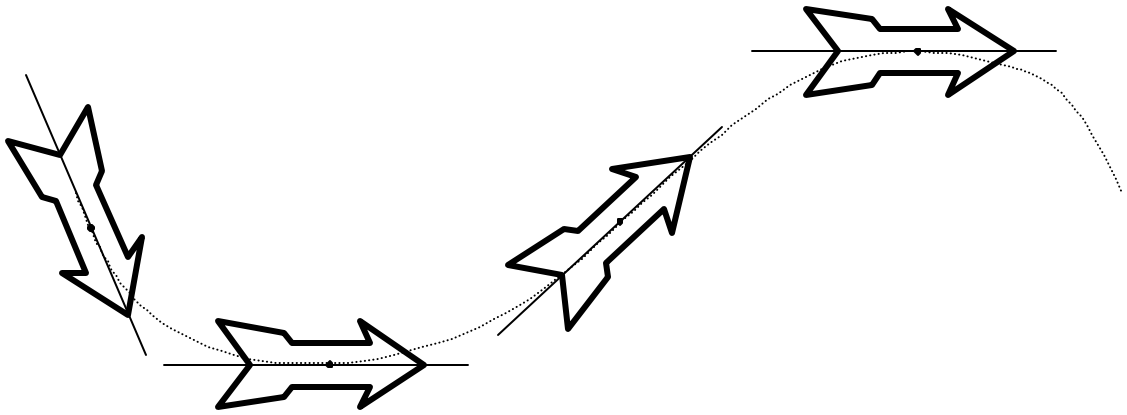
While he paints with his right hand, he twirls the knob with his left. If the arrow rotates simultaneously (apparently) with the knob rotation about point *P* which translates simultaneously (apparently) with stylus translation, we will say he is painting with a brush which rotates in realtime.

The knob, or potentiometer, used in this example could just as well have been a joystick, a footpedal, a strain gauge, a voice coil, a keyboard, etc. Or even another tablet or another stylus on the same tablet. The point is only that some other device is used for control. Alternatively, another dimension of one device could be used, if available–e.g., a strain gauge mounted on the stylus so it detects pressure horizontally as well as vertically.

The table paint implementation of the rotating brush is simple. At some earlier time, *n* rotations of the arrow are computed. These *n* brushes are stored in the auxiliary framebuffer, the *table buffer*. In this example, the brushes are so simple that a runcoded version of each could be stored in the table buffer so that a large number of them may be stored there. The paint program selects the particular brush it uses at any moment to be that indexed by a number generated with the auxiliary control device. A simple mapping of these numbers into the numbers 0, 1, 2, ..., *n*-1 suffices.

A Simple Variation: The Tracking Brush

A variation on the rotating brush which does not even need the auxiliary control device is the *tracking brush*. Here the paint program approximates the direction of motion of the painter's stroke from position information only and uses this direction to select that particular brush rotation which causes the brush to lie tangent to the motion curve:



For example, a local interpolating spline might be passed through the last *k* tablet points, the tangent computed at *m* points along this curve, and *m* rotations of the brush written into the framebuffer, chosen to align with the tangents.

A Wealth of Variations

Here is a short list of table paint possibilities, just to indicate the many interpretations the schemata enjoys:

(1) The table buffer holds dots of many diameters. A pressure sensitive control causes the brush stroke to change thickness (in realtime, of course) in direct proportion with the artist's touch.
(2) The color of the brush changes smoothly according to the speed of movement of the stylus. This variation could be implemented without any additional equipment.
(3) The density of dots in the brush changes according to the displacement of a footpedal while the pattern of dot dispersion changes according to the angle of a joystick (cf. airbrushing).
(4) The brush animates because the table buffer holds the inbetweens of a simple animation. A set of lips, for example, could open and close according to the amplitude of the user's voice.
(5) Rubberstamping a smooth shaded sphere, say, could be accomplished such that the highlights are always correctly placed with respect to a given light source.
(6) The contents of the principal framebuffer or of a second additional framebuffer is used as an index into the table buffer instead of using an auxiliary analog control device (suggested by Lance Williams).

Final Remarks

Table paint is an example of how to interpret ordinary general purpose graphics equipment in an extraordinary, but simple, way and gain considerably thereby. A framebuffer is used to hold a table of pictures (i.e., brushes) instead of a single picture. Then simple table lookup makes a variety of apparently difficult brush dynamics quite easy to implement without special purpose equipment. Large amounts of cheap memory will make table lookup algorithms, such as the

table paint class, particularly attractive to computer graphics installations from here on.

Only a few of the many possible table paint variations have been implemented. I implemented realtime rotating and swelling brushes at NYIT in 1978 and Garland Stern implemented an animated brush there in 1979.

References

[1] Alvy Ray Smith, *Paint*[2], NYIT Technical Memo No. 7, Jul 1978 (also in SIGGRAPH '78 and SIGGRAPH '79 Tutorial Notes, Aug 1978 and 1979 [and at SIGGRAPHs '80-'82 too]). [Published in: **Tutorial: Computer Graphics**, edited by John C Beatty and Kellogg S Booth, IEEE Computer Society Press, Silver Spring, MD, 2nd Edition, 1982, 501-515.]

[2] Nicholas P Negroponte, *The Return of the Sunday Painter* in **The Computer Age: A Twenty Year View** (Eds. Michael L Dertouzos and Joel Moses) MIT Press, Cambridge, MA, 1979.

[2] Called *PAINT* in original paper.