Texas
(Preliminary Report)
With Addendum of 21 Aug 1979

Alvy Ray Smith
Computer Graphics Lab
New York Institute of Technology

Technical Memo No. 10
27 July 1979

Published as Tutorial Notes at SIGGRAPH '79 (without Addendum). This document was reentered by the author in Microsoft Word on Nov 5, 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.

Introduction

Texas—from TEXture Applying System—is a set of programs for moving pictures about in 3-space to obtain other pictures. Texas makes pictures comprised of other pictures embedded in 3-space. Here a *picture*, or *texture*, may be thought of as the contents of a framebuffer, although it may be stored temporarily in a disk file or disguised as a mipmap. Texas takes as input a stageset, which is a set of flats, where a *flat* is a picture painted (mapped) onto the surface enclosed by a planar, convex quadrilateral. Its output is a 2-dimensional rendering of a stageset as viewed from some vantage point in 3-space.

Texas features incremental rendering of textures in perspective, depth darkening, front and back textures for each flat, z buffering or priority sorting for hidden surface removal, antialiasing of edges or tag buffering for ex post facto dekinking. It uses the Picture System II driver by Jim Clark, the mipmap utilities of Lance Williams and Dick Lundin, and has an interface to Garland Stern's 3-dimensional animation program Arbor (or Boop) and to Ed Catmull's polygon rendering program Render.

Texas is intended to be a generalization of the multiplane camera. The current version lacks transparency [but see Appendix], however, so cannot yet do multiplaning. Another inadequacy at present is the priority sort algorithm which is simple minded. Intended additions are light sources, shadows, global partial transparencies for individual flats, correct pixel preimage averaging, as well as local transparency and a better sort.

Usage Manual

Texas

An interactive tablet-driven program for creating a stageset on the PSII (on the [DEC PDP] ll/45 only). If you have never run this program before, you should initialize your .texastate in some way. The following will work:

```
cp /usr/alvy/.texastate usr/[yourname]/.texastate
```

(See Edtexstate below.) Then Texas comes up at the PSII station by typing “texas” at the terminal. The complete usage is given by :

```
texas [-] [-e[oi]] [-f]
```

where,

`-`	alone causes the usage message to be printed
`-e`	puts eases on both ends of animation created by Texas (see “animate stageset” below)
`-ei`	puts an ease in to the animation only
`-eo`	puts an ease out from the animation only
`-f`	displays a fieldguide on the nonmenu portion of the PSII display for positioning

All of the menus are arranged on a tree with “Texas Main Menu” at the root node. Each menu has a “return” button which returns you up one level in the tree (assume the root node is at the top) and an “exit” button which exits the program (down and out on the stylus also exits the program).

To get started, you should push the “create directory” button and type in the name of a directory where you want your stagesets to be stored. This button does not create the directory. It merely enters its name in your .texastate. You will henceforth come up with this directory assumed. For example, the “select stageset” button will automatically list the contents of this directory on the PSII.

If there are no stagesets in your stageset directory, you may place one there with the “create stageset” button. “Delete stageset” works as advertised.

The main entry point to the subtrees of Texas is the button marked “manipulate stageset”. If you have just created a stageset, then the button of most interest at this level is the “create flat” button. Each time you press this button, a canonical flat is appended to your stageset. Appended flats are named by default `f0`, `f1`, `f2`, ... These names may be altered with Editset without ill effect. If you want to type in your own flat coordinates, you should create a dummy stageset, with this button, which has as many flats as you will need. Then use Editset to enter the desired coordinates.

The individual flats can be moved about in 3-space with the “transform flat” button, or the entire stageset can be moved about in 3-space as a unit with the “transform stageset” button.

Two Important Points:

(1) WHAT YOU SEE IS WHAT YOU GET. The outermost box on the PSII display represents the edges of a framebuffer. If you were to render the current stageset as seen by the current camera (see the “camera” button in the main menu) into a framebuffer using Tender, you would get the flats in the arrangement shown on the display, clipped as shown. This is true regardless of what the numbers in the stageset might indicate (see Prset below).

(2) “UPDATE STATESET” IS THE BUTTON WHICH COUNTS. All changes displayed on the monitor are on a local copy of your stageset until you press the “update stageset” button. So the “transform” buttons are like the Unix text editor which edits a local copy of your file until you give it the “w” command. Since a

lot of work can be lost if you forget to perform a permanent update, Texas puts in some reminders here and there.

The buttons “transform flat” and “delete flat” will not work unless a particular flat has been selected with the “select flat” button. This level provides two ways for selecting a flat. One is “hit” for hit testing. The other is useful when the stageset is so complex that hit testing becomes ambiguous. It is the “step” mode. Each time you press the stylus down, the next flat in the stageset is selected and brightened on the display. If the “verbose” switch (up a level) is on, pertinent data about the selected flat is printed on the terminal.

Both the “transform” levels will spew out copious information on the terminal if the “verbose” switch is on. This is useful for making precise adjustments to a flat’s or stageset’s position. For exact placement, however, Editset should be used after getting the flats roughly in position. All the transformation controls (“scale”, “rotate”, “move”) operate relative the position where the stylus is first pressed down. They all respond to horizontal motions only (even the “move” “y”!), and these must be in the outlined section of the display (the nonmenu section).

Since flat intersections are computed in software, turning the “intersections” switch off will always speed up display. The intersections are quite useful for positioning of flats however.

The “manipulate stageset” section of Texas is used for changing the contents of a stageset. The “camera” section is used for changing how a stageset is viewed, but does not affect its contents at all. Changes effected in the “camera” subtree alter the contents of the user’s .texastate. (This is clumsy and will be changed in future versions.) Specifically, Texas provides four cameras which may be independently moved about a stageset for different views, and a fifth camera, called the “oracle” which looks at the currently selected camera looking at the current stageset. The parameters describing these five cameras are stored in .texastate and are used by Tweenset, Tweenex, and Cpcam. The viewing frustum of the current camera is displayed when the “oracle” switch is on. The oracle is useful for setting near and far clipping planes. The most effective use of the intensity depth cueing of the PSII requires that the near and far clipping planes be as tightly placed around the stageset as possible without (undesirable) clipping.

A useful button for getting started at the “camera” level is the “initialize” button. This causes reseting of the current camera’s parameters to a canonical set of values.

WARNING! At several levels in Texas the current camera’s view can be altered in azimuth and elevation by moving, respectively, horizontally and vertically with the stylus in pressure in the outlined section of the display. These moves are also recorded in .texastate.

The “animate stageset” button at the main menu level enters a crude animation level permitting animation between two current camera views. This is not yet a fully developed section of the program. It is intended for animation between camera views only. For animation of the stageset itself, or of flats within a stageset, Arbor or Boop should be used. (See Flip and Unflip below.) The buttons for “eases” are not yet implemented. Use instead the Shell level `-e[oi]` options.

Tweenset

This program is similar to the "animate stageset" button in the main menu of Texas, which may be thought of as a preview button for Tweenset. Tweenset generates the files needed by Tender for rendering. If Tweenset is to be executed on the [DEC PDP] 11/70, your .texastate should be copied over from the 11/45 when you bring your stageset over. It should be installed in the directory of the same name on both machines. Tweenset uses the camera parameters stored in .texastate.

Here is a usage summary:

`tweenset [stageset name i j n] [opts]`

`i`, `j` = camera numbers (0-4)
`n` = number of frames
inbetweens of stageset are placed in `name`.0, `name`.1, …, `name`.(`n`-1)

and `[opts]` are:

`-c`	clear fb (or Tek4014) between each frame
`-d [t]`	draw each frame (`t` implies in the 4014) (NB, no clipping done when drawing to 4014)
`-e`	puts eases on both ends of motion
`-ei`	ease in only
`-eo`	ease out only
`-f` *m*	generate frame number *m* only
`-r` *x y z*	rotate stageset *x* (degrees) about x axis, *y* about y, *z* about z
`-s` *x y z*	scale the stageset
`-t` *x y z*	translate the stageset

Each file generated by Tweenset is a stageset with a set of camera parameters appended. Prset can be used to print the stageset contents of these files and Editset to modify the stageset section.

The –`r`, `-s`, `-t` arguments are applied in that order, and are hence not very powerful. Once again, for other than simple motions, a 3-dimensional animation program should be used.

Tender

This is the rendering program for Texas. It renders a stageset with a camera description appended, as generated by Tweenset. So each frame of an animation of a stageset requires another invocation of Tender. As currently conceived, the program will use anywhere from 0 to 10 framebuffers. The most complex version so far, however, uses only 3 for target, 1 for source, 2 for z-buffer, and 1 for tag buffer, for a total of 7 framebuffers. Local transparency will require another 1 for source, and correct pixel preimage computation will require yet another 2 framebuffers for source.

Here is a usage summary:

`tender [filename] [opts]`

`filename` contains a count, a list of flats, and a set of camera parameters

`[opts]` are:

`-`	print usage message
`-a`	do antialiasing
`-A`	generate alpha, or matte, buffer (antialiased if `-a` given) (forces `-t` flag but all objects are tagged at 255)
`-b` *backname*	render *backname* texture on backfacing flats
`-B`	debug flag
`-c`	clear target buffers(s) to 0
`-d[t]`	causes outlines to be drawn (`t` implies Tex4014)
`-D[t]`	draw outlines only (forces `-d[t]` flag)
`-f` *flatnum*	render given flat only
`-h` *hither yon*	set hither and yon (default = -1., 1.) (forces `-z` flag)
`-H` *f*	set hither and yon to min and max z (forces `-z` flag) (max darkness occurs at *f* times total darkness (*f* = 1. typically, 0 *f* 1))
`-i`	initializes z-buffer (if present) and tag buffer (if present)
`-m`	source includes a matte buffer
`-M`	choose average of xinc and yinc as d coordinate (max is default) [xinc, yinc, d are mipmap parameters]
`-n`	do not use pictures specified in stageset
`-p`	use pixel preimages, not mipmap approximations
`-P`	point sampling (no mip, no antialiasing, no preimage computation)
`-s`	use priority sort instead of z-buffer
`-t`	turn on object tag buffer
`-T` *tag*	set initial tag to given number (forces `-t` flag)
`-v`	verbose flag
`-Y`	luminance only (do not use with `-P`)
`-z`	turns on depth darkening
`-Z`	turn on z-buffer calculations

target uses first 1-3 FBs, source the next 1-4, `z` the next 2, `t` the next 1.

The priority sort used with the `–s` flag is simply: Render all backfacing flats first then all frontfacing flats. It is not difficult to create stagesets for which this ordering fails. Future versions of Tender will have more sophisticated ordering algorithms employed.

The texture, or picture, attached to each flat in a stageset may be set with the “assign texture” button in Texas (“manipulate stageset” level) or with Editset. The textures thus fixed to a stageset may be ignored, however, by use of the `–n` option of Tender.

There is a flag associated with each flat in a stageset called the “active flag”. Usually this flag is TRUE, but if it is FALSE the corresponding flat is ignored by Tender. Editset can be used to change this flag.

Antialiasing of the edges of a flat may be accomplished with the `–a` flag if priority ordering is used. If z-buffering is used for hidden surface calculations, then approximate antialiasing may be effected by using the `–t` option to generate a tag buffer. Then a pass with Dekink (by Lance Williams) smooths the edges.

A note on mipmaps is appropriate here. In general, a pixel to be rendered in the target picture has a preimage in the source picture which partially covers a set of pixels in the source. The complete correct computation of the average color of this preimage is quite expensive. In his PhD dissertation work, Ed Catmull came up with the scheme of preaveraging the source picture to cut down this computation time. In this scheme, if you know that each of your target pixels has a preimage of roughly $m \times n$ source pixels, you generate an averaged version of the source picture with is $\frac{1}{m} \times \frac{1}{n}$ the size of the original. Typically, a set of such preaveraged sources is needed for good rendering.

Lance Williams invented the elegant format embodying this notion known as the "mipmap" of the source. In this format, three-fourths of one framebuffer holds the red, green, and blue components (one fourth each) of the source averaged down by a factor of 2. Three-fourths of the remaining one quarter of the framebuffer holds the red, green, and blue components of the source averaged down by a factor of 4. And so fourth(!) Incidentally, "mip" stands for "multum in parvo", or "many things in a small place". So not only does the mipmap cut down on computation time if the same texture is to be used as a source many times, but it also cuts down on framebuffer requirements. A full-color (24-bit, RGB) source picture fits in one framebuffer. Tender assumes mipmap format (except for the `-P` option) and uses the assembly coded routines written for mipmap access by Dick Lundin.

The mipmap format has two drawbacks: Its approximation to a pixel preimage is highly inaccurate if the correct preimage is elongated. The highest resolution available is half that of the true source. For these reasons, a future version of Tender will compute exact pixel preimages, despite the cost in space and time, if the user so desires.

Prset

Usage of this program is given by:

```
prset stageset [all]
```

where presence of the `all` argument (eg, the letter 'a' will do) causes inactive flats to be printed also. See file /usr/alvy/header/flatdefs for the structure of a flat.

Editset

Usage of this utility program is:

```
editset stageset [all]
```

where the `all` flag permits editing of the inactive flats also. Thus inactive flats can be made active. Editset is a one-pass editor—ie, you cannot back up a line. However, you may exit with a control-C at any time and have the stageset updated as indicated up to the time of exit. This is true even if the end of file has not been reached.

Vertexstate

The user's .texastate (/usr/[username]/.texastate) is printed on his terminal. See files tstatedef and kamdefs in directory /usr/alvy/header for details of the

.texastate structure. The principal motivation for this file is continuity between successive uses of Texas.

Edtexstate

The operation of this editor is similar to Editset above but for the user's .texastate instead of a stageset.

Cpcam

Usage:

```
cpcam i j
```

copies the parameters stored for camera `i` in the user's .texastate to the parameters stored there for camera `j`.

Flip

This program converts a stageset to the format required by Arbor or Boop for 3-dimensional animation with spline interpolated parameters. It is used as follows:

```
flip [stageset name] [-]
```

It outputs one file named `stageset`.a and also one file for each flat named `name`0.v, `name`1.v, … . Thus a stageset is interpreted as a root node with *m* leaves if there are *m* flats. Fancier interpretations are possible, of course, but Flip will not generate them. For example, a stageset might be thought of as comprising two substagesets (subsets). This implies a 3-level tree for Arbor.

Unflip

Arbor (or Boop) can be used to generate from one (Flip'd) stageset many frames of animation. The inbetweens generated by Arbor are called "frames". Unflip converts frames back into stagesets suitable for rendering by Tender. It is called as follows:

```
unflip [-] arbordir framename stagename stageproto camnum
```

where

`arbordir` = Arbor directory where *.v files are located
`framename` = an Arbor frame filename
`stagename` = the equivalent stageset filename
`stageproto` = prototype stageset (ie, with texture names)
`camnum` = camera (0-4) transform to be appended to each stageset

Arbor uses the PSII for animating a stageset. As of this writing, Arbor does not give you what you see on the PSII. Thus the last argument is required to cause the stagesets generated by Unflip'ing frames to fill the framebuffer viewport as desired. The desired camera view should be set up before passing a stageset to Arbor.

Tweenex

This program is similar to Tweenset above:

```
tweenex [stageset name i j n] [-e] [-t]
```

where,

`i`, `j` = camera numbers (0-4)
`n` = number of frames
`-e` = put eases on both ends of all motions
`-t` = draw each frame on a Tek4014

It generates a sequence of files with names `name`.0, `name`.1, … which are in the format required by Ed Catmull's polygon rendering routine Render. The files must be passed through Ed's program Ysort before being sent to Render.

Addendum 1—8/21/79

The Texas rendering program Tender has been updated from that described in the preceding document by the addition of local transparency. This and several other additions will be described briefly here.

Local transparency is implemented via another source framebuffer, the alpha buffer. Thus, if the `-m` flag is specified, the source picture is assumed to consist of two mipmaps. The first is as before. The second is a mipmap of the alpha, or matte, buffer describing the transparencies of the pixels in the first mipmap. In this version, it is up to the user to fetch the desired alpha mipmap into the alpha buffer.

The priority ordering algorithm has been improved slightly to omit completely obscured flats if the `-s` option is specified. Since this may give undesired results for the case of local transparency (`-m` flag), it is not done in this case unless the new `-C` flag is specified. The `-C` flag forces the covering step of the priority algorithm which is normally turned off if `-m` is given.

Just the alpha matte for the target picture is generated if `-j` is given. No target picture is generated.

The clear option `-c` has been changed to require a numerical argument (`-c` *pv*). This allows initial clearing of the target framebuffers to pvalue *pv*.

Some general words about the Texas system: It is intended for a small number of complexly rendered flats. It should handle 50 flats but will be tediously slow for such a large number unless they are quite small, do not need z-buffering or tag buffering, and need only a few different textures. Although local transparency and antialiasing are implemented for the case of z- and tag buffering, they are not theoretically valid in this context and should be used with this fully in mind.

Summary

Texas consists of the following programs:

Texas
: Picture System II (PSII) program for creating a stageset

Tender
: (texas render) renders a stageset into a framebuffer

Prset
: prints out stageset to a terminal

Editset
: allows editing of a stageset

Tweenset
: generates inbetweens between two views of a stageset

Vertexstate
: verifies (prints out) the user's .texastate file

Edtexstate
: allows editing of .texastate

Cpcam
: (copy camera) a commonly used .texastate edit

Flip
: converts a stageset to format required by Arbor (or Boop)

Unflip
: converts an Arbor file to stageset format

Tweenex
: converts stagesets into the format expected by Ed Catmull's polygon rendering program Render and does inbetweening (cf, Tweenset for Tender)