PHILIPS

PHILIPS INTERACTIVE MEDIA of America

Technical Note #84

ArtSpace Animation Conversion Production Pathway

Blake Senftner

April 2, 1993

This note describes the use of the program **anim_converter**. This is a UNIX-based program that takes a series of TGA or RGB888 images and converts them to a set of RL7 images suitable for compiling as an animation to be used as part of a CD-I title.

ArtSpace Animation Conversion Production Pathway

This document describes the use of a program called **anim_converter**. This UNIX based program takes a series of TGA or RGB888 images as input and converts them to a set of RL7 images suitable for compiling as an animation to be played within a CD-I title.

This program is capable of both creating "video windows" for play on top of still image backgrounds and freeform animated "accents" for playing over still images, such as an animated leaf blowing down a long, windy road. When non-video window animations are processed with **anim_converter**, two types of anti-aliasing are available to minimize edging between foreground and background elements.

Why Not Use Photoshop?

Imagine a conservative title that employs only ten animated video clips, each one playing at 15 frames per second and lasting for 30 seconds. The typical production team's plan is to use Photoshop, after capturing the video with a VideoSpigot card, and then to use the Optimage Conversion Stack to create the 4500 (10 * 15 * 30) single image frames; these then need to be masked with the transparent color, scaled, and placed by one of the production's graphic artists. Assuming the artist is using a fast Macintosh computer, he or she will spend a minimum of 3 minutes per frame. Based on that three-minute figure, the graphic artist will have to spend about 225 hours converting the captured frames for use in the CD-I title.

It is hard to image an artist who would enjoy the of repetitive task of hand converting 4500 images, not to mention the human errors that are generated by such repetition. This is what I call "data slavery." It is the worst example of what a job using a computer can become. The beleaguered graphic artist who is stuck with that task should be replaced with software. Products like DeBabelizer do exist on the Macintosh, but with in house source code and the speed of our SPARC stations, it became a question of our desired level of control.

Faced with the situation of our **FLIGHT** disc, currently in production at ArtSpace, which has 30 video window clips, each about 15 seconds, and an additional 25 animations, 8 seconds each, I wrote **anim_converter** for use by the production team. Anim_Converter features speed beyond that available on the Mac, antialiasing filters that have been optimized for "cel" and "background" images to be composited at run time via the two plane technique, and control over the CLUT creation beyond that currently available on the Mac or other platforms.

Anim_Converter Requirements

The program **anim_converter** is UNIX based. It reads in a text script file of instructions. The program requires two flags, in addition to the script file name, when being executed. The first flag specifies where the **tools** directory can be located. The second flag specifies a work area that has sufficient free space.

The **tools** directory is a directory that contains the following image conversion tools:

`vist2rgb`	Converts a TGA (Vista) file to an RGB888 file
`rgb_deinterlace`*	De-interlaces RGB888 images captured from interlaced video sources
`rgb_makmask`*	Creates an image that is black at transparent pixels and white where not transparent
`rgbtodyuv`	Converts an RGB888 file to a DYUV file
`rgbscale`	Scales down an RGB888 image
`rgb_defringe2`*	Compares scaled foreground, background and mask images and then makes corrections
`rgb_lessnoise`*	Reduces "noise" in RGB888 images intended for conversion to RL7
`rgb_contrast`*	Contrast enhancement filter for RGB888 color or gray scale images
`toccir`	CCIR filter for RGB888 images
`rgb_place`*	Places one RGB888 image inside another RGB888 image
`rgb_antialias`*	Antialiases one partially transparent RGB888 against a background RGB888
`rg8glue`	"Glues" a series of RGB888 images together
`findclut`	Given an RGB888 image, creates CLUT file from parameters
`rgbtoclut`	Converts an RGB888 to a CLUT image

* *These image filters were written specifically for use by **anim_converter**.

The **anim_converter** program requires a work area at least as large as the size of the entire animation saved as RGB888 images at CD-I resolution, plus space for the background image in RGB888 format. To explain the size of the required work area, the process/pathway for **anim_converter** is detailed in the next section. A sample calculation for the disk space requirement for a video clip composed of 200 frames at NTSC resolution follows:

(200 + 1) * 384 * 240 * 3 = 53 megabytes

where the **(200 + 1)** represents the 200 video frames, plus the background image, and the 384 * 240 * 3 represents the size of each image.

What Does anim_converter Do?

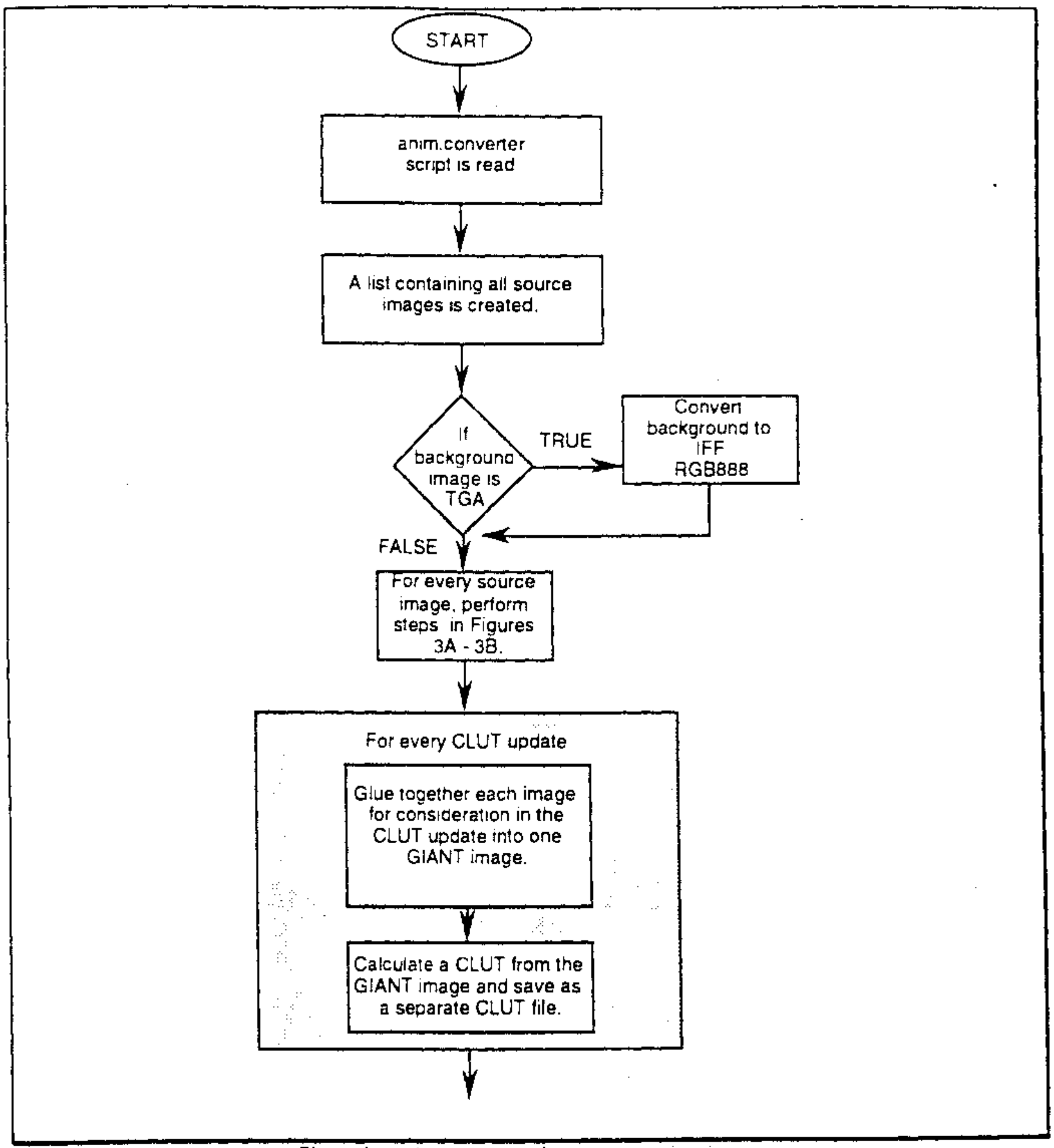

START
anim.converter script is read
A list containing all source images is created.
If background image is TGA
TRUE
Convert background to IFF RGB888
FALSE
For every source image, perform steps in Figures 3A - 3B.
For every CLUT update
Glue together each image for consideration in the CLUT update into one GIANT image.
Calculate a CLUT from the GIANT image and save as a separate CLUT file.

*Figure 1: **anim.converter** begins processing images.*

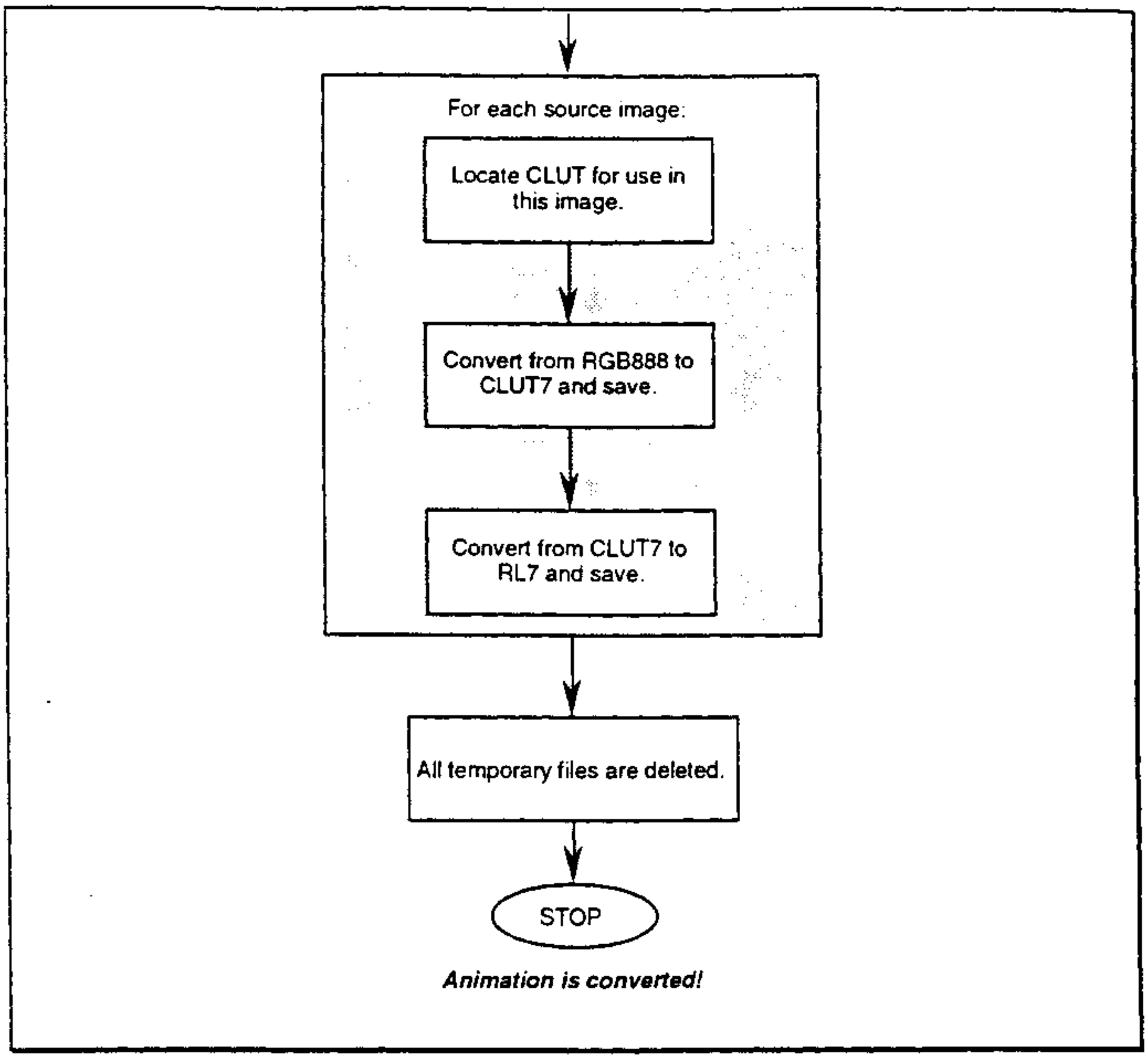
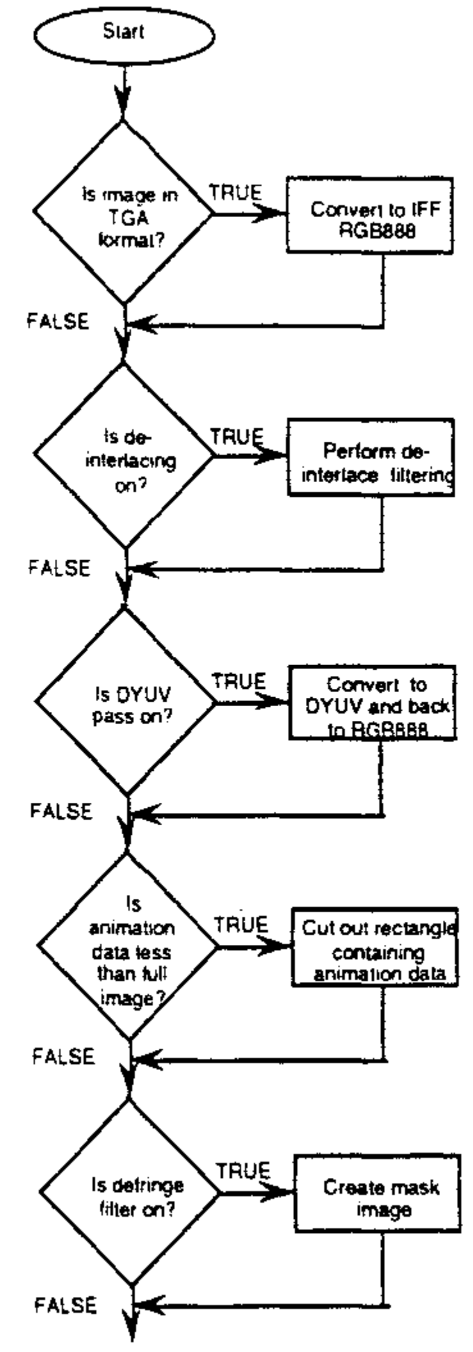

For each source image:
Locate CLUT for use in this image.
Convert from RGB888 to CLUT7 and save.
Convert from CLUT7 to RL7 and save.
All temporary files are deleted.
STOP
Animation is converted!

Figure 2: anim.converter completes processing images.

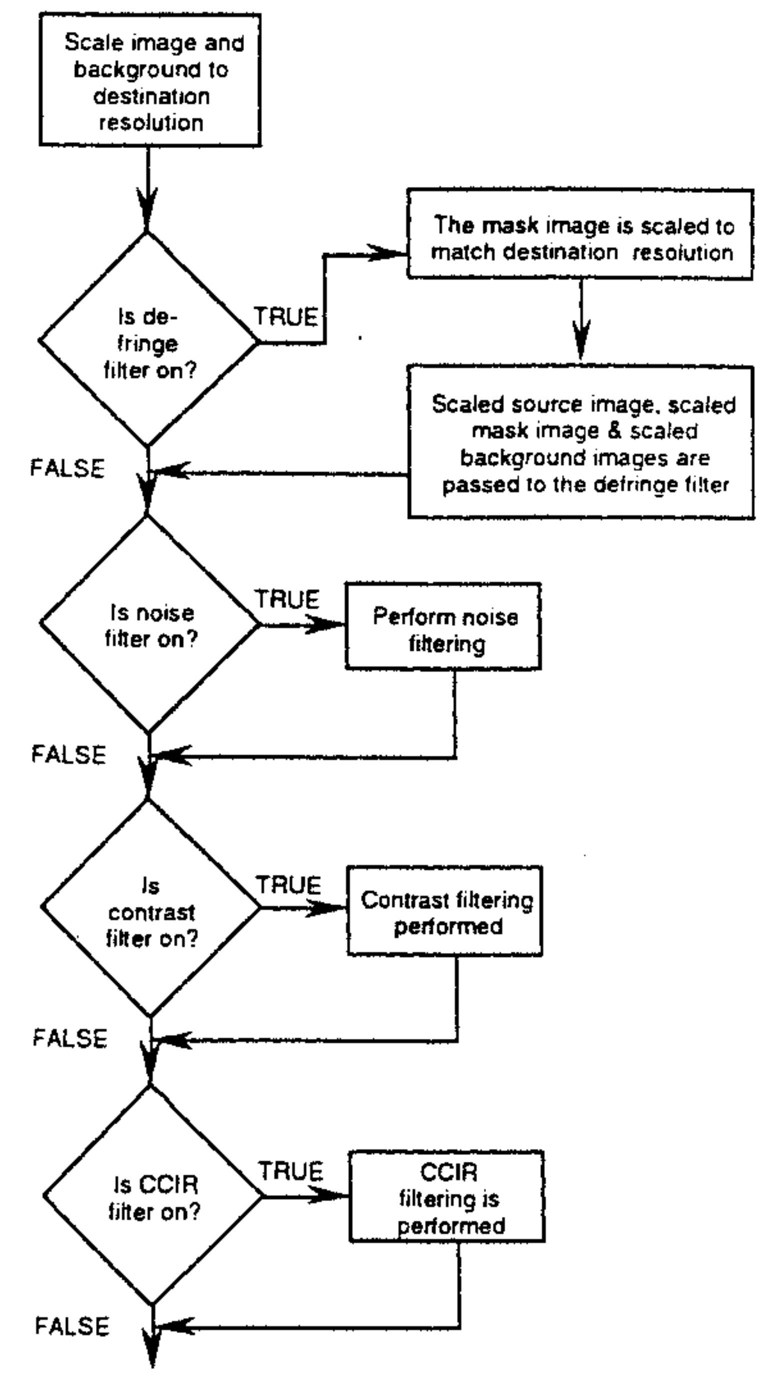
It can be seen that for the CLUT selection operation to work, the entire animation must be available as RGB888 images at the destination resolution and filtering. This is the reason the work area needs to be large enough to hold the entire animation as RGB888 images in CD-I resolution.



Continued in Figure 3B.

Figure 3A: Steps performed on every image.

Continued from Figure 3A.


Scale image and background to destination resolution
Is de-fringe filter on?
TRUE
The mask image is scaled to match destination resolution
Scaled source image, scaled mask image & scaled background images are passed to the defringe filter
FALSE
Is noise filter on?
TRUE
Perform noise filtering
FALSE
Is contrast filter on?
TRUE
Contrast filtering performed
FALSE
Is CCIR filter on?
TRUE
CCIR filtering is performed
FALSE

Continued in Figure 3C.

Figure 3B: Continuation of steps performed on every image.

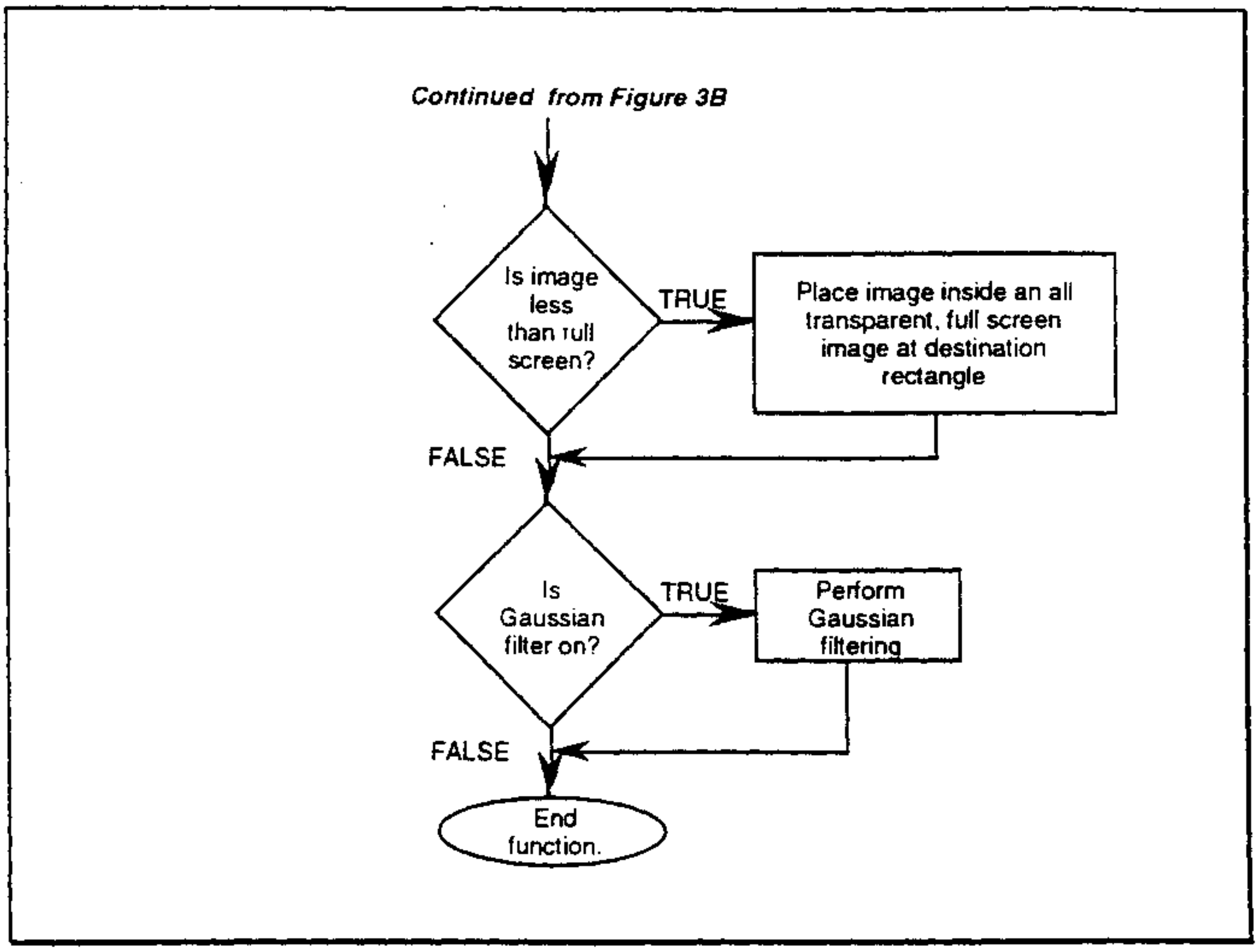

Continued from Figure 3B
Is image less than full screen?
TRUE
Place image inside an all transparent, full screen image at destination rectangle
FALSE
Is Gaussian filter on?
TRUE
Perform Gaussian filtering
FALSE
End function.

Figure 3C: Final steps performed on every image.

Anim_Converter Syntax

An **anim_converter** script requires at least five commands. These required commands are the first commands detailed below. The rest of the commands are not specified in alphabetical order; rather, they are listed in the order in which they are executed. This way, one can logically follow the results of using one filter after another.

* * *

`source    tga|rgb  /wild/card/directory/specification/*.tga`

This is the first required command. It specifies the location and file format of the source images. For example:

```
source tga /op/grey_filter/tga_frames/*.tga
```

This command specifies that all the files that fit the specified wild card are of TGA format and are to be the input frames. Because the frames are TGA format, they will need to be converted to RGB888 during pass one of the program.

* * *

`dest_name      ntsc|pal /directory/specification/with/prefix`

This is the second required command. It specifies the filename prefix for each animation, as well as the "compatibility" of the image. For example:

```
dest_name ntsc /op/grey_filter/rl7_frames/clip1
```

This specifies that each RL7 file name will look like **clip10001_n.rl7**. The prefix **clip1** becomes the first part of the file name; the directory specification determines the location where is the image is to be saved, and "compatability" places an _n for NTSC images and an _p for PAL images.

* * *

`bkgnd            tga|rgb  /directory/filename`

This is the third required command. It specifies the background image's location and file format of the image. For example:

```
bkgnd rgb /op/grey_filter/backgrounds/anim_clip1_bkgnd.rg8
```

This specifies that the named image is an RGB888 image for use as the background image. This image has two purposes: first it determines the destination resolution of the animation conversion and second, it serves as input to the antialias filter.

* * *

```
src_rect       left      top        right     bottom
```

This is the fourth required command. It specifies the source rectangle that is to be cut out of the source frames to serve as the actual animation frame. This command is critical for captured video, because most video capture hardware leaves a scan line or two of garbage at the top or bottom of the image. The left, top, right and bottom parameters are numbers. For example:

```
src_rect  8  1  753 482
```

This specifies the rectangle to cut out when using a Vista Board in an IBM PC, because the Vista Board leaves 8 pixels of junk on the right edge as well as in other locations. So, this command directs the program to use the subsection of the images that do not contain video garbage.

* * *

```
dest_rect      tga|cdi  left      top  right     bottom
```

This is the final required command. It specifies the destination resolution of the animation and the location at which the animation is to be placed inside the background image. Note the **tga|cdi** flag in this command; it allows for the coordinates to be specified in "virtual Targa" resolution or actual CD-I resolution. This "virtual Targa" capability was added because the ArtSpace graphic artists work only with TGA resolution, and do not identify with CD-I resolution. For example:

```
dest_rect tga  35  271  309  448
```

This specifies that the animation is to be placed at that location inside a TGA resolution file. The program converts these numbers to CD-I resolutions to get the actual location of the destination rectangle.

* * *

```
deinterlace   off|on
```

This turns the de-interlace filter on or off. The default value for this is **off**. Thus, if this command is not included in your script, the filter is assumed to be off. The method of de-interlacing employed is to interpolate between even scanlines to create the odd scanlines.

* * *

```
thru_dyuv     off|on
```

This turns the function that passes RGB888 images through the DYUV file format on or off. This is useful when an animation must match edit with a DYUV frame at the beginning or end of the animation. All this does is convert the images to DYUV and back to RGB888.

* * *

`noise_filter %age`

This activates the RL7 noise suppression filter. If this parameter is not specified, this filter defaults to **off**. This filter looks at every pixel in the image, comparing it to the pixels on the immediate right and left. If the pixels to the right and left are the same color, then the pixel being examined is made the same color. The percentage parameter specifies how much difference there can be between the pixels before they are determined to be the same color. For example:

```
noise filter 2
```

This specifies that the RL7 noise filter is to be used and that all single pixels with less than 2% difference in color from their immediate neighbors will be suppressed.

* * *

`contrast [grey]        clipmin  clipmax  scalemin scalemax`

This activates the contrast filter. If this parameter is not specified, the contrast filter is not used. The optional "grey" parameter instructs the filter to force the images to NTSC black and white. NTSC black and white consists of the formula

```
grey = 0.3 * red + 0.59 * green + 0.11 * blue
```

The **clipmin** and **clipmax** parameters specify the range of allowed values for a color. If a color is below **clipmin**, it is set to **clipmin**; likewise, if a color has a value above **clipmax**, then it is set to **clipmax**. The **scalemin** and **scalemax** parameters specify the values to which the range of colors is to be scaled. For example:

```
contrast  grey  40 200 16 225
```

This specifies for the images to be converted to NTSC greyscale, the colors clipped to 40 as a minimum and 200 as a maximum and then scaled so that the colors that were 40 through 200 are now 16 through 225.

* * *

`ccir    on|off`

This turns the CCIR filter on or off. The default setting is **off**. In the case where the contrast filter has been used, this filter is not needed because the scale factors for the contrast filter can be set to insure that all the colors lie within CCIR ranges.

* * *

`trans_col    red    green    blue`

This specifies the transparent color for the source frames. If this is not specified, the values are assumed to be:

```
    red      =    0
    green    =    0
    blue     =    0
```

* * *

`antialias    off|defringe|gaussian`

This turns off the anitaliasing filter or selects between two possible modes of operation. The default setting is **off**.

When the **defringe** option is selected, a mask image is created before the image is scaled to CD-I resolution. When the image is scaled down, the mask is also scaled down. This results in a mask that contains grey levels equal to the amount of mixing between the source imagery and the transparent background. Given these values, the transparent color mixed into the source image is removed and the equal amount of background color is added back in. The creates "blue screen" style animations that have background color data antialiased into the foreground animating data.

When the **gaussian** option is selected, the transparency edge in each source frame is identified, and a gaussian filter is applied between the edge of the foreground image and the background image. In effect, the outer edge of the animation data is blurred with the background data.

While both methods work, **defringe** requires that the foreground (animation) data must be scaled for the mask to have grey levels in it to provide the anti-aliasing parameters. The "gaussian" filter may not work satisfactorily if the foreground data has been scaled; this is why the "defringe" method was created. If scaling is to occur, best results can be achieved with the largest possible resolution for the source frames and defringe chosen.

* * *

`first_frame   number`

This specifies the frame number of the file name of the first converted image. For example:

```
first_frame 10
```

This specifies that the first frame will be named **clip10010_n.rl7**, rather than the default setting, which would result in the file name of **clip10001_n.rl7**. Of course, the prefix of the animation file name and compatibility setting are specified in the **dest_name** command.

* * *

`frame_inc      number`

This specifies the increment between frames. If not specified, the default is one.

* * *

```
clut_updates          number
update_frame     size_of_clut  <<   frame1  frame2  ...  frameN
```

This multi-line command specifies the number of CLUT updates in the final animation, the frame number of each CLUT update, the size of the CLUT for the update, and the frames to be used to create that CLUT. For example:

```
clut_updates    1
  1  32   <<   1 10 200 500
```

This specifies that the animation has a single CLUT update. The update takes place on the first frame (frame 1) and the size of that CLUT is 32 colors. The source frames 1, 10, 200, and 500 will be used to create the CLUT.

If no CLUT update is specified, a default update is assumed on the first frame, created from the first frame, and of size 128.