DTA Help Contents

About DTA

Topics:

Overview of DTA

DTA is a command-line utility for creating Autodesk Animator .FLI and .FLC animation files from:

- .TGA files as created by the POV-Ray and POLYRAY ray-tracers.
- .IMG files as created by the Vivid ray-tracer.
- .PCX files. .DIB or .BMP files
- .DIB or .BMP files
- .GIF files.
- Other .FLI or .FLC files.
- VistaPro .VAN animation files
- Presidio .ANI animation files

DTA can also perform a wide range of post-processing functions on image files including:

- Creating a single optimal 256-color palette from a series of truecolor pictures, and then creating an Autodesk Animator .FLI file out of them.
- Save the palette as a .MAP (PICLAB, FRACTINT) or .COL (Autodesk Animator) palette file.
- Convert pictures to a bunch of different still image formats.
- Read in a palette file in either .COL or .MAP format and animating a bunch of pictures using that palette.
- Arbitrary rotation.
- Scaling.
- Multi-layer compositing.
- Averaging images together for a variety of effects, including simulated motion blur and red/blue 3D.
- And more...

About DTA Help

CompuServe ID: 76546,1321

This Help File was produced and edited by Robert McGregor at Screaming Tiki (tm).

CompuServe ID: Robert McGregor 73122,3125

Memory Problems and DTA

IF YOU ARE HAVING MEMORY PROBLEMS WITH DTA, READ THIS

Real Mode vs. Protected Mode

DTA used to come in two varieties: a real mode version and a protected mode version. From now on there's only a protected mode version. There's no more DTAX.EXE.

DTA will let you access up to 16MB of extended memory (either raw or configured as XMS with HIMEM.SYS or QEMM386.SYS or whatever). It requires at least 1 or 2 MB of RAM to run at all.

Don't delete the files DPMI16BI.OVL, RTM.EXE, and DPMIINST.EXE. These make up the protected mode drivers for Borland Pascal 7.0 (the compiler used to build DTA). DTA won't run without them.

Topics

Shareware Information

The Rules

Feel free to re-upload DTA to other bulletin board systems in its ***original, unmodified*** form. You may not repackage it with other programs. You may **not** repackage it with your own tutorial. I feel like an idiot having to make these demands, but I've been finding archives on BBSs going by names like MORPHPAK.ZIP which are collections of DTA, my other program DMorf, sometimes also Richard Godoeken's program RMORPH, plus Trilobyte's PLAY. That's not kosher. I and those other folks distributed our programs the way we did for reasons.

Do not include DTA on a disk along with any magazine, book, hardware product, or other software product without my permission.

If you place DTA on an Internet FTP site, please drop me a note (**76546.1321@compuserve.com**) so I'll know how to answer when people ask me *"Where can I find DTA on the net?"*

Disclaimer

If you use DTA, you do so at your own risk. I won't be held responsible if it screws anything up.

Support

If you've got any requests/bug reports/suggestions, send a message to **David Mason** on one of the following:

- The **"You Can Call Me Ray"** BBS, (708) 358-5611
- **"The Graphics Alternative"**, (510) 524-2780, and on **"Channel 1"** BBS, (617) 354-8873.
- **CompuServe ID: 76546,1321**
- **From the Internet: 76546.1321@compuserve.com**

You'll probably get some kind of a response (maybe sooner, maybe later)...

Money Matters

DTA is a shareware program. If you think DTA is worth it, send some money or some computer hardware or something to:

David K. Mason
P.O. Box 181015
Boston, MA 02118

I think $35 is an appropriate amount, but feel free to send more or less.

Blatant Shameless Plugs

Other shareware programs:

- **Dave's Morphing program (DMorf),** the first shareware morphing program for the PC.
- **Dave's Flic Viewer (DFV),** an animation player that can handle DTA's FLH and FLT files in addition to conventional FLI and FLC files. (no shareware registration required for registered users of DTA or DMorf)
- **Dave's Self-Viewing Flic Builder (BUILDSV),** which converts flics into executable programs by appending a copy of the flic to a special version of DFV (no shareware registration required for registered users of DTA or DMorf).

For more information on using DTA and related programs, see my books:

"Making Movies on Your PC"
by David K. Mason and Alexander Enzmann
Waite Group Press, $34.95 USA, ISBN 1-878739-41-7
Covers the creation of 3-D animation sequences using Polyray and DTA. The DTA reference info is up to date through release 1.8g.

"Morphing on Your PC"
by David K. Mason
Waite Group Press, $29.95 USA, ISBN 1-878739-53-0
Covers building animated morph sequences with DMorf and DTA. The DTA reference info is up to date through release 2.0.

Shareware Information

Input Formats

DTA supports the following file formats:

.TGA	Targa 16-,24-,32-bit compressed/uncompressed, or 8-bit grayscale uncompressed.
.GIF	GIF (87a or 89a).
.PCX	PC Paintbrush 256-color or 24-bit PCX images.
.DIB/BMP	Microsoft Windows device-independent bitmap files.
.IMG	Vivid raytracer 24-bit RLE output.
.FLI and .FLC	Autodesk Animator/Animator Pro animation files.
.VAN	VRLI VistaPro 256-color animation files.
.ANI	Presidio PC Animate Plus animation files.
.RLE	MindImage 2 SIRDS run-length encoded depth files.
.IFF, .LBM and .ILB	Amiga IFF/ILBM pictures.
.LZH	LHArc compressed archive containing files in the above picture formats.
.ZIP	PKZIP compressed archive containing files in the above picture formats.
@script	Process all files listed in a text file called "script.SCR".

To specify a particular picture in an archive:

```
"ARCHIVE.ZIP:PICTURE.TGA".
```

To specify particular frames in an animation:

```
"ANIM.FLI:first[,last]".
```

Command Line Switches

Syntax

```
DTA (filenames) (switches)
```

Filename Options

/O(name) Specify an output filename.

/FO[#[,#]] Always save to the original filename.

Output Format Options

By default, DTA will create a flic (.FLI/.FLC/.FLH/.FLT) animation file. Specify a different format with one of these switches:

(default) Create a .FLI animation file. [default]

/FB Create Windows .BMP file[s].

/FC Create an Autodesk Animator .COL palette file.

/FD Create Windows .DIB file[s].

/FG Create CompuServe .GIF file[s].

/FI Create grayscale .TIF file[s]

/FM Create a PICLAB .MAP palette file.

/FP Create .PCX file[s].

/FR Create MindImage .RLE file[s].

/FT Create Targa .TGA file[s].

/FV Create a VistaPro .VAN animation file.

/B## **Bits Per Pixel**. Legal values and *defaults*:
TGA: 8 16 *24* 32
Flics: *8*/16/24
PCX: *8* 24 all others: 8
BMP/DIB: 8 *24*

Output Resolution Options

By default, DTA will create output files with the same dimensions as as the input pictures. When creating an animation, it will use the dimensions of the first input picture.

This is sometimes undesirable, as when you"ve got multiple pictures of different sizes, or when you"re compositing pictures.

So, you can override this with one of these switches:

/R# specify output resolution:

1 320x200

2 320x240

3 320x400

4	**320x480**
5	**360x480**
6	**640x480**
7	**640x400**
8	**800X600**
9	**N/A**
10	**1024x768**
11	**N/A**
12	**1280x1024**

/R#,# **specify resolution exactly**

Flic Output Options

/FLX Build a Tempra .FLX file instead of default .FLH (16-bit only).

/FLC build .FLC even when the size is 320x200 (8-bit only).

/P Play in ping-pong, or pendulum, oscillation order.

/Snnn Specify display speed of animation playback (default=0).

/TO# Flic Tolerance; causes DTA to ignore pixel differences between frames when the distance in color-space is less than or equal to a value that you supply.

/TX# Include a complete, no-tolerance frame once every # frames (use only in conjunction with /TO). /TO and /TX work only with FLT and FLH files.

.GIF Output Options

/IG Interlace .GIF files

.TGA Output Options

/NC Do NOT compress .TGA file

/BU Save .TGA files bottom-up instead of top-down

Palette Options for Color-mapped Output

/G Use greyscale instead of a palette. /G represents a 256-level grayscale except when the output format is a 320x200 .FLI file, when it represents a 64-level grayscale. That's because .FLI won't support higher than 64 shades.

/G32 Use 32-level greyscale instead of a palette (there's also a **/G128**, **/G64**, **/G32**, **/G16**, **/G8** and **/G4**). **/G128** represents a 128-level grayscale, **/G64** a 64-level grayscale, etc. Most VGA and SVGA monitors can only display 64 shades of gray, so it's wasteful to use more than that if your picture will only be displayed on a monitor. Some laptop VGA displays can only display 32 shades. If you're going to be printing your image or using it as a bump map or height

field with a rendering program, then the more shades the better.

/332 Use 3/3/2 palette.

/Upalname Use existing .MAP or .COL file for palette.

/M# Set maximum colors.

/MN Do NOT remap colormapped input.

/PO Force single optimal palette generation even when creating GIF files.

/C# Scan only one frame per # for palette.

Dithering Options

/DF Dither with Floyd-Steinberg filter.

/DS Dither with Sierra-Lite filter.

/DO[1-6] Ordered dither (digit represents dither strength).

/DR[#] Random noise dither.

Frame Averaging Options

/A[#|A] Average <number> TGAs per output frame (A=All).

/T[#|A] Trail <number> TGAs per output frame (A=All).

/X[num] eXpand <number>. Create and insert <number> averaged frames between each pair of regular frames.

/WF Weight of first frame (default=1).

/WI Weight increment (default=0).

Composition Options

/L Separates output layers. Picture files specified in later layers get overlayed on top of previous layers. If the overlayed pictures contain transparency information (32-bit TGA files only), then the pictures underneath will show through..

/CK#,#,# Chroma-key, to make all occurrences of a color transparent for compositing. The three numbers represent the red,green, and blue components of the key color.

/CI#,#,# Inverse chroma-key.

/CT Chroma-key tolerance. Specifies how close to the key color a pixel has to be to make it transparent.

/CC#,#,#,# Chroma-key color -- specify an RGBa value to replace chroma-keyed pixels with (default=0, 0, 0, 0).

/RED, /GREEN, /BLUE Make a grayscale image from one of a pictures color channels.

/ALPHA Make a grayscale image from a picture's alpha channel.

Geometrical Transformation Options

/CL#,#[,#,#]	Clip the input picture. The first two numbers specify the top-left, and the other two numbers control the width and depth..
/CLP#,#[,#,#]	Clip from top-left (after scaling).
/LOC#,#[,#]	Specify where to place image (default is 0,0). The final number specifies time.
/LT	Make /LOC base vertical coords from the top edge instead of the center.
/LB	Make /LOC base vertical coords from the bottom edge.
/LL	Make /LOC base horizontal coords from the left edge instead of the center
/LR	Make /LOC base horizontal coords from the right edge.
/ROT#[,#]	Rotation -- first number represents angle, second is for time.
/SC[#,#]	Rescale pictures to screen resolution or specified size.
/SCF[#,#]	Fast (but dumb) rescale.

Other Options

/K#	Use only one frame per # picture files. Skip the rest.
/I[#]	Process total number of picture files.
/GA#	Gamma-correct picture. Try specifying a number between 1.0 and 2.0 to brighten, between 0.5 and 1.0 to darken.
/INV[R,G,B,A]	Invert colors or individual color channel.

Command Line Options

Scripts

You tell DTA to process all the files listed in a text file by specifying the name of the text file preceded by a "@" (you know, like "@pics.scr"). A script file can contain as many file specifications as you want. It can also contain any of the above-mentioned switches. Each must be on a line of its own.

Just make sure that you've got a new file name or switch on each line of the script file. You can even include the name of a new script file...and try to avoid recursive script files...

Don't put "@script1.scr" inside of SCRIPT1.SCR, and **don't** put "@script1.scr" inside of SCRIPT2.SCR if SCRIPT2.SCR is called by SCRIPT1.SCR. This will cause DTA to go into an infinite loop and hang up!

Here's an example script called test.scr:

```
back.tga
/l
*.tga
/a2
/sc640,480
/l
signat.gif
/lr
/lb
/fo2
```

Typing "dta @test" in this case would cause DTA to perform exactly the same as if you instead typed:

```
dta back.gif /l *.tga /a2 /sc640,480 /l signat.gif /lr /lb /fo2
```

Scripts can come in handy if:

- your command is just getting too darn long and complicated to fit on a DOS command line. A script can be as long as you want.

- you have to use the same command over and over again. Why remember all that stuff or use a crib sheet when you can just put it all in a script just once and then use it over and over.

  You can also use a script in addition to other filenames and switches. Say you had a script file called **whirl.scr** that looked like this:

  ```
  /l
  logo.gif
  /rot0,0
  /rot360,100
  /ch0,0,0
  /loc400,350
  ```

  then instead of typing

  ```
  dta x*.tga /l logo.gif /rot0,0 /rot360,100 /ch0,0,0 /loc400,350
  ```

  you could just type:

  ```
  dta x*.tga @whirl
  ```

Speed Concerns

DTA is fastest when you use grayscale instead of a palette.

If you let DTA generate its own palette, DTA is still relatively speedy. It gets **a lot** slower if you dither.

If you create an animation from .TGA files stored in an archive file (.LZH, .ZIP, .ARJ), then DTA is REAL slow, because it takes time to shell out to the de-archiving program. ZIP files compress the least, but PKUNZIP is the fastest dearchiver. ARJ files compress really well, but the dearchiver is the slowest. LZH files compress well, but not as well as ARJ, and LHA's speed is somewhere between ARJ and PKUNZIP.

If you average multiple files or rescale pictures, that is going to slow things down.

Credits

To create palettes and select colors from palettes, DTA uses an algorithm that I found in an article called "A Simple Method for Color Quantization: Octree Quantization," by Michael Gervautz and Werner Purgathofer, which can be found in a book called "Graphics Gems," edited by Andrew S. Glassner. I recommend this book highly.

The .FLI file format was explained in a document called FLIDOC.TXT from Jim Kent's FLILIB. I wasn't able to use this C library in DTA, since I do my coding in Pascal, but the document contained all the info I needed. (Jim Kent is also the guy who, with Tom Hudson, wrote TGAFLI, and who developed Animator and Animator Pro.)

Special Thanks To:

- Borland International and TurboPower Software for creating some great programming tools.
- Jim Kent for his info about 640x480 .FLIs on BIX.
- to Dan Farmer, Alexander Enzmann, Jeff Bowermaster, and others for their excellent suggestions.
- Dan Farmer for sprucing up the ASCII documentation.
- Rob McGregor for producing and editing this Help File.
- Tim Wegner for talking me into finally adding GIF89a input.
- John Jordan for suggesting alpha insertion.
- The Cafe Pamplona in Harvard Square for being such a cool place to swill coffee and discuss ray-tracing and animation.

How To...

Close

Create an 8-bit (256 color) Flic

Create an 8-bit (256-color) flic file (.FLI or .FLC) from a bunch of .TGA files.

```
DTA ROCKET*.TGA
```

Close

Create a 16-bit Flic

Create a 16-bit flic file (.FLH) from a bunch of .TGA files.

```
DTA ROCKET*.TGA /B16
```

Close

Create a 24-bit Flic

Create a 24-bit flic file (.FLT) from a bunch of .TGA files.

```
DTA ROCKET*.TGA /B24
```

Close

Make an Animator Palette File

Make a palette for all .TGA files starting with "ROCKET". Output the palette to an Animator palette file called "ROCKET.COL".

```
DTA ROCKET*.TGA /FC /OROCKET
```

Close

Build a Multilayer Flic

Build a multilayer flic. SKY.GIF gets put in the background, and the pictures FIRST.TGA, a bunch of TGAs beginning with the string MORF, and LAST.TGA get superimposed over it. SKY.GIF shows through any parts of the other pictures which are transparent (32-bit .TGA files can contain a transparency value for each pixel called an alpha channel).

This example assumes that the pictures BEFORE.TGA and AFTER.TGA had transparency added to them with my other program, DMorf (Rel. 1.1 or higher), and that the MORF*.TGA files are morphed pictures built with DMorf.

```
DTA SKY.GIF /L BEFORE.TGA MORF*.TGA AFTER.TGA
```

Close

Make a PICLAB Palette File

Make a palette for .TGA files "PIC1.TGA", "PIC2.TGA", and "PIC3.TGA" Output the palette to a PICLAB palette file called "XYZ.MAP".

```
DTA PIC1 PIC2 PIC3 /FM /OXYZ
```

Close

Create a Dithered Animation

Make input from all .TGA files beginning with "ROCKET". Output to Make an animation file called "ROCKET.FLI". Dither the frames with Floyd-Steinberg error-diffusion.

```
DTA ROCKET* /OROCKET /DF
```

Close

Create a Dithered Gray Scale Animation

Make input from all .TGA files beginning with "ROCKET". Map colors to grayscale. Dither, using ordered dither strength 2. Ping-pong it. Output to an animation with the default name of "ANIM.FLI".

```
DTA ROCKET* /G /DO2 /P
```

Close

Create a Default Animation With .TGA

Make input from ROCKET.TGA. Output to an animation with the default name of "ANIM.FLI".

```
DTA ROCKET
```

Close

Create an Animation From a Text File List

Make input from files listed in a text file called "WING2.LST". Set playback speed to "10". Output to an animation with the name of "WING2.FLI".

```
DTA @WING2.LST /S10 /OWING2.FLI
```

Close

Use Wildcards For Animation Files

Make input from all .TGA files whose name begins with "BBB", and "AAA". Read in an Animator palette file called "PAL1.COL", and merge it with a Piclab palette file called "PAL2.MAP". Use this combined palette for the animation. Output to an animation with the name of "FRED.FLI".

```
DTA BBB*.TGA AAA*.TGA /UPAL1.COL /UPAL2.MAP /OFRED
```

NOTE: The "BBB" files will appear in the animation before the "AAA" files. The "BBB" files will be sorted alphabetically, and so will the "AAA" files, but separately.

Convert .TGA To .GIF

Make .GIFs from all .TGAs in the current directory.

```
DTA *.TGA /FG
```

Convert .TGA to Dithered Grayscale .GIF

Make a dithered grayscale .GIF from a file called SOMBRERO.TGA.

```
DTA SOMBRERO /FG /DR2 /G
```

Create a 3D Animation

Make a red/blue 3D .FLI file (the kind that requires funny glasses). Use the .TGA files beginning with "LEFT" as the red component and the ones beginning with "RIGHT" as the blue component. Output to an animation with the default filename of "ANIM.FLI"

```
dta /chr left*.tga /ga0.8 /chb right*.tga
```

The **/ga** is to adjust the brightness of the red component so it doesn't drown out the blues. Some experimentation may be necessary to get the balance just right... if somebody comes up with an ideal brightness combination, let me know and I'll put it in the doc.

NOTE: For this to work right, there must be an equal number of "LEFT" and "RIGHT" .TGAs.

Create an Animation With Three Sets of Pictures.

This is a fun one...

The ones called XXX*.TGA are the background, and the YYY*.TGA pictures are the foreground. But, the ZZZ*.TGA pictures are used for the alpha channel of the foreground, dictating which parts of the YYY pictures will be transparent and how much (dark parts of ZZZ will make YYY very transparent, while bright parts will be less so.

```
DTA XXX*.TGA /L YYY*.TGA /CHA ZZZ*.TGA
```

Average Multiple Files

Make input from .TGAs beginning with "XXX". Average all files. Output to a new Targa file called "YY.TGA".

```
DTA XXX* /FTYY /AA
```

Averaging 5 input pictures with "/A2" option:

Average:	**To produce:**
file 1, file 2	frame 1
file 3, file 4	frame 2
file 5, file 1	frame 3

Use Trailing Mode

Make input from .TGAs beginning with "XXX". Trail all files. Output to a new Targa file called "YY.TGA".

```
DTA XXX* /FTYY /TA
```

Trailing 4 input pictures with "/T2" option:

Average:	To produce:
file 1, file 2	frame 1
file 2, file 3	frame 2
file 3, file 4	frame 3
file 4, file 1	frame 4

Trailing 5 input pictures with "/T3" option:

Average:	To produce:
file 1, file 2, file 3	frame 1
file 2, file 3, file 4	frame 2
file 3, file 4, file 1	frame 3
file 4, file 5, file 1	frame 4
file 5, file 1, file 2	frame 5

Create an Animation With Ordered Dithering

Make input from all .TGA files beginning with "XXX". Trail 3 frames. Use ordered dithering, strength 3. Output an animation called "ZZZ.FLI"

```
DTA XXX*.TGA /T3 /DO3 /OZZZ
```

Expand the Length of an Animation

Make input from all .TGA files beginning with "XXX". Expand 3. Assuming 5 .TGA input files, this would create a 15 frame animation. Output an animation called "ANIM.FLI"

```
DTA XXX*.TGA /X3
```

Expanding 2 input pictures with "/X1" option:

```
frame 1 = (100% of FILE 1 ) + (  0% of FILE 2 )
frame 2 = ( 50% of FILE 1 ) + ( 50% of FILE 2 )
frame 3 = (  0% of FILE 1 ) + (100% of FILE 2 )
```

Expanding 2 input pictures with /X2" option:

```
frame 1 = (100% of FILE 1 ) + (  0% of FILE 2 )
frame 2 = ( 66% of FILE 1 ) + ( 33% of FILE 2 )
frame 3 = ( 33% of FILE 1 ) + ( 66% of FILE 2 )
frame 4 = (  0% of FILE 1 ) + (100% of FILE 2 )
```

Create an Animation With .GIFs

Creates a file called ANIM.FLI or ANIM.FLC from all .GIF files in the current directory. The extension depends on the resolution of the first picture. If its 320x200, then DTA will create a .FLI file. Otherwise, itll create a .FLC file.

```
DTA *.GIF
```

Limit Frames

Assuming 60 .TGA files, create a flic containing only 30 frames. Generate the palette from only 15 frames.

```
DTA *.TGA /K2 /C2
```

Create and Rescale an Animation

Create a flic from a bunch of .GIF files, set the screen size to 320 by 200, and rescale all the images to the screen size.

```
DTA *.GIF /R320,200 /SC
```

Create a 640x480 .FLC From 320x200 Images

Create a 640x480 .FLC file from a bunch of .TGA files. Rescale each picture.

This one is useful when you want to turn a bunch of 320x200 pictures into a 640x480 flic because 320x200 screens use a different aspect ratio than 640x480.

320x200 pictures look a bit squat when you display them in 640x480 mode, but 320x240 looks correct.

```
DTA *.TGA /R6 /SD320,240
```

Append Two Animation Files

Create a new flic by appending two existing flics. Unlike my old FLAP program, DTA will create a new palette.

```
DTA FLIC1.FLI FLIC2.FLI /OFLIC3
```

Create .GIFs From .FLI Frames

Create GIF files from the frames in FLIC.FLI. The **/NM** is required to tell DTA ***not*** to generate an optimized palette. That would be a waste of time, since a FLIC file usually contains an optimized palette.

```
DTA FLIC.FLI /FG /NM
```

Rescale and Rotate a Flic 90 Degrees

Use "`/ROT90`", "`/ROT180`", "`/ROT-90`", "`/ROT270`" etc.

Read an existing flic, scale each of the frames to a 120x100 square, rotate the result 90 degrees to the right, and create a new flic. The **/R1** tells DTA to make a 320x200 .FLI file instead of a 120x100 .FLC file.

The little 120x100 square gets placed in the middle of the screen.

```
DTA FLIC.FLI /SC120,100 /ROT90 /ONEWFLIC /R1 /NM
```

This is the order that operations get done:

1) Rotation
2) Scaling
3) Clipping

So if you did "`/cl100,100 /sc200,200 /rot90`" then DTA will first rotate the picture 90 degrees to the right, then scale it to 200x200, and then clip off the top half and left half of the scaled, rotated image.

Rescale and Rotate a Flic Incrementally

This one is much like **Rescale and Rotate a Flic 90 Degrees**, but a bit funkier. The first frame (0%) will be rotated 0 degrees (not at all). The last frame (100%) will be rotated 360 degrees (a complete rotation). The frames in between will be rotated in imcrements between 0 and 360 degrees. (If you had 362 frames in the animation, then the second frame would be rotated 1 degree, the second 2 degrees, etc.)

Note we did ***not*** use the **/NM** switch to turn off palette optimization. Scaling and rotation *require* that the palette be remapped because the processes create new colors. On the other hand, fast scaling (/SCF) and 90-degree rotations do not require an optimized palette.

```
DTA FLIC.FLI /SC100,100 /ROT0,0 /ROT360,100 /ONEWFLIC /R1
```

Rescale and Rotate a Flic Incrementally With Acceleration

Not quite the same as **Rescale and Rotate a Flic Incrementally**. Instead of rotating the frames at a constant rate the rotation starts slowly, until at two-thirds (66.666%) of the way through the images have rotated 180 degrees. The rotation speeds up so that by the time it finishes (hits 100%) it's completed a full rotation.

```
DTA FLIC.FLI /SC100,100 /ROT0,0 /ROT180,66.666 /ROT360,100 /ONF /R1
```

Read Frames From a .ZIP File

Read frame 20 of a flic that's stored in a ZIP file. Clip a 90x80-pixel square from location 100,120 of the image and save it as a TGA file.

```
DTA BACKUP.ZIP:FLIC.FLI:20 /CL110,120,90,80 /FT
```

Contacting Dave...

If you've got any requests/bug reports/suggestions, send a message to **David Mason** on one of the following:

- The **"You Can Call Me Ray"** BBS, (708) 358-5611
- **"The Graphics Alternative"**, (510) 524-2780, and on **"Channel 1"** BBS, (617) 354-8873.
- **CompuServe ID: 76546,1321**
- **From the Internet: 76546.1321@compuserve.com**

Or via snailmail to:

David K. Mason
P.O. Box 181015
Boston, MA 02118

You'll probably get some kind of a response (maybe sooner, maybe later)...

What's New in DTA...

Rel 2.1.0 (04/23/94)

Changes:

- Fixed flic-reading code so it reads the bottom row of pixels properly.

- Rewrote a lot of the internal picture-file handling code. All pictures are now read into memory buffers, and all operations are performed on the memory buffers in sequence (formerly, DTA read most pictures in a line at a time, and did all the operations on each line of pixels...it kept lots of picture files open simultaneously).

  The side effects of these changes are:

  1) Basic operations use more memory.

  2) Real complicated operations (averaging, compositing, etc.) use less memory.

  3) Most everything works at least a little bit faster, and some operations are a lot faster.

  4) There's no longer any limit on the number of pictures that you can average. (DTA used to fail if you tried averaging more than 10 or 12 pictures.)

  5) I was able to reshuffle the order of operations. Now, the order is: chroma-key, averaging, clipping, scaling, post-scale clipping, rotation, compositing. (Now that I've gotten this far, I've realized that this still stinks...what's really needed is a user-defined order.)

  6) Made it possible to add arbitrary rotations.

  7) Processing a frame takes more steps, so the "percent finished" information isn't as useful as it used to be.

- Had trouble reading some TGA files produced with RMORF 0.4 because of incorrect header information. DTA will now assume that a TGA file is 320x200 resolution if the image size fields are left blank.

- Added built-in support for ZIP and LZH files, thanks to TurboPower Software's incredible programming toolkits. No more shelling to PKUNZIP or LHA required. As a result, extraction from archives is a lot faster, and there's no need to put up with LHA's weird screen access anymore. Unfortunately, because of this change I had to drop support for the ARJ archiver.

- When building 8-bit flics, DTA used to always use the 320x200 resolution some another resolution was selected using the /R switch. The reason for that was that 320x200 is the only supported resolution for the original Autodesk Animator .FLI format. From now on, it'll use the actual resolution from the first input picture file, like the now-obsolute /RA switch used to do. So unless your input pictures are already 320x200, you'll get .FLC files instead of .FLI files. You can make DTA create a .FLI file even with other resolution input by specifying /R1 or /R320,200. You'll also need to use /R if you want to create larger images for compositing...

- When DTA makes a 320x200 flic, it always creates a .FLI file instead of a .FLC file. Unless you use the /FLC switch.

- Reworked grayscale options...now there's **/G**, **/G128**, **/G64**, and **/G32**, **/G16**, **/G8**, and **/G4**.

/G represents a 256-level grayscale except when the output format is a 320x200 .FLI file, when it represents a 64-level grayscale. That's because .FLI won't support higher than 64 shades.

/G128 represents a 128-level grayscale, **/G64** a 64-level grayscale, etc. The lower numbers will get you a lower-quality image, which will almost always compress better. Most VGA and SVGA monitors can only display 64 shades of gray, so it's wasteful to use more than that if your picture will only be displayed on a monitor. Some laptop VGA displays can only display 32 shades. If you're going to be printing your image or using it as a bump map or height field with a rendering program, then the more shades the better.

- Renamed /CH (chroma-key) to /CK so that /CH would be free for a new option.

- Got rid of /3D... I pulled it out to make some of the rewrite easier. You can now get a similar 3d effect using /CHR and /CHB (see below).

- Added /BU, which makes DTA save .TGA files bottom-up (or upside-down, if you prefer) instead of top-down, as some applications don't understand top-down .TGAs.

- Added /IG, which maked DTA save .GIF files in interlaced format. It can also read interlaced .GIFs now.

- When you create an 8-bit TGA file, DTA will now produce a colormapped TGA, not a greyscale TGA. If you want a greyscale TGA, use /G instead of /B8. DTA can now read colormapped TGA files, too.

- Added **/CH[R,G,B,A]** which stand for **CH**annel insertion, which should let you create some truly weird effects if you're so inclined. This is a pretty hard concept to explain, so bear with me...and remember that you don't have to use it, just like you don't have to use **/L**... DTA still understands simple command lines like:

```
"dta *.tga"
```

You can specify /CHR (red channel), /CHG (green channel), /CHB (blue channel), and /CHA (alpha channel) to cause a gray-scale version of an image file to be inserted into a specific channel of the current layer. You can use this to specify a matte for an image that doesn't contain an alpha channel, like this:

```
dta background.tga /l pics*.tga /cha masks*.tga
```

or to get a red/blue 3d effect (the kind that requires funny glasses):

```
dta /chr left*.tga /ga0.8 /chb right*.tga
```

The **/ga** is to adjust the brightness of the red component so it doesn't drown out the blues. Some experimentation may be necessary to get the balance just right... if somebody comes up with an ideal brightness combination, let me know and I'll put it in the doc.

/CH works much like **/L**, except **/CH** is subordinate to **/L**.

/CH merging (insertion) takes place after picture averaging, gamma correction, and chroma-keying, but before clipping, rotation, scaling, etc. **/L** merging (compositing) takes place after all of the above. So, you can have separate chroma-key, gamma, averaging settings for each **/CH** layer, but only one **/ROT**, **/LOC**, etc. setting in each **/L** layer regardless of how many **/CH** layers you've got. And some other switches, like **/F**, **/O**, **/R**, etc. can only be specified once per command line, no matter how many **/L** and **/CH** layers you've got.

Note that if you specify "`pics*.tga /cha masks*.tga`", you're really specifying two **/CH** levels, not just one. The first, without a **/CH** parameter, contains `pics*.tga`, and there's no

special processing of the color channels. The second gets converted to a grayscale and is inserted into the alpha channel. (Thanks to John Jordan for suggesting alpha insertion in the first place...)

- When converting from one file format to another, DTA will usually use the original picture's filename, but with a new extension (unless you specify a different filename with /O). But, if you're creating an output picture from multiple input layers, or multiple input channels, or averaged images, or if input file and output file are in the same format, then DTA will invent new output filenames, like "OUT00000".

  You can now force DTA to use the original filename even in those situations if you use the new /FO (force original filename) switch. DTA might still not pick the right filename when you're using multiple layers, because it'll always use the first filename in the first channel group in the first layer.

  If, for example, you're overlaying a signature or log over a bunch of input pictures, like this:

  ```
  DTA PICS*.TGA /L SIGN.TGA /FG /FO
  ```

  it'll be okay because each .GIF file will use the filename from the first layer (PICS*.TGA). But if you're overlaying a bunch of pictures onto a single background, like this:

  ```
  DTA BACK.TGA /L PICS*.TGA /FG /FO
  ```

  it won't be okay, because each output picture will get saved as BACK.GIF. In that circumstance tell DTA to use the filenames from the second layer, like this:

  ```
  DTA BACK.TGA /L PICS*.TGA /FG /FO2
  ```

  You can also specify a specific channel group, like this:

  ```
  DTA /CHR X*.TGA /CHG Y*.TGA /CHB Z*.TGA /FG /FO1,2
  ```

  The **1** means the first layer...you have to specify a layer when you're specifying a channel, even if there's only one. The **2** means the second channel group in the command, green in this case. You can use **/FO** to force DTA to replace your original files after processing, like this:

  ```
  DTA *.GIF /ROT90 /FG /FO
  ```

- Added read/write support for MindImage .RLE files. (These are the .RLE files used for the random dot stereogram program *MindImage*...DTA can ***NOT*** read any of the many other file formats that also use the extention .RLE)

  You read one of these files just like you would any other, by using the filename in the command line. For example, to convert a RLE to a GIF file, type:

  ```
  DTA HUMAN.RLE /FG
  ```

  To write a RLE, use the new **/FR** switch. Be warned that MindImage expects pictures' dimensions to be powers of two with a maximum of 512. So, DTA will always force pictures to have horizontal and vertical dimensions of 64, 128, 256, or 512. If a dimension is larger than 512, then it'll be chopped down to 512. Otherwise, unless a dimension is exactly equal to one of the allowed dimensions, it'll be increased to match. If the dimensions that are selected for you are different from the original dimensions, DTA will crop or extend the picture.

  For example, if you start with a 640x480 picture, DTA will chop off a little from the left and right to pare the 640 down to 512, and add enough black border to the top and bottom to extend the 480 to 512.

Make DTA rescale the image to the new dimensions with the **/SC** switch. In the 640x480 example:

```
DTA 640.GIF /FR /SC
```

will cause the 640x480 image to be rescaled to exactly 512x512.

Using:

```
DTA 640.GIF /FR /R128,128
```

will result in a 128x128 image chopped from the middle of the starting image. The next two commands:

```
DTA 640.GIF /FR /SC128,128
```

and

```
DTA 640.GIF /FR /R128,128 /SC
```

both result in a 128x128 image with the whole rescaled to fit exactly.

- Added read-only support for IFF/ILBM files, 24-bit variety only. Files with extensions of .IFF and .LBM will be recognized.

- Changed the default 16-bit flic format from .FLX to .FLH. .FLX is still supported to make life easier for users of Mathematica's Tempra Turbo Animator (the program that first supported that format). Just use the /FLX switch to tell DTA to use .FLX instead of .FLH. The basic (only?) difference between .FLX and .FLH is that they compress the first frame differently. .FLX uses a byte-oriented scheme which does not compress 16-bit pictures very well. .FLH uses a word-oriented scheme which works much better.

NOTE: My DFV flic player can read either. Tempra Turbo Animator will not be able to read the .FLH files.

- Added support for .FLT (24-bit) flics. Just use the **/B24** switch to tell it to build a .FLT in stead of .FLC. At the moment, my DFV player is the only player that can can view this variation of the flic format...if anybody out there wants some info on the modifications I made to the flic format to support 24-bit, let me know. They're not too complicated, and they're not a secret.

- Because 24-bit flics can get so ***huge***, I added some tricks for minimizing the size. First, the **/TO** (flic tolerance) switch, which causes DTA to ignore pixel differences between frames when the distance in color-space is less than or equal to a value that you supply. I know that explanation leaves something to be desired -- to understand what's going on, try it out by adding "`/TO15`" at the end of a command line. The result will usually be a smaller flic that plays more quickly and doesn't look quite as good. This trick works better with digitized video than it does with rendered animation.

  Sometimes this gimmick produces a side effect of faint trails on the screen when you animate anti-aliased images. Try the **/TX** switch (which isn't really an acronym of anything) with a number, for example "`/TX5`". In that case, one frame in every 5 will be a full difference frame with no tolerance, and the trails will disappear.

/TO and /TX work only with 24-bit flics. I'll probably add it later for 8-bit and 16-bit flics.

- Added **/INV** which inverts either whole colors or individual color channels. You can do **/INVR**, **/INVG**, **/INVB**, **/INVA**, or combinations of them. Specifying just "/INV" is the same as specifying "`/INVR /INVG /INVB`". If you use any **/INV** option with **/NM** (no remapping of mapped files) then color indexes will be inverted instead of color components.

- Added **/RED**, **/GREEN**, **/BLUE**, and **/ALPHA** switches which let you extract a particular channel from

an image. So, you can:

```
DTA PICTURE /RED
```

to create a grayscale image from just the red component, or:

```
DTA X*.TGA /CHR X*.TGA /BLUE /CHB Y*.TGA /RED
```

to first load in each picture as a regular 32-bit image, then replace the red channel with the original blue channel and then replace the blue channel with the red channel from a completely different set of pictures. This is fairly useless most of the time, but it should make some really wild effect possible.

- Added **/SCF** (fast scaling). Use it like **/SC**. It works a ***lot*** faster than **/SC**, but looks really bad, because it doesn't use any interpolation.

- Added **/CLP** (post-scale clipping) which works just like **/CL** but it clips the picture after scaling takes place. If you type:

```
dta x.tga /sc500,500 /cl120,120,100,100 /pcl140,140,150,150
```

DTA will read in the picture and clip out a 100x100 window from the picture (at coordinates 120,120), then rescale the clipped window to 500x500, and then clip out a 150x150 window from the scaled picture (at coordinates 140,140).

- Fixed a bug: DTA would ignore **/332** and **/G** palette options if you were outputting to a GIF file.

- Got rid of **/ST** (start) parameter, replaced it with **/LOC**. Syntax is now `/LOC#,#,#` `(default=0,0,0).` The first two numbers are the x,y coords on the screen (from the center) where the center of the image should go.... Unless you specify **/LL** (which tells DTA to base the location on the left edge of the screen and image), **/LR** (right edge), **/LT** (top edge), or **/LB** (bottom edge). The third number is a time variable, specified as a percentage.

  If you specify `/LOC0,0,0` that means put the image at 0,0 at the beginning of the animation. Add `/LOC25,25,50` and the image will be put at coords 25,25 at exactly halfway through the animation. If you don't specify a `#,#,100`, then DTA will assume you want the final location to loop back to the starting point. You can enter as many or few **/LOC** parameters per layer as you want... DTA will use an interpolating spline to figure out the actual locations per frame. Unlike the old **/ST**, pictures can be placed at non-integer locations. If you do, your picture will be resampled.

- Added arbitrary rotation. So you can specify `/rot45` (or `/rot110.25` for that matter). You can add time values after multiple /ROT switches just like you can after **/LOC**. To get a full rotation, you could do `"/ROT0,0 /ROT360,100"`. If rot[100%]-rot[0] is a multiple of 360, then the final rotation value will not be used ... this helps produce a smooth loop. Rotation uses smooth resampling.

- Input pictures that are larger than the output picture are now centered... instead of only the top corner showing, the center will show.

- Added read-only support for Type-1 TGA files (8-bit, colormapped).

- Added **/CC** (chroma-key color), which allows you to specify what color DTA should replace matching pixels with. The default is still 0,0,0,0 (black, fully transparent).

- Averaging and scaling now handle 32-bit pixels correctly ... previously they didn't take the alpha byte into account properly. As a result, colors (in areas where alpha-blending was taking place) would sometimes come out too dark or too light. As a result, these two functions are slower than they used to

be... but I think the other speedups will cancel it out.

- Made DTA write 8- and 24-bit PCX files (it did a couple years ago, but only 8-bit and then I pulled it out to save room. Now that DTA's protected mode, there's no reason not to support this output format).

- Made DTA read and write Vistapro .VAN animation files.

- DTA can now write as well as read BMP files, with **/FB** switch. **/FD** switch creates the same type of file, but uses the .DIB extension instead of .BMP By default, it creates 24-bit BMPs, but you can make it create 8-bit BMPs by adding the **/B8** switch.

- Fixed a bug in flic-reading... if a flic contained a partial palette (fairly rare), DTA would get confused and crash.