

TURBO GRAPHIX r.59 MEL COMMAND LISTING:

Turbo Graphix is Copyright 7thGate Software. LLC 2006-2008

This is a straightforward listing of MEL commands added by the Turbo Graphix and Turbo Graphix User Interface plug-ins.

(C): means "create" mode, no extra flag is needed to specify create mode.

(E): means "edit" mode, you must specify "-e" or "-edit" to enter edit mode.

(Q): means "query" mode, you must specify "-q" or "-query" to enter query mode.

(M): means "multi" capable, a flag that is multi-capable can be used multiple times in one statement.

If you find any inconsistencies or errors with this document please email: feedback@7thgatesoft.com

Turbo Graphix.mll :

Command to add/modify/remove image files that are processed in a list :

```
TG_ImageList : flags supported : [flags can support (Q)(E)(C)(M) modes ]
// adds an image to the list (will create a unique layer name)
-a -addImage [(C)(M)] params : {string "imagepath"} returns : none
// splits and adds a layer from a layered image to the list (PSD file)
// the separated layer is stored in the waste gate(temp folder) as a PNG file
-apl -addPsdLayer [(C)] params : {string "sourcepsdpath" string "layername"} returns : none
// records the source of your PSD image data. An empty string "" unless an image was
// extracted from a PSD file. Then it takes the format "psd/path.psd|layername|X|Y" OR separated
// the X,Y at the end of the string specifies the original location of the layer relative to PSD
-psl -psdSourceLayer [(E)(Q)] params : {int "layerindex" string "psd/path.psd|layername|X|Y"}
// returns : string[] "psd/path.psd|layername|X|Y"
// used to discover/change the name of a specific layer (layer names must be unique)
-ln -layerName [(E)(Q)] params : {int "layerindex" string "newname"} returns : string[] "lyrstrings"
// finds the file names of all images
-in -imageNames [(Q)] params : {none} returns : string[] "namesofimages.ext"
// finds the full file path of all images
-ip -imagePaths [(Q)] params : {none} returns : string[] "full/path/image.ext"
// removes a specific image by index
-r -remove [(C)(M)] params : {int "layerindex"} returns : none
// clears the image list of all entries
-ra -removeAll [(C)] params : { none } returns : none
// forces the list to only have 1 of any particular file name
-u -unique [(C)] params : { none } returns : none
// edits an images tiling on the x
-tw -tileWidth [(E)(Q)] params : {int "layerindex" int "newwidth"} returns : int[] "tilewidths"
// edits an images tiling on the y
-th -tileHeight [(E)(Q)] params : {int "layerindex" int "newheight"} returns : int[] "tileheights"
// edits an images tiling offset on the x
-tox -tileOffsetX [(E)(Q)] params : {int "layerindex" int "gridxoffset"} returns : int[] "tilexoffsets"
// edits an images tiling offset on the y
-toy -tileOffsetY [(E)(Q)] params : {int "layerindex" int "gridyoffset"} returns : int[] "tileyoffsets"
// edits an images x pixel offset
-iox -imageOffsetX [(E)(Q)] params : {int "layerindex" int "newxoffset"} returns : int[] "xoffsets"
// edits an images y pixel offset
-ioy -imageOffsetY [(E)(Q)] params : {int "layerindex" int "newyoffset"} returns : int[] "yoffsets"
// edits an layers x pixel center (pivot point)
-lcx -layerCenterX [(E)(Q)] params : {int "layerindex" int "newxorigin"} returns : int[] "xorigins"
// edits an layers y pixel center (pivot point)
-lcy -layerCenterY [(E)(Q)] params : {int "layerindex" int "newyorigin"} returns : int[] "yorigins"
// switches the position of two entries in the ImageList
-sw -swap [(C)] params : {int "layer_1" int "layer_2"} returns : none
// returns the size of the ImageList
-s -size [(Q)] params : {none} returns : int "listsize"
```

Command that will process the images added by TG_ImageList

TG_ProcessImages : flags supported : [flags can support (Q)(E)(C)(M) modes]
// You must provide the output dir for processing
-o -outputDirectory [(E)(Q)(C)] params : {string "outputdir"} returns : "outputdir"
// name of the build (used for naming iri files & textures created)
-b buildName [(E)(Q)(C)] params : {string "buildname"} returns : "current name"
// if this flag is set, a new directory will be generated inside "outputdir" with the name "buildname"
-n -newDirectory [(E)(Q)(C)] params : {bool "true/false"} returns : 1 if true 0 if false
// x pixel origin of ALL layers being built
-xo -xOrigin [(E)(Q)(C)] params : {int "newxorigin"} returns : "xorigin"
// y pixel origin of ALL layers being built
-yo -yOrigin [(E)(Q)(C)] params : {int "newyorigin"} returns : "yorigin"
// determines the maximum width of output textures (power of 2)
-mtw -maxTextureWidth [(E)(Q)(C)] params : {int "newwidth"} returns : "maxtexwidth"
// determines the maximum height of output textures (power of 2)
-mth -maxTextureHeight [(E)(Q)(C)] params : {int "newheight"} returns : "maxtexheight"
// the image format to use when outputting the processed RGBA data.
// supports : (.png, .bmp, .jpg, .tga, .tif, .dds)// revised with DeviL
-f -format [(E)(Q)(C)] params : {string ".newextension"} returns : ".extension"
// command will overwrite existing data without prompting for user confirmation
-r -replace [(C)] params : none returns : none
// flag will determine whether or not to compensate for pwr-of-2 to alleviate mipmapping artifacts
-eb -edgeBleed [(E)(Q)(C)] params : {bool "true/false"} returns : true/false
// If enabled, pattern data collected will remain connected during export to produce higher quality
-ki -keepIntact [(E)(Q)(C)] params : {bool "true/false"} returns : true/false
// If enabled, patterns that border each other will connect into larger patterns reducing poly counts
-rp -reducePatterns [(E)(Q)(C)] params : {bool "true/false"} returns : true/false

// Command that permanently modifies images held in the image list (Cannot be Undone)

TG_ModifyImage : flags supported : [flags can support (Q)(E)(C)(M) modes]
// will crop the specified image automatically by removing extra surrounding transparency
// Once the image is cropped, both position and tessellation will be properly offset to compensate
-ac -autoCrop [(C)] params : {uint "imagelistindex"} returns : uint[] newWidthHeight
// if this flag is set when -autoCrop is specified, instead of re-centering an images pivot
// location during a crop, they will be kept stationary in their exact pixel location
-kp -keepPivot [(C)] params : {none} returns : none
// allows you to determine or edit the width and height of an image in the image list.
// Both query and command modes are very cpu intensive. (command mode will perform bilinear scaling)
// Query returns ALL widths and heights in 1 interleaved array.(widthA[0]heightA[1]widthB[2]...)
-s -size [(Q)(C)] params : {uint "index" uint "new_width" uint "new_height"} returns : uint[] ImgWs+Hs
// Determines if an image was extracted from an external PSD. if so... will re-extract
// that image and overwrite the currently referenced file. if not... it does nothing
// the re-extracted image will be readjusted to remain stationary pixel for pixel
// this is based on the recorded X,Y offset in -psl.
// Re-extractions are not cropped by default, call -autoCrop to re-crop an extracted image
-ufs -updateFromSource [(C)] params : {uint "imagelistindex"} result : bool success

// Command that will scan your current selection of meshes and attempt to make them easier to deform

TG_T_EdgeSolver : flags supported : [flags can support (Q)(E)(C)(M) modes]
// Apply this command to eliminate T edges that are generated during image->mesh conversion
// T Edges allow for free floating vertexes that causes tearing when deforming mesh topology
// Due to the nature of the operation, multiple passes may be needed to eliminate all T edges
// flag specifies the minimum distance before new vertexes are merged. see ("polyMergeVertex")
-t -tolerance [(C)] params : {float "merge_tolerance"} returns : int num_solved
// determines whether or not to retain construction history when solving edges
// a setting of false is recommended, this will delete history and not pollute DAG with splitnodes
-h -history [(C)] params : {bool "keep"} returns : none

// Command that will convert an IRI file into a AFF file in memory

TG_LoadIRI : flags supported : [flags can support (Q)(E)(C)(M) modes]
// source IRI to be converted
-iri -imageRebuildInstructions [(C)] params : {string "fullpathofIRI"} returns : none
// determines how much to shrink the UV's to alleviate edge mipmapping artifacts
-to -texelOffset [(C)(E)(Q)] params : {double subpixels} returns : double subpixels
// determines if model geometry should be split by alpha distinctions by texture
// if not, a single mesh will be used using the minimum transparency setting per texture
-sm -singleMesh [(C)(E)(Q)] params : {bool "single"} returns : bool "single"
// detetmines the final 3d scale of the model by the ratio of 1 pixel to 1 3d unit
-pr -pixelRatio [(C)(E)(Q)] params : {double pixelratio} returns : double pixelratio

// Command will load an existing AFF file into memory

TG_LoadAFF : flags supported : [flags can support (Q)(E)(C)(M) modes]
// full path where to load the aff file ex: "C:/proj/hippo.aff"
-fp -fullPath [(C)] params : {string "fullpath"} returns : none
// you may provide the texture folder if textures are located separately
-tp -texturePath [(C)] params : {string "texturefolder"} returns : none

// Command actually generates textures & geometry in maya from the AFF currently in memory

// Will return the full DAG path to the parent transform of the newly created model in create mode [(C)]

TG_CreateModel : flags supported : [flags can support (Q)(E)(C)(M) modes]
// determines wether or not to support display layers
-dl -displayLayers [(C)(E)(Q)] params : {bool enable} returns : bool
// determines wether or not to support render layers
-rl -renderLayers [(C)(E)(Q)] params : {bool enable} returns : bool
// should we auto build a shading network for the new geometry? (recommended)
-sn -shadingNetwork [(C)(E)(Q)] params : {bool enable} returns : bool
// if we don't use custom hardware shaders, use this to avoid Z fighting
-ls -layerSpacing [(C)(E)(Q)] params : {double spacing} returns : double spacing
// if this flag is enabled, layers consisting of multiple meshes will
// be visually merged in Maya's DAG for easier editing of verts + faces.
// This complicates the DAG structure for developers needing to export.
// However it can allow the artist to be far more productive.
-af -artistFriendly [(C)(E)(Q)] params : {bool enable} returns : bool

// Command that will edit and query the state of a Turbo Graphix license

TG_LicenseManager : flags supported : [flags can support (Q)(E)(C)(M) modes]
// Use this to send in your license key you got from www.turbo-graphix.com
-l -license [(C)] params : {string "license" string "user/company"} returns : none
// This returns the state of your license (used for display purposes)
-v -valid [(Q)] params : {none} returns : bool valid
// returns the number of seats supported by this license (0 == Unlimited)
// The demo version is obviously unlimited, most registered versions support 1 seat
-s -seats [(Q)] params : {none} returns : int seats
// returns the registration ID of this license, -1 if not registered
-r -regID [(Q)] params : {none} returns : int registration#
// returns the name of the individual or company that registered the product
-n -name [(Q)] params : {none} returns : string user/company name
// how long has the user had the application installed? (in hours).(Demo+Registered)
-t -time [(Q)] params : {none} returns : int hours_installed
// what version are we using? r. stands for revision.
-vr -version [(Q)] params : {none} return : string revision_string

Turbo Graphix UI.mll :

// Command used augment the overloaded hardware renderer and MEL scripts used by Turbo Graphix's UI

```
TG_UserInterface : flags supported : [flags can support (Q)(E)(C)(M) modes ]
// used to find or change what items are selected
-s -select [(C)(M)(Q)] params : {int "index" boolean "select?"} returns : int[] "selected"
// used to refresh the visual display, attempts to keep the selection
// if reload is true, all files will be reloaded in case the files have changed
-r -refresh [(C)] params : {bool reload} returns : none
// used to auto place and edit the selected items.
// alignment and centering strings must be one of the following : "left" "top" "right" "bottom" "center"
// division level must be a value greater than 0 and less than 101,
// minpattern size must be greater than 15 and less than 129
-aa -autoAdjust [(C)] params : { string "Xalign" string "Yalign" string "Xcenter" string "Ycenter"
                                unsigned divisionlevel unsigned minpatternsizedivisionlevel } returns : none
// controls how transparent non-selected images are. valid range is (0.0-1.0)
-xr -xRay [(C)(Q)] params : {double transparency} returns : double transparency
// controls the level of zoom on the camera 10% - 1000%
-z -zoom [(C)(Q)] params : {double zoomlevel} returns : double zoomlevel
// Makes it so the UI will Focus the camera on selected Items
// if no items are selected it will recenter the camera
// if zoom is true, it will auto-adjust the zoom level to best fit the selection
-f -focus [(C)] params : {bool zoom} returns : none
// allows you to manually change the location (in pixel space) of the camera.
-cx -cameraX [(C)(Q)] params : {int x} returns : int x
-cy -cameraY [(C)(Q)] params : {int y} returns : int y
// frees up all the memory being used by the image data for display in the GUI
// this does not affect the base plugin in any way. (ie : your image list will be kept)
// use the refresh command if you wish to repopulate the GUI.
-fm -freeMemory [(C)] params : {none} returns : none
```