

Appendix

Appendix A: Tcl interface

The following section describes 3D-Equalizer's tcl interface. Tcl is a widespread, very powerful scripting language developed by John K. Ousterhout at University of California, Berkley.

The major application of 3D-Equalizer's tcl interface is for customizing the graphical user interface and for exchanging data with other software tools. A good source of further information about tcl can be found at <http://www.tcltk.com>. A few 3DE example scripts can be found in the directory "<3de install directory>/user_data/tcl_archive/".

Datatypes

<vector_2d> = [list <x> <y>]

<vector_3d> = [list <x> <y> <z>]

<matrix_3d> = [list [list <m00> <m01> <m02>]
[list <m10> <m11> <m12>]
[list <m20> <m21> <m22>]]

<curve_2d> = [list <vector_2d> <vector_2d> <vector_2d>...]

Command: get3DEInstallPath

Synopsis

<path> get3DEInstallPath

Parameters

<path> - an absolute filesystem path

Description

Executing this command causes 3D-Equalizer to return the path to its installation directory.

Example

```
# include "veclib.tcl"
set path [get3DEInstallPath]
source $path/user_data/tcl_archive/veclib.tcl
```

Command: get3DEVersion

Synopsis

<string> get3DEVersion

Parameters

<string> - text string

Description

Executing this command causes 3D-Equalizer to return its version.

Example

```
# print 3DE's version
set version_string [get3DEVersion]
open3DEConsole
print3DEConsole "Current version: " $version_string
```

Command: open3DEConsole

Synopsis

open3DEConsole

Description

Executing this command causes 3D-Equalizer to open the tcl console window.

Example

```
# open console and print error message
open3DEConsole
print3DEConsole "ERROR!!!\n"
```

See also

close3DEConsole, print3DEConsole, flush3DEConsole

Command: close3DEConsole

Synopsis

`close3DEConsole`

Description

Executing this command causes 3D-Equalizer to close the tcl console window.

Example

```
close3DEConsole
```

See also

`open3DEConsole`, `print3DEConsole`, `flush3DEConsole`

Command: print3DEConsole

Synopsis

```
print3DEConsole <string1> <string2>...
```

Parameters

<string> - Argument that is interpreted as a string

Description

Executing this command causes 3D-Equalizer to print out the supplied parameters in the tcl console window. The actual output will take place by the time the tcl script is finished or if the command “flush3DEConsole” is executed.

Example

```
set a 1
print3DEConsole "a = " $a "\n"
```

See also

close3DEConsole, open3DEConsole, flush3DEConsole

Command: flush3DEConsole

Synopsis

flush3DEConsole

Description

Executing this command causes 3D-Equalizer to flush the tcl console window output.

Example

```
# display the numbers 1..100
for {set i 1} {$i<=100} {set i [expr $i+1]} \
{
  print3DEConsole "i = " $i "\n"
  flush3DEConsole
}
```

See also

close3DEConsole, open3DEConsole, print3DEConsole

Command: post3DEInfoRequester

Synopsis

```
post3DEInfoRequester <message>
```

Parameters

<message> - text message string

Description

On execution, a requester window opens on the screen, displaying the specified text message. The execution of the TCL script stops until the user clicks on the Ok button.

Example

```
# post info requester  
post3DEInfoRequester "Selected pointgroup deleted!"
```

See also

post3DEQuestionRequester, post3DEPromptRequester, post3DEFileRequester

Command: `post3DEQuestionRequester`

Synopsis

```
<bool> post3DEQuestionRequester <message> <ok_label> <cancel_label>
```

Parameters

<code><bool></code>	- boolean value: 0, 1
<code><message></code>	- text message string
<code><ok_label></code>	- text string of Ok button
<code><cancel_label></code>	- text string of Cancel button

Description

On execution, a requester window opens on the screen, displaying the specified text message. The execution of the TCL script stops until the user clicks on a button. If the user clicks on the left button (Ok button), the command returns 1. Otherwise, 0 is returned.

Example

```
# post question requester asking to delete a pointgroup
set ok [post3DEQuestionRequester "Delete?" "Yes" "No"]
if {$ok}\
{
  set pg [getSelectedPGroup]
  deletePGroup $pg
}
```

See also

`post3DEInfoRequester`, `post3DEPromptRequester`, `post3DEFileRequester`

Command: `post3DEPromptRequester`

Synopsis

```
<string> post3DEPromptRequester <message>
```

Parameters

```
<string>          - text string  
<message>        - text message string
```

Description

On execution, a requester window opens on the screen, displaying the specified text message. Furthermore, the requester contains a textfield, an Ok button and a Cancel button. The user can enter some text into the textfield. The execution of the TCL script stops until the user clicks on a button. If the user clicks on the Ok button, the command returns the string that has been entered into the textfield. Otherwise, an empty string ("") is returned.

Example

```
# post prompt requester to ask for new pointgroup name  
set name [post3DEPromptRequester "Enter name:"]  
if {$name!=""}\  
{  
  set pg [getSelectedPGroup]  
  setPGroupName $pg $name  
}
```

See also

`post3DEInfoRequester`, `post3DEQuestionRequester`, `post3DEFileRequester`

Command: `post3DEFileRequester`

Synopsis

`<path> post3DEFileRequester <message> <filter_pattern>`

Parameters

`<path>` - text string containing an absolute path
`<message>` - text message string
`<filter_pattern>` - string that defines a file pattern

Description

On execution, a standard 3D-Equalizer file requester window opens on the screen. The execution of the TCL script stops until the user closes this file requester. If the user has specified a filename, the absolute path of this file is returned. If the user clicks on the Cancel button to close the requester, an empty string ("") is returned.

Example

```
# post file requester to ask for a file
set path [post3DEFileRequester "Save As..." "*"]
if {$path!=""}\
{
  open3DEConsole
  print3DEConsole "Saving data to file: " $name "...\\n"
  flush3DEConsole

  ...
}
```

See also

`post3DEInfoRequester`, `post3DEQuestionRequester`, `post3DEPromptRequester`

Command: getSelectedPGroup

Synopsis

<pgroup_id> getSelectedPGroup

Parameters

<pgroup_id> - identifier of a pointgroup object

Description

On execution this command returns the pointgroup id of the pointgroup object that is currently selected in the Pointgroups Window. If no pointgroup object is currently selected, a warning message is displayed in the tcl console window.

Example

```
set pg [getSelectedPGroup]
```

See also

deletePGroup, getNextPGroup, getPGroupName, getPGroupType

Command: getSelectedFobj

Synopsis

<fobj_id> getSelectedFobj

Parameters

<fobj_id> - identifier of a frame object

Description

On execution this command returns the frame object id of the sequence or reference frame object that is currently selected in the Frames Window. If no frame object is currently selected, a warning message is displayed in the tcl console window.

Example

```
set fobj [getSelectedFobj]
```

See also

deleteFobj, getNextFobj, getFobjName

Command: `getSelectedPoint`

Synopsis

```
<point_id> getSelectedPoint
```

Parameters

<point_id> - identifier of a point object

Description

On execution this command returns the point id of the point object that is currently selected in the Pointgroups Window. If no point object is currently selected, a warning message is displayed in the tcl console window.

This command is obsolete. Use *getPointSelectionFlag* instead!

Example

```
set point [getSelectedPoint]
```

See also

`deletePoint`, `getNextPoint`, `getPointName`, `getPointSelectionFlag`

Command: getSelectedCamera

Synopsis

<camera_id> getSelectedCamera

Parameters

<camera_id> - identifier of a camera object

Description

On execution this command returns the camera id of the camera object that is currently selected in the Camera Adjustment Window. If no camera object is currently selected, a warning message is displayed in the tcl console window.

Example

```
set camera [getSelectedCamera]
```

See also

getNextCamera, getCameraName

Command: getSelectedFrame

Synopsis

<frame> getSelectedFrame

Parameters

<frame> - frame number

Description

On execution this command returns the number of the currently selected frame specified with the Main Window's frame slider. If no frame object is currently selected, a warning message is displayed in the tcl console window.

Example

```
set frame [getSelectedFrame]
```

Command: createPGroup

Synopsis

```
<pgroup_id> createPGroup <pgroup_type>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<pgroup_type>	- constant that defines the pointgroup type: CAMERA, OBJECT, MOCAP

Description

On execution this command creates a new pointgroup object and returns its pointgroup id.

Example

```
# create a new object pointgroup  
set pg [createPGroup OBJECT]
```

See also

deletePGroup, getNextPGroup, getPGroupName

Command: deletePGroup

Synopsis

```
deletePGroup <pgroup_id>
```

Parameters

<pgroup_id> - identifier of a pointgroup object

Description

On execution this command deletes the specified pointgroup object.

Example

```
# delete the currently selected pointgroup
set pg [getSelectedPGroup]
deletePGroup $pg
```

See also

createPGroup, getNextPGroup, getPGroupName

Command: getPGroupType

Synopsis

<pgroup_type> getPGroupType <pgroup_id>

Parameters

<pgroup_type> - constant that defines the pointgroup type: CAMERA, OBJECT, MOCAP
<pgroup_id> - identifier of a pointgroup object

Description

On execution this command returns the type of the specified pointgroup object.

Example

```
# display type of the currently selected pointgroup
set pg [getSelectedPGroup]
set pg_type [getPGroupType $pg]
print3DEConsole "Pointgroup type: " $pg_type "\n"
```

See also

createPGroup, deletePGroup, getNextPGroup, getPGroupName

Command: getFirstPGroup

Synopsis

<pgroup_id> getFirstPGroup

Parameters

<pgroup_id> - identifier of a pointgroup object

Description

On execution this command returns the pointgroup id of the first pointgroup object. If there are no pointgroup objects available, the string "0" is returned.

Example

```
set pg [getFirstPGroup]
```

See also

createPGroup, deletePGroup, getNextPGroup, getPGroupName

Command: getNextPGroup

Synopsis

<pgroup_id> getNextPGroup <pgroup_id>

Parameters

<pgroup_id> - identifier of a pointgroup object

Description

On execution this command returns the pointgroup id of the pointgroup object that comes next after the specified pointgroup object. If the specified pointgroup object is the last one, the string "0" is returned.

Example

```
# display the names of all pointgroups
for {set pg [getFirstPGroup]} \
  {$pg != 0} \
  {set pg [getNextPGroup $pg]} \
  {
    set name [getPGroupName $pg]
    print3DEConsole "Pointgroup name: " $name "\n"
  }
```

See also

createPGroup, deletePGroup, getPGroupName

Command: getNoPGroups

Synopsis

<number_of_pgroups> getNoPGroups

Parameters

<number_of_pgroups> - integer that specifies the number of pointgroup objects

Description

On execution this command returns the number of available pointgroup objects.

Example

```
set no_pgroups [getNoPGroups]
```

See also

getIndexPGroup

Command: `getIndexPGroup`

Synopsis

`<pgroup_id> getIndexPGroup <pgroup_index>`

Parameters

`<pgroup_id>` - identifier of a pointgroup object
`<pgroup_index>` - integer that specifies a pointgroup object

Description

On execution this command returns the pointgroup id of a pointgroup object specified by the pointgroup index parameter. The first pointgroup object is defined by an index of "0".

Example

```
# display the names of all pointgroups
for {set index 0} \
    {$index < [getNoPGroups]} \
    {set index [expr $index+1]} \
    {
    set pg [getIndexPGroup $index]
    set name [getPGroupName $pg]
    print3DEConsole "Pointgroup name: " $name "\n"
    }
```

See also

`getNoPGroups`

Command: getPGroupName

Synopsis

<name> getPGroupName <pgroup_id>

Parameters

<name> - name string
<pgroup_id> - identifier of a pointgroup object

Description

On execution this command returns the name of the pointgroup object specified by the supplied pointgroup id.

Example

```
# display name of the currently selected pointgroup
set pg [getSelectedPGroup]
set name [getPGroupName $pg]
print3DEConsole "Pointgroup name: " $name "\n"
```

See also

setPGroupName

Command: setPGroupName

Synopsis

```
setPGroupName <pgroup_id> <name>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<name>	- name string

Description

On execution this command sets the name of the given pointgroup object to the supplied name string.

Example

```
# set name of the currently selected pointgroup to "wurst"  
set pg [getSelectedPGroup]  
setPGroupName $pg "wurst"
```

See also

getPGroupName

Command: createFobj

Synopsis

<fobj_id> createFobj <fobj_type>

Parameters

<fobj_id>	- identifier of a frame object
<fobj_type>	- constant that defines the frame object type: SEQUENCE, REF_FRAME

Description

On execution this command creates a new frame object and returns its id.

Example

```
# create a new sequence
set fobj [createFobj SEQUENCE]
```

See also

deleteFobj, getNextFobj, getFobjName

Command: deleteFobj

Synopsis

deleteFobj <fobj_id>

Parameters

<fobj_id> - identifier of a frame object

Description

On execution this command deletes the specified frame object.

Example

```
# delete the currently selected frame object
set fobj [getSelectedFobj]
deleteFobj $fobj
```

See also

createFobj, getNextFobj, getFobjName

Command: getFobjType**Synopsis**

```
<fobj_type> getFobjType <fobj_id>
```

Parameters

```
<fobj_type>      - constant that defines the frame object type:  
                  SEQUENCE, REF_FRAME  
<fobj_id>       - identifier of a pointgroup object
```

Description

On execution this command returns the type of the specified frame object.

Example

```
# display type of the currently selected frame object  
set $fobj [getSelectedFobj]  
set $fobj_type [getFobjType $fobj]  
if {$fobj_type=="SEQUENCE"} then \  
  {  
    print3DEConsole "This is an image sequence!\n"  
  } \  
else \  
  {  
    print3DEConsole "This is a reference frame!\n"  
  }  
}
```

See also

createFobj, deleteFobj, getNextFobj, getFobjName

Command: getFirstFobj

Synopsis

<fobj_id> getFirstFobj

Parameters

<fobj_id> - identifier of a frame object

Description

On execution this command returns the frame object id of the first frame object. If there are no frame objects available, the string "0" is returned.

Example

```
set fobj [getFirstFobj]
```

See also

createFobj, deleteFobj, getNextFobj, getFobjName

Command: getNextFobj**Synopsis**

```
<fobj_id> getNextFobj <fobj_id>
```

Parameters

<fobj_id> - identifier of a frame object

Description

On execution this command returns the frame object id of the frame object that comes next after the specified frame object. If the specified frame object is the last one, the string "0" is returned.

Example

```
# display the names of all frame objects
for {set fobj [getFirstFobj]} \
    {$fobj != 0} \
    {set fobj [getNextFobj $fobj]} \
    {
    set name [getFobjName $fobj]
    print3DEConsole "Frame object name: " $name "\n"
    }
```

See also

createFobj, deleteFobj, getFobjName

Command: getNoFobjs

Synopsis

<number_of_fobjs> getNoFobjs

Parameters

<number_of_fobjs> - integer that specifies the number of frame objects

Description

On execution this command returns the number of available frame objects.

Example

```
set no_fobjs [getNoFobjs]
```

See also

getIndexFobj

Command: getIndexFobj**Synopsis**

```
<fobj_id> getIndexFobj <fobj_index>
```

Parameters

```
<fobj_id>          - identifier of a frame object  
<fobj_index>      - integer that specifies a frame object
```

Description

On execution this command returns the frame object id of a frame object specified by the frame object index parameter. The first frame object is defined by an index of "0".

Example

```
# display the names of all frame objects  
for {set index 0} \  
    {$index < [getNoFobjs]} \  
    {set index [expr $index+1]} \  
    {  
    set fobj [getIndexFobj $index]  
    set name [getFobjName $fobj]  
    print3DEConsole "Frame object name: " $name "\n"  
    }
```

See also

getNoFobjs

Command: getFobjName

Synopsis

<name> getFobjName <fobj_id>

Parameters

<name> - name string
<fobj_id> - identifier of a frame object

Description

On execution this command returns the name of the frame object specified by the supplied frame object id.

Example

```
# display name of the currently selected frame object
set fobj [getSelectedFobj]
set name [getFobjName $fobj]
print3DEConsole "Frame object name: " $name "\n"
```

See also

setFobjName

Command: setFobjName

Synopsis

```
setFobjName <fobj_id> <name>
```

Parameters

```
<fobj_id> - identifier of a frame object  
<name> - name string
```

Description

On execution this command sets the name of the given frame object to the supplied name string.

Example

```
# set name of the currently selected frame object to "willi"  
set fobj [getSelectedFobj]  
setFobjName $fobj "willi"
```

See also

```
getFobjName
```

Command: getFobjNoFrames

Synopsis

```
<no_frames> getFobjNoFrames <fobj_id>
```

Parameters

```
<no_frames>      - integer value  
<fobj_id>       - identifier of a frame object
```

Description

On execution this command returns the number of frames of the frame object specified by the supplied frame object id. If the frame object is a reference frame then “1” is returned.

Example

```
# display number of frames of the  
# currently selected frame object  
set fobj [getSelectedFobj]  
set no_frames [getFobjNoFrames $fobj]  
print3DEConsole "Number of frames: " $no_frames "\n"
```

See also

setFobjSequenceAttr, getFobjSequenceAttr

Command: getFobjPath**Synopsis**

```
<path> getFobjPath <fobj_id>
```

Parameters

```
<path>      - absolute filename path starting from "/"  
<fobj_id>   - identifier of a frame object
```

Description

On execution this command returns the absolute image filename path of the frame object specified by the supplied frame object id. All indexing numbers in the path are replaced by '#' characters, similar to the "Sequence Settings" requester.

Example

```
# display the path of the currently selected frame object  
set fobj [getSelectedFobj]  
set path [getFobjPath $fobj]  
print3DEConsole "Image path: " $path "\n"
```

See also

```
setFobjPath
```

Command: setFobjPath

Synopsis

```
setFobjPath <fobj_id> <path>
```

Parameters

<fobj_id> - identifier of a frame object
<path> - absolute filename path starting from “/”

Description

On execution this command redefines the absolute image filename path of the frame object specified by the supplied frame object id. All indexing numbers in the path must be replaced by ‘#’ characters, similar to the “Sequence Settings” requester.

Example

```
# set image filename path of the currently selected  
# frame object to '/data/images/dog.####.jpg'  
set fobj [getSelectedFobj]  
setFobjPath $fobj "/data/images/dog.####.jpg"
```

See also

getFobjPath

Command: getFobjSequenceAttr**Synopsis**

```
[list <start> <end> <step>] getFobjSequenceAttr <fobj_id>
```

Parameters

<start> - integer value defining the start index of an image sequence
<end> - integer value defining the end index of an image sequence
<step> - integer value defining the index increment of an image sequence
<fobj_id> - identifier of a frame object

Description

On execution this command returns indexing information of the frame object specified by the supplied frame object id. The return value consists of a list containing three integer numbers, representing the content of the Start, End, Step textfields found in the “Sequence Settings” requester.

Example

```
# display the sequence attributes of the
# currently selected frame object
set fobj [getSelectedFobj]
set attr_list [getFobjSequenceAttr $fobj]
print3DEConsole "Start index: " [lindex $attr_list 0] "\n"
print3DEConsole "End index: " [lindex $attr_list 1] "\n"
print3DEConsole "Stepsize: " [lindex $attr_list 2] "\n"
```

See also

setFobjSequenceAttr

Command: setFobjSequenceAttr

Synopsis

```
setFobjSequenceAttr <fobj_id> <start> <end> <step>
```

Parameters

<fobj_id> - identifier of a frame object
<start> - integer value defining the start index of an image sequence
<end> - integer value defining the end index of an image sequence
<step> - integer value defining the index increment of an image sequence

Description

On execution this command redefines the indexing attributes of the frame object specified by the supplied frame object id. Executing this command on a reference frame has no effect.

Example

```
# set the indexing attributes of the currently selected  
# frame object to 1 - 55, stepsize 2  
set fobj [getSelectedFobj]  
setFobjSequenceAttr $fobj 1 55 2
```

See also

getFobjSequenceAttr

Command: getFobjImageType**Synopsis**

```
<image_type> getFobjImageType <fobj_id>
```

Parameters

<image_type>	- constant that defines the image type of a frame object: NON_LACE, LACE_EVEN, LACE_ODD, FIELD, ANAMORPHIC
<fobj_id>	- identifier of a frame object

Description

On execution this command returns the image type of the frame object specified by the supplied frame object id.

Example

```
# display the image type of the currently  
# selected frame object  
set fobj [getSelectedFobj]  
set imgtype [getFobjImageType $fobj]  
print3DEConsole "Image type: " $imgtype "\n"
```

See also

setFobjImageType

Command: setFobjImageType

Synopsis

```
setFobjImageType <fobj_id> <image_type>
```

Parameters

<fobj_id>	- identifier of a frame object
<image_type>	- constant that defines the image type of a frame object: NON_LACE, LACE_EVEN, LACE_ODD, FIELD, ANAMORPHIC

Description

On execution this command redefines the image type of the frame object specified by the supplied frame object id.

Example

```
# set the image type of the currently selected  
# frame object to "Noninterlace"  
set fobj [getSelectedFobj]  
setFobjImageType $fobj NON_LACE
```

See also

getFobjImageType

Command: getFobjCamera**Synopsis**

```
<camera_id> getFobjCamera <fobj_id>
```

Parameters

```
<camera_id>    - identifier of a camera object  
<fobj_id>      - identifier of a frame object
```

Description

On execution this command returns the identifier of the camera object that is linked to the frame object specified by the supplied frame object id.

Example

```
# display the name of the camera object linked to the  
# currently selected frame object  
set fobj [getSelectedFobj]  
set camera [getFobjCamera $fobj]  
set name [getCameraName $camera]  
print3DEConsole "Camera name: " $name "\n"
```

See also

getFirstCamera, getNextCamera

Command: getFobjZoomingFlag

Synopsis

<bool> getFobjZoomingFlag <fobj_id>

Parameters

<bool> - boolean value: 0, 1
<fobj_id> - identifier of a frame object

Description

On execution this command returns the status of the “Variable Focal Length” flag of the frame object specified by the supplied frame object id.

Example

```
# display whether zooming is enabled or disabled with the
# currently selected frame object
set fobj [getSelectedFobj]
set zooming [getFobjZoomingFlag $fobj]
if {$zooming} then \
{
  print3DEConsole "Zooming enabled!\n"
} \
else \
{
  print3DEConsole "Zooming disabled!\n"
}
```

See also

getFobjZoomFactor

Command: getFobjZoomFactor**Synopsis**

```
<float> getFobjZoomFactor <fobj_id> <frame>
```

Parameters

```
<float>    - floating point value
<fobj_id> - identifier of a frame object
<frame>   - integer value that defines a frame number
```

Description

On execution this command returns the zoom factor of the specified frame object in the specified frame. Zoom factor values must be multiplied with the focal length value of a camera object to obtain the “real” zoomed focal length.

Example

```
# display all zoom factors of the
# currently selected sequence
set fobj [getSelectedFobj]
for {set frame 1} \
    {$frame<=[getNoFrames $fobj]} \
    {set frame [expr $frame+1]} \
    {
    set zoom_factor [getFobjZoomFactor $fobj $frame]
    print3DEConsole "Frame: " $frame
    print3DEConsole ", zoom factor: " $zoom_factor "\n"
    }
```

See also

```
getFobjZoomingFlag
```

Command: getFobjImageWidth

Synopsis

<image_width> getFobjImageWidth <fobj_id>

Parameters

<image_width> - integer value representing a horizontal image resolution in pixel
<fobj_id> - identifier of a frame object

Description

On execution this command returns the horizontal image resolution of the specified frame object.

Example

```
# display the image width of the selected frame object
set fobj [getSelectedFobj]
set width [getFobjImageWidth $fobj]
print3DEConsole "Image width: " $width "\n"
```

See also

getFobjImageHeight

Command: getFobjImageHeight

Synopsis

<image_height> getFobjImageHeight <fobj_id>

Parameters

<image_height> - integer value representing a vertical image resolution in pixel
<fobj_id> - identifier of a frame object

Description

On execution this command returns the vertical image resolution of the specified frame object.

Example

```
# display the image height of the selected frame object
set fobj [getSelectedFobj]
set height [getFobjImageHeight $fobj]
print3DEConsole "Image height: " $height "\n"
```

See also

getFobjImageWidth

Command: getFobjImageWidthFOV

Synopsis

```
<image_width_fov> getFobjImageWidthFOV <fobj_id>
```

Parameters

```
<image_width_fov> - floating point value representing a horizontal FOV size in pixel  
<fobj_id>          - identifier of a frame object
```

Description

On execution this command returns the horizontal FOV size of the specified frame object.

Example

```
# display the fov width of the selected frame object  
set fobj [getSelectedFobj]  
set width [getFobjImageWidthFOV $fobj]  
print3DEConsole "FOV width: " $width "\n"
```

See also

getFobjImageHeightFOV

Command: getFobjImageHeightFOV

Synopsis

<image_height_fov> getFobjImageHeightFOV <fobj_id>

Parameters

<image_height_fov> - floating point value representing a vertical FOV size in pixel
<fobj_id> - identifier of a frame object

Description

On execution this command returns the vertical FOV size of the specified frame object.

Example

```
# display the fov height of the selected frame object
set fobj [getSelectedFobj]
set height [getFobjImageHeightFOV $fobj]
print3DEConsole "FOV height: " $height "\n"
```

See also

getFobjImageWidthFOV

Command: createPoint

Synopsis

```
<point_id> createPGroup <pgroup_id>
```

Parameters

```
<point_id>      - identifier of a point object  
<pgroup_id>    - identifier of a pointgroup object
```

Description

On execution this command creates a new point object in the specified pointgroup and returns its point object id.

Example

```
# create a new point in the currently selected pointgroup  
set pg [getSelectedPGroup]  
set point [createPoint $pg]
```

See also

deletePoint, getNextPoint, getPointName

Command: deletePoint

Synopsis

```
deletePoint <pgroup_id> <point_id>
```

Parameters

```
<pgroup_id>    - identifier of a pointgroup object  
<point_id>     - identifier of a point object
```

Description

On execution this command deletes the specified point object.

Example

```
# delete the currently selected point  
set pg [getSelectedPGroup]  
set point [getSelectedPoint]  
deletePGroup $pg $point
```

See also

createPoint, getNextPoint, getPointName

Command: **getFirstPoint**

Synopsis

```
<point_id> getFirstPoint <pgroup_id>
```

Parameters

```
<point_id>      - identifier of a point object  
<pgroup_id>    - identifier of a pointgroup object
```

Description

On execution this command returns the point object id of the first point in the specified pointgroup. If there are no point objects available, the string "0" is returned.

Example

```
set point [getFirstPoint $pg]
```

See also

createPoint, deletePoint, getNextPoint, getPointName

Command: getNextPoint**Synopsis**

```
<point_id> getNextPoint <pgroup_id> <point_id>
```

Parameters

```
<point_id>      - identifier of a point object  
<pgroup_id>    - identifier of a pointgroup object
```

Description

On execution this command returns the point id of the point that comes next after the specified point in the specified pointgroup. If the specified point is the last one, the string "0" is returned.

Example

```
# display the names of all points of the selected pointgroup  
set pg [getSelectedPGroup]  
for {set point [getFirstPoint $pg]} \  
    {$point != 0} \  
    {set point [getNextPoint $pg $point]} \  
{  
    set name [getPointName $pg $point]  
    print3DEConsole "Point name: " $name "\n"  
}
```

See also

createPoint, deletePoint, getPointName

Command: getNoPoints

Synopsis

```
<number_of_points> getNoPoints <pgroup_id>
```

Parameters

<number_of_points>- integer that specifies the number of point objects
<pgroup_id> - identifier of a pointgroup object

Description

On execution this command returns the number of available points in the specified pointgroup.

Example

```
set no_points [getNoPoints $pg]
```

See also

getIndexPoint

Command: getIndexPoint**Synopsis**

```
<point_id> getIndexPoint <pgroup_id> <point_index>
```

Parameters

```
<point_id>          - identifier of a point object  
<pgroup_id>        - identifier of a pointgroup object  
<point_index>      - integer that specifies a point object
```

Description

On execution this command returns the id of the point specified by the point index parameter in the specified pointgroup. The first point object is defined by an index of "0".

Example

```
# display the names of all points in the selected pointgroup  
set pg [getSelectedPGroup]  
for {set index 0} \  
    {$index < [getNoPoints $pg]} \  
    {set index [expr $index+1]} \  
    {  
        set point [getIndexPGroup $pg $index]  
        set name [getPointName $pg $point]  
        print3DEConsole "Point name: " $name "\n"  
    }  
}
```

See also

getNoPoints

Command: findPointByName

Synopsis

<point_id> findPointByName <pgroup_id> <name>

Parameters

<point_id> - identifier of a point object
<pgroup_id> - identifier of a pointgroup object
<name> - name string

Description

On execution this command returns the id of the point in the specified pointgroup that matches the specified name. If no point object could be found, a warning message is displayed and the string "0" is returned.

Example

```
# search point "wurst" in the selected pointgroup
set pg [getSelectedPGroup]
set point [findPointByName $pg "wurst"]
if {$point!=0}\
{
  print3DEConsole "Point wurst has been found!\n"
} \
else \
{
  print3DEConsole "Point wurst has not been found!\n"
}
```

See also

setPointName, getPointName

Command: getPointName**Synopsis**

```
<name> getPointName <pgroup_id> <point_id>
```

Parameters

<name>	- name string
<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object

Description

On execution this command returns the name of the specified point in the specified pointgroup.

Example

```
# display name of the currently selected point
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set name [getPointName $pg $point]
print3DEConsole "Point name: " $name "\n"
```

See also

setPointName

Command: setPointName

Synopsis

```
setPointName <pgroup_id> <point_id> <name>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<name>	- name string

Description

On execution this command sets the name of the specified point in the specified pointgroup to the supplied name string.

Example

```
# set name of the currently selected point to "honsel"  
set pg [getSelectedPGroup]  
set point [getSelectedPoint]  
setPointName $pg $point "honsel"
```

See also

getPointName

Command: `getPointSelectionFlag`**Synopsis**

```
<bool> getPointSelectionFlag <pgroup_id> <point_id>
```

Parameters

```
<bool>           - boolean value: 0, 1  
<pgroup_id>     - identifier of a pointgroup object  
<point_id>      - identifier of a point object
```

Description

On execution this command returns the string “1”, if the specified point is currently selected. Otherwise string “0” is returned.

Example

```
# display the names of all selected points in the  
# currently selected pointgroup  
set pg [getSelectedPGroup]  
for {set point [getFirstPoint $pg]} \  
    {$point!=0} \  
    {set point [getNextPoint $pg $point]} \  
    {  
    if {[getPointSelectionFlag $pg $point]} \  
    {  
        set name [getPointName $pg $point]  
        print3DEConsole "Point " $name " is selected.\n"  
    }  
    }  
}
```

See also

`getSelectedPoint`

Command: `getPointPosition3D`

Synopsis

```
<vector_3d> getPointPosition3D <pgroup_id> <point_id>
```

Parameters

<code><vector_3d></code>	- three-dimensional floating point vector
<code><pgroup_id></code>	- identifier of a pointgroup object
<code><point_id></code>	- identifier of a point object

Description

On execution this command returns the user-defined 3D point position of the specified point in the specified pointgroup.

Example

```
# display the user-defined 3D position of the currently
# selected point
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set vec3d [getPointPosition3D $pg $point]
print3DEConsole "3D point position: " $vec3d "\n"
```

See also

`setPointPosition3D`

Command: setPointPosition3D**Synopsis**

```
setPointPosition3D <pgroup_id> <point_id> <vector_3d>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<vector_3d>	- three-dimensional floating point vector

Description

On execution this command redefines the user-defined 3D point position of the specified point in the specified pointgroup.

Example

```
# set the user-defined 3D position of the currently
# selected point to (0,0,0)
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set vec3d [list 0 0 0]
setPointPosition3D $pg $point $vec3d
```

See also

```
getPointPosition3D
```

Command: `getPointCalcPosition3D`

Synopsis

```
<vector_3d> getPointCalcPosition3D <pgroup_id> <point_id>
```

Parameters

```
<vector_3d>      - three-dimensional floating point vector  
<pgroup_id>     - identifier of a pointgroup object  
<point_id>      - identifier of a point object
```

Description

On execution this command returns the calculated 3D point position of the specified point in the specified pointgroup.

Example

```
# display the calculated 3D position of the currently  
# selected point  
set pg [getSelectedPGroup]  
set point [getSelectedPoint]  
set vec3d [getPointCalcPosition3D $pg $point]  
print3DEConsole "3D point position: " $vec3d "\n"
```

See also

```
setPointCalcPosition3D
```

Command: setPointCalcPosition3D**Synopsis**

```
setPointPosition3D <pgroup_id> <point_id> <vector_3d>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<vector_3d>	- three-dimensional floating point vector

Description

On execution this command redefines the calculated 3D point position of the specified point in the specified pointgroup.

Example

```
# set the calculated 3D position of the currently  
# selected point to (0,0,0)  
set pg [getSelectedPGroup]  
set point [getSelectedPoint]  
set vec3d [list 0 0 0]  
setPointCalcPosition3D $pg $point $vec3d
```

See also

```
getPointCalcPosition3D
```

Command: `getPointMoCapCalcPosition3D`

Synopsis

`<vector_3d> getPointMoCapCalcPosition3D <pgroup_id> <point_id> <fobj_id> <f>`

Parameters

<code><vector_3d></code>	- three-dimensional floating point vector
<code><pgroup_id></code>	- identifier of a pointgroup object
<code><point_id></code>	- identifier of a point object
<code><fobj_id></code>	- identifier of a frame object
<code><f></code>	- integer value that defines a frame number

Description

On execution this command returns the calculated 3D point position of the specified point in the specified mocap pointgroup, tracked in the specified frame of the specified sequence.

Example

```
# display the calculated 3D position of the currently
# selected mocap point, in frame 1 of the first sequence
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getFirstFobj]
set vec3d [getPointMoCapCalcPosition3D $pg $point $fobj 1]
print3DEConsole "3D point position: " $vec3d "\n"
```

See also

`getPGroupType`

Command: getPointSurveyType**Synopsis**

```
<survey_type> getPointSurveyType <pgroup_id> <point_id>
```

Parameters

<survey_type>	- constant that defines a point's survey type: SURVEY_FREE, SURVEY_APPROX, SURVEY_EXACT
<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object

Description

On execution this command returns the survey type of the specified point in the specified pointgroup.

Example

```
# display the survey type of the currently selected point
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set survey_type [getPointSurveyType $pg $point]
print3DEConsole "Survey type: " $survey_type "\n"
```

See also

setPointSurveyType

Command: **setPointSurveyType**

Synopsis

```
setPointSurveyType <pgroup_id> <point_id> <survey_type>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<survey_type>	- constant that defines a point's survey type: SURVEY_FREE, SURVEY_APPROX, SURVEY_EXACT

Description

On execution this command redefines the survey type of the specified point in the specified pointgroup.

Example

```
# set the survey type of the currently selected point to  
# 'exactly surveyed point'  
set pg [getSelectedPGroup]  
set point [getSelectedPoint]  
setPointSurveyType $pg $point SURVEY_EXACT
```

See also

getPointSurveyType

Command: `getPointPosition2D`

Synopsis

```
<vector_2d> getPointPosition2D <pgroup_id> <point_id> <fobj_id> <frame>
```

Parameters

<code><vector_2d></code>	- two-dimensional floating point vector; (0,0) means the lower left image corner; (1,1) means the upper right image corner
<code><pgroup_id></code>	- identifier of a pointgroup object
<code><point_id></code>	- identifier of a point object
<code><fobj_id></code>	- identifier of a frame object
<code><frame></code>	- integer value that defines a frame number

Description

On execution this command returns the 2D tracking point position of the specified point in the specified pointgroup, tracked in the specified frame of the specified frame object. This command is not very efficient. For handling large amounts of 2D tracking data use “`get/setPointPosition2DBlock`” instead.

Example

```
# display the 2D tracking curve of the selected point
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getSelectedFobj]
for {set frame 1} {$frame<=[getNoFrames $fobj]} \
  {set frame [expr $frame+1]} \
  {
    set vec2d [getPointPosition2D $pg $point $fobj $frame]
    print3DEConsole "Frame: " $frame "," vec2d "\n"
  }
}
```

See also

`setPointPosition2D`, `setPointPosition2DBlock`, `getPointPosition2DBlock`

Command: setPointPosition2D

Synopsis

```
setPointPosition2D <pgroup_id> <point_id> <fobj_id> <frame> <vector_2d>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<fobj_id>	- identifier of a frame object
<frame>	- integer value that defines a frame number
<vector_2d>	- two-dimensional floating point vector; (0,0) means the lower left image corner; (1,1) means the upper right image corner

Description

On execution this command sets the 2D tracking position of the specified point in the specified pointgroup, tracked in the specified frame of the specified frame object, to the specified 2D vector. This command is not very efficient. For handling large amounts of 2D tracking data use “get/setPointPosition2DBlock” instead.

Example

```
# set the 2D tracking point position of the selected point
# in frame 13 of the selected frame object to (0.45,0.74)
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getSelectedFobj]
setPointPosition2D $pg $point $fobj 13 [list 0.45 0.74]
```

See also

getPointPosition2D, setPointPosition2DBlock, getPointPosition2DBlock

Command: `getPointPosition2DBlock`

Synopsis

```
<curve_2d> getPointPosition2DBlock <pgroup_id> <point_id> <fobj_id> <start> <end>
```

Parameters

<code><curve_2d></code>	- list of two-dimensional floating point vectors; a (-1,-1) 2D vector element represents an invalid tracking point
<code><pgroup_id></code>	- identifier of a pointgroup object
<code><point_id></code>	- identifier of a point object
<code><fobj_id></code>	- identifier of a frame object
<code><start></code>	- integer value that defines a frame number
<code><end></code>	- integer value that defines a frame number

Description

On execution this command returns a list of 2D tracking point positions of the specified point in the specified pointgroup, tracked in the specified frame object. The first 2D vector in the list represents the tracking position of the specified start frame number.

Example

```
# display the 2D tracking curve of the selected point
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getSelectedFobj]
set n [getNoFrames $fobj]
set curve [getPointPosition2DBlock $pg $point $fobj 1 $n]
print3DEConsole "Tracking curve: " $curve "\n"
```

See also

`setPointPosition2D`, `getPointPosition2D`, `setPointPosition2DBlock`

Command: setPointPosition2DBlock

Synopsis

```
setPointPosition2DBlock <pgroup_id> <point_id> <fobj_id> <start> <curve_2d>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<fobj_id>	- identifier of a frame object
<start>	- integer value that defines a frame number
<curve_2d>	- list of two-dimensional floating point vectors; a (-1,-1) 2D vector element represents an invalid tracking point

Description

On execution this command copies the 2D tracking point positions contained in the specified curve to the tracking curve of the specified point in the specified pointgroup, of the specified frame object. The 2D tracking curve is copied to the point's curve starting at the specified start frame number.

Example

```
# copy the 2D tracking curve in frame object fobj from  
# point p1 to point p2 (pointgroup pg)  
set n [getNoFrames $fobj]  
set curve [getPointPosition2DBlock $pg $p1 $fobj 1 $n]  
setPointPosition2DBlock $pg $p2 $fobj 1 $curve
```

See also

setPointPosition2D, getPointPosition2D, getPointPosition2DBlock

Command: isPointPos2DValid

Synopsis

```
<bool> isPointPos2DValid <pgroup_id> <point_id> <fobj_id> <frame>
```

Parameters

<bool>	- boolean value: 0, 1
<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<fobj_id>	- identifier of a frame object
<frame>	- integer value that defines a frame number

Description

On execution this command returns the visibility status of the 2D tracking curve of the specified point in the specified pointgroup, tracked in the specified frame object. If the tracking point in that frame is visible, the string "1" is returned. Otherwise string "0" is returned.

Example

```
# display the visibility status of the selected point's
# 2D tracking curve
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getSelectedFobj]
for {set frame 1} {$frame<=[getNoFrames $fobj]} \
  {set frame [expr $frame+1]} \
  {
    if {[isPointPos2DValid $pg $point $fobj $frame]} then\
      print3DEConsole "Frame: " $frame " is valid.\n"
  }
}
```

See also

setPointPosition2D, getPointPosition2D

Command: deletePointCurve2D

Synopsis

```
deletePointCurve2D <pgroup_id> <point_id> <fobj_id>
```

Parameters

<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<fobj_id>	- identifier of a frame object

Description

On execution this command deletes the entire 2D tracking curve of the specified point in the specified pointgroup, tracked in the specified frame object.

Example

```
# remove the selected point's 2D tracking curve
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getSelectedFobj]
deletePointCurve2D $pg $point $fobj
```

See also

setPointPosition2DBlock, getPointPosition2DBlock

Command: isPointCalculated3D**Synopsis**

```
<bool> isPointCalculated3D <pgroup_id> <point_id>
```

Parameters

```
<bool>           - boolean value: 0, 1  
<pgroup_id>     - identifier of a pointgroup object  
<point_id>      - identifier of a point object
```

Description

On execution this command returns the calculation status of the specified point in the specified pointgroup. If the point's 3D position has been calculated, the string "1" is returned. Otherwise string "0" is returned.

Example

```
# display the calculation status of all points in the  
# currently selected pointgroup  
set pg [getSelectedPGroup]  
for {set point [getFirstPoint $pg]} \  
    {$point!=0} \  
    {set point [getNextPoint $pg $point]} \  
{  
  if {[isPointCalculated3D $pg $point]} \  
  {  
    set name [getPointName $pg $point]  
    print3DEConsole "Point " $name " is calculated.\n"  
  }  
}
```

See also

setPointCalcPosition3D, getPointCalcPosition3D

Command: `getPointDeviation`

Synopsis

```
<double> getPointDeviation <pgroup_id> <point_id> <fobj_id> <frame>
```

Parameters

<double>	- floating point value
<pgroup_id>	- identifier of a pointgroup object
<point_id>	- identifier of a point object
<fobj_id>	- identifier of a frame object
<frame>	- integer value that defines a frame number

Description

On execution this command returns the deviation of the specified point in the specified pointgroup, tracked in the specified frame object.

Example

```
# display the status of the selected point's
# 2D tracking curve
set pg [getSelectedPGroup]
set point [getSelectedPoint]
set fobj [getSelectedFobj]
for {set frame 1} {$frame<=[getNoFrames $fobj]} \
  {set frame [expr $frame+1]} \
  {
    set dev [getPointDeviation $pg $point $fobj $frame]
    print3DEConsole "Frame: " $frame ": " $dev "\n"
  }
```

Command: `getPointColor2D`**Synopsis**

```
<color_index> getPointColor2D <pgroup_id> <point_id>
```

Parameters

```
<color_index>    - integer value: 0,...,6  
<pgroup_id>     - identifier of a pointgroup object  
<point_id>      - identifier of a point object
```

Description

On execution this command returns the tracking points' color index of the specified point in the specified pointgroup.

Example

```
# display the 2D color index of all points  
set pg [getSelectedPGroup]  
for {set point [getFirstPoint $pg]} \  
    {$point!=0} \  
    {set point [getNextPoint $pg $point]} \  
    {  
        set col [getPointColor2D $pg $point]  
        set name [getPointName $pg $point]  
        print3DEConsole "Point " $name ": " $col "\n"  
    }  
}
```

See also

`setPointColor2D`

Command: setPointColor2D

Synopsis

<color_index> setPointColor2D <pgroup_id> <point_id>

Parameters

<color_index> - integer value: 0,...,6
<pgroup_id> - identifier of a pointgroup object
<point_id> - identifier of a point object

Description

On execution this command sets the tracking points' color index of the specified point in the specified pointgroup to the specified value.

Example

```
# set all selected points to green
set pg [getSelectedPGroup]
for {set point [getFirstPoint $pg]} \
    {$point!=0} \
    {set point [getNextPoint $pg $point]} \
    {
    if {[getPointSelectionFlag $pg $point]} \
    {
        setPointColor2D $pg $point 1
    }
    }
```

See also

getPointColor2D

Command: getScenePosition3D

Synopsis

<vector_3d> getScenePosition3D

Parameters

<vector_3d> - three-dimensional floating point vector

Description

On execution this command returns the 3D position of the scene node object displayed in the Orientation Window.

Example

```
set vec3d [getScenePosition3D]
```

See also

getSceneRotation3D, getSceneScale3D

Command: `getSceneRotation3D`

Synopsis

`<matrix_3d> getSceneRotation3D`

Parameters

`<matrix_3d>` - three-dimensional floating point matrix

Description

On execution this command returns the 3D orientation of the scene node object displayed in the Orientation Window.

Example

```
set mtx3d [getSceneOrientation3D]
```

See also

`getScenePosition3D`, `getSceneScale3D`

Command: getSceneScale3D

Synopsis

<float> getSceneScale3D

Parameters

<float> - floating point value

Description

On execution this command returns the scale value of the scene node object displayed in the Orientation Window.

Example

```
set scale [getSceneScale3D]
```

See also

getScenePosition3D, getSceneRotation3D

Command: getPGroupPosition3D

Synopsis

<vector_3d> getPGroupPosition3D <pgroup_id> <fobj_id> <frame>

Parameters

<vector_3d>	- three-dimensional floating point vector
<pgroup_id>	- identifier of a pointgroup object
<fobj_id>	- identifier of a frame object
<frame>	- integer value that defines a frame number

Description

On execution this command returns the reconstructed 3D position of the specified pointgroup tracked in the specified frame of the specified frame object. If the pointgroup is of type “camera group” the return value represents the camera’s position in 3D space.

Example

```
# display the 3D position of the selected pointgroup in the
# selected frame object in the current frame
set pg [getSelectedPGroup]
set fobj [getSelectedFobj]
set frame [getSelectedFrame]
set vec3d [getPGroupPosition3D $pg $fobj $frame]
print3DEConsole "3D Position: " $vec3d "\n"
```

See also

getPGroupRotation3D, getPGroupScale3D

Command: getPGroupRotation3D**Synopsis**

```
<matrix_3d> getPGroupRotation3D <pgroup_id> <fobj_id> <frame>
```

Parameters

<matrix_3d>	- three-dimensional floating point matrix
<pgroup_id>	- identifier of a pointgroup object
<fobj_id>	- identifier of a frame object
<frame>	- integer value that defines a frame number

Description

On execution this command returns the reconstructed 3D orientation of the specified pointgroup tracked in the specified frame of the specified frame object. If the pointgroup is of type “camera group” the return value represents the camera’s orientation in 3D space.

Example

```
# display the 3D rotation of the selected pointgroup in the
# selected frame object in the current frame
set pg [getSelectedPGroup]
set fobj [getSelectedFobj]
set frame [getSelectedFrame]
set mtx3d [getPGroupRotation3D $pg $fobj $frame]
print3DEConsole "3D Rotation: " $mtx3d "\n"
```

See also

getPGroupPosition3D, getPGroupScale3D

Command: getPGroupScale3D

Synopsis

```
<float> getPGroupScale3D <pgroup_id>
```

Parameters

<float>	- three-dimensional floating point value
<pgroup_id>	- identifier of a pointgroup object

Description

On execution this command returns the overall scale value of the specified pointgroup.

Example

```
# display the 3D scale of the selected pointgroup
set pg [getSelectedPGroup]
set scale [getPGroupScale3D $pg]
print3DEConsole "Scale value: " $scale "\n"
```

See also

getPGroupPosition3D, getPGroupRotation3D

Command: getFirstCamera

Synopsis

<camera_id> getFirstCamera

Parameters

<camera_id> - identifier of a camera object

Description

On execution this command returns the camera id of the first camera object. If there are no camera objects available, the string "0" is returned.

Example

```
set camera [getFirstCamera]
```

See also

getNextCamera, getCameraName

Command: getNextCamera

Synopsis

<camera_id> getNextCamera <camera_id>

Parameters

<camera_id> - identifier of a camera object

Description

On execution this command returns the camera id of the camera object that comes next after the specified camera object. If the specified camera object is the last one, the string "0" is returned.

Example

```
# display the names of all cameras
for {set camera [getFirstCamera]} \
    {$camera != 0} \
    {set camera [getNextCamera $camera]} \
    {
    set name [getCameraName $pg]
    print3DEConsole "Camera name: " $name "\n"
    }
```

See also

getFirstCamera, getCameraName

Command: getNoCameras

Synopsis

<number_of_cameras> getNoCameras

Parameters

<number_of_cameras> - integer that specifies the number of camera objects

Description

On execution this command returns the number of available camera objects.

Example

```
set no_cameras [getNoCameras]
```

See also

getIndexCamera, getCameraName

Command: `getIndexCamera`

Synopsis

`<camera_id> getIndexCamera <camera_index>`

Parameters

`<camera_id>` - identifier of a camera object
`<camera_index>` - integer that specifies a camera object

Description

On execution this command returns the camera id of a camera object specified by the camera index parameter. The first camera object is defined by an index of "0".

Example

```
# display the names of all cameras
for {set index 0} \
    {$index < [getNoCameras]} \
    {set index [expr $index+1]} \
{
    set camera [getIndexCamera $index]
    set name [getCameraName $camera]
    print3DEConsole "Camera name: " $name "\n"
}
```

See also

`getNoCameras`, `getCameraName`

Command: getCameraName

Synopsis

<name> getCameraName <camera_id>

Parameters

<name> - name string
<camera_id> - identifier of a camera object

Description

On execution this command returns the name of the specified camera.

Example

```
# display name of the currently selected camera
set camera [getSelectedCamera]
set name [getCameraName $camera]
print3DEConsole "Camera name: " $name "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraFBackWidth

Synopsis

<float> getCameraFBackWidth <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the filmback width value of the specified camera.

Example

```
# display filmback width of the currently selected camera
set camera [getSelectedCamera]
set fback_w [getCameraFBackWidth $camera]
print3DEConsole $fback_w "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraFBackHeight

Synopsis

<float> getCameraFBackHeight <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the filmback height value of the specified camera.

Example

```
# display filmback height of the currently selected camera
set camera [getSelectedCamera]
set fback_h [getCameraFBackHeight $camera]
print3DEConsole $fback_h "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: `getCameraFocalLength`

Synopsis

```
<float> getCameraFocalLength <camera_id>
```

Parameters

```
<float>          - floating point value  
<camera_id>     - identifier of a camera object
```

Description

On execution this command returns the focal length value of the specified camera.

Example

```
# display focal length of the currently selected camera  
set camera [getSelectedCamera]  
set focal [getCameraFocalLength $camera]  
print3DEConsole $focal "\n"
```

See also

`getFirstCamera`, `getNextCamera`, `getNoCameras`, `getIndexCamera`

Command: getCameraFilmAspect

Synopsis

<float> getCameraFilmAspect <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the film aspect value of the specified camera.

Example

```
# display film aspect of the currently selected camera
set camera [getSelectedCamera]
set film_aspect [getCameraFilmAspect $camera]
print3DEConsole $film_aspect "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraLensCenterX

Synopsis

<float> getCameraLensCenterX <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the lens center x offset value of the specified camera.

Example

```
# display lens center x of the currently selected camera
set camera [getSelectedCamera]
set lcenter_x [getCameraLensCenterX $camera]
print3DEConsole $lcenter_x "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraLensCenterY**Synopsis**

```
<float> getCameraLensCenterY <camera_id>
```

Parameters

```
<float>          - floating point value  
<camera_id>     - identifier of a camera object
```

Description

On execution this command returns the lens center y offset value of the specified camera.

Example

```
# display lens center y of the currently selected camera  
set camera [getSelectedCamera]  
set lcenter_y [getCameraLensCenterY $camera]  
print3DEConsole $lcenter_y "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraHorizAngle

Synopsis

<float> getCameraHorizAngle <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the horizontal angle value of the specified camera.

Example

```
# display horizontal angle of the currently selected camera
set camera [getSelectedCamera]
set h_angle [getCameraHorizAngle $camera]
print3DEConsole $h_angle "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraVertiAngle**Synopsis**

```
<float> getCameraVertiAngle <camera_id>
```

Parameters

```
<float>          - floating point value  
<camera_id>     - identifier of a camera object
```

Description

On execution this command returns the vertical angle value of the specified camera.

Example

```
# display vertical angle of the currently selected camera  
set camera [getSelectedCamera]  
set v_angle [getCameraVertiAngle $camera]  
print3DEConsole $v_angle "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraPixelAspect

Synopsis

<float> getCameraPixelAspect <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the pixel aspect value of the specified camera.

Example

```
# display pixel aspect of the currently selected camera
set camera [getSelectedCamera]
set pixel_aspect [getCameraPixelAspect $camera]
print3DEConsole $pixel_aspect "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraLensDistortion**Synopsis**

<float> getCameraLensDistortion <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the lens distortion value of the specified camera.

Example

```
# display lens distortion of the currently selected camera
set camera [getSelectedCamera]
set distortion [getCameraLensDistortion $camera]
print3DEConsole $distortion "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraAnamorphicSqueeze

Synopsis

<float> getCameraAnamorphicSqueeze <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the anamorphic squeeze value of the specified camera.

Example

```
# display anamorphic squeeze of the  
# currently selected camera  
set camera [getSelectedCamera]  
set anamorph [getCameraAnamorphicSqueeze $camera]  
print3DEConsole $anamorph "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraCurvatureX**Synopsis**

```
<float> getCameraCurvatureX <camera_id>
```

Parameters

```
<float>          - floating point value  
<camera_id>     - identifier of a camera object
```

Description

On execution this command returns the curvature x value of the specified camera.

Example

```
# display curvature x of the currently selected camera  
set camera [getSelectedCamera]  
set curve_x [getCameraCurvatureX $camera]  
print3DEConsole $curve_x "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: **getCameraCurvatureY**

Synopsis

<float> getCameraCurvatureY <camera_id>

Parameters

<float> - floating point value
<camera_id> - identifier of a camera object

Description

On execution this command returns the curvature y value of the specified camera.

Example

```
# display curvature y of the currently selected camera
set camera [getSelectedCamera]
set curve_y [getCameraCurvatureY $camera]
print3DEConsole $curve_y "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Command: getCameraQuarticDistortion**Synopsis**

```
<float> getCameraQuarticDistortion <camera_id>
```

Parameters

```
<float>          - floating point value  
<camera_id>     - identifier of a camera object
```

Description

On execution this command returns the quartic distortion value of the specified camera.

Example

```
# display quartic distortion of the current camera  
set camera [getSelectedCamera]  
set q_distort [getCameraQuarticDistortion $camera]  
print3DEConsole $q_distort "\n"
```

See also

getFirstCamera, getNextCamera, getNoCameras, getIndexCamera

Appendix B: A collection of vector, matrix and quaternion operations in Tcl

The file “veclib.tcl” in the 3DE distribution (directory “<3de install dir>/sys_data/tcl_archive”) is a collection of Tcl procedures, which are based on the vector, matrix and quaternion types used for 3DE’s Tcl interface. You will find a more detailed description and examples in the file “veclib_doc.pdf”. In the list below we used the following conventions:

- The entries for each procedure in the list consist of the return type of the procedure (in <angle> brackets), the procedure name and the list of parameters.
- The possible return types are:
 - <real> for real numbers
 - <vec2> for two-dimensional vectors
 - <vec3> for two-dimensional vectors
 - <quat> for quaternions
 - <mat2> for 2x2-matrices
 - <mat3> for 3x3-matrices
 - <mat4> for 4x4-matrices
 - <srt2> for two-dimensional general transformations
 - <srt3> for three-dimensional general transformations
 - <boole> boolean return type: “0” or “1”
- <vec> stands for all types of vectors (i.e. vec2, vec3 and the unsupported type vec4)
- <mat> stands for all types of matrices (i.e. mat2, mat3 and the unsupported type mat4)
- <srt> stands for all types of general transformations (i.e. srt2 and srt3)
- The command names are constructed as follows
 - Binary operators match the pattern <type1><op><type2>, e.g. vec2+vec2, mat3*vec3
 - Unary operators match the pattern <op><type>, e.g. -quat
 - Functions mapping from an object of type <type> and possible parameters to anywhere match the pattern <type><func-name>, e.g. mat2invert, mat3subcyc
 - Functions testing the type match the pattern is<type>, e.g. ismat3
 - Type conversions / constructors match the pattern <type1>=<type2>, e.g. quat=vec3,
 - The nomenclatura of get/set functions is not necessarily regular, e.g. mgetrow, qgetvec...

- The list of parameters obey the following convention:
 - vectors are denoted by $v, v1, v2, v3\dots$
 - matrices are denoted by $m, m1, m2, m3\dots$
 - integers are denoted by $i, j, k\dots$
 - angles (radian!) are denoted by w, w^*
 - numbers (scalars) are denoted by $s, s1, s2, s3\dots$
 - general transformations (srt) are denoted by $l, l1, l2, l3\dots$
 - parameters are written as if they were given to the procedure by variables (with or without '\$')

In Tcl there are no elementary data types except for strings. We use strings and Tcl lists in order to construct vectors, matrices, quaternions and transformations. Examples:

```
wish>set va {1 2 3}
wish>set vb {4 5 6}
wish>puts [vec3+vec3 $va $vb]
5 7 9
```

```
wish>set ma {{1 2} {3 4}}
wish>set mb {{5 6} {7 8}}
wish>puts [mat2*mat2 $ma $mb]
{19 22} {43 50}
```

```
wish>set q {1 {2 3 4}}
wish>puts $q
1 {2 3 4}
```

```
wish>set m {{11 12 13} {21 22 23} {31 32 33}}
wish>puts $m
{11 12 13} {21 22 23} {31 32 33}
```

```
wish>set l [list $v $m]; # or: set l [srt=vecmat $v $m]
wish>puts $l
{ 5 7 9 } {{11 12 13} {21 22 23} {31 32 33}}
```

Commands

get/set functions

<real>	vget \$v \$i	Component i of vector v.
<vec>	vset v \$i \$s	Set component i of vector v to s. Returns v.
<vec>	mgetrow \$m \$i	Row-vector i of matrix m.
<mat>	msetrow m \$i \$v	Set row-vector i of matrix m to v. Returns m.
<real>	mget \$m \$i \$j	Component (i,j) of matrix m.
<mat>	mset m \$i \$j \$s	Set component (i,j) of matrix m to s.
<real>	qget \$q \$i	Component i of quaternion q.
<quat>	qset q \$i \$s	Set component i of quaternion q to s.
<real>	qgets \$q	Scalar part of quaternion q (component 0).
<quat>	qsets q \$s	Set scalar part of quaternion q (component 0) to s.
<vec3>	qgetvec \$q	Vector part of quaternion q.
<quat>	qsetvec q \$v	Set vector part of quaternion q to v.
<vec>	srtgetvec \$l	Vector part of transformation l.
<srt>	srtsetvec l \$v	Set vector part of transformation l to v. Returns l.
<mat>	srtgetmat \$l	Matrix part of transformation l.
<srt>	srtsetmat l \$m	Set matrix part of transformation l to m. Returns l.

vec2

<vec2>	vec2+vec2 \$v1 \$v2	Sum of vectors v1 and v2.
<vec2>	vec2-vec2 \$v1 \$v2	Difference of vectors v1 and v2.
<vec2>	-vec2 \$v	Negative of vector v.
<real>	vec2*vec2 \$v1 \$v2	Scalar product of vectors v1 and v2.
<vec2>	vec2*s \$v \$s	Component-wise product of vector v and scalar s.
<vec2>	s*vec2 \$s \$v	Component-wise product of vector v and scalar s.
<vec2>	vec2/s \$v \$s	Component-wise quotient of vector v and scalar s.
<real>	vec2^vec2 \$v1 \$v2	Skew-symmetric spinor product of vectors v1 and v2.
<mat2>	vec2&vec2 \$v1 \$v2	Tensor product of vectors v1 and v2.
<real>	vec2normquad \$v	Square of the euclidian norm of vector v.
<real>	vec2norm \$v	Euclidian norm of vector v.
<vec2>	vec2unit \$v	Vector v normalized.
<boole>	isvec2 \$a	a is a list that consists of two components? 1:0.

vec3

<vec3>	vec3+vec3 \$v1 \$v2	Sum of vectors v1 and v2.
<vec3>	vec3-vec3 \$v1 \$v2	Difference of vectors v1 and v2.
<vec3>	-vec3 \$v	Negative of vector v.
<real>	vec3*vec3 \$v1 \$v2	Scalar product of vectors v1 and v2.
<vec3>	vec3*s \$v \$s	Component-wise product of vector v and scalar s.
<vec3>	s*vec3 \$s \$v	Component-wise product of vector v and scalar s.
<vec3>	vec3/s \$v \$s	Component-wise quotient of vector v and scalar s.
<vec3>	vec3^vec3 \$v1 \$v2	Skew-symmetric vector product of vectors v1 and v2.
<mat3>	vec3&vec3 \$v1 \$v2	Tensor product of vectors v1 and v2.
<real>	vec3normquad \$v	Square of the euclidian norm of vector v.
<real>	vec3norm \$v	Euclidian norm of vector v.
<vec3>	vec3unit \$v	Vector v normalized.
<mat3>	vec3dual \$v	Dual matrix of vector v.
<boole>	isvec3 \$a	a is a list that consists of three components? 1:0.

quat

<quat>	quat+quat \$q1 \$q2	Sum of quaternions q1 and q2.
<quat>	quat-quat \$q1 \$q2	Difference of quaternions q1 and q2.
<quat>	-quat \$q	Negative of quaternions.
<quat>	quat^quat \$q1 \$q2	Non-commutative product of quaternions q1 and q2.
<real>	quat*quat \$q1 \$q2	Real-valued scalar product of quaternions q1 and q2.
<quat>	quat*s \$q \$s	Product of quaternion q and scalar s.
<quat>	s*quat \$s \$q	Product of quaternion q and scalar s.
<quat>	quat/s \$q \$s	Quotient of quaternion q and scalar s.
<quat>	quat=s \$s	Generates a quaternion out of scalar s.
<quat>	quat=vec3 \$v	Generates a quaternion out of vector v.
<real>	quatnormquad \$q	Square of the euclidian norm of quaternion q.
<real>	quatnorm \$q	Euclidian norm of quaternion q.
<quat>	quatunit \$q	Quaternion q normalized.
<quat>	quatconj \$q	Conjugate of quaternion q.
<quat>	quatinvert \$q	Inverse quaternion of q.
<mat3>	quatmaptomat3 \$q	Maps quaternion q into a 3x3-matrix. See explanation in file "veclib_doc.pdf".

mat2

<mat2>	mat2+mat2 \$m1 \$m2	Sum of matrices m1 and m2.
<mat2>	mat2-mat2 \$m1 \$m2	Difference of matrices m1 and m2.
<mat2>	-mat2 \$m	Negative of matrix m.
<mat2>	mat2*mat2 \$m1 \$m2	Product of matrices m1 and m2.
<mat2>	mat2*s \$m \$s	Component-wise product of matrix m and scalar s.
<mat2>	s*mat2 \$s \$m	Component-wise product of matrix m and scalar s.
<vec2>	mat2*vec2 \$m \$v	Product of matrix m and vector v.
<mat2>	mat2/s \$m \$s	Component-wise quotient of matrix m and scalar s.
<mat2>	mat2=s \$s	Generates a matrix out of scalar s.
<mat2>	mat2=diag \$s1 \$s2	Generates a diagonal matrix out of scalars s1 and s2.
<real>	mat2trace \$m	Trace of matrix m.
<real>	mat2det \$m	Determinant of matrix m.
<mat2>	mat2trans \$m	Matrix m transposed.
<real>	mat2subcyc \$m \$i \$j	Adjoint element of matrix m with respect to component (i,j).
<mat2>	mat2invert \$m	Inverse of matrix m.
<real>	mat2rotangle \$m	Calculate rotation angle of a rotation matrix m. See explanation in file "veclib_doc.pdf".
<mat2>	mat2rotmatrix \$w	Calculate rotation matrix out of angle w. See explanation in file "veclib_doc.pdf".
<boole>	ismat2 \$a	a is a list of two vec2 objects? 1:0.

mat3

<mat3>	mat3+mat3 \$m1 \$m2	Sum of matrices m1 and m2.
<mat3>	mat3-mat3 \$m1 \$m2	Difference of matrices m1 and m2.
<mat3>	-mat3 \$m	Negative of matrix m.
<mat3>	mat3*mat3 \$m1 \$m2	Product of matrices m1 and m2.
<mat3>	mat3*s \$m \$s	Component-wise product of matrix m and scalar s.
<mat3>	s*mat3 \$s \$m	Component-wise product of matrix m and scalar s.
<vec3>	mat3*vec3 \$m \$v	Product of matrix m and vector v.
<mat3>	mat3/s \$m \$s	Component-wise quotient of matrix m and scalar s.
<mat3>	mat3=s \$s	Generates a matrix out of scalar s.
<mat3>	mat3=diag \$s1 \$s2 \$s3	Generates a diagonal matrix out of scalars s1, s2 and s3.
<real>	mat3trace \$m	Trace of matrix m.
<real>	mat3det \$m	Determinant of matrix m.
<mat3>	mat3trans \$m	Matrix m transposed.

- <mat2> **mat3subcyc** \$m \$i \$j Adjoint cyclic submatrix of matrix m with respect to component (i,j).
- <mat3> **mat3invert** \$m Inverse of matrix m.
- <vec3> **mat3dual** \$m Dual vector of matrix m.
- <quat> **mat3maptoquat** \$m Maps matrix m into a quaternion. See explanation in file “veclib_doc.pdf”.
- <vec3> **mat3rotangles** \$m \$i \$j \$k Calculate rotation angles of a rotation matrix m. See explanation in file “veclib_doc.pdf”.
- <mat3> **mat3rotmatrix** \$i \$j \$k \$wi \$wj \$wk Calculate rotation matrix out of angles wi, wj, wk. See explanation in file “veclib_doc.pdf”.
- <boole> **ismat3** \$a a is a list of three vec3 objects? 1:0.

mat4

- <mat4> **mat4*mat4** \$m1 \$m2 Product of matrices m1 and m2.

srt2

- <vec2> **srt2*vec2** \$l \$v Applies transformation l to vector v.
- <mat2> **srt2*mat2** \$l \$m Applies transformation l to matrix m.
- <srt2> **srt2*srt2** \$l1 \$l2 Product of transformations l1 and l2.
- <srt2> **srt2invert** \$l Inverse of transformation l.
- <mat3> **mat3=srt2** \$l Generate a 3x3-matrix representation out of transformation l.

srt3

- <vec3> **srt3*vec3** \$l \$v Applies transformation l to vector v.
- <mat3> **srt3*mat3** \$l \$m Applies transformation l to matrix m.
- <srt3> **srt3*srt3** \$l1 \$l2 Product of transformations l1 and l2.
- <srt3> **srt3invert** \$l Inverse of transformation l.
- <mat4> **mat4=srt3** \$l Generate a 4x4-matrix representation out of transformation l.