# *Chapter 1.*
# Scheme Extensions

Topic: Ignore

Scheme is a public domain programming language, based on the LISP language, that uses an interpreter to run commands. ACIS provides extensions (written in C++) to the native Scheme language that can be used by an application to interact with ACIS through its Scheme Interpreter. The C++ source files for ACIS Scheme extensions are provided with the product. *Spatial*'s Scheme based demonstration application, Scheme ACIS Interface Driver Extension (Scheme AIDE), also uses these Scheme extensions and the Scheme Interpreter. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

## gl:cull–mode

Scheme Extension: Rendering Control

Action: Gets the backface culling mode used when rendering entities.

Filename: gl/gl_scm/gl_scm.cxx

APIs: api_gl_get_cull_mode

Syntax: (**gl:cull–mode**)

Arg Types: None

Returns: string

Errors: None

Description: This extension sets the backface culling mode OpenGL uses when rendering entities. When backface culling is active, polygons whose normals point away from the eyepoint are not rendered. This is most obvious when the fill style is set to "edge" or "point". Normally you should use "auto" as GL renders faster if it can cull polys that face the wrong way.

The previous mode is returned. Culling modes are:

| | |
|---|---|
| auto | backface culling depends on the sidedness of entities and faces. Double–sided faces are not culled, single sided are. |
| always | cull back facing faces regardless of sidedness. |
| never | cull back facing faces regardless of sidedness. |

Limitations: Supported on NT platforms only.

Example:

```
; gl:cull–mode
; Get the backface culling mode and fill style
; to display edges hidden by forward faces.
(view:gl)
;; #[view 41943049]
(gl:fill–style)
;; ”filled”
(gl:cull–mode)
;; ”auto”
```

# gl:fill–style

Scheme Extension: Rendering Control

Action: Gets the fill style that OpenGL uses when rendering polygons.

Filename: gl/gl_scm/gl_scm.cxx

APIs: api_gl_get_fill_style

Syntax: (**gl:fill–style**)

Arg Types: api_gl_get_fill_style

Returns: string

Errors: None

Description: This extension gets the fill mode that OpenGL uses when rendering polygons.

Fill styles are:

| | |
|---|---|
| filled | renders polygons fully. |
| edge | renders only polygon edges. |
| point | renders only polygon vertices. |

Limitations: Supported on NT platforms only.

Example:
```
; gl:fill–style
; Get the polygon fill style.
(view:gl)
;; #[view 41943049]
; Create a sphere
(define sphere1(solid:sphere (position 0 0 0) 40))
;; sphere1
(gl:fill–style)
;; ”filled”
```

# gl:get–pixel–format

Scheme Extension: Rendering Control, OpenGL

Action: Prints the pixel format used for the GL view.

Filename: gl/gl_scm/gl_scm.cxx

APIs: None

Syntax: (**gl:get–pixel–format** [view])

Arg Types: view view

Returns: view

Errors: None

Description: The pixel format can be used to determine if hardware acceleration is being used.

Limitations: Supported on NT platforms only.

Example:
```
; gl:get–pixel–format
; Print the pixel format.
(define GL (view:gl))
; Create a cylinder.
(define cyl1
    (solid:cylinder (position 10 –15 0)
    (position 20 10 10) 22))
;; cyl1
; Output the cylinder to printer.
(gl:get–pixel–format GL)
;; GL view is using PixelFormat 4
;; ”gl_context”
```

# gl:metafile

Scheme Extension: Viewing

Action: Writes a view to a file in Windows Enhanced Metafile format.

Filename: gl/gl_scm/gl_scm.cxx

APIs: None

Syntax:
```
(gl:metafile [filename] [view] [emf–wmf=#t]
    [header=#f])
```

| Arg Types: | | |
|---|---|---|
| | filename | string |
| | view | view |
| | emf–wmf | boolean |
| | header | boolean |

Returns: view

Errors: None

Description: When running on the Windows NT platform, this extension writes a view in Windows Enhanced Metafile format to a file or to the clipboard and returns the specified view to the output filename. gl:metafile sends the image to the output file, while render:metafile renders the image sent to the output file.

The emf–wmf argument controls whether an old style metafile (wmf) is made or the new enhanced format (emf) is made. If an old style is specified (#t), the header argument is used to add some extra header information that is required from some Windows programs such as Word. The metafile type defaults to #t which is an enhanced metafile. Header type defaults to #f which does not add the extra header.

Limitations: Supported on NT platforms only.

Example:
```
; gl:metafile
; Define a view
(define view1 (view:gl))
;; view1
(define title1 (view:set–title "view1" view1))
;; title1
(define view2( view:gl))
;; view2
(define title2 (view:set–title "view2" view2))
;; title2
(define body1 (solid:subtract
```

```
    (solid:cylinder
    (position 0 0 –10) (position 0 0 10) 10)
    (solid:sphere (position 0 0 20) 20)
    (solid:cylinder
    (position 10 0 –10) (position 10 0 10) 5)
    (solid:cylinder
    (position –10 0 0) (position 10 0 0) 3)
    (solid:cylinder
    (position 0 –10 0) (position 0 10 0) 3)))
;; body1
; These are the metafile test cases.
; The following generates an enhanced metafile
(gl:metafile "c:/temp/gltest1.emf")
;; #[view 184420622]
; The following generates an enhanced metafile
; for view2
(gl:metafile "c:/temp/gltest2.emf" view2)
;; #[view 117705038]
; The following generates an enhanced metafile as
; specified from emf–wmf
(gl:metafile "c:/temp/gltest3.emf" #t)
;; #[view 184420622]
; The following generates an enhanced metafile
; for view2 as specified by the header argument.
(gl:metafile "c:/temp/gltest4.emf" view2 #t)
;; #[view 117705038]
; The following generates an old style metafile
; format without the header information
(gl:metafile "c:/temp/gltest5.wmf" #f)
;; #[view 184420622]
; The following generates an old style metafile
; format for view2 without the header information
; for view2
(gl:metafile "c:/temp/gltest6.wmf" view2 #f)
;; #[view 117705038]
; The following generates an old style metafile
; format for view2 with the extra header information
(gl:metafile "c:/temp/gltest7.wmf" #f #t)
;; #[view 184420622]
; The following generates an old style metafile
; format for view2 with the extra header information
(gl:metafile "c:/temp/gltest8.wmf" view2 #f #t)
;; #[view 117705038]
; The following generates an enhanced metafile
; that is sent to the clipboard (suggested format
```

```
; for clipboard transfers)
(gl:metafile)
;; #[view 184420622]
```

# gl:print

Scheme Extension: Viewing, OpenGL

Action: Prints a view.

Filename: gl/gl_scm/gl_scm.cxx

APIs: None

Syntax: (**gl:print** [view])

Arg Types: view view

Returns: view

Errors: None

Description: Change the printer setup information by selecting the Print Setup button. Modify page orientation and paper size to accommodate specific requirements. The image is sent to the specified printer and an image file is created and stored in the current directory; the default is PostScript. This extension also returns the view where the file was saved.

This extension requires Windows and a Personal Computer (PC). If the PC is not attached to a printer, a Print To File window automatically displays to specify an output filename.

Limitations: Supported on NT platforms only.

Example:

```
; gl:print
; Print a view in Windows.
(define GL (view:gl))
;; GL
; Create a cylinder.
(define cyl1
    (solid:cylinder (position 10 –15 0)
    (position 20 10 10) 22))
;; cyl1
; Output the cylinder to printer.
(gl:print GL)
;; #[view 223283532]
```

# gl:rebuild

Scheme Extension: Rendering Control, OpenGL

Action: Deletes the OpenGL display lists regenerated on the next refresh.

Filename: gl/gl_scm/gl_scm.cxx

APIs: None

Syntax: (**gl:rebuild**)

Arg Types: None

Returns: string

Errors: None

Description: This extension can be used if the OpenGL view does not display entities correctly. The string “Lists Deleted” is returned.

Limitations: Supported on NT platforms only.

Example:

```
; gl:rebuild
; Define a view
(define view1 (view:gl))
;; view1
; Create a cylinder.
(define cylinder1 (solid:cylinder
    (position 10 –15 0) (position 20 10 10) 22))
;; cylinder1
; Refresh the view.
(gl:rebuild)
;; “Lists Deleted”
```

# gl:set–cull–mode

Scheme Extension: Rendering Control, OpenGL

Action: Sets the backface culling mode used when rendering entities.

Filename: gl/gl_scm/gl_scm.cxx

APIs: api_gl_get_cull_mode, api_gl_set_cull_mode

Syntax: (**gl:set–cull–mode** mode)

Arg Types: mode string

Returns: string

Errors: None

Description: This extension sets the backface culling mode Microsoft OpenGL uses when rendering entities. When backface culling is active, polygons whose normals point away from the eyepoint are not rendered. This is most obvious when the fill style is set to “edge” or “point”. Normally you should use “auto” as GL renders faster if it can cull polys that face the wrong way. The previous mode is returned. Returns the string identifying the previous mode.

Culling modes are:

| | |
|---|---|
| auto ..... | backface culling depends on the sidedness of entities and faces. Double–sided faces are not culled, single sided are. |
| always ..... | cull back facing faces regardless of sidedness. |
| never ..... | cull back facing faces regardless of sidedness. |

Limitations: Supported on NT platforms only.

Example:

```
; gl:set–cull–mode
; Set the backface culling mode and fill style
; to display edges hidden by forward faces.
(view:gl)
;; #[view 41943049]
(define sphere1 (solid:sphere (position 0 0 0) 40))
;; sphere1
(gl:set–fill–style ”edge”)
;; ”edge”
(view:refresh)
;; #[view 115869988]
(gl:set–cull–mode ”never”)
;; ”never”
(entity:display sphere1)
;; #[entity 2 1]
```

# gl:set–fill–style

Scheme Extension: Rendering Control, OpenGL

Action: Sets the fill style that Microsoft OpenGL uses when rendering polygons.

Filename: gl/gl_scm/gl_scm.cxx

APIs: api_gl_get_fill_style, api_gl_set_fill_style

Syntax: (**gl:set-fill-style** style)

Arg Types:

| | |
|---|---|
| style | string |

Returns: string

Errors: api_gl_get_fill_style, api_gl_set_fill_style

Description: This extension sets the fill mode that Microsoft OpenGL uses when rendering polygons. The choices allow full rendering of polygons, edge rendering of polygons, or rendering of polygon vertices only. The previous style is returned.

Fill styles are:

| | |
|---|---|
| filled ..... | renders polygons fully. |
| edge ..... | renders only polygon edges. |
| point ..... | renders only polygon vertices. |

Limitations: Supported on NT platforms only.

Example:

```
; gl:set-fill-style
; Set the polygon fill style.
(view:gl)
;; #[view 41943049]
(define sphere1 (solid:sphere (position 0 0 0) 40))
;; sphere1
(gl:set-fill-style "edge")
;; "filled"
(view:refresh)
;; #[view 41943049]
(gl:set-fill-style "point")
;; "edge"
(view:refresh)
;; #[view 41943049]
```

# gl:view

Scheme Extension: Viewing, OpenGL

Action: Creates a Microsoft OpenGL view and attaches it to a gl_context.

Filename: gl/gl_scm/gl_scm.cxx

APIs: None

Syntax: `(gl:view [x–window y–window width height] [part])`

Arg Types:

| | |
|---|---|
| x–window | integer |
| y–window | integer |
| width | integer |
| height | integer |
| part | part |

Returns: view

Errors: None

Description: This extension creates a view and attaches it to a gl_context. The optional x–window specifies the *x* location of the upper left–hand corner of the view window on the screen. The optional y–window specifies the *y* location of the upper left–hand corner of the view window on the screen. The optional width specifies the horizontal size of the window in pixels. The optional height specifies the vertical size of the window in pixels. If no arguments are specified, a window is created based on the size of the display used. The default for the extension (gl:view) displays the view where the *y*–axis is up, the *x*–axis is to the right, and the *z*–axis is coming out from the screen.

Limitations: Supported on NT platforms only.

Example:

```
; gl:view
; Create a new OpenGL view.
(gl:view)
;; #[view 41943049]
(define view1 (gl:view 0 0 100 100))
;; view 1
```

# view:gl

Scheme Extension: Viewing, OpenGL

Action: Creates a Microsoft OpenGL view and attaches it to a Microsoft OpenGL context.

Filename: gl/gl_scm/gl_scm.cxx

APIs: None

Syntax: `(view:gl [x–window y–window width height] [part])`

| | | |
|---|---|---|
| Arg Types: | x–window | integer |
| | y–window | integer |
| | width | integer |
| | height | integer |
| | part | part |

Returns: view

Errors: None

Description: This extension creates a view and attaches it to a Microsoft OpenGL context. The optional x–window specifies the x location of the upper left–hand corner of the view window on the screen. The optional y–window specifies the y location of the upper left–hand corner of the view window on the screen. The optional width specifies the horizontal size of the window in pixels. The optional height specifies the vertical size of the window in pixels. If no arguments are specified, a window is created based on the size of the display used. The default for the extension (gl:view) displays the view where the y–axis is up, the x–axis is to the right, and the z–axis is coming out from the screen.

Limitations: Supported on NT platforms only.

Example:

```
; view:gl
; Create a new OpenGL view.
(view:gl)
;; #[view 41943049]
(define view1 (view:gl 0 0 100 100))
;; view 1
```