Chapter 3.
Scheme Extensions Ma thru Zz

Topic: Ignore

material

Scheme Extension: Materials

Action: Creates a material for rendering.

Filename: rbase/rnd_scm/mat_scm.cxx

APIs: api_rh_create_material

Syntax: (**material**)

Arg Types: None

Returns: material

Errors: None

Description: The default material is “plain”.

Limitations: None

Example:

```
; material
; Create a material.
(define mat1 (material))
;; mat1
; Create a solid block.
(define block1
    (solid:block (position 0 0 0)
    (position 15 10 5)))
;; block1
; Assign the material to the block.
(entity:set–material block1 mat1)
;; ()
; Remove the material attribute.
(entity:set–material block1 #f)
;; ()
```

material:color–props

Scheme Extension: Materials, Colors

Action: Gets the properties of a material's color component.

Filename: rbase/rnd_scm/mclr_scm.cxx

APIs: api_rh_get_color_comp

Syntax: (**material:color–props** material)

Arg Types: material material

Returns: ((string . real | integer | string | color) ...)

Errors: None

Description: This extension returns a list containing pairs of the property name and its present value.

When a new "plain" color type material is created, the default reflection type is "phong". It has the following properties:

"ambient factor" 1
"diffuse factor" 0.75
"specular factor" 0.5
"exponent" 10
"specular color" . . . . . #[color 1 1 1]

A high ambient factor causes the image to appear bright white in views regardless of the color that has been set. To reveal the color, reset the reflection properties to less extreme values. See material:set–color–prop for more details.

material specifies the color material entity.

Limitations: None

Example:

```
; material:color–props
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 30 10 3) 20))
;; cyl1
; Determine if the cylinder has a material.
(entity:material cyl1)
;; #f
; Create a material.
(define mat1 (material))
;; mat1
; Get the default material color type.
(material:color–type mat1)
;; “plain”
; Get the default material color properties.
(material:color–props mat1)
;; ((”color” . #[color 1 1 1]))
; Set the material.
(entity:set–material cyl1 mat1)
;; ()
; Set the color of the material.
(entity:set–material–color cyl1
    (color:rgb 1 1 0))
;; ()
; Get the color properties of the material.
(material:color–props mat1)
;; ((”color” . #[color 1 1 0]))
```

material:color–type

Scheme Extension: Materials, Colors

Action: Gets the type of a material’s color component.

Filename: rbase/rnd_scm/mclr_scm.cxx

APIs: api_rh_get_color_comp

Syntax: (**material:color–type** material)

Arg Types:	material	material

Returns: string

Errors: None

Description: This extension returns the type of color shader assigned to the material as a string.

material specifies the color material entity to be inquired.

Limitations: None

Example:

```
; material:color–type
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 30 10 3) 20))
;; cyl1
; Determine if the cylinder has a material.
(entity:material cyl1)
;; #f
; Create a material.
(define mat1 (material))
;; mat1
; Get the default material color type.
(material:color–type mat1)
;; “plain”
; Set the color type of the material.
(material:set–color–type mat1 ”wrapped polka”)
;; ()
; Get the color type of the material.
(material:color–type mat1)
;; ”wrapped polka”
```

material:color–types

Scheme Extension: Materials, Colors

Action: Gets the list of valid color component types.

Filename: rbase/rnd_scm/mclr_scm.cxx

APIs: api_rh_get_color_comp_list

Syntax: (**material:color–types**)

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns all valid color material types as a string list.

Limitations: None

Example:
```
; material:color-types
; Get a list of valid color shader types.
(material:color-types)
;; (”base” ”blue marble” ”chrome” ”cubes” ”plain”
;; ”marble” ”simple wood” ”solid clouds”
;; ”solid polka” ”wrapped brick” ”wrapped checker”
;; ”wrapped diagonal” ”wrapped grid” ”wrapped image”
;; ”wrapped polka” ”wrapped s stripe”
;; “wrapped t stripe” “wrapped textured brick”)
```

material:displacement–props

Scheme Extension: Materials, Displacement

Action: Gets the type of a material’s displacement component.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_get_displace_comp

Syntax: (**material:displacement–props** material)

Arg Types: material material

Returns: ((string . integer | real | color) ...)

Errors: None

Description: This extension returns a list containing pairs of the property name and its present value.

material specifies the displacement material entity.

Limitations: None

Example:
```
; material:displacement-props
; Create a material.
(define mat1 (material))
;; mat1
; Set the displacement type of the material.
(material:set-displacement-type mat1 ”casting”)
;; ()
; Get the displacement properties of the material.
(material:displacement-props mat1)
;; ((”scale” . 1) (”casting amplitude” . 0.05)
;; (”dented amplitude” . 0.5) (”dented scale” . 1)
;; (”dented threshold” . 0.8) (”detail” . 2))
```

material:displacement–status

Scheme Extension: Materials, Displacement

Action: Gets the on–off status of a material's displacement component.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_get_displace_status

Syntax: (**material:displacement–status** material)

Arg Types:	material	material

Returns: boolean

Errors: None

Description: Displacement status is only applicable to Advanced Rendering. Displacement properties cannot be displayed when using Basic Rendering.

material specifies the displacement material entity.

Limitations: None

Example:

```
; material:displacement–status
; Create a material.
(define mat1 (material))
;; mat1
; Set the displacement type of the material.
(material:set–displacement–type mat1 ”casting”)
;; ()
; Set the displacement status of the material.
(material:set–displacement–status mat1 #f)
;; ()
; Get the displacement status of the material;
(material:displacement–status mat1)
;; #f
```

material:displacement–type

Scheme Extension: Materials, Displacement

Action: Gets the type of a material's displacement component.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_get_displace_comp

Syntax: `(material:displacement-type material)`

Arg Types: material material

Returns: string

Errors: None

Description: This extension returns the type of displacement shader assigned to the material as a string.

material specifies the displacement material entity.

Limitations: None

Example:

```
; material:displacement-type
; Create a material.
(define mat1 (material))
;; mat1
; Get the displacement type of the material.
(material:displacement-type mat1)
;; "none"
; Set the displacement type of the material.
(material:set-displacement-type mat1 "casting")
;; ()
; Get the displacement type of the material.
(material:displacement-type mat1)
;; "casting"
```

material:displacement–types

Scheme Extension: Materials, Displacement

Action: Gets the list of valid displacement component types.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_get_displace_comp_list

Syntax: `(material:displacement-types )`

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns all valid displacement material types as a string list.

Limitations: None

Example:
```
; material:displacement–types
; Get the list of valid displacement shader types.
(material:displacement–types)
;; (”casting” ”none” ”rough” ”wrapped bump map”
;; ”wrapped dimple” ”wrapped knurl” ”wrapped rough”
;; ”wrapped tread plate”)
```

material:reflection–props

Scheme Extension: Materials, Reflectance

Action: Gets the properties of a material’s reflection component.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_get_reflect_comp

Syntax: (**material:reflection–props** material)

Arg Types: material material

Returns: ((string . integer | real | color) ...)

Errors: None

Description: This extension returns a list containing pairs of the property name and its present value.

material specifies the reflection material entity.

Limitations: None

Example:
```
; material:reflection–props
; Create a material.
(define mat1 (material))
;; mat1
; Get the default reflection shader properties.
(material:reflection–props mat1)
;; ((”ambient factor” . 1) (”diffuse factor” . 0.75)
;; (”specular factor” . 0.5) (”exponent” . 10)
;; (”specular color” . #[color 1 1 1]))
```

material:reflection–status

Scheme Extension: Materials, Reflectance

Action: Gets the status of a material’s reflection component.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_get_reflect_status

Syntax: `(material:reflection–status material)`

Arg Types:	material	material

Returns: boolean

Errors: None

Description: The extension returns #t for on and #f for off.

material specifies the reflection material entity to retrieve the status information.

Limitations: None

Example:

```
; material:reflection–status
; Create a material.
(define mat1 (material))
;; mat1
; Get the current reflection status.
(material:reflection–status mat1)
;; #f
; Set the reflection status off.
(material:set–reflection–status mat1 #f)
;; ()
(material:reflection–status mat1)
;; #f
```

material:reflection–type

Scheme Extension: Materials, Reflectance

Action: Gets the type of a material's reflection component.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_get_reflect_comp

Syntax: `(material:reflection–type material)`

Arg Types:	material	material

Returns: string

Errors: None

Description: This extension returns the type of reflection shader assigned to the material as a string.

material specifies the reflectance material entity.

Limitations: None

Example:
```
; material:reflection–type
; Create a material.
(define mat1 (material))
;; mat1
; Get the default reflection shader type.
(material:reflection–type mat1)
;; ”phong”
```

material:reflection–types

Scheme Extension: Materials, Reflectance

Action: Gets the list of valid reflection component types.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_get_reflect_comp_list

Syntax: (**material:reflection–types**)

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns all valid reflection material types as a string list.

Limitations: None

Example:
```
; material:reflection–types
; Get a list of valid reflection shader types.
(material:reflection–types)
;; (”chrome 2D” ”conductor” ”constant” ”dielectric”
;; ”environment” ”glass” ”matte” ”metal” ”mirror”
;; ”phong” ”plastic”)
```

material:set–color–prop

Scheme Extension: Materials, Colors

Action: Sets a property of a material’s color component.

Filename: rbase/rnd_scm/mclr_scm.cxx

APIs: api_rh_get_color_comp, api_rh_set_color_arg

Syntax: `(material:set-color-prop material name value)`

Arg Types:

material	entity
name	string
value	real

Returns: unspecified

Errors: None

Description: The following color properties and default values are available. Wrapping an image, such as a checkerboard, around an object provides a different appearance from creating an image from a solid block of alternating colored cubes.

Note *Only color property names and values associated with a specific material entity can be set. Nearly all of the rendering modes are only available with the* Advanced Rendering Component.

"base" copies the input color to the output color. The input color is available if colors are assigned to the vertices of faceted geometry. If colors are not available at vertices, the input color is black.

"blue marble" provides a blue marble appearance. The complexity of the pattern is controlled by the detail positive integer. The overall scale of the pattern is controlled by the scale positive real. The default values are:

scale . . . . . 1.0 (real)
detail . . . . . 3 (int)

"chrome" provides simple chrome–like reflections. The source has a base color, specified by the base color, which is mixed with bands of colors based on the orientation of the face–surface relative to a direction. The orientation of the face–surface is specified by the vector argument. The argument mix specifies the degree to which the colors are mixed: 0 corresponds to no base color while 1 corresponds to no reflection color. The default values are:

base color . . . . . #[color 1 1 1]
vector . . . . . #[gvector 0 1 0.5]
mix . . . . . 0.5 (real)

"cubes" corresponds to a 3D–lattice of cubes with alternating colors. It appears as if it were carved out of a block composed of congruent cubes, where each congruent cube has alternating colors. The size of the cubes is controlled by the size positive real, while the colors are controlled by the color arguments. The default values are:

size . . . . . 1.0 (real)
odd color . . . . . #[color 1 1 1] (white)
even color . . . . . #[color 0 0 0] (black)

“marble” provides a veined marble appearance. The complexity of the pattern is controlled by the detail argument. The scale argument controls the scale of the pattern. The ground color argument specifies the base color, while the vein color argument specifies color for the veins running through the marble. The marble has a crystalline structure where the argument grain size specifies the size of the crystals. The default values are:

scale 1.0
detail 4
ground color . . . . . (1.00, 0.75, 0.85)
vein color . . . . . (0.775, 0.63, 0.875)
vein contrast 0.1
grain 1.0
grain scale 0.1

“plain” provides a plain, uniform color to each face. This is the only material color type displayed when using the Basic Rendering Component. The default values are:

color . . . . . #[color 1 1 1] (white)

“simple wood” provides a simple wood pattern. The wood is based on having a tree trunk centered on a given axis with concentric rings of light and dark wood colors. The fuzziness of the boundaries between light and dark wood is specified using the noise argument. The size of the rings is determined by the argument scale. The default values are:

scale . . . . . 1.0 (real)
light wood color . . . . . #[color 1 0.965 0.8]
dark wood color . . . . . #[color 0.98 0.85 0.53]
point on axis . . . . . #[gvector 0 0 0]
axis direction . . . . . #[gvector1 0 0]
noise . . . . . 0.5 (real)

“solid clouds” provides an appearance as if carved out of a 3D arrangement of clouds. The complexity of the texture is controlled by detail. The default values are:

scale . . . . . 1.0 (real)
background color . . . . . #[color 0 0 1]
clouds color . . . . . #[color 1 1 1]
detail . . . . . 2 (int)

“solid polka” provides an appearance as if carved out of a 3D arrangement of spheres, resulting in a polka dot pattern. The argument radius specifies the size of the spheres, while the argument separation the distance between neighboring spheres. The sharpness between sphere boundaries is given by the argument edge softness, where 0 means sharp boundaries and 1 means a continuously blended color between spheres and background. The default values are:

scale . . . . . 1.0 (real)
separation . . . . . 1.0 (real)
radius . . . . . 0.4 (real)
edge softness . . . . . 0.1 (real)
background color . . . . . #[color 1 1 1]
spot color . . . . . #[color 1 0 0]

“wrapped brick” provides an appearance as if carved out of a block of simple bricks. The bricks have a uniform color and are separated by mortar of another color. If the material is applied to an entity with no associated texture space, a default texture space of type “z plane” is used, given with the extension texture–space:set–prop. The default values are:

scale . . . . . 1.0 (real)
brick width . . . . . 1.0 (real)
brick height . . . . . 0.5 (real)
mortar size . . . . . 0.1 (real)
brick color . . . . . #[color 1 0.58 0.19]
mortar color . . . . . #[color 0.58 0.58 0.58]

“wrapped checker” provides an appearance to the given face as if a checkerboard of alternating colors were wrapped around the object. (Wrapping a checkerboard around an object is different from solid cubes.) The default values are:

size . . . . . 1.0 (real)
odd color . . . . . #[color 1 1 1] (white)
even color . . . . . #[color 0 0 0] (black)

“wrapped diagonal” provides a diagonal stripe in texture–space. The stripe is applied to each face to which the material is applied. The width of the stripe is specified as a fraction of the pattern size, so it will be a real number between 0 and 1. The sharpness of the stripe edges is specified with the fuzz argument, where 0 is sharp and 1 is blended with the background color. If the material is applied to an entity with no associated texture space, a default texture space of type “z plane” is used, given with the extension texture–space:set–prop. The default values are:

size . . . . . 1.0 (real)
width . . . . . 0.2 (real)
background color . . . . . #[color 1 1 1]
stripe color . . . . . #[color 1 0 0]
fuzz . . . . . 0.1 (real)

“wrapped grid” provides a grid pattern in texture–space. It is applied to each face to which the material is applied. If the material is applied to an entity with no associated texture space, a default texture space of type “z plane” is used, given with the extension texture–space:set–prop. The default values are:

scale . . . . . 1.0 (real)
width . . . . . 0.8 (real)
height . . . . . 0.8 (real)
grid size . . . . . 0.2 (real)
grid color . . . . . #[color 1 0 0]
background color . . . . . #[color 1 1 1]

“wrapped image” provides image mapping. The image is either in .lwi or .bmp format. If the material is applied to an entity with no associated texture space, a default texture space of type “z plane” is used, given with the extension texture–space:set–prop. The default values are:

filename . . . . . string

“wrapped polka” provides a polka dot pattern. Applies pattern to each face to which material is applied. The texture corresponds to a grid of circles of one color against a background of another color. The distance between the centers of neighboring disks is specified by the separation property. The sharpness of the disk edges is controlled by the edge softness property: 0 corresponds to sharp edges and 1 corresponds to blend of color across edges. If the material is applied to an entity with no associated texture space, a default texture space of type “z plane” is used, given with the extension texture–space:set–prop. The default values are:

scale . . . . . 1.0 (real)
separation . . . . . 1.0 (real)
radius . . . . . 0.4 (real)
edge softness . . . . . 0.1 (real)
background color . . . . . #[color 1 1 1]
spot color . . . . . #[color 1 0 0]

“wrapped s stripe” provides a stripe as a line of constants in texture–space. The sharpness of the edges is controlled by the fuzz argument, where 0 is sharp and 1 is continuous color with background. The default values are:

size . . . . . 1.0 (real)
width . . . . . 0.2 (real)
background color . . . . . #[color 1 1 1]
stripe color . . . . . #[color 1 0 0]
fuzz . . . . . 0.1 (real)

"wrapped t stripe" provides a stripe as a line of constant t in texture–space. The sharpness of the edges is controlled by the fuzz argument, where 0 is sharp and 1 is continuous color with background. The default values are:

size . . . . . 1.0 (real)
width . . . . . 0.2 (real)
background color . . . . . #[color 1 1 1]
stripe color . . . . . #[color 1 0 0]
fuzz . . . . . 0.1 (real)

material specifies a color material entity.

name specifies the property name to attach to the material entity.

value specifies the value associated with the property name.

Limitations: None

Example:

```
; material:set-color-prop
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Create a material.
(define mat1 (material))
;; mat1
; Set the color shader type.
(material:set-color-type mat1 "solid polka")
;; ()
; Set the properties of the color shader type.
(material:set-color-prop mat1
    "background color" 4)
;; ()
; Assign the material to the cylinder.
(entity:set-material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set-mode "full")
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

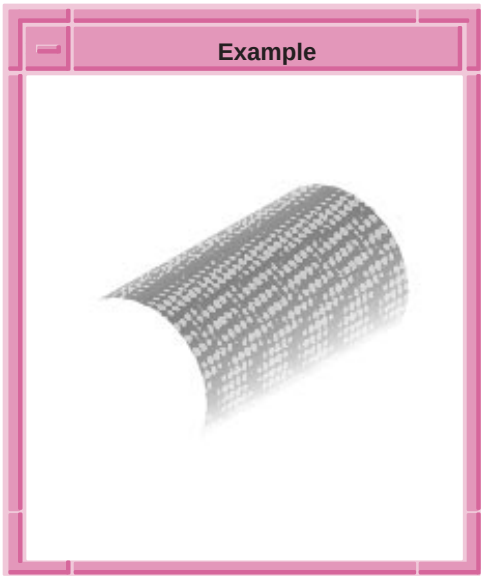

Figure 3-1. material:set–color–prop

material:set–color–type

Scheme Extension: Materials, Colors

Action: Sets the type of a material's color component.

Filename: rbase/rnd_scm/mclr_scm.cxx

APIs: api_rh_set_color_comp

Syntax: `(material:set-color-type material color-type)`

Arg Types:	material	material
	color–type	string

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies the color material entity.

color–type specifies the color type to set to the material entity. The available color types include "base", "blue marble", "chrome", "cubes", "marble", "plain", "simple wood", "solid clouds", "solid polka", "wrapped brick", "wrapped checker", "wrapped diagonal", "wrapped grid", "wrapped image", "wrapped polka", "wrapped s stripe", and "wrapped t stripe".

Limitations: None

Example:

```
; material:set–color–type
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Create a material.
(define mat1 (material))
;; mat1
; Set the color shader type.
(material:set–color–type mat1 ”solid polka”)
;; ()
; Set the properties of the color shader type.
(material:set–color–prop mat1
    ”background color” 5)
;; ()
; Assign the material to the cylinder.
(entity:set–material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set–mode ”full”)
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

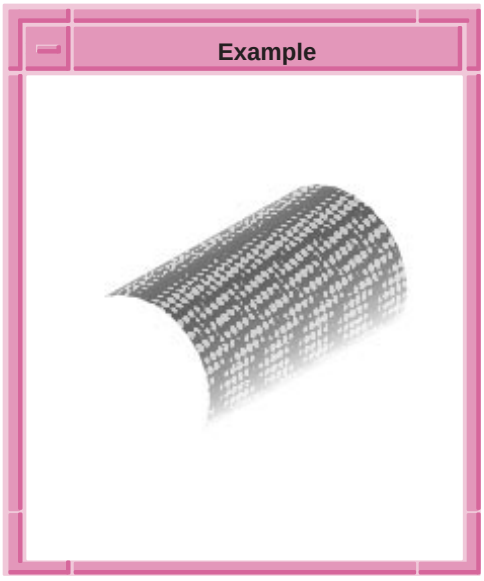

Figure 3-2. material:set–color–type

material:set–displacement–prop

Scheme Extension: Materials, Displacement

Action: Sets a property of a material's displacement component.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_get_displace_comp, api_rh_set_displace_arg

Syntax: `(material:set–displacement–prop material name value)`

Arg Types:		
	material	material
	name	string
	value	real

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies a displacement material entity.

name specifies the property name to attach to the material entity.

value specifies the value associated with the property name.

Note *Only displacement property names and values associated with a specific material entity can be set.*

"casting" provides an irregular casting pattern. This pattern derives from two superimposed noise functions providing displacements and indentations. The indentations scale the displacements. The default values are:

scale . . . . . 1.0 (real)
casting amplitude . . . . . 0.05 (real)
dented amplitude . . . . . 0.5 (real)
dented scale . . . . . 1.0 (real)
dented threshold . . . . . 0.8 (real)
detail . . . . . 2 (int)

"None" does not make any displacement.

"rough" provides a rough, cast metal–like finish. The appearance of the roughness is precisely controlled. Amplitude specifies the magnitude of the surface perturbations, expressed as a factor, typically in the range 0 to 1. The default values are:

scale . . . . . 1.0 (real)
amplitude . . . . . 0.1 (real)
detail . . . . . 3 (int)
sharpness . . . . . 1 (int)

"wrapped bump map" provides a regular dimpled appearance to a material. This displacement is considered a grid of spheres that protrudes above a material's surface. The default values are:

file name . . . . . string
scale . . . . . 1.0 (real)
amplitude . . . . . 0.1 (real))

"wrapped dimple" provides a regular dimpled appearance to a material. This displacement is considered a grid of spheres that protrudes above a material's surface. The default values are:

scale . . . . . 1.0 (real)
separation . . . . . 1.0 (real)
radius . . . . . 0.5 (real)
centre depth . . . . . 0.25 (real)
blend . . . . . 0.1 (real)

"wrapped rough" provides a rough, cast metal–like finish. The magnitude of the surface perturbations, expressed as a factor, typically, in the range 0 to 1. The default values are:

scale . . . . . 1.0 (real)
amplitude . . . . . 0.1 (real)
detail . . . . . 3 (int)
sharpness . . . . . 1 (int)

wrapped knurl provides a knurled pattern similar to grips used on tool handles. This displacement is considered a grid of indented pyramids on the material's surface. A blend is applied to the edges of the pyramids with the size defined by the value of blend, ranging from 0 to 1, where 0 denotes no blend and 1 denotes maximum blend. The default values are:

scale . . . . . 1.0 (real)
blend . . . . . 0.1 (real)
amplitude . . . . . 1.0 (real)

wrapped tread plate provides a regular tread plate pattern to a material. This displacement is considered as cylinders with rounded (spherical) ends that protrude above the material's surface. The default values are:

scale . . . . . 1.0 (real)
radius . . . . . 0.4 (real)
amplitude . . . . . 1.0 (real)
blend . . . . . 0.2 (real)

Limitations: None

Example:

```
; material:set-displacement-prop
; Create a material.
(define mat1 (material))
;; mat1
; Set the displacement type of the material.
(material:set-displacement-type mat1 "casting")
;; ()
; Set the properties of the displacement type.
(material:set-displacement-prop mat1
    "detail" 1)
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set-material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set-mode "full")
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

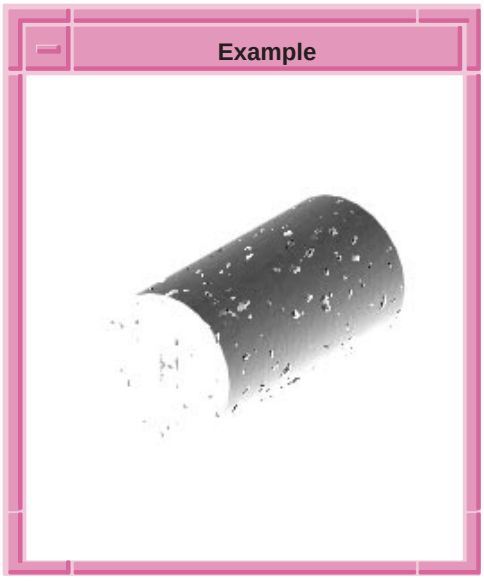

Figure 3-3. material:set–displacement–prop

material:set–displacement–status

Scheme Extension: Materials, Displacement

Action: Sets the on–off status of a material's displacement component.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_set_displace_status

Syntax: `(material:set-displacement-status material status)`

Arg Types:	material	material
	status	boolean

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies the displacement material entity.

status specifies either to enable (#t) or disable (#f) the displacement. When enabled, the argument's displacement properties are displayed in the rendered images of entities to which it is applied.

Limitations: None

Example:

```
; material:set-displacement-status
; Create a material.
(define mat1 (material))
;; mat1
; Set the displacement type of the material.
(material:set-displacement-type mat1 "casting")
;; ()
; Set the properties of the displacement type.
(material:set-displacement-prop mat1
    "detail" 1)
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set-material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set-mode "full")
;; ()
; Set the status of the displacement shader to off.
(material:set-displacement-status mat1 #f)
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

Example

Figure 3-4. material:set–displacement–status

material:set–displacement–type

Scheme Extension: Materials, Displacement

Action: Sets the type of a material's displacement component.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_set_displace_comp

Syntax: `(material:set-displacement-type material displacement-type)`

Arg Types:		
	displacement–type	string
	material	material

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies the displacement material entity.

displacement–type specifies the displacement type to be assigned to the material entity. The available displacement types include "casting", "None", "rough", "wrapped bump map", "wrapped dimple", and "wrapped rough".

Note *Displacement types can be set but not displayed.*

Limitations: None

Example:

```
; material:set–displacement–type
; Create a material.
(define mat1 (material))
;; mat1
; Set the displacement type of the material.
(material:set–displacement–type mat1 ”casting”)
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set–material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set–mode ”full”)
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

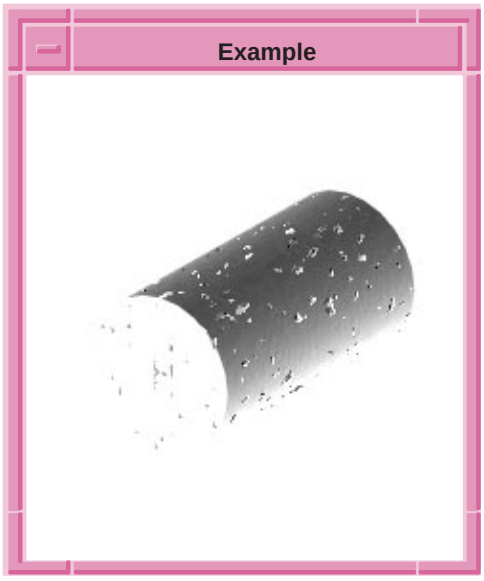

Figure 3-5. material:set–displacement–type

material:set–reflection–prop

Scheme Extension: Materials, Reflectance

Action: Sets a property of a material's reflection component.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_get_reflect_comp, api_rh_set_reflect_arg

Syntax: `(material:set-reflection-prop material name value)`

Arg Types:		
	material	material
	name	string
	value	real | color

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies a reflectance material entity.

name specifies the property name that attaches to the material entity. The following reflection properties and associated values are available.

Note *Only reflection property names and values associated with a specific material entity can be set.*

"chrome 2D" provides a chrome–like effect that generates a reflection pattern from a 2D array of colors. The map is projected onto surfaces using a spherical mapping. The default values are:

ambient factor . . . . . 1.0 (real)
diffuse factor . . . . . 0.5 (real)
specular factor . . . . . 0.5 (real)
chrome factor . . . . . 0.3 (real)
roughness . . . . . 0.1 (real)

"conductor" provides secondary mirrored views through ray tracing. Fresnel filtering is incorporated for accurate modeling of light interaction on the surfaces of objects. This model is appropriate for simulating metallic surface finishes accurately. The default values are:

ambient factor . . . . . 0.1 (real)
diffuse factor . . . . . 0.1 (real)
specular factor . . . . . 0.8 (real)
mirror factor . . . . . 0.8 (real)
roughness . . . . . 0.1 (real)
refraction red . . . . . 0.13764 (real)
refraction green . . . . . 0.128079 (real)
refraction blue . . . . . 0.156556 (real)
absorption red . . . . . 3.75633 (real)
absorption green . . . . . 4.37743 (real)
absorption blue . . . . . 2.3437 (real)

"constant" provides a constant color. This model ignores the effect of all light sources and gives the same result as an environment containing a single white ambient source with intensity 1.

"dielectric" provides an accurate simulation of dielectric (glass–like) materials that have both reflective and transmissive properties. Secondary mirrored and transmitted views are incorporated by ray tracing. Fresnel filtering is incorporated for the specular reflection, the mirrored view, and the transmitted view for accurate modeling of light interaction on the surfaces of objects. The default values are:

ambient factor . . . . . 0.0 (real)
diffuse factor . . . . . 0.0 (real)
specular factor . . . . . 0.9 (real)
transmission factor . . . . . 0.9 (real)
mirror factor . . . . . 0.9 (real)
roughness . . . . . 0.1 (real)
refraction . . . . . 1.59144 (real)

"environment" provides environment mapping. The default values are:

ambient factor . . . . . 1.0 (real)
diffuse factor . . . . . 0.5 (real)
specular factor . . . . . 0.5 (real)
environment factor . . . . . 0.3 (real)
roughness . . . . . 0.1 (real)
angle scale . . . . . 1.0 (real)

"glass" provides an approximation of glass–like materials that have both reflective and transmissive properties. Secondary mirrored and transmitted views are incorporated by ray tracing. Reflectance coefficients specify the amount of the specular light component, and the contribution made by light from the mirror and transmission directions. This model is appropriate for approximating glass surface finishes. The default values are:

specular factor . . . . . 0.5 (real)
transmission factor . . . . . 0.9 (real)
mirror factor . . . . . 0.1 (real)
roughness . . . . . 0.1 (real)
refraction . . . . . 1.59144 (real)

"matte" makes the given entity appear dull matte. This type of model is suitable for non–glossy surfaces, such as brick or fabric. The default values are:

ambient factor . . . . . 1.0 (real)
diffuse factor . . . . . 1.0 (real)

"metal" makes the given entity appear metallic. The default values are:

ambient factor . . . . . 1.0 (real)
specular factor . . . . . 1.0 (real)
roughness . . . . . 0.25 (real)

"mirror" supports secondary mirrored views through ray tracing. Reflectance coefficients specify the amount of ambient light, diffuse light diffuse factor, specular light, and the contribution made by light reflected in the mirror direction. This model is appropriate for representing mirror–like surface finishes. The default values are:

ambient factor . . . . . 0.1 (real)
diffuse factor . . . . . 0.1 (real)
specular factor . . . . . 0.8 (real)
mirror factor . . . . . 0.8 (real)
roughness . . . . . 0.0625 (real)

"phong" makes the given entity appear with phong–like reflections that are greatest in the mirror direction of the face–surface opposite the view direction with respect to the face normal. This model is most suitable for highly polished surfaces. The default values are:

ambient factor . . . . . 1.0 (real)
diffuse factor . . . . . 0.75 (real)
specular factor . . . . . 0.5 (real)
exponent . . . . . 10.0 (real)
specular color . . . . . #[color 1 1 1]

"plastic" makes the given entity a plastic appearance. This model has same type of behavior as the phong model. This model is most suitable for shiny or highly polished surfaces. The default values are:

ambient factor . . . . . 1.0 (real)
diffuse factor . . . . . 0.75 (real)
specular factor . . . . . 0.5 (real)
roughness . . . . . 0.1 (real)
specular color . . . . . #[color 1 1 1]

value specifies the value associated with the property name.

Limitations: None

Example:

```
; material:set-reflection-prop
; Create a material.
(define mat1 (material))
;; mat1
; Set the reflection shader type.
(material:set-reflection-type mat1 "mirror")
;; ()
; Set the property of the reflection shader type.
(material:set-reflection-prop mat1
    "roughness" 0.1)
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set-material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set-mode "full")
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

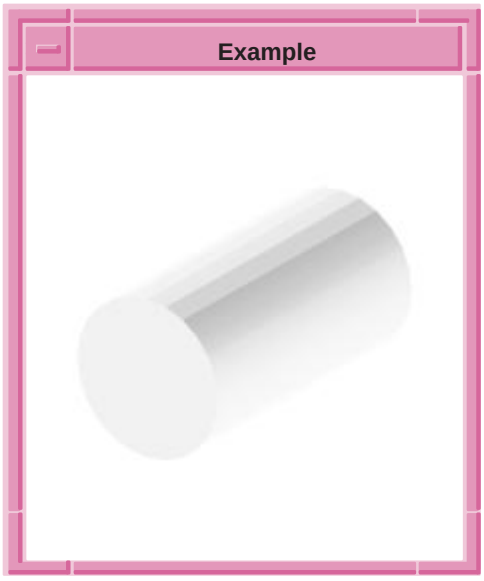

Figure 3-6. material:set–reflection–prop

material:set–reflection–status

Scheme Extension: Materials, Reflectance

Action: Sets the status of a material's reflection component.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_set_reflect_status

Syntax: `(material:set-reflection-status material status=#t)`

Arg Types:		
	material	material
	status	boolean

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies the reflection material entity.

status specifies enable (#t) or disable (#f) of the reflection. When enabled, it displays the material's reflection properties.

Limitations: None

Example:
```
; material:set-reflection-status
; Create a material.
(define mat1 (material))
;; mat1
; Set the reflection shader status on.
(material:set-reflection-status mat1 #t)
;; ()
```

material:set–reflection–type

Scheme Extension: Materials, Reflectance

Action: Sets the type of a material's reflection component.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_set_reflect_comp

Syntax: (**material:set-reflection-type** material reflection-type)

Arg Types:		
	material	material
	reflection–type	string

Returns: unspecified

Errors: None

Description: Refer to Action.

mat specifies the type of reflectance material entity.

reflection–type specifies the desired reflection material type. The following reflection types are available: "chrome 2D", "conductor", "constant", "dielectric", "environment", "glass", "matte", "metal", "mirror", "phong", and "plastic".

Note *Reflection types can be set but not displayed.*

Limitations: None

Example:

```
; material:set-reflection-type
; Create a material.
(define mat1 (material))
;; mat1
; Set the reflection shader type.
(material:set-reflection-type mat1 "mirror")
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set-material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set-mode "full")
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

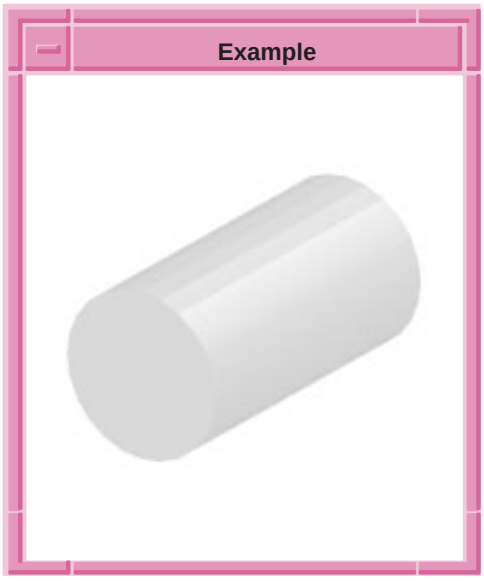

Figure 3-7. material:set–reflection–type

material:set–transparency–prop

Scheme Extension: Materials, Transparency

Action: Sets a property of a material's transparency component.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_get_transp_comp, api_rh_set_transp_arg

Syntax: `(material:set-transparency-prop material name value)`

Arg Types:		
	material	material
	name	string
	value	real | color

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies a transparency material entity.

name specifies the property name to attach to the material entity. The following property names are available.

Note *Only transparency property names and values associated with a specific material entity can be set.*

"base" copies the input transparency to the output transparency. The input transparency is available if transparencies are assigned to the vertices of polygonal geometry. If transparencies are not available at vertices, the input transparency is black, implying that the surface is opaque. It depends on data generated by the faceter.

"eroded" creates the illusion of erosion on a surface. The default values are:

scale . . . . . 1.0 (real)
coverage . . . . . 0.5 (real)
fuzz . . . . . 0.1 (real)

"None" No transparency (opaque).

"plain" provides a plain, uniform transparency. The transparency is specified as a color filter value by color. This simulates colored transparency and translucency, such as colored glass. The default values are:

color ..... #[color 1 1 1]

“wrapped grid” provides a grid pattern in texture–space. The grid appears opaque with holes visible between the grid lines. The default values are:

scale ..... 1.0 (real)
width ..... 0.8 (real)
height ..... 0.8 (real)
grid size ..... 0.2 (real)
transparency ..... 0.0 (real)

“wrapped image” provides image mapping for transparency shading. The name of the file containing the image data is provided as a string to filename. The default values are:

filename ..... string

value specifies the value associated with the property name.

Limitations: None

Example:

```
; material:set–transparency–prop
; Create a material.
(define mat1 (material))
;; mat1
; Set the reflection shader type.
(material:set–transparency–type mat1
    ”wrapped grid”)
;; ()
; Set the property of the reflection shader type.
(material:set–transparency–prop mat1
    ”grid size” 0.3)
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set–material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set–mode ”full”)
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

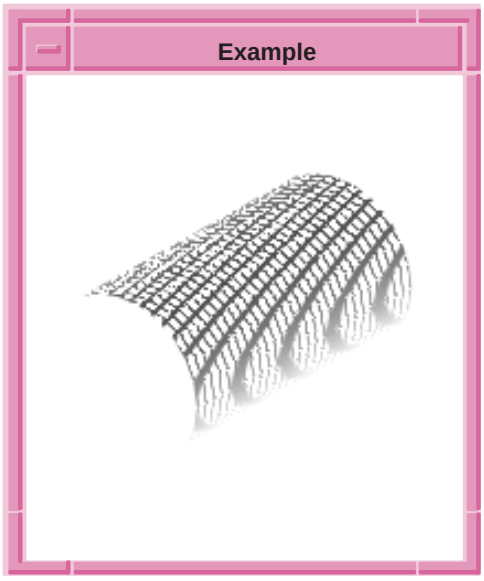

Figure 3-8. material:set–transparency–prop

material:set–transparency–status

Scheme Extension: Materials, Transparency

Action: Sets the status of a material's transparency component.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_set_transp_status

Syntax: `(material:set-transparency-status material status=#t)`

Arg Types:		
	material	material
	status	boolean

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies the transparency material entity.

status specifies either to enable (#t) or disable (#f) the transparency. When enabled, it displays the material's transparency properties.

Limitations: None

Example:
```
; material:set-transparency-status
; Create a material.
(define mat1 (material))
;; mat1
; Set the transparency shader status on.
(material:set-transparency-status mat1 #t)
;; ()
```

material:set–transparency–type

Scheme Extension: Materials, Transparency

Action: Sets the type of a material's transparency component.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_set_transp_comp

Syntax: (**material:set-transparency-type** material transparency–type)

Arg Types:		
	material	material
	transparency–type	string

Returns: unspecified

Errors: None

Description: Refer to Action.

material specifies the transparency material entity.

transparency–type specifies the desired transparency material type. The available transparency types are: "base", "eroded", "None", "plain", "wrapped grid", and "wrapped image".

Limitations: None

Example:

```
; material:set-transparency-type
; Create a material.
(define mat1 (material))
;; mat1
; Set the reflection shader type.
(material:set-transparency-type mat1
    "wrapped grid")
;; ()
; Create a solid cylinder.
(define cyl1
    (solid:cylinder (position 0 0 0)
    (position 5 30 0) 8))
;; cyl1
; Assign the material to the cylinder.
(entity:set-material cyl1 mat1)
;; ()
; Set the appropriate render mode.
(render:set-mode "full")
;; ()
; Render and view the results.
(render)
;; ()
; OUTPUT Example
```

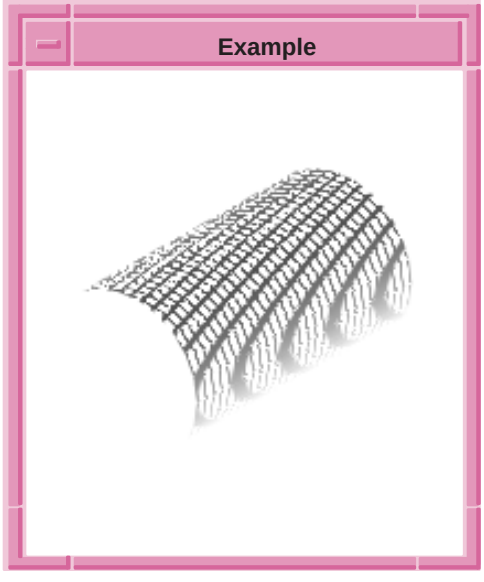

Figure 3-9. material:set–transparency–type

material:transparency–props

Scheme Extension: Materials, Transparency

Action: Gets the properties of a material's transparency component.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_get_transp_comp

Syntax: (**material:transparency–props** material)

Arg Types: material material

Returns: ((string . integer | real | color) ...)

Errors: None

Description: This extension returns a list containing pairs of the property name and its present value.

material specifies the transparency material entity.

Limitations: None

Example:

```
; material:transparency–props
; Create a material.
(define mat1 (material))
;; mat1
; Set the type of transparency for the material.
(material:set–transparency–type mat1
    ″wrapped grid″)
;; ()
; Get the properties of the transparency type.
(material:transparency–props mat1)
;; ((″scale″ . 1) (″width″ . 0.8) (″height″ . 0.8)
;; (″grid size″ . 0.2) (″transparency″ . 0))
```

material:transparency–status

Scheme Extension: Materials, Transparency

Action: Gets the status of a material's transparency component.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_get_transp_status

Syntax: (**material:transparency–status** material)

Arg Types: material material

Returns: boolean

Errors: None

Description: The extension returns #t for on and #f for off.

material specifies the transparency material entity to retrieve the status information.

Limitations: None

Example:

```
; material:transparency–status
; Create a material.
(define mat1 (material))
;; mat1
; Get the current transparency status.
(material:transparency–status mat1)
;; #f
; Set the transparency status off.
(material:set–transparency–status mat1 #f)
;; ()
(material:transparency–status mat1)
;; #f
```

material:transparency–type

Scheme Extension: Materials, Transparency

Action: Gets the type of a material’s transparency component.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_get_transp_comp

Syntax: (**material:transparency–type** material)

Arg Types: material material

Returns: string

Errors: None

Description: This extension returns the type of transparency shader assigned to the material as a string.

material specifies the transparency material entity.

Limitations: None

Example:
```
; material:transparency–type
; Create a material.
(define mat1 (material))
;; mat1
; Get the default transparency type.
(material:transparency–type mat1)
;; ″none″
```

material:transparency–types

Scheme Extension: Materials, Transparency

Action: Gets a list of valid transparency component types.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_get_transp_comp_list

Syntax: (**material:transparency–types**)

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns all valid transparency material types as a string list.

Limitations: None

Example:
```
; material:transparency–types
; Get a list of all valid transparency shaders.
(material:transparency–types)
;; (″base″ ″eroded″ ″none″ ″plain″ ″wrapped grid″
;; ″wrapped image″)
```

material?

Scheme Extension: Materials

Action: Determines if a Scheme object is a material.

Filename: rbase/rnd_scm/mat_scm.cxx

APIs: None

Syntax: `(material? object)`

Arg Types: object — scheme–object

Returns: boolean

Errors: None

Description: This extension returns #t if the object is a material; otherwise, it returns #f.

object specifies the scheme–object that has to be queried for a material.

Limitations: None

Example:

```
; material?
; Create a material.
(define mat1 (material))
;; mat1
; Determine if the object is actually a material.
(material? mat1)
;; #t
```

render

Scheme Extension: Rendering Control, Viewing

Action: Displays an entity, a list of entities, or entities in specified view as shaded solids.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax: `(render [entity–list=all] [view=active])`

Arg Types: entity–list — entity | (entity ...)
view — view

Returns: unspecified

Errors: None

Description: If any of the entities are not already faceted, facets are applied according to refinement properties.

Rendered images are displayed against a white background unless specified otherwise by render:set–background. Light entities established by light:list affect how the rendering looks. If no lights are specified, then a default eye (camera) light is used. render:mode returns the current rendering mode.

entity–list specifies an entity or list of entities to render. If entity–list is not specified, all entities are rendered in the active view.

view specifies the view in which to render the entities; the default is the active view.

Limitations: None

Example:

```
; render
; Create a solid block.
(define block1
    (solid:block (position –30 –30 –30)
    (position 0 0 0)))
;; block1
; Create a solid sphere.
(define sphere1 (solid:sphere (position 5 5 5) 25))
;; sphere1
; Render the block and the sphere.
(render)
;; ()
; OUTPUT Example
```

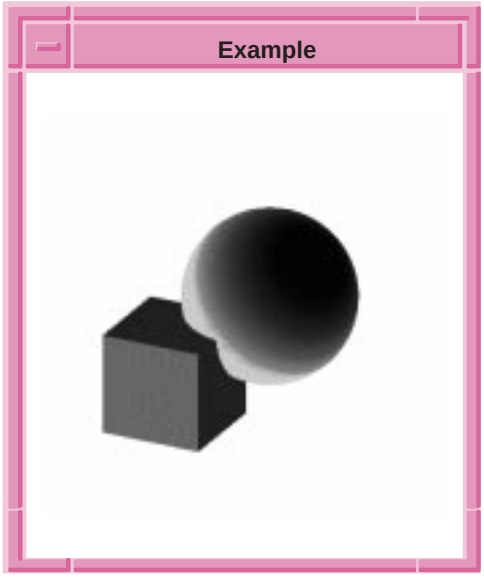

Figure 3-10. render

render:background

Scheme Extension: Backgrounds and Foregrounds, Rendering Control

Action: Gets the current background.

Filename: rbase/rnd_scm/bkgd_scm.cxx

APIs: api_rh_get_background

Syntax: (**render:background**)

Arg Types: None

Returns: background | boolean

Errors: None

Description: This extension returns the current background. If there is no current background active, it returns boolean #f.

Limitations: None

Example:

```
; render:background
; Create a solid block.
(define block1
     (solid:block (position –30 –30 –30)
    (position 0 0 0)))
;; block1
; Create a ”clouds” background.
(define back1 (background ”clouds”))
;; back1
; Set the background.
(render:set–background back1)
;; ()
; Get the current background.
(render:background)
;; #[entity 3 1]
(render)
;; ()
; OUTPUT Example
```

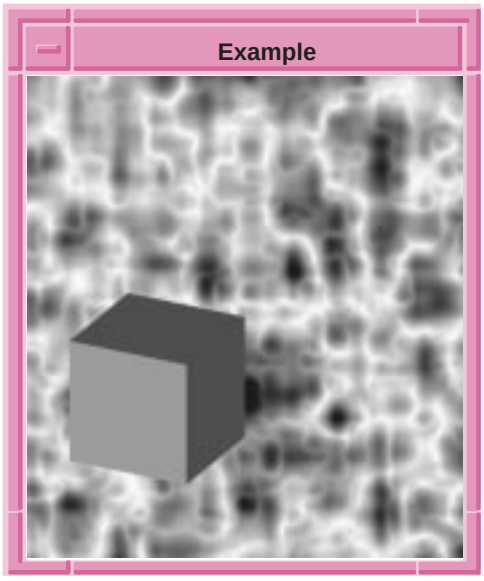

Figure 3-11. render:background

render:control–variable

Scheme Extension: Rendering Control

Action: Gets the value of a rendering control variable.

Filename: rbase/rnd_scm/avh_scm.cxx

APIs: api_rh_get_control_variable

Syntax: `(render:control–variable name)`

Arg Types:	name	string

Returns: integer | real

Errors: None

Description: Refer to Action.

name specifies the name of the control variable. Valid names for the control variables are pixel threshold, min pixel contrib, raytrace level, octtree occupancy, and octtree depth. The control variables are set using render:set–control–variable.

Limitations: None

Example:
```
; render:control–variable
; Get a list of valid rendering control variables.
(render:control–variables)
;; (″pixel threshold″ ″min pixel contrib″
;; ″raytrace level″ ″octtree occupancy″
;; ″octtree depth″)
; Get the value of a rendering control variable.
(render:control–variable ″pixel threshold″)
;; 0.1
```

render:control–variables

Scheme Extension: Rendering Control

Action: Gets a list of valid rendering control variables.

Filename: rbase/rnd_scm/avh_scm.cxx

APIs: None

Syntax: (**render:control–variables**)

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns a list of all valid control variable names.

Limitations: None

Example:
```
; render:control–variables
; Get a list of valid rendering control variables.
(render:control–variables)
;; (″pixel threshold″ ″min pixel contrib″
;; ″raytrace level″ ″octtree occupancy″
;; ″octtree depth″)
```

render:edge–width

Scheme Extension: Rendering Control

Action: Gets the edge width to use when building “edges” for display in the solid renderers.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax: (**render:edge–width** [view=active])

Arg Types: view view

Returns: unspecified

Errors: None

Description: Retrieves the current setting of the rendering edge width. The solid interactive renderers (OpenGL, etc.) need to have a width value to use when building display data for edges. The view "line width" is not used, because it cannot be set on a view–by–view basis in these renderers, and a view may not be available when the data is created.

If the width is not specified, the value 2.0 is used. If the view is not specified, the active view is used.

After the render:edge–width command is given, perform an entity:display on each entity to see the results of the changes.

view specifies the view to use. If view is not specified active view is taken.

Limitations: None

Example:
```
; render:edge–width
; Get the present edge width.
(render:edge–width)
;; 1
```

render:environment–map

Scheme Extension: Rendering Control, Environment Maps

Action: Gets the active environment map used for rendering reflections.

Filename: rbase/rnd_scm/emap_scm.cxx

APIs: api_rh_get_environment_map

Syntax: (**render:environment–map**)

Arg Types: None

Returns: environment–map

Errors: None

Description: This extension returns the currently active environment map; otherwise, it returns #f if no environment map is currently active.

An environment map is used for special rendering features, such as reflections. An environment map is not supported in Basic Rendering Component. In Advanced Rendering Component, the render mode should be “full” or better.

Limitations: None

Example:

```
; render:environment–map
; Create a procedurally defined environment map.
(define emap1 (environment–map:stripe 3 4 5))
;; emap1
; Set the active environment map.
(render:set–environment–map emap1)
;; ()
; Get the active environment map.
(render:environment–map)
;; #[entity 2 1]
```

render:foreground

Scheme Extension: Backgrounds and Foregrounds, Rendering Control

Action: Gets the current foreground.

Filename: rbase/rnd_scm/frgd_scm.cxx

APIs: api_rh_get_foreground

Syntax: (**render:foreground**)

Arg Types: None

Returns: foreground

Errors: None

Description: This extension returns the current foreground. foreground is an entity that specifies a foreground shader. These shaders filter and transform the color of a model’s pixels. The available foreground types established through the Scheme extension foreground:set–prop are “none”, “depth cue”, “fog”.

Limitations: None

Example:
```
; render:foreground
; Get a list of valid foreground types.
(foreground:types)
;; (”fog” ”depth cue” ”none”)
; Create a ”none” foreground.
(define fore1 (foreground ”none”))
;; fore1
; Set the foreground.
(render:set–foreground fore1)
;; ()
; Get the current foreground.
(render:foreground)
;; #[entity 2 1]
```

render:interleaf

Scheme Extension: Image Output, Rendering Control

Action: Renders the view or entities to an Interleaf file.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax:
```
(render:interleaf [entity–list=all]
    [filename=plotfile.il] [x–size=215.9mm
    y–size=279.4mm] [view=active])
```

Arg Types:		
	entity–list	entity | (entity ...)
	filename	string
	x–size	real
	y–size	real
	view	view

Returns: string

Errors: None

Description: This extension returns the filename. The default values are:

entity–list	all displayed entities
filename	plotfile.il
x–size	215.9mm (8.5 in)
y–size	279.4mm (11.0 in)
view	active

dl:interleaf sends the wireframe image to the output file, while render:interleaf renders the image sent to the output file.

entity–list specifies an entity or list of entities to be included in the render file.

filename specifies the name of the file to send the images; the default is plotfile.il.

x–size and y–size specify the *x*–coordinate and *y*–coordinate in pixels.

view specifies which view to print.

Limitations: None

Example:

```
; render:interleaf
; Create a solid cylinder.
(define cyl1 (solid:cylinder (position –10 –10 –10)
    (position 30 30 30) 20))
;; cyl1
; Render the cylinder.
(render cyl1)
;; ()
; Render the cylinder to a file
; in Interleaf format.
(render:interleaf cyl1 ”cyl1” 100 100)
;; ”cyl1”
```

render:metafile

Scheme Extension: Image Output, Rendering Control

Action: Renders a view or entities in Windows Enhanced Metafile format to a file or to the clipboard.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax:

```
(render:metafile [filename=plotfile.il]
    [entity–list=all] [view=active])
```

Arg Types:		
	filename	string
	entity–list	entity | (entity ...)
	view	view

Returns: view

Errors: None

Description: This extension renders a view in Windows Enhanced Metafile format to a file or to the clipboard.

dl:metafile sends the wireframe image to the output file, while render:metafile renders the image sent to the output file.

filename specifies the name of the file containing the image data. If no filename is specified, the image copies to the clipboard.

entity–list comprises any entity or list of entities included in the metafile; the default is all entities.

view specifies the view to be included in the metafile; the default is the active view.

Limitations: NT platforms only.

Example:

```
; render:metafile
; Create a solid cylinder.
(define cyl1 (solid:cylinder (position –10 –10 –10)
    (position 30 30 30) 20))
;; cyl1
; Render the cylinder.
(render cyl1)
; Render the cylinder to a file in Windows
; enhanced metafile format.
(render:metafile ”cyl1” cyl1)
;; ()
```

render:mode

Scheme Extension: Rendering Control

Action: Gets the current rendering mode.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: api_rh_get_render_mode

Syntax: (**render:mode**)

Arg Types: None

Returns: string

Errors: None

Description: This extension returns the current mode of rendering as a string. Rendering modes are set using render:set–mode.

Limitations: None

Example:
```
; render:mode
; Get a list of valid rendering modes.
(render:modes)
;; (”lambert” ”raytrace–full” ”raytrace–preview”
;; ”full” ”preview” ”phong” ”gouraud” ”flat”)
; Get the current rendering mode.
(render:mode)
;; ”gouraud”
```

render:modes

Scheme Extension: Rendering Control

Action: Gets a list of valid render modes.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax: (**render:modes**)

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns all valid rendering modes as a list of strings. Rendering modes are set using the extension render:set–mode.

Limitations: None

Example:
```
; render:modes
; Get a list of valid rendering modes.
(render:modes)
;; (”lambert” ”raytrace–full” ”raytrace–preview”
;; ”full” ”preview” ”phong” ”gouraud” ”flat”)
```

render:pnm

Scheme Extension: Image Output, Rendering Control

Action: Renders the view or entities to a PNM (Portable Any Map) file.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax:
```
(render:pnm [entity–list=all]
    [filename=plotfile.pnm]
    [binary=#t [color=#t]] [view=active])
```

Arg Types:		
	entity–list	entity | (entity ...)
	filename	string
	binary	boolean
	color	boolean
	view	view

Returns: string

Errors: None

Description: This extension produces a file containing a rendered image in PNM or PPM format, which is a neutral raster image format used by the PBMPLUS image conversion utilities. This extension returns the filename.

entity–list specifies the entities to be rendered to produce the image; the default is to display all entities.

filename specifies the name of the file to be produced; the default is plotfile.pnm.

binary argument indicates whether to produce PNM data in binary format (#t) or ASCII format (#f); the default is #t.

color argument controls color output; The default, #t, produces color PPM output; #f produces PGM greyscale output.

view specifies the view from which orientation, clipping, and resolution information is obtained; the default is the active view.

Limitations: None

Example:
```
; render:pnm
; Create a solid cylinder.
(define cyl1 (solid:cylinder (position –10 –15 –10)
    (position 10 20 30) 10))
;; cyl1
; Render the cylinder.
(render cyl1)
;; ()
; Render the cylinder to a file in PNM format.
(render:pnm ”image1.pnm”)
;; ”image1.pnm”
```

render:postscript

Scheme Extension: Image Output, Rendering Control

Action: Renders the view or entities to a PostScript file.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax:
```
(render:postscript [entity–list=all]
    [filename=plotfile.ps] [color=#f]
    [x–size=195.9mm y–size=259.4mm] [view=active])
```

Arg Types:		
	entity–list	entity | (entity ...)
	filename	string
	color	boolean
	x–size	real
	y–size	real
	view	view

Returns: string

Errors: None

Description: This extension returns the filename.

The default values are:

entity–list	all displayed entities
filename	plotfile.ps
color	#f (black on white)
x–size	195.9mm (7.75 in)
y–size	259.4mm (10.25 in)
view	active

dl:postscript sends the wireframe image to the output file, while render:postscript renders the image sent to the output file.

entity–list specifies an entity or list of entities to be included in the render file.

filename specifies the filename where images are sent; the default filename is plotfile.ps.

color specifies a logical (#t or #f) whether the color is sent to the file.

x–size and y–size specify the dimensions in millimeters of a rectangle into which you want the printed image of the viewport to fit as tightly as possible without distortion.

view specifies the view to print.

Limitations: None

Example:
```
; render:postscript
; Create a solid cylinder.
(define cyl1 (solid:cylinder (position –20 –25 –15)
    (position 25 35 20) 17.5))
;; cyl1
; Render the cylinder.
(render cyl1)
;; ()
; Render the cylinder to a file
; in PostScript format.
(render:postscript cyl1 ”cyl1” 100 100)
;; ”cyl1”
```

render:print

Scheme Extension: Rendering Control, Viewing

Action: Renders the view or entities to the Windows system printer.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax:
```
(render:print [entity–list=all] [x–resolution
    y–resolution] [view=active])
```

Arg Types:		
	entity–list	entity | (entity ...)
	x–resolution	integer
	y–resolution	integer
	view	view

Returns: view

Errors: None

Description: This extension prints a rendered image to the Windows system printer and displays a Print window requesting printer information.

dl:print sends the wireframe image to the output file, while render:print renders the image sent to the output file.

entity–list comprises any entity or list of entities included in the file; the default is all entities.

x–resolution specifies the number of dots per inch in the *x*–direction based on the type of printer being used.

y–resolution specifies the number of dots per inch in the *y*–direction based on the type of printer being used. Default resolutions are taken from the view.

view specifies the view to be included in the file; the default is the active view.

Limitations: NT platforms only.

Example:

```
; render:print
;; Create a solid cylinder.
(define cyl1 (solid:cylinder (position 10 –15 0)
    (position 20 10 10) 22))
;; cyl1
; Render the cylinder to Windows printing system.
(render:print (entity 1))
;; ()
```

render:set–background

Scheme Extension: Backgrounds and Foregrounds

Action: Sets the current background.

Filename: rbase/rnd_scm/bkgd_scm.cxx

APIs: api_rh_set_background

Syntax: (**render:set–background** background)

Arg Types:	background	background | boolean

Returns: unspecified

Errors: None

Description: Refer to Action.

background specifies a background entity. If boolean #f is specified, the system is returned to its initial state of no active background.

Limitations: None

Example:

```
; render:set-background
; Create a solid block.
(define block1
    (solid:block (position -30 -30 -30)
    (position 0 0 0)))
;; block1
; Create a "clouds" background.
(define back1 (background "clouds"))
;; back1
; Set the background.
(render:set-background back1)
;; ()
; Get the current background.
(render:background)
;; #[entity 3 1]
(render)
;; ()
; OUTPUT Example
```

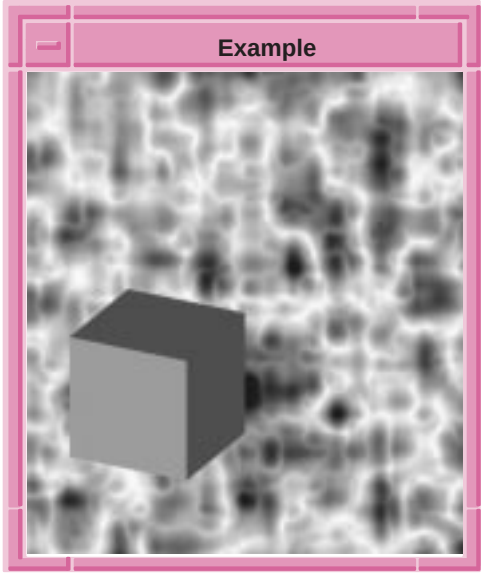

Figure 3-12. render:set–background

render:set–control–variable

Scheme Extension: Rendering Control, Rendering Control

Action: Sets the value of a rendering control variable.

Filename: rbase/rnd_scm/avh_scm.cxx

APIs: api_rh_set_control_variable

Syntax: `(render:set-control-variable name value)`

Arg Types:

name	string
value	real | integer

Returns: unspecified

Errors: None

Description: Refer to Action.

name specifies the name of the control variable.

value specifies the new value associated with the named control variable.

Valid names and value ranges follow:

pixel threshold controls anti–aliasing. It determines when to adaptively super–sample the image. The image is sampled until adjacent color samples differ in the largest of their red, green and blue components by an amount not exceeding the pixel threshold. Range is 0.0 to 1.0. The default is 0.1.

min pixel contrib is used for determining whether the contribution made by a secondary ray can be ignored. A combination of this control variable and the level variable define the recursive ray tracing termination criteria. The range is 0.0 to 1.0. The default is 0.05.

raytrace level is the maximum level of recursion that is enforced when tracing secondary rays. The default is 16.

octtree occupancy is the maximum occupancy of a leaf cell of the octtree data structure constructed for ray tracing. The occupancy corresponds to the number of facets contained in an octtree leaf cell. An octtree is constructed when rendering using ray tracing, either in a ray tracing mode, or when ray tracing is required by a shader in the preview or full rendering modes. The default is 8.

octtree depth is the maximum depth of the octtree data structure constructed during ray tracing. An octtree is constructed when rendering using ray tracing, either in a ray tracing mode, or when ray tracing is required by a shader in preview or full rendering modes. The default is 8.

Limitations: None

Example:

```
; render:set-control-variable
; Get a valid list of rendering control variables.
(render:control-variables)
;; ("pixel threshold" "min pixel contrib"
;; "raytrace level" "octtree occupancy"
;; "octtree depth")
; Set the value of a rendering control variable.
(render:set-control-variable "raytrace level" 1.0)
;; ()
```

render:set–default–color–type

Scheme Extension: Rendering Control, Colors

Action: Sets the default color type for newly created materials.

Filename: rbase/rnd_scm/mclr_scm.cxx

APIs: api_rh_def_color_comp

Syntax: (**render:set-default-color-type** color-type)

Arg Types: color–type string

Returns: unspecified

Errors: None

Description: Refer to Action.

color–type specifies the color material type to set to default status. The available color types include "base", "blue marble", "chrome", "cubes", "marble", "plain", "simple wood", "solid clouds", "solid polka", "wrapped brick", "wrapped checker", "wrapped diagonal", "wrapped grid", "wrapped image", "wrapped polka", "wrapped s stripe", and "wrapped t stripe".

Limitations: None

Example:

```
; render:set-default-color-type
; Set the default color shader type.
(render:set-default-color-type "blue marble")
;; ()
```

render:set–default–displacement–type

Scheme Extension: Displacement, Rendering Control

Action: Sets the default displacement type for newly created materials.

Filename: rbase/rnd_scm/bump_scm.cxx

APIs: api_rh_def_displace_comp

Syntax: `(render:set–default–displacement–type displacement–type)`

Arg Types: displacement–type string

Returns: unspecified

Errors: None

Description: Refer to Action.

displacement–type specifies the default displacement material type. The available displacement types include “casting”, “None”, “rough”, “wrapped bump map”, “wrapped dimple”, and “wrapped rough”.

Limitations: None

Example:
```
; render:set–default–displacement–type
; Set the default displacement shader type.
(render:set–default–displacement–type ”casting”)
;; ()
```

render:set–default–reflection–type

Scheme Extension: Reflectance, Rendering Control

Action: Sets the default reflection type for newly created materials.

Filename: rbase/rnd_scm/rflt_scm.cxx

APIs: api_rh_def_reflect_comp

Syntax: `(render:set–default–reflection–type reflection–type)`

Arg Types: reflection–type string

Returns: unspecified

Errors: None

Description: Refer to Action.

reflection–type specifies the default reflection material type. Available reflection types include “chrome 2D”, “conductor”, “constant”, “dielectric”, “environment”, “glass”, “matte”, “metal”, “mirror”, “phong”, and “plastic”.

Limitations: None

Example:
```
; render:set-default-reflection-type
; Set the default reflection shader type.
(render:set-default-reflection-type "glass")
;; ()
```

render:set–default–transparency–type

Scheme Extension: Transparency, Rendering Control

Action: Sets the default transparency type for newly created materials.

Filename: rbase/rnd_scm/trsp_scm.cxx

APIs: api_rh_def_transp_comp

Syntax: (**render:set-default-transparency-type** transparency-type)

Arg Types:

transparency–type	string

Returns: unspecified

Errors: None

Description: Refer to Action.

transparency–type specifies the transparency material type to be set to the default. Available transparency types include "base", "eroded", "None", "plain", "wrapped grid", and "wrapped image".

Limitations: None

Example:
```
; render:set-default-transparency-type
; Set the default transparency shader type.
(render:set-default-transparency-type "base")
;; ()
```

render:set–edge–width

Scheme Extension: Rendering Control

Action: Sets the edge width to use when building "edges" for display in the solid renderers.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax: `(render:set–edge–width [width] [view=active])`

Arg Types:		
	width	integer
	view	view

Returns: unspecified

Errors: None

Description: The solid interactive renderers (OpenGL, etc.) need to have a width value to use when building display data for edges. The view "line width" is not used, because it cannot be set on a view by view basis in these renderers, and a view may not be available when the data is created.

After the render:edge–width command is given, perform an entity:display on each entity to see the results of the changes.

width specifies the edge width. If the width is not specified, the value 2.0 is used.

view specifies the view to use. If the view is not specified, the active view is used.

Limitations: None

Example:

```
; render:set–edge–width
; Set the rendering edge width.
(render:set–edge–width 2.0)
;; #[view 1075915872]
```

render:set–environment–map

Scheme Extension: Rendering Control, Environment Maps

Action: Sets the currently active environment map.

Filename: rbase/rnd_scm/emap_scm.cxx

APIs: api_rh_set_environment_map

Syntax: `(render:set–environment–map environment–map)`

Arg Types:		
	environment–map	environment–map | boolean

Returns: environment–map

Errors: None

Description: This extension sets the environment–map to be used for rendering. An environment map is used for special rendering features, such as reflections.

An environment map is not supported in Basic Rendering Component. In Advanced Rendering Component, the render mode should be “full” or better.

environment–map specifies the environment map to set. Specifying #f deactivates the environment map and returns renderer to the default state.

Limitations: None

Example:
```
; render:set–environment–map
; Set the environment map to be used in
; rendering objects with materials with
; environment mapped reflections.
(define emap1 (environment–map:render
    (part:entities) 256 (position 0 0 0)))
;; emap1
(render:set–environment–map  emap1)
;; ()
```

render:set–foreground

Scheme Extension: Backgrounds and Foregrounds

Action: Sets the current foreground.

Filename: rbase/rnd_scm/frgd_scm.cxx

APIs: api_rh_set_foreground

Syntax: (**render:set–foreground** foreground)

Arg Types:	foreground	foreground

Returns: unspecified

Errors: None

Description: Refer to Action.

foreground is an entity that specifies a foreground shader. These shaders filter and transform the color of a model’s pixels. The available foreground types established through the Scheme extension foreground:set–prop are “none”, “depth cue”, “fog”.

Limitations: None

Example:
```
; render:set–foreground
; Get a valid list of foreground types.
(foreground:types)
;; (”fog” ”depth cue” ”none”)
; Create a ”none” foreground.
(define fore1 (foreground ”none”))
;; fore1
; Set the foreground.
(render:set–foreground fore1)
;; ()
```

render:set–mode

Scheme Extension: Rendering Control

Action: Sets the current render mode.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: api_rh_set_render_mode

Syntax: (**render:set–mode** mode)

Arg Types: mode string

Returns: unspecified

Errors: None

Description: The default render mode is gouraud.

mode specifies the render mode. Any subsequent rendering operations use the specified mode. The following modes are available:

flat renders the model by shading it with a constant color across the facets.

gouraud renders the model by shading each surface with a single, uniform base color. Curved surfaces are approximated by a mesh of polygonal facets, and in the shaded image, colors are calculated at each vertex of each facet. These colors are blended across the facets to create smooth shading of curved surfaces (intensity interpolation).

phong renders the model by interpolating the normals across a facet surface along a scanline, generating a more realistic image (vector–normal interpolation).

full renders the model by scanline mode that eliminates aliasing artifacts on object silhouettes. All surfaces that contribute to the intensity of a pixel are correctly determined and accounted for.

preview renders the model by scanline mode where geometry is sampled at a rate of one sample per rendered pixel, to produce object silhouettes with a jagged appearance.

raytrace–preview renders the model by scanline preview mode, but determines all visibility by ray tracing.

raytrace–full renders the model where visibility is determined at a rate sufficient to suppress the appearance of re–aliased artifacts as discontinuities between neighboring pixels.

Limitations: None

Example:

```
; render:set–mode
; Get a list of valid rendering modes.
(render:modes)
;; (”lambert” ”raytrace–full” ”raytrace–preview”
;; ”full” ”preview” ”phong” ”gouraud” ”flat”)
; Get the current rendering mode.
(render:mode)
;; ”gouraud”
; Set the rendering mode.
(render:set–mode ”flat”)
;; ()
; Get the rendering mode.
(render:mode)
;; ”flat”
```

render:targa

Scheme Extension: Image Output, Rendering Control

Action: Renders the view to a Truevision TARGA format file.

Filename: rbase/rnd_scm/rndr_scm.cxx

APIs: None

Syntax:

```
(render:targa [entity–list=all] [filename]
    [x–size y–size]
    [compress [color]] [view=active])
```

Arg Types:	entity–list	entity | (entity ...)
	filename	string
	x–size	real
	y–size	real
	compress	boolean
	color	boolean
	view	view

Returns: string

Errors: None

Description: This extension produces a file containing a rendered image in TARGA format entity–list. This extension returns the filename.

entity–list indicates the entities to render to produce the image.

filename specifies the name of the file to be produced.

compress indicates whether to produce compressed (#t) or uncompressed (#f) TARGA data.

color controls color output. The default, #t, produces color output; #f produces greyscale output. If two Boolean arguments are specified, the first controls compress and the second controls color.

view specifies the view of orientation, clipping, and resolution information is obtained.

Limitations: None

Example:

```
; render:targa
; Create a solid block.
(define block1 (solid:block (position 0 0 0)
    (position 10 10 10)))
;; block1
; Render the block.
(render block1)
;; ()
; Render the block to a file in Targa format.
(render:targa block1 ”entities1.tga” #f #t)
;; ”entities1.tga”
```

texture–space

Scheme Extension: Texture Spaces

Action: Creates a texture space.

Filename: rbase/rnd_scm/tmap_scm.cxx

APIs: api_rh_create_texture_space

Syntax: (**texture–space** type)

Arg Types: type string

Returns: texture–space

Errors: None

Description: Texture space controls how a a wrapped material entity is applied to a face or solid body. A texture space is applied to an entity using entity:set–texture–space.

The Advanced Rendering Component is required to display texture space on wrapped materials. However, in the Basic Rendering Component, the texture space may be created, its properties changed, and the texture space entity associated with a model entity. However, Basic Rendering Component does not display the results.

type is a string specifying the texture space type. The available texture space types include “arbitrary plane”, “auto axis”, “cylindrical”, “spherical”, “uv”, “x plane”, “y plane”, and “z plane”.

Limitations: None

Example:

```
; texture–space
; Create a solid block.
(define block1 (solid:block
    (position 0 0 0) (position 5 10 16)))
;; block1
; Create a solid sphere.
(define sphere1 (solid:sphere
    (position –10 –10 –10) 10))
;; sphere1
; Create two texture spaces.
(define texture1 (texture–space ”auto axis”))
;; texture1
(define texture2 (texture–space ”spherical”))
;; texture2
; Set the textures onto the solids.
(entity:set–texture–space block1 texture1)
;; ()
(entity:set–texture–space sphere1 texture2)
;; ()
```

texture–space:props

Scheme Extension: Texture Spaces

Action: Gets the properties of a texture space.

Filename: rbase/rnd_scm/tmap_scm.cxx

APIs: api_rh_get_texture_space_args

Syntax: `(texture–space:props texture–space)`

Arg Types: texture–space texture–space

Returns: ((string . real | integer | gvector) ...)

Errors: None

Description: This extension returns a list containing pairs of the property type and their current values.

The Advanced Rendering Component is required to display texture space on wrapped materials. However, in the Basic Rendering Component, the texture space may be created, its properties changes, and the texture space entity associated with a model entity. However, Basic Rendering Component does not display the results.

texture–space specifies a texture space to be queried.

Limitations: None

Example:

```
; texture–space:props
; Create a texture space.
(define texture1 (texture–space ”auto axis”))
;; texture1
; Get the properties of the texture space.
(texture–space:props texture1)
;; ((”scale” . 1))
```

texture–space:set–prop

Scheme Extension: Texture Spaces

Action: Sets a property of a texture space.

Filename: rbase/rnd_scm/tmap_scm.cxx

APIs: api_rh_get_texture_space_args, api_rh_set_texture_space_arg

Syntax: `(texture-space:set-prop texture-space name value)`

Arg Types:

texture–space	entity
name	string
value	integer | real | gvector

Returns: unspecified

Errors: None

Description: Texture space controls how a a wrapped material entity is applied to a face or solid body. A texture space is applied to an entity using entity:set–texture–space.

texture–space specifies the texture space entity.

name specifies the texture space property name. The properties are:

"ambient" illuminates all surfaces equally regardless of orientation. The intensity of the source (as a scalar quantity) is determined by intensity and color. The default values are:

intensity	1.0 (real)
color	#[color 1 1 1]

"arbitrary plane" is point mapping onto an arbitrary plane. The default values are:

scale	1.0 (real)
aspect ratio	1.0 (real)
origin	#[gvector 0 0 0]
normal vector	#[gvector 0 0 1]
up vector	#[gvector 0 1 0]

"auto axis" is point mapping by one of the three coordinate axes (*x*, *y*, and *z* axes).

"cylindrical" is point mapping onto a cylinder. The default values are:

scale around axis	1.0 (real)
scale along axis	1.0 (real)
centre point	#[gvector 0 0 0]
axis direction	#[gvector 0 0 1]
origin	#[gvector 1 0 0]

"spherical" is point mapping onto a sphere. The default values are:

latitude scale	1.0 (real)
longitude scale	1.0 (real)
centre point	#[gvector 0 0 0]
origin	#[gvector 1 0 0]
axis direction	#[gvector 0 0 1]

"uv" maps parametric coordinate system to the texture space. The default values are:

u scale . . . . . 1.0 (real)
v scale . . . . . 1.0 (real)

"x plane" provides point mapping onto a plane of constant *x*. The default value is:

scale . . . . . 1.0 (real)

"y plane" provides point mapping onto a plane of constant *y*. The default value is:

scale . . . . . 1.0 (real)

"z plane" provides point mapping onto a plane of constant *z*. The default value is:

scale . . . . . 1.0 (real)

value specifies the value of the associated property name.

The Advanced Rendering Component is required to display texture space on wrapped materials. However, in the Basic Rendering Component, the texture space may be created, its properties changes, and the texture space entity associated with a model entity. However, Basic Rendering Component does not display the results.

Limitations: None

Example:

```
; texture-space:set-prop
; Get a list of valid texture space shader types.
(texture-space:types)
;; ("arbitrary plane" "auto axis" "cylindrical"
;; "spherical" "uv" "x plane" "y plane" "z plane")
; Create a texture space.
(define texture1 (texture-space "arbitrary plane"))
;; texture1
; Set a property of the texture space.
(texture-space:set-prop texture1
    "aspect ratio" 0.75)
;; ()
```

texture–space:type

Scheme Extension: Texture Spaces

Action: Gets the type of a texture space.

Filename: rbase/rnd_scm/tmap_scm.cxx

APIs: api_rh_get_texture_space_args

Syntax: (**texture–space:type** texture–space)

Arg Types: texture–space texture–space

Returns: string

Errors: None

Description: Refer to Action.

texture–space specifies a texture space to be queried.

Limitations: None

Example:
```
; texture–space:type
; Get a list of valid texture space shader types.
(texture–space:types)
;; (”arbitrary plane” ”auto axis” ”cylindrical”
;; ”spherical” ”uv” ”x plane” ”y plane” ”z plane”)
; Create a texture space.
(define texture1 (texture–space ”auto axis”))
;; texture1
; Get the type of texture space.
(texture–space:type texture1)
;; ”auto axis”
```

texture–space:types

Scheme Extension: Texture Spaces

Action: Gets a list of valid texture space types.

Filename: rbase/rnd_scm/tmap_scm.cxx

APIs: api_rh_get_texture_space_types

Syntax: (**texture–space:types**)

Arg Types: None

Returns: (string ...)

Errors: None

Description: This extension returns all valid texture space types in a list of strings.

The Advanced Rendering Component is required to display texture space on wrapped materials. However, in the Basic Rendering Component, the texture space may be created, its properties changes, and the texture space entity associated with a model entity. However, Basic Rendering Component does not display the results.

Limitations: None

Example:
```
; texture–space:types
; Get a list of valid texture space shader types.
(texture–space:types)
;; (″arbitrary plane″ ″auto axis″ ″cylindrical″
;; ″spherical″ ″uv″ ″x plane″ ″y plane″ ″z plane″)
```

texture–space?

Scheme Extension: Texture Spaces

Action: Determines if a Scheme object is a texture space.

Filename: rbase/rnd_scm/tmap_scm.cxx

APIs: None

Syntax: (**texture–space?** object)

Arg Types: object scheme–object

Returns: boolean

Errors: None

Description: Refer to Action.

object specifies the scheme–object that has to be queried for a texture space.

Limitations: None

Example:
```
; texture–space?
; Create a texture space.
(define texture1 (texture–space ″auto axis″))
;; texture1
; Determine if the texture space is actually
; a texture space.
(texture–space? texture1)
;; #t
```