Command reference

Object appearance

Saving the default appearance: `gsave3d()`

Description

Saves the current default appearance by pushing it onto the appearance stack.

Restoring the default appearance: `grestore3d()`

Description

Restores the default appearance which was last saved on the appearance stack.

Set default color: `color3d(<colorvec>)`

Description

Sets the default color for all types of objects.

`<colorvec>` the default color to set

Set default point color: `pointcolor3d(<colorvec>)`

Description

Sets the default color for points.

`<colorvec>` the default point color to set

Set default line color: `linecolor3d(<colorvec>)`

Description

Sets the default color for lines.

`<colorvec>` the default line color to set

Set default surface color: `surfacecolor3d(<colorvec>)`

Description

Sets the default surface color.

`<colorvec>` the default surface color to set

Set default opacity: `alpha3d(<real>)` or `surfacealpha3d(<real>)`

Description

Sets the default opacity value for surfaces.

`<real>` the default opacity value to set, range [0,1]

Set default shininess: `shininess3d(<real>)`

Description

Sets the default shininess factor for all types of objects.

`<real>` the default shininess factor to set

Set default point shininess: `pointshininess3d(<real>)`

Description

Sets the default shininess factor for points.

`<real>` the default point shininess factor to set

Set default line shininess: `lineshininess3d(<real>)`

Description

Sets the default shininess factor for lines.

`<real>` the default line shininess factor to set

Set default surface shininess: `surfaceshininess3d(<real>)`

Description

Sets the default shininess factor for surfaces.

`<real>` the default surface shininess factor to set

Set default size: `size3d(<real>)`

Description

Sets the default size for points and lines.

`<real>` the default point and line size to set

Set default point size: `pointsize3d(<real>)`

Description

Sets the default size for points.

`<real>` the default point size to set

Set default line size: `linesize3d(<real>)`

Description

Sets the default size for lines.

`<real>` the default line size to set

Drawing

Drawing points: `draw3d(<point>)`

Description

Draws a point.

`<point>` the position of the point

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the point size
color	`[<real1>,<real2>,<real3>]`	sets the point color to an RGB value
shininess	`<real>`	sets the shininess

Drawing lines: `draw3d(<point1>,<point2>)`

Description

Draws an infinite line, ray or segment. The type of line to be drawn is specified by the "type" modifier. When no type is given, a segment is drawn by default. The two arguments are interpreted depending on the line type:

Line type	`<point1>`	`<point2>`
Segment	First end point	Second end point
Ray	Ray origin	Arbitrary point on ray, different from origin
Line	Point on line	Point on line, different from first point

Modifiers

This function can handle the following modifiers:

Modifier	**Parameter**	**Effect**
type	`<string>`	specifies the line type (values `"segment"`, `"ray"`, and `"line"` allowed)
size	`<real>`	sets the line size
color	`[<real1>,<real2>,<real3>]`	sets the line color to an RGB value
shininess	`<real>`	sets the shininess

Example

The following example illustrates drawing different line types:

```
// draws a red segment, between (0,0,0) and (1,0,0)
draw3d([0,0,0],[1,0,0],color->[1,0,0]);
// draws a green segment, between (0,0,0) and (0,1,0)
draw3d([0,0,0],[0,1,0],type->"segment",color->[0,1,0]);
// draw a blue ray starting at (0,0,0), extending along the positive z axis
draw3d([0,0,0],[0,0,1],type->"ray",color->[0,0,1]);
// draw a yellow line passing through (1,1,1) and (2,1,1)
draw3d([1,1,1],[2,1,1],type->"line",color->[1,1,0]);
```

[screenshot]

Connecting points: `connect3d(<list>)`

Description

Draws line segments connecting a sequence of points.

`<list>` the list of points to connect. For a list of n points, n-1 line segments are drawn.

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the line size
color	`[<real1>,<real2>,<real3>]`	sets the line color to an RGB value
shininess	`<real>`	sets the shininess

Drawing a polygon outline: `drawpoly3d(<list>)`

Description

Draws the outline of a polygon.

`<list>` the vertices (corner points) defining the polygon

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the line size
color	`[<real1>,<real2>,<real3>]`	sets the line color to an RGB value
shininess	`<real>`	sets the shininess

Drawing a filled polygon: `fillpoly3d(<list>)`

Description

Draws a filled polygon.

`<list>` the vertices (corner points) defining the polygon

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the surface size
color	`[<real1>,<real2>,<real3>]`	sets the surface color to an RGB value
shininess	`<real>`	sets the shininess
alpha	`<real>`	sets the opacity

Drawing a filled polygon with custom normals: `fillpoly3d(<list1>,<list2>)`

Description

Draws a filled polygon with user-defined normals.

`<list1>` the vertices (corner points) defining the polygon

`<list2>` the normal vectors of the polygon's vertices. The lengths of `<list1>` and `<list2>` must match.

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the surface size
color	`[<real1>,<real2>,<real3>]`	sets the surface color to an RGB value
shininess	`<real>`	sets the shininess
alpha	`<real>`	sets the opacity

Drawing a filled circle: `fillcircle3d(<point>,<vec>,<real>)`

Description

Draws a filled circle.

`<point>` the center of the circle

`<vec>` the normal vector of the circle

`<real>` the radius of the circle

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the surface size
color	`[<real1>,<real2>,<real3>]`	sets the surface color to an RGB value
shininess	`<real>`	sets the shininess
alpha	`<real>`	sets the opacity

Drawing a sphere: `drawsphere3d(<point>,<real>)`

Description

Draws a sphere.

`<point>` the center of the sphere

`<real>` the radius of the sphere

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
size	`<real>`	sets the surface size
color	`[<real1>,<real2>,<real3>]`	sets the surface color to an RGB value
shininess	`<real>`	sets the shininess
alpha	`<real>`	sets the opacity

Drawing a mesh: `mesh3d(<int1>,<int2>,<list>)`

Description

Draws a grid-based mesh. The vertices are organized in a regular grid of m rows and n columns. Neighbouring vertices in a row or column are connected by edges. Quadliteral faces are formed by combining the edges of each grid cell. To simplify the rendering of meshes, each quadliteral face is split along one of its diagonals into two triangles.

The behaviour at the borders of the surface can be specified by the "topology" modifier. When this modifier is not present, an open topology is assumed.

Topology	Description
Open	No additional edges or faces, resulting in a total total of $(m-1) \times (n-1)$ quadliteral faces. The surface has two sides and one border.
Close rows	Additional edges are introduced connecting the first and last vertex of each row. Also corresponding faces are generated, resulting in a total of $(m-1) \times n$ quadliteral faces. The surface has two sides and two borders.
Close columns	Additional edges are introduced connecting the first and last vertex of each column. Also corresponding faces are generated, resulting in a total of $m \times (n-1)$ quadliteral faces. The surface has two sides and two borders.
Close both	Additional edges are introduced connecting the first and last vertex of each row as well as the first and last vertex of each column. Also corresponding faces are generated, resulting in a total of $m \times n$ quadliteral faces. The surface has two sides and no border.

The way surface normals are computed is specified by the "normaltype" modifier. When this modifier is not present, per face normals are computed.

Normal type	Description
Per face	The normal at each surface point is the normal of the triangular face it belongs to. This can result in shading discontinuities along face edges, revealing the underlying grid structure.
Per vertex	The normal at each grid vertex is the average normal of the adjacent faces. For all other surface points the normal is computed by taking the three normals of the grid vertices forming the triangular face the point belongs to, and doing a linear combination of them with the barycentric coordinates of the point as coefficients. This results in smooth shading, hiding the underlying grid structure to a certain degree.

`<int1>` the number of grid rows, m

`<int2>` the number of grid columns, n

`<list>` the vertices of the grid, in row-major order. The length of this list must equal $m \times n$.

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
normaltype	`<string>`	specifies the normal type (values `"perFace"` and `"perVertex"` allowed)
topology	`<string>`	specifies the grid topology (values `"open"`, `"closeRows"`, `"closeColumns"`, and `"closeBoth"` allowed)
size	`<real>`	sets the surface size
color	`[<real1>,<real2>,<real3>]`	sets the surface color to an RGB value
shininess	`<real>`	sets the shininess
alpha	`<real>`	sets the opacity

Example

Drawing a mesh with custom normals:
`mesh3d(<int1>,<int2>,<list1>,<list2>)`

Description

Draws a grid-based mesh with user-defined normals. For a description of how the grid is formed, refer to mesh3d(<int1>,<int2>,<list>).

`<int1>` the number of grid rows, m

`<int2>` the number of grid columns, n

`<list1>` the vertices of the grid, in row-major order. The length of this list must equal $m \times n$.

`<list2>` the normals for each vertex, in row-major order. The length of this list must equal $m \times n$.

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
topology	`<string>`	specifies the grid topology (values `"open"`, `"closeRows"`, `"closeColumns"`, and `"closeBoth"` allowed)
size	`<real>`	sets the surface size
color	`[<real1>,<real2>,<real3>]`	sets the surface color to an RGB value
shininess	`<real>`	sets the shininess

... continued on next page

Modifier	Parameter	Effect
alpha	`<real>`	sets the opacity

Example

Lighting and scene appearance

Set background color: `background3d(<colorvec>)`

Description

Sets the scene background color to an RGB value.

`<colorvec>` the scene background color to set

Position the camera: `lookat3d(<point1>,<point2>,<vec>)`

Description

Sets the position, look at point and up vector of the camera.

`<point1>` the position of the camera

`<point2>` the look at point of the camera

`<vec>` the up vector of the camera

Set field of view: `fieldofview3d(<real>)`

Description

Sets the field of view of the camera.

`<real>` horizontal field of view of the camera. Must be in range $]0, \pi[$

Set visible depth range: `depthrange3d(<real1>,<real2>)`

Description

Sets the minimum and maximum camera depth. The camera depth of a point is its distance to the camera plane (the plane through the camera and orthogonal to the viewing direction). All objects or parts thereof which don't fall into the current camera depth range are not visible.

Set rendering hints: `renderhints3d()`

Description

Sets hints for various aspects of the rendering process. The rendering hints are specified by the following four modifiers:

- The "quality" modifier allows to select from a fixed set of predefined quality levels. Quality level 0 is the worst quality but needs the least resources. The highest quality level is 8, which provides a very good quality at the cost of high resource requirements. The predefined quality levels are a simple way to influence the overall rendering quality without having to specify seperate rendering hints. When the requested quality level is not supported (e.g. due to hardware limitations or resource constraints), *Cindy3D* might fall back to a lower level.

- The "renderMode" modifier specifies how the objects are rendered. When it is "simple", all objects are rendered as triangle meshes. In this modethe shading is done per vertex, resulting in shading artifacts. When the render mode is set to "raycasted", points, lines, and spheres are rendered as continuous surfaces using ray casting. Also the shading is done per pixel. The "raycasted" render mode produces higher quality results, but might be slower depending on the graphics hardware.

- The "screenError" modifier sets the allowed screen space error for the level of detail algorithm. In the "simple" render mode, points, lines, and spheres are approximated by triangle meshes. As an optimization, small or far away objects are represented by meshes with fewer triangles to reduce rendering time. This is called "level of detail". *Cindy3D* uses a fixed number of triangle meshes with different triangle counts for each primitive type. To determine which of these meshes to draw for a specific primitive, each mesh is virtually projected onto the screen and the maximum triangle size in pixels is measured. The smallest mesh for which the maximum projected triangle size is below the "screenError" is then used for rendering the primitive. This means that lower values of "screenError" result in higher quality, at the cost of rendering time.

  This modifier has no effect when used in conjunction with the "raycasted" render mode.

- The "samplingRate" modifier influences the smoothness of object silhouettes. The sampling rate defines the number of samples that are taken for each pixel of the output image. The final pixel color is an average of all its samples. The higher the sampling rate, the smoother the object silhouettes appear, at the cost of increased memory and time consumption. When the requested sampling rate is not supported (e.g. due to hardware limitations or resource constraints), *Cindy3D* might fall back to a lower sampling rate.

Modifiers

The function can handle the following modifiers:

Modifier	Parameter	Effect
quality	`<int>`	selects one of the predefined quality levels (values in [0,8] allowed)
renderMode	`<string>`	sets the rendering mode (values `"simple"` and `"raycasted"` allowed)
samplingRate	`<int>`	sets the number of samples taken per pixel (integer values from 1)
screenError	`<real>`	specifies maximum allowed screen space error in pixels (must be larger than 0)

Set up a point light: `pointlight3d(<int>)`

Description

Creates or modifies a point light source. If the light source at the given index is already a point light, overrides the properties specified by the modifiers and enables the light. Otherwise, replaces the light source at the given index by a new point light with the properties from the modifiers. In absence of a modifier, its default value is used.

`<int>` light source index. Must be in range [0,7].

Modifiers

The function can handle the following modifiers:

Modifier	Parameter	Effect
ambient	`[<real1>,<real2>,<real3>]`	sets the ambient light color to an RGB value (default: [0,0,0])
diffuse	`[<real1>,<real2>,<real3>]`	sets the diffuse light color to an RGB value (default: [1,1,1])
specular	`[<real1>,<real2>,<real3>]`	sets the specular light color to an RGB value (default: [1,1,1])
position	`<point>`	sets the light position (default: [0,0,0])
frame	`<string>`	specifies whether the position is relative to the camera frame or absolute (values `"camera"` and `"world"` allowed, default: `"camera"`)

Set up a directional light: `directionallight3d(<int>)`

Description

Creates or modifies a directional light source. If the light source at the given index is already a directional light, overrides the properties specified by the modifiers and enables the light. Otherwise, replaces the light source at the given index by a new directional light with the properties from the modifiers. In absence of a modifier, its default value is used.

`<int>` light source index. Must be in range [0,7].

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
ambient	`[<real1>,<real2>,<real3>]`	sets the ambient light color to an RGB value (default: [0,0,0])
diffuse	`[<real1>,<real2>,<real3>]`	sets the diffuse light color to an RGB value (default: [1,1,1])
specular	`[<real1>,<real2>,<real3>]`	sets the specular light color to an RGB value (default: [1,1,1])
direction	`<vec>`	sets the light direction (default: [0,-1,0])
frame	`<string>`	specifies whether the direction is relative to the camera frame or absolute (values `"camera"` and `"world"` allowed, default: `"camera"`)

Set up a spot light: `spotlight3d(<int>)`

Description

Creates or modifies a spot light source. If the light source at the given index is already a spot light, overrides the properties specified by the modifiers and enables the light. Otherwise, replaces the light source at the given index by a new directional light with the properties from the modifiers. In absence of a modifier, its default value is used.

`<int>` light source index. Must be in range [0,7].

Modifiers

This function can handle the following modifiers:

Modifier	Parameter	Effect
ambient	`[<real1>,<real2>,<real3>]`	sets the ambient light color to an RGB value (default: [0,0,0])
diffuse	`[<real1>,<real2>,<real3>]`	sets the diffuse light color to an RGB value (default: [1,1,1])
specular	`[<real1>,<real2>,<real3>]`	sets the specular light color to an RGB value (default: [1,1,1])
position	`<point>`	sets the light position (default: [0,0,0])
direction	`<vec>`	sets the light direction (default: [0,-1,0])
cutoffAngle	`<real>`	sets the cutoff angle of the spot cone (in radians, values in $[0, \frac{\pi}{2}]$ allowed, default: $\frac{\pi}{4}$)
exponent	`<real>`	sets the attenuation exponent (values in [0,128] allowed, default: 0)
frame	`<string>`	specifies whether position and direction are relative to the camera frame or absolute (values `"camera"` and `"world"` allowed, default: `"camera"`)

Disable a light source: `disablelight3d(<int>)`

Description

Disables the light at the given index.

`<int>` light source index. Must be in range [0,7].