MATLAB Lighting Commands

MATLAB provides commands that enable you to position light sources and adjust the characteristics of lit objects. These commands include the following.

Command	Purpose
`camlight`	Create or move a light with respect to the camera position
`lightangle`	Create or position a light in spherical coordinates
`light`	Create a light object
`lighting`	Select a lighting method
`material`	Set the reflectance properties of lit objects

MATLAB **`camlight`** (1/2)

Create or move a light object in camera coordinates

Syntax

```
camlight headlight
camlight right
camlight left
camlight
camlight(az,el)
camlight(...'style')
camlight(light_handle,...)
light_handle = camlight(...)
```

Description

`camlight('headlight')` creates a light at the camera position.
`camlight('right')` creates a light right and up from camera.
`camlight('left')` creates a light left and up from camera.
`camlight` with no arguments is the same as **`camlight('right')`**.

MATLAB **`camlight`** (2/2)

`camlight(az,el)` creates a light at the specified azimuth (**`az`**) and elevation (**`el`**) with respect to the camera position. The camera target is the center of rotation and **`az`** and **`el`** are in degrees.

`camlight(...,'style')` The style argument can take on two values:
local (default) -- The light is a point source that radiates from the location in all directions.
infinite -- The light shines in parallel rays.

`camlight(light_handle,...)` uses the light specified in **`light_handle`**.
`light_handle = camlight(...)` returns the light's handle.

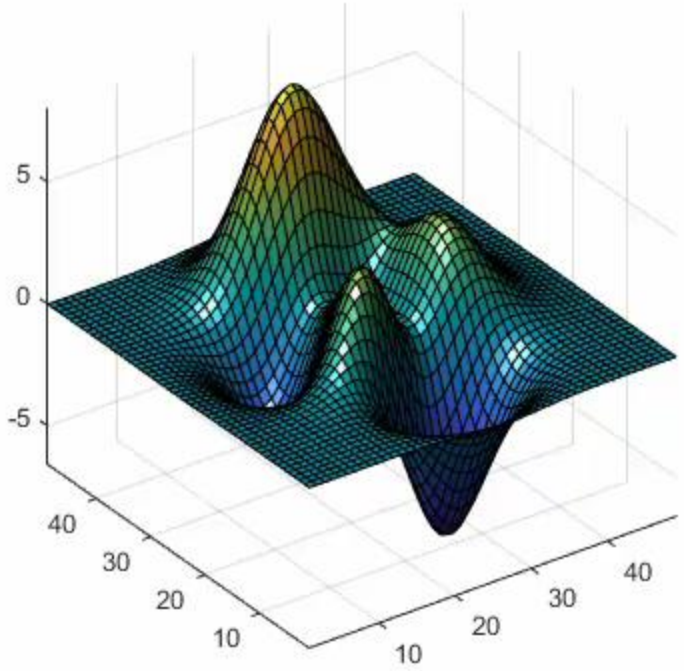
Remarks

`camlight` sets the light object Position and Style properties. A light created with **`camlight`** will not track the camera. In order for the light to stay in a constant position relative to the camera, you must call **`camlight`** whenever you move the camera.

Example 1

```
surf(peaks)
axis vis3d
axis tight
h = camlight('left');
set(gcf,'Color','w');

v = VideoWriter('example1.mp4','MPEG-4');
open(v);
for i = 1:80;
   camorbit(2,0)
   drawnow;
   pause(0.01);
   F = getframe(gcf);
   writeVideo(v,F);
end
close(v);
```



Shading Algorithms in MATLAB

MATLAB supports three different algorithms for lighting calculations, selected by setting the **`FaceLighting`** and **`EdgeLighting`** properties of each patch and surface object in the scene. Each algorithm produces somewhat different results:

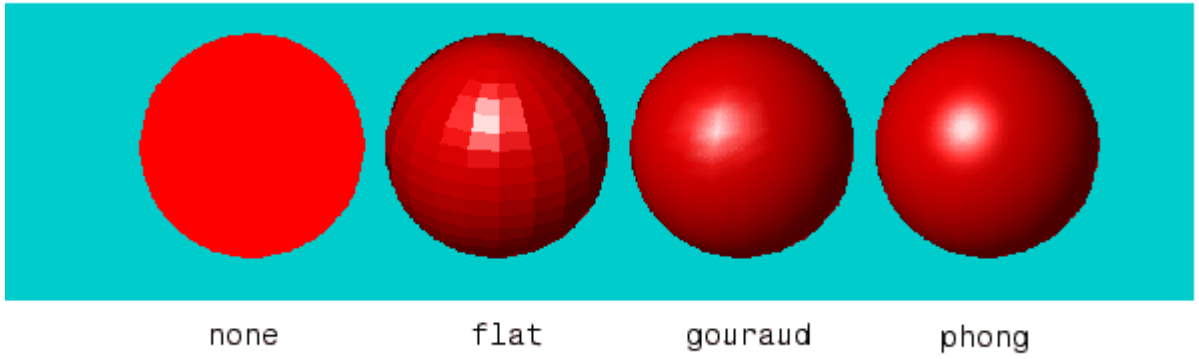
Flat lighting – Produces uniform color across each of the faces of the object. Select this method to view faceted objects.

Gouraud lighting – Calculates the colors at the vertices and then interpolates colors across the faces. Select this method to view curved surfaces.

Phong lighting – Interpolates the vertex normals across each face and calculates the reflectance at each pixel. Select this choice to view curved surfaces.

Shading Example

Phong lighting generally produces better results than Gouraud lighting, but takes longer to render. This illustration shows how a red sphere looks using each of the lighting methods with one white light source. The **`lighting`** command (as opposed to the **`light`** function) provides a convenient way to set the lighting method. (see next slide)



MATLAB **`lighting`** Function

`lighting` selects the algorithm used to calculate the effects of light objects on all surface and patch objects in the current axes.

`lighting flat` selects flat lighting.
`lighting gouraud` selects gouraud lighting.
`lighting phong` selects phong lighting.
`lighting none` turns off lighting

Light Objects

You create a light object using the light function. Three important light object properties are

`Color` – Color of the light cast by the light object

`Style` – Either infinitely far away (the default) or local

`Position` – Direction (for infinite light sources) or the location (for local light sources)

The **`Color`** property determines the color of the directional light from the light source. The color of an object in a scene is determined by the color of the object and the light source. The **`Style`** property determines whether the light source is a point source (Style set to local), which radiates from the specified position in all directions, or a light source placed at infinity (Style set to infinite), which shines from the direction of the specified position with parallel rays. The **`Position`** property specifies the location of the light source in axes data units. In the case of a light source at infinity, Position specifies the direction to the light source. Lights affect surface and patch objects that are in the same axes as the light. These objects have a number of properties that alter the way they look when illuminated by lights.

MATLAB **`light`** Function

`light` creates a light object in the current axes. Lights affect only patch and surface objects.

`light('PropertyName',PropertyValue,...)` creates a light object using the specified values for the named properties. MATLAB parents the light to the current axes unless you specify another axes with the Parent property.

`handle = light(...)` returns the handle of the light object created.

You cannot see a light object per se, but you can see the effects of the light source on patch and surface objects. You can also specify an axes-wide ambient light color that illuminates these objects. However, ambient light is visible only when at least one light object is present and visible in the axes. You can specify properties as property name/property value pairs, structure arrays, and cell arrays.

See also the patch and surface **`AmbientStrength`**, **`DiffuseStrength`**, **`SpecularStrength`**, **`SpecularExponent`**, **`SpecularColorReflectance`**, and **`VertexNormals`** properties. Also see the **`lighting`** and **`material`** commands.

Light **`style`** Property

`Style` **`{infinite} | local`**

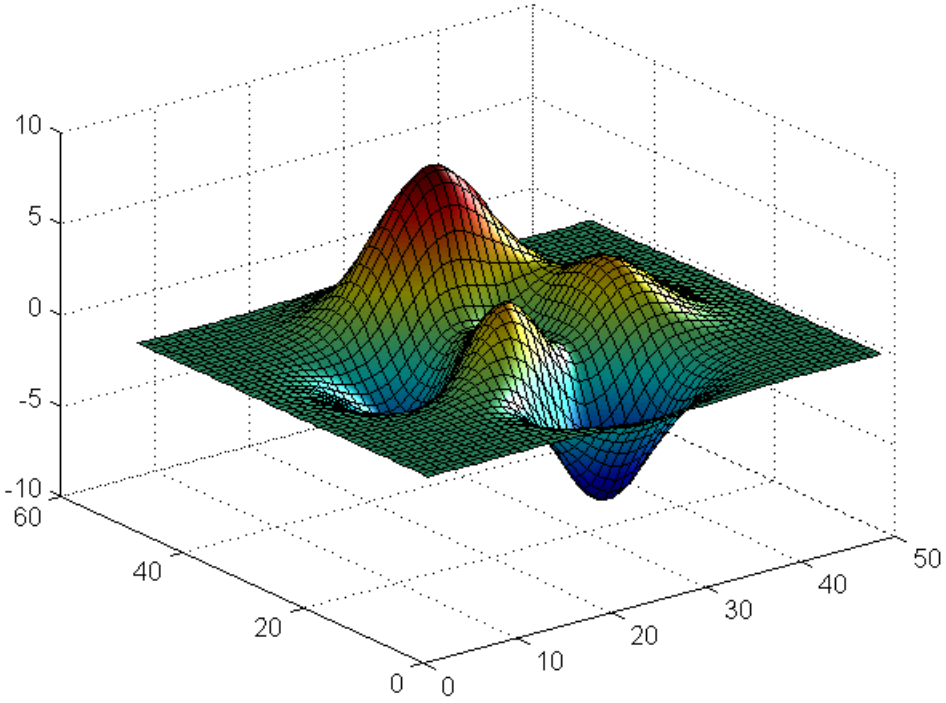
The style property defines a parallel or divergent light source. This property determines whether MATLAB places the light object at infinity, in which case the light rays are parallel, or at the location specified by the **`Position`** property, in which case the light rays diverge in all directions.

If the **`Style`** property is set to **`local`**, **`Position`** specifies the actual location of the light (which is then a point source that radiates from the location in all directions).

If the **`Style`** property is set to **`infinite`**, **`Position`** specifies the direction from which the light shines in parallel rays.

Example 2

```
h = surf(peaks);
set(h,'FaceLighting','phong','FaceColor','interp',...
'AmbientStrength',0.5,'EdgeColor','none')
light('Position',[1 0 0],'Style','infinite');
```



MATLAB **`lightangle`** function

`lightangle(az,el)` creates a light at the position specified by azimuth and elevation. **`az`** is the azimuthal (horizontal) rotation and **`el`** is the vertical elevation (both in degrees). The interpretation of azimuth and elevation is the same as that of the view command.

`light_handle = lightangle(az,el)` creates a light and returns the handle of the light in **`light_handle`**.

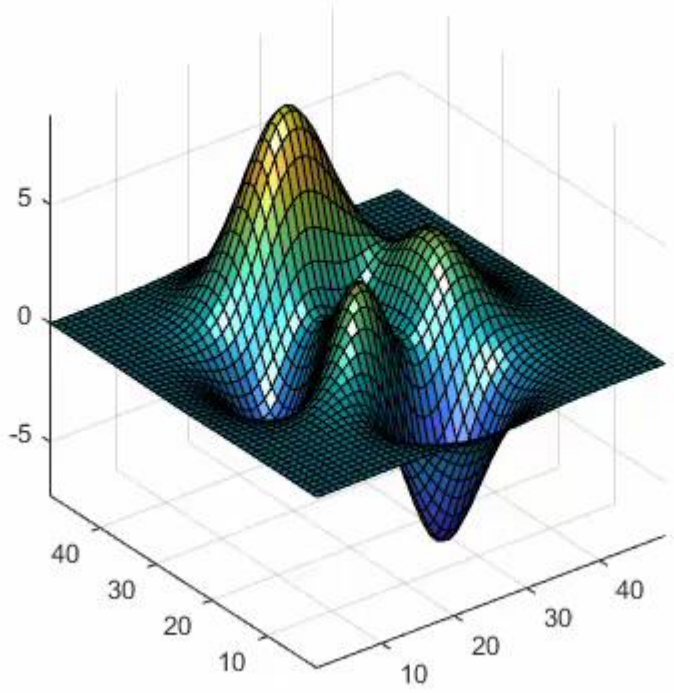
`lightangle(light_handle,az,el)` sets the position of the light specified by **`light_handle`**.

`[az,el] = lightangle(light_handle)` returns the azimuth and elevation of the light specified by **`light_handle`**.

By default, when a light is created, its style is **`infinite`**. If the light handle passed in to **`lightangle`** refers to a local light, the distance between the light and the camera target is preserved as the position is changed.

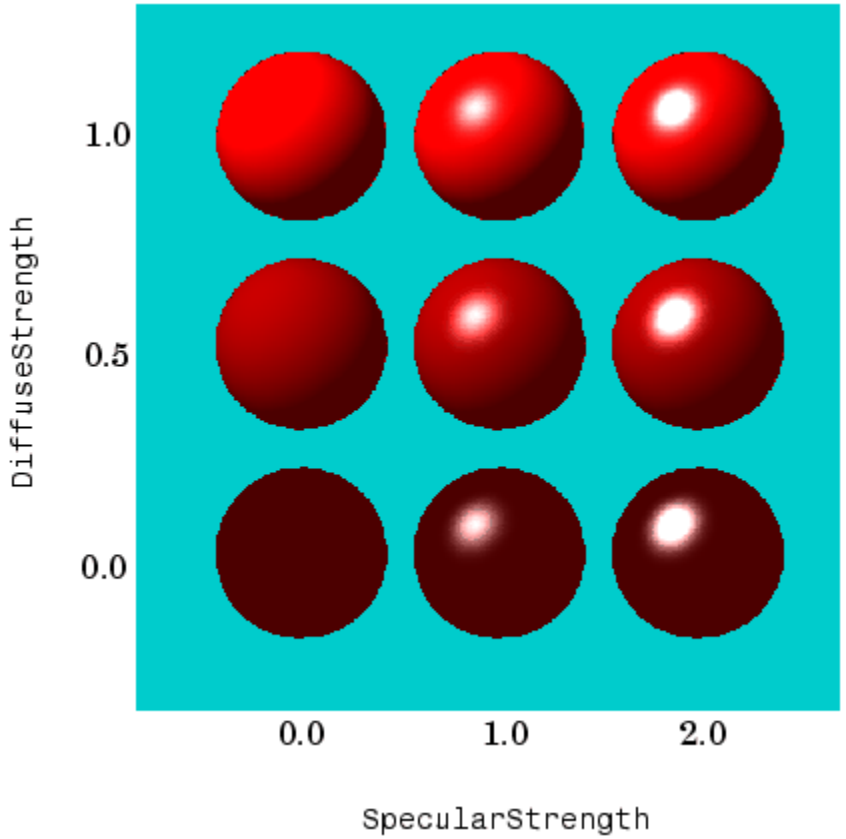
Example 3

```
surf(peaks)
axis vis3d
h = light;
set(gcf,'Color','w’);
v = VideoWriter('example3.mp4','MPEG-4');
open(v);
for az = -50:0.5:50
   lightangle(h,az,30)
   drawnow
   pause(0.01);
   F = getframe(gcf);
   writeVideo(v,F);
 end
close(v);
```



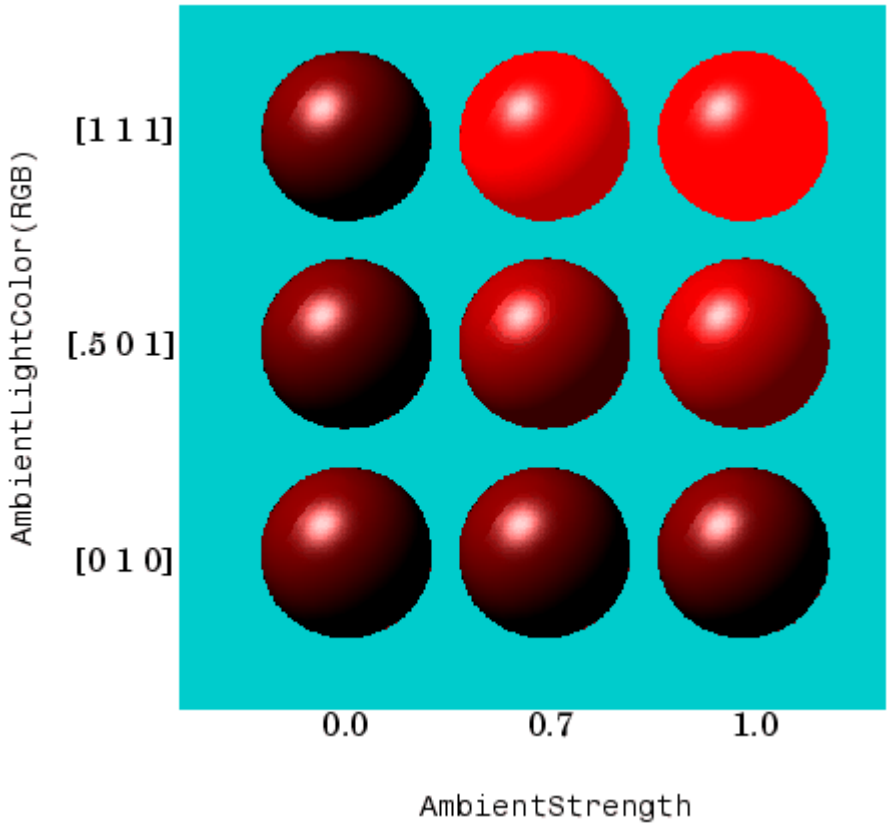
Specular and Diffuse Reflection

You can control the amount of specular and diffuse reflection from the surface of an object by setting the **`SpecularStrength`** and **`DiffuseStrength`** properties. This picture illustrates various settings.



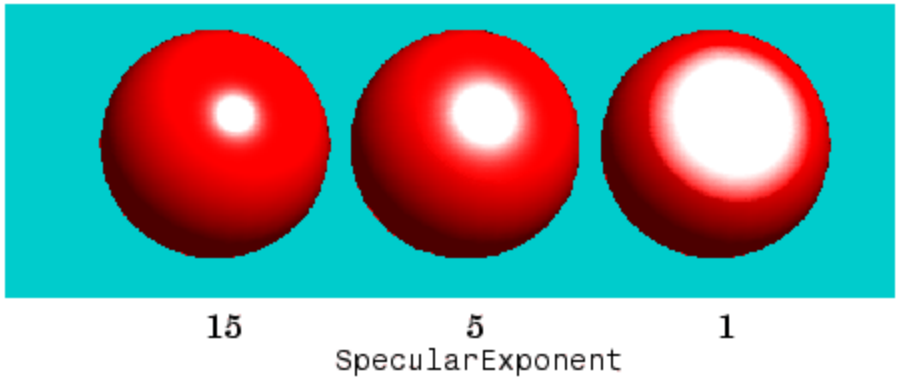
Ambient Light

Ambient light is a directionless light that shines uniformly on all objects in the scene. Ambient light is visible only when there are light objects in the axes. There are two properties that control ambient light --
`AmbientLightColor` is an axes property that sets the color, and **`AmbientStrength`** is a property of patch and surface objects that determines the intensity of the ambient light on the particular object. This illustration shows three different ambient light colors at various intensities. The sphere is red and there is a white light object present.



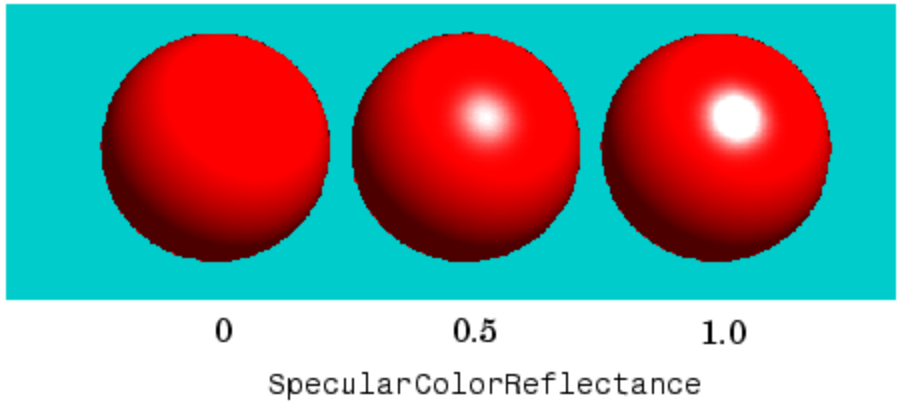
Specular Exponent

The size of the specular highlight spot depends on the value of the patch and surface object's **`SpecularExponent`** property. Typical values for this property range from 1 to 500, with normal objects having values in the range 5 to 20. This illustration shows a red sphere illuminated by a white light with three different values for the **`SpecularExponent`** property.



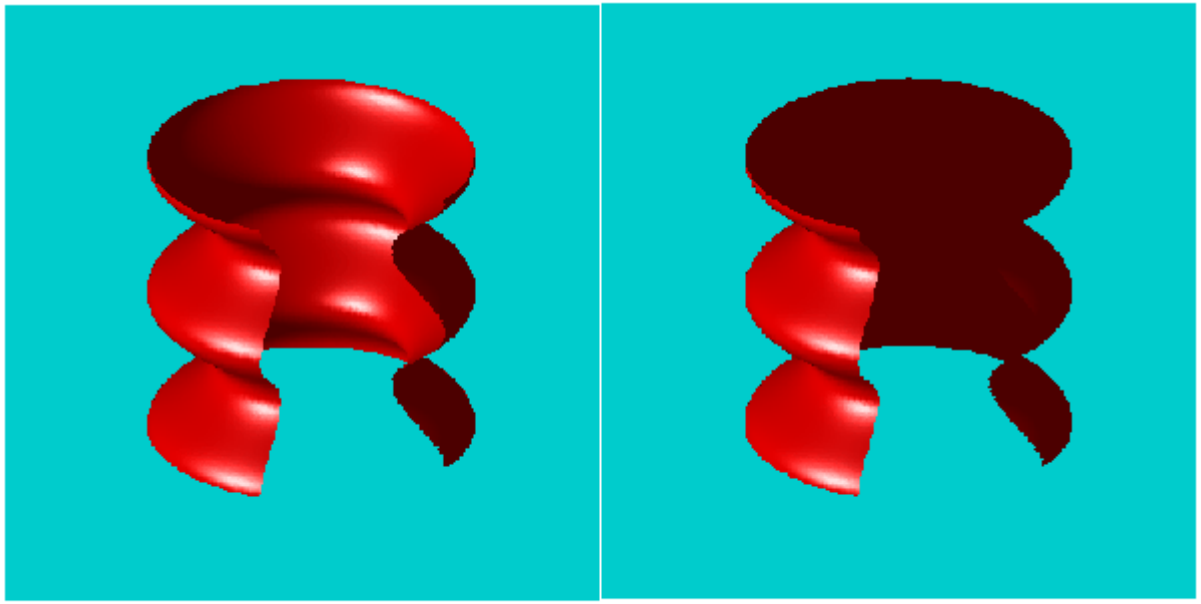
Specular Color Reflectance

The color of the specularly reflected light can range from a combination of the color of the object and the color of the light source to the color of the light source only. The patch and surface **`SpecularColorReflectance`** property controls this color. This illustration shows a red sphere illuminated by a white light. The values of the **`SpecularColorReflectance`** property range from 0 (object and light color) to 1 (light color).



Back Face Lighting

Back face lighting is useful for showing the difference between internal and external faces. These pictures of cut-away cylindrical surfaces illustrate the effects of back face lighting.



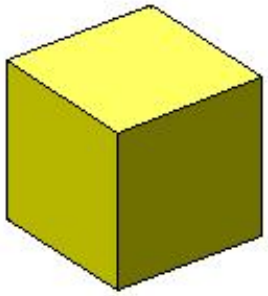
BackFaceLighting = reverselit

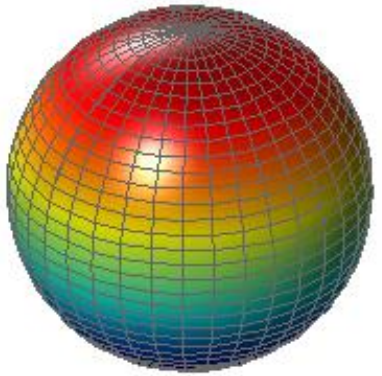
BackFaceLighting = unlit

The default value for **`BackFaceLighting`** is **`reverselit`**. This setting reverses the direction of the vertex normals that face away from the camera, causing the interior surface to reflect light towards the camera. Setting **`BackFaceLighting`** to **`unlit`** disables lighting on faces with normals that point away from the camera.

Example 4

```
vert = [ 1 1 1; 1 2 1; 2 2 1; 2 1 1; 1 1 2; 1 2 2; 2 2 2; 2 1 2];
fac = [ 1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
sphere(36);
h = findobj('Type','surface');
set(h,'FaceLighting','phong',...
   'FaceColor','interp',... 'EdgeColor',[.4 .4 .4],...
   'BackFaceLighting','lit')
hold on
patch('faces',fac,'vertices',vert,'FaceColor','y');
light('Position',[1 3 2]);
light('Position',[-3 -1 3]);
material shiny
axis vis3d off
hold off
```





All faces of the cube have **`FaceColor`** set to yellow. The sphere function creates a spherical surface and the handle of this surface is obtained using **`findobj`** to search for the object whose Type property is **`surface`**. The light functions define two white (the default color) light objects located at infinity in the direction specified by the **`Position`** vectors. These vectors are defined in axes coordinates [x, y, z]. The patch uses flat **`FaceLighting`** (the default) to enhance the visibility of each side. The surface uses phong **`FaceLighting`** because it produces the smoothest interpolation of lighting effects. The **`material shiny`** command affects the reflectance properties of both the cube and sphere (although its effects are noticeable only on the sphere because of the cube's flat shading). Because the sphere is closed, the **`BackFaceLighting`** property is changed from its default setting, which reverses the direction of vertex normals that face away from the camera, to normal lighting, which removes undesirable edge effects.

MATLAB **`material`** Command (1/2)

`material` sets the lighting characteristics of surface and patch objects.

`material shiny` sets the reflectance properties so that the object has a high specular reflectance relative to the diffuse and ambient light, and the color of the specular light depends only on the color of the light source.

`material dull` sets the reflectance properties so that the object reflects more diffuse light and has no specular highlights, but the color of the reflected light depends only on the light source.

`material metal` sets the reflectance properties so that the object has a very high specular reflectance, very low ambient and diffuse reflectance, and the color of the reflected light depends on both the color of the light source and the color of the object.

	`ka`	**`kd`**	**`ks`**	**`n`**	**`sc`**
`shiny`	0.3	0.6	0.9	20	1.0
`dull`	0.3	0.8	0	10	1.0
`metal`	0.3	0.3	1.0	25	0.5

`sc` = specular color

MATLAB **`material`** Command (2/2)

`material([ka kd ks])` sets the ambient/diffuse/specular strength of the objects.

`material([ka kd ks n])` sets the ambient/diffuse/specular strength and specular exponent of the objects.

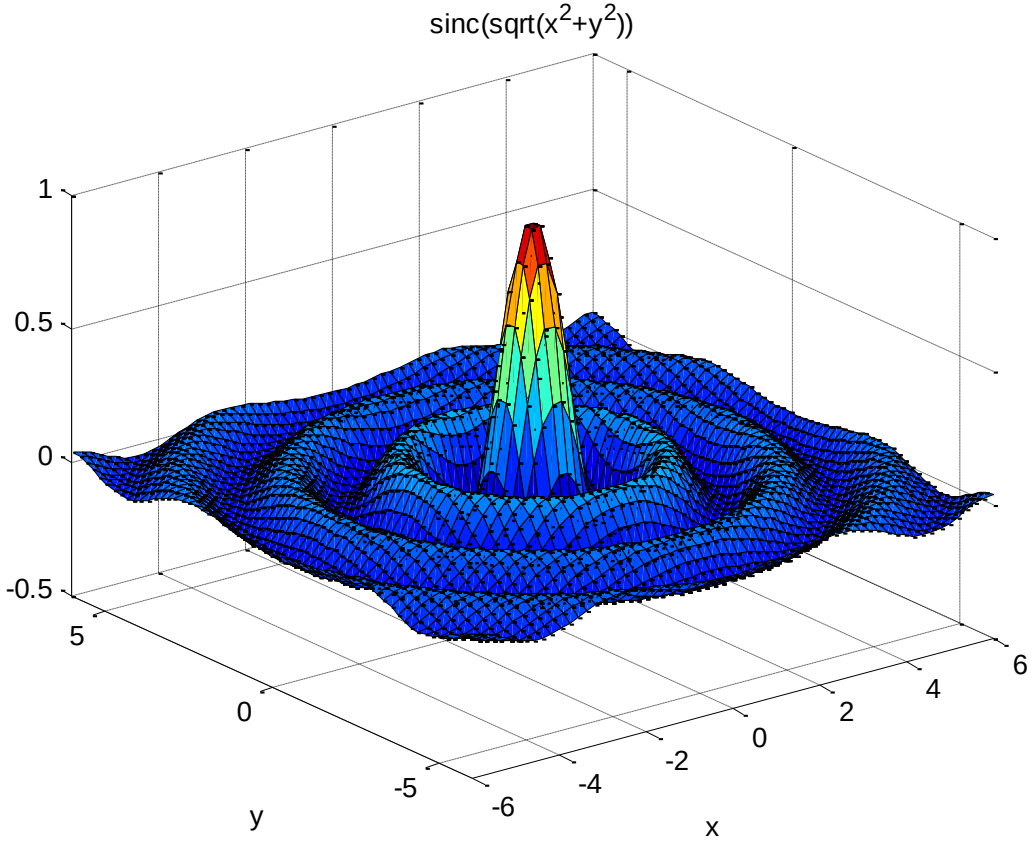
`material([ka kd ks n sc])` sets the ambient/diffuse/specular strength, specular exponent, and specular color reflectance of the objects.

`material default` sets the ambient/diffuse/specular strength, specular exponent, and specular color reflectance of the objects to their defaults.

The **`material`** command sets the material properties of all surface and patch objects in the axes. There must be visible light objects in the axes for lighting to be enabled. Look at the materal.m M-file to see the actual values set (enter the command **`type material`**).

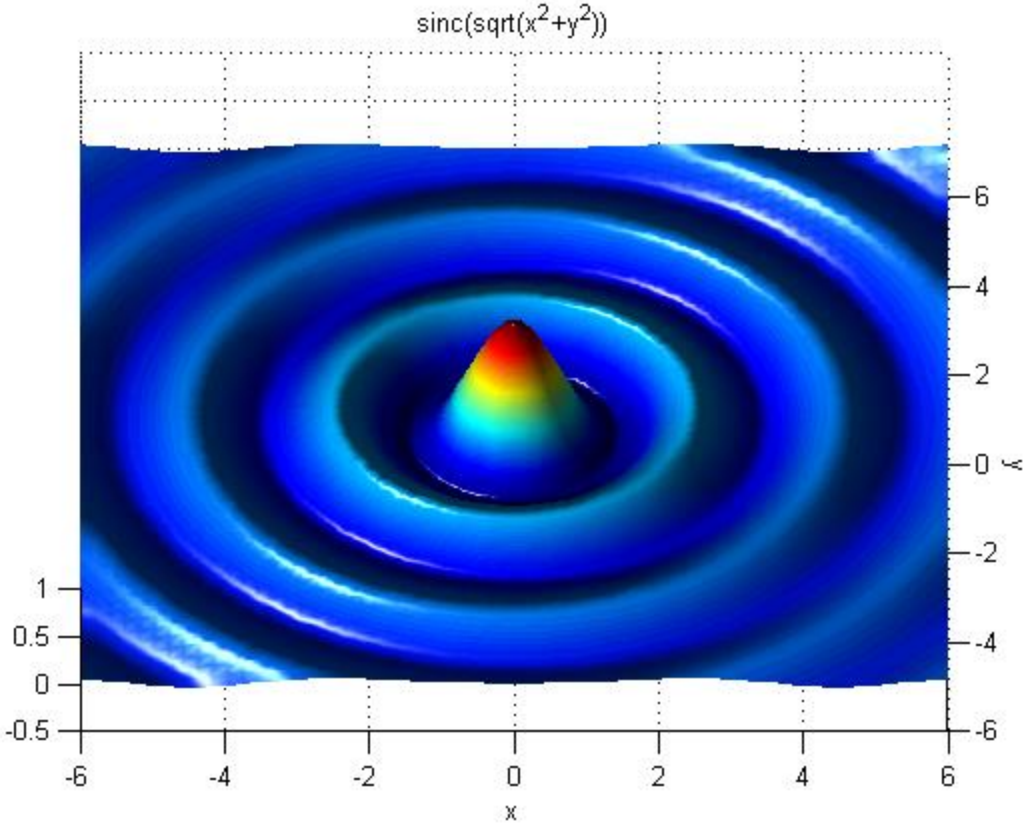
Mathematical Surface via **ezsurf**

```
ezsurf('sinc(sqrt(x^2+y^2))',[-6 6]);
```



Example 5

```
ezsurf('sinc(sqrt(x^2+y^2))',[-6 6]); view(0,75)
shading interp
lightangle(-45,30) set(gcf,'Renderer','zbuffer')
set(findobj(gca,'type','surface'),...
   'FaceLighting','phong',...
  'AmbientStrength',.3,'DiffuseStrength',.8,...
   'SpecularStrength',.9,'SpecularExponent',25,...
   'BackFaceLighting','unlit')
```



After obtaining the surface object's handle using **`findobj`**, you can set properties that affect how the light reflects from the surface.