fig2pov documentation

# Additional Povray options

In addition to a straightforward conversion from the MATLAB figure to a ray-traced image, it is possible to specify additional properties and options for the graphics objects. This will enable rendering options that are not natively supported in MATLAB itself, such as adding texture to patch objects, adding shadowing, etc ...

Each graphics object has a property called ‘UserData’, which allows any type of data to be attached to the object. The appearance a graphics object can be changed by adding a structure named 'povray' to the UserData of that object.

For instance, adding the following to a graphics object with handle h:

h.UserData.povray.Texture = ‘T\_Stone1’;

will indicate to fig2pov that the (Povray) texture T\_Stone1 has to be applied to that object.

## Additional Povray options for object of type ‘axes’

Plane

PlaneColor

PlaneTexture

Define

## Additional Povray options for object of type ‘patch’

Texture

InteriorTexture

TextureScale

ShadowLess

drawAsSphere

drawAsCylinder

FaceColor

FaceAlpha

drawEdges

EdgeColor

EdgeTexture

MarkerFaceColor

MarkerTexture

## Additional Povray options for object of type ‘surface’

**Texture**

See ‘Additional Povray options for object of type ‘patch’’

**InteriorTexture**

See ‘Additional Povray options for object of type ‘patch’’

**TextureScale**

See ‘Additional Povray options for object of type ‘patch’’

ShadowLess

**drawAsSphere**

See ‘Additional Povray options for object of type ‘patch’’

**drawAsCylinder**

See ‘Additional Povray options for object of type ‘patch’’

FaceColor

FaceAlpha

MeshOn

## Additional Povray options for object of type ‘line’

Povray is a ray-tracing program. As such, it has no concept of "lines". Instead, lines are converted into thin cylinders.

Texture

InteriorTexture

TextureScale

SmoothingOn

### Lines

### Mix of objects

## Using Povray options

Each graphics object in MATLAB has a property/field call 'UserData'. This field has no predefined structure and can be used to add data of any kind to the object. Note that each object (inclduing figures and axes) has its own 'UserData'. To provide povray-specific instructions for a graphics object, add a field 'povray' to UserData. UserData.povray is interpreted as a structure and its fields specify additional rendering instructions.

Some examples are given below.

### Adding Texture

MATLAB can add texture to surface objects, but no to other graphics objects. Povray has very rich texturing capabilities.

Let's start again from the cube:

```

cube = patch('Vertices',[0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1], ...

'Faces',[1 2 4 3; 5 6 8 7; 1 2 6 5; 3 4 8 7; 1 3 7 5; 2 4 8 6], ...

'FaceColor', [1 0 0]);

view(3)

axis equal

```

Adding texture using povray is done by adding the proper instructions to cube.UserData:

```

cube.UserData.povray.Texture = 'T\_Stone21';

```

'T\_Stone21' is one of the many texture recognized by Povray. A full list can be found in xxxx, including stone, wood, metal, glass, ... finishes.

Once the structure 'povray' with field 'Texture' is added to cube, fig2pov can be called to generate the povray script:

```

fig2pov(gca, 'cube.pov')

```

Executing 'cube.pov' in Povray gives the following figure:

<img src="cube\_povray\_texture1.png" width="500" title="hover text">

### Adding background

By default, the color of the background will be the same as the color used for the axes object.

### Adding a plane

### Optimizing rendering of spheres and cylinders

In MATLAB, patch and surface objects are really collections of polygons, even if the intention is to draw a sphere or a cylinder. The appearance of "roundness" is obtained by using large numbers of faces.

Povray natively supports commands that draw smooth spheres or cylinders (or any object with a rotation axis). If we know that an object is a sphere or has an symmetry axis, we can include this information in UserData.povray. fig2pov will then use the appropriate povray commands, rather than treating the object as a collection of faces.

Consider a red sphere in MATLAB:

```

[x,y,z]=sphere;

h\_sphere = surf(x, y, z, 'FaceColor', 'r')

view(3)

axis equal

```

![matlab sphere](sphere.png)

The faces are clearly visible and will remain visible when using fig2pov:

![povray sphere](sphere1.png)

We can let fig2pov know to treat the graphics object as a sphere by adding:

```

h\_sphere.UserData.povray.drawAsSphere = true;

```

With this, the rendered image will look like:

![povray sphere smooth](sphere\_drawassphere.png)

Likewise, a similar command can be used for cylinders. Note that the MATLAB command 'cylinder' means any object that is formed by rotation a curve around the z-axis. For example:

```

t = 0:pi/10:2\*pi;

[x, y, z] = cylinder(0.5+0.1\*cos(t));

surf(x, y, z, 'FaceColor', 'r')

axis equal

```

![cylinder1](cylinder1.png)

![cylinder povray](cylinder\_povray.png)

![cylinder povray drawascylinder](cylinder\_povray\_drawascylinder.png)