Defining Patches

You define a patch by specifying the coordinates of its vertices and some form of color data. Patches support a variety of coloring options that are useful for visualizing data superimposed on geometric shapes. There are two ways to specify a patch:

- By specifying the coordinates of the vertices of each polygon, which MATLAB connects to form the patch
- By specifying the coordinates of each unique vertex and a matrix that specifies how to connect these vertices to form the faces

The second technique is preferred for multifaceted patches because it generally requires less data to define the patch; vertices shared by more than one face need be defined only once.

Matlab **`patch`** Function

Create patch graphics object

Syntax

```
patch(X,Y,C)
patch(X,Y,Z,C)
patch(FV)
patch(...'PropertyName',PropertyValue...)
patch('PropertyName',PropertyValue...)
handle = patch(...)
```

PN/PV pairs only

Description

`patch` is the low-level graphics function for creating patch graphics objects. A patch object is one or more polygons defined by the coordinates of its vertices. You can specify the coloring and lighting of the patch.

Polygon **`patch`**

`patch(X,Y,C)` adds the filled two-dimensional patch to the current axes. The elements of **`X`** and **`Y`** specify the vertices of a polygon. If **`X`** and **`Y`** are matrices, MATLAB draws one polygon per column. **`C`** determines the color of the patch. It can be a single **`ColorSpec`**, one color per face, or one color per vertex. If **`C`** is a 1-by-3 vector, it is assumed to be an RGB triplet, specifying a color directly.

`patch(X,Y,Z,C)` creates a patch in three-dimensional coordinates.

You must specify color data so MATLAB can determine what type of coloring to use. If you do not specify color data, MATLAB returns an error.

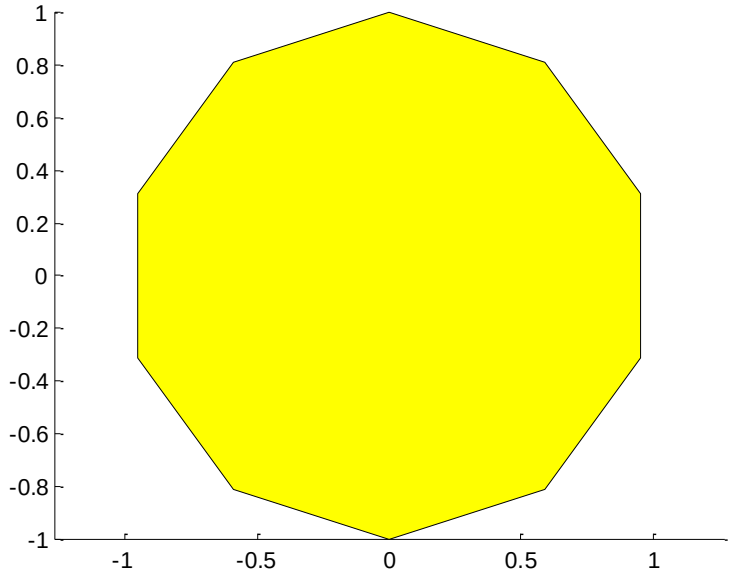
Polygon Example

A polygon is simply a patch with one face. To create a polygon, specify the coordinates of the vertices and color data with a statement of the form

```
patch(x-coordinates,y-coordinates,[z-coordinates],colordata)
```

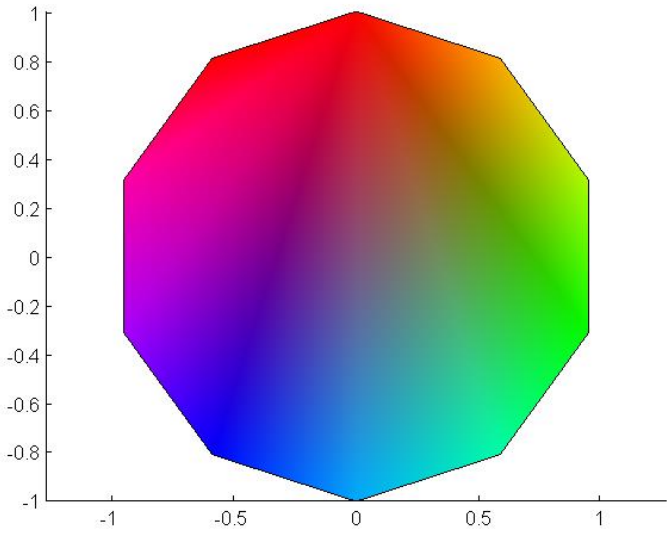
For example, these statements display a 10-sided polygon with a yellow face enclosed by a black edge. The axis equal command produces a correctly proportioned polygon.

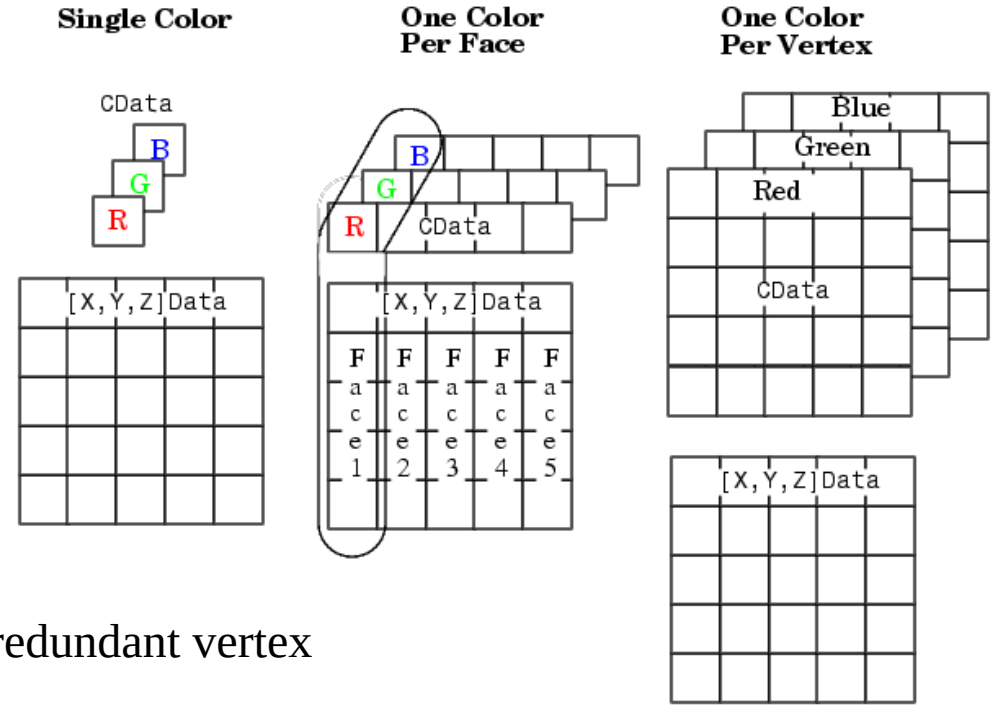
```
t = 0:pi/5:2*pi;
patch(sin(t),cos(t),'y')
axis equal
```



The first and last vertices need not coincide; MATLAB automatically closes each polygonal face of the patch. In fact, it is generally better to define each vertex only once.

Interpolating Vertex Colors





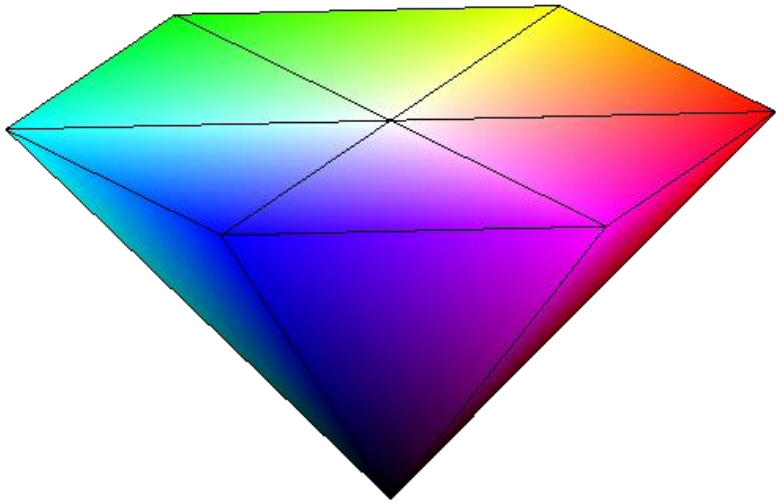
Remove redundant vertex

```
t = t(1:length(t)-1);
h = linspace(0,1,10);
s = ones(10,1);
v = s;
hsv = cat(3,h',s,v);
rgb = hsv2rgb(hsv);
close
patch(sin(t),cos(t),rgb);
axis equal
```

You can specify color for each vertex, each face, or a single color for the entire patch.

RGB Hexcone

These matrices specify the cube using Vertices and Faces. Using the vertices/faces technique can save a considerable amount of computer memory when patches contain a large number of faces. This technique requires the formal patch function syntax, which entails assigning values to the Vertices and Faces properties explicitly.

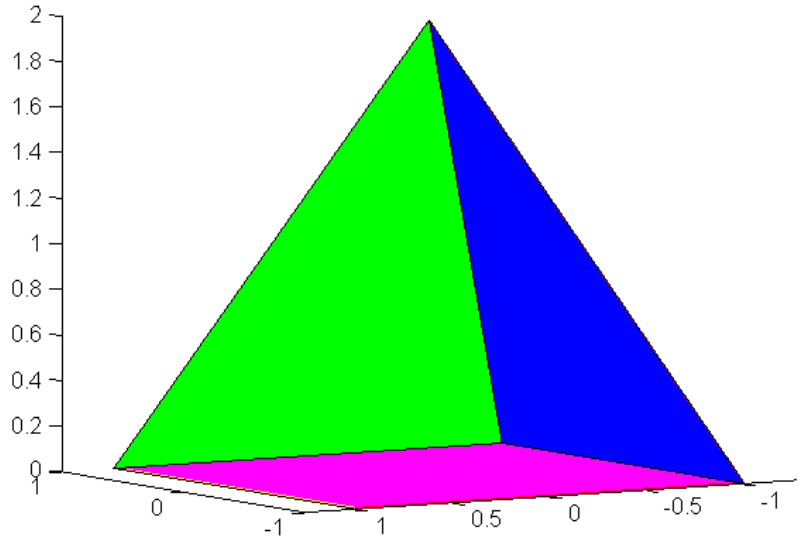


Because the high-level syntax does not automatically assign face or edge colors, you must set the appropriate properties to produce patches with colors other than the default white face color and black edge color.

```
close
t = (0:5)*pi/3;
x = [cos(t) 0 0]';
y = [sin(t) 0 0]';
z = [ones(7,1); 0];
vert = [x y z];
faces = [1 2 7; 2 3 7; 3 4 7; 4 5 7; 5 6 7; 6 1 7; ...
    8 1 2; 8 2 3; 8 3 4; 8 4 5; 8 5 6; 8 6 1];
rgb = [ 1 0 0; 1 1 0; 0 1 0; 0 1 1; 0 0 1; 1 0 1; 1 1 1; 0 0 0];
patch('Vertices',vert,'Faces',faces,'FaceVertexCData',rgb, …
    'FaceColor','interp','EdgeColor','w');
```

Faces with Varying Numbers of Vertices

MATLAB does not require each face to have the same number of vertices. In cases where they do not, pad the Faces matrix with NaNs. To define a patch with faces that do not close, add one or more NaNs to the row in the Vertices matrix that defines the vertex you do not want connected




```
close
vert = [ -1 -1 0; 1 -1 0; 1 1 0; -1 1 0; 0 0 2; NaN NaN NaN];
faces = [1 2 5 NaN; 2 3 5 NaN; 3 4 5 NaN; 4 1 5 NaN; 1 2 3 4];
rgb = [ 1 0 0; 1 1 0; 0 1 0; 0 0 1; 1 0 1];
patch('Vertices',vert,'Faces',faces,'FaceVertexCData',rgb,'Face
Color','flat');
axis equal
```

I added an extra row of vertices, otherwise **`faces`** and **`vert`** are the same length, and MATLAB gets confused about whether to do vertex coloring or face coloring.