Handle Graphics

- **Handle Graphics** refers to the numerous low level features that specifies how graphics are to behave and be displayed in MatLab.
- Handle Graphics objects (such as plots and graphs) can have their properties changed from their default values.
- These properties can be changed in **Figure** windows via various menus, toolbars, browsers and editors available within the windows. Hence, little knowledge of Handle Graphics is required when using these interactive tools.
- However, when manipulating graphics objects non-interactively in M files, we need to have some knowledge of Handle Graphics.

- There is such a wide diversity and breath of graphics features that it would impossible to give comprehensive overview of them in this lecture.

Objects

- Every component of a MatLab graphics is an **object**, each object has a **handle** or unique identifier (i.e. a **pointer**) associated with it and each object has **properties** that can be modified. Thus following the object oriented (OO) nomenclature, an object is a collection of data and functions that form a whole entity.

- All plotting and graphics in MatLab create graphics objects and manipulating their property values change the appearance of the plots and other graphics.

- Each object has a handle associated with it, which is a double precision number. The command `Hf_fig=figure` creates a new figure window and returns its handle in the `Hf` variable. Figure handles are usually small integers and usually displayed in the figure's window title bar. Other handles are large integers or even non-integer (double) numbers.

- A MatLab suggestion: start handle variable names with a capital H, followed one or more letters identifying the object type, then an underscore and finally one or more descriptive characters. Thus `Hf_fig` might be the handle variable for a figure, `Ha_ax1` might be a handle variable for an an axis object, `Ht_title` might be handle variable for a title object, `Hls` might be a handle variable for a lineseries plot object and `Hs` a handle variable for surface object.

Object Hierarchy

- Graphical objects are arranged into a parent-object hierarchy.
- The hierarchy starts with the **root** object, which is the MatLab session. When you start a MatLab session, a root object is immediately instantiated.
- All child objects must have an existing parent object to be created.
- For example, if you want to create an **image** object, first a **root** object, then a **figure** object, followed by an **axes** object, must exist.

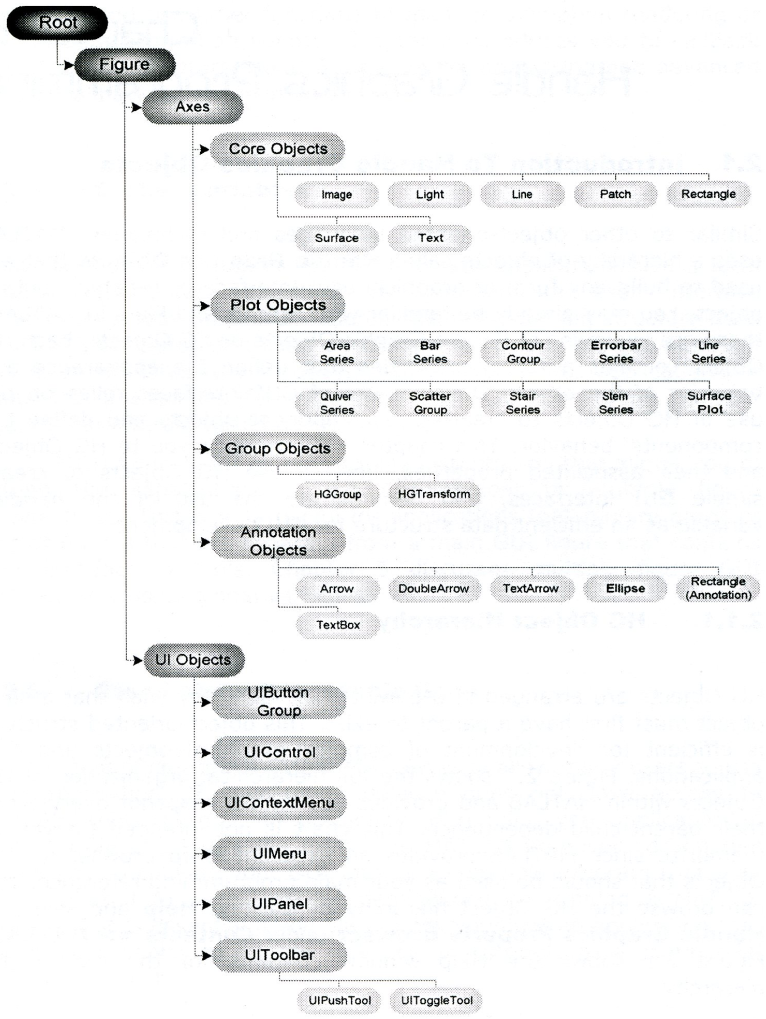
- This object-oriented structure is used to develop more complex graphical objects (especially for GUI applications).

- MatLab can create **composite** or **core** objects. Composite objects consists of other composite objects and/or core objects. Core objects are self-contained, i.e. they do not consist of other objects,

- All objects that are children of the figure object, except for the core objects, are composite objects.

- Core objects consist of image, light, line, patch, rectangle, surface and text objects.

- The figure below shows the MatLab handle graphics hierarchy:



MatLab Handle Graphics Hierarchy (from “MatLab Advanced GUI Development” by Scott T. Smith, page 10).

Object Properties

- All objects have properties that define their appearance and you can change these to change the appearance of an object.
- Object properties consist of property **names** and property **values**. Property names are always character strings.
- When an object is created it is initialized with default property values.
- Object creation functions can be issued with:

  ```
  'Property-name' - Property value
  ```

  tuples directly or indirectly by using the handle of an object and the **get** and **set** functions to modify these tuples.

- For example, `Hf_fig1=figure('color','yellow')` results in the figure:

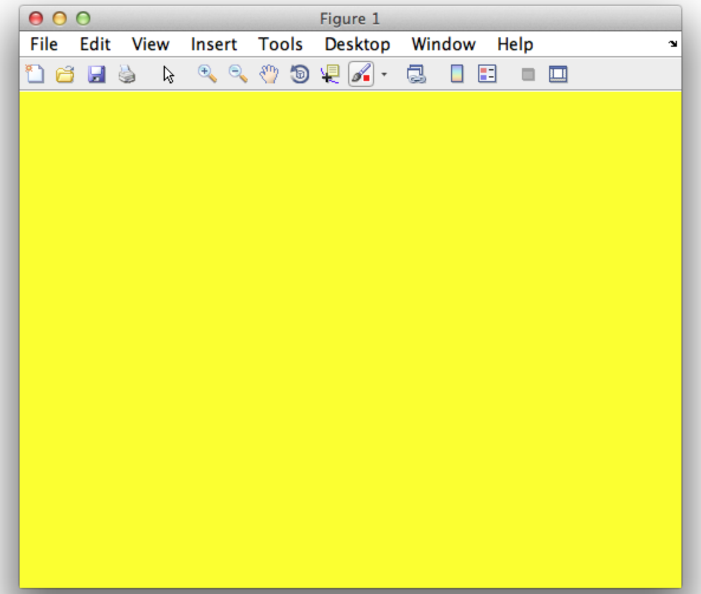
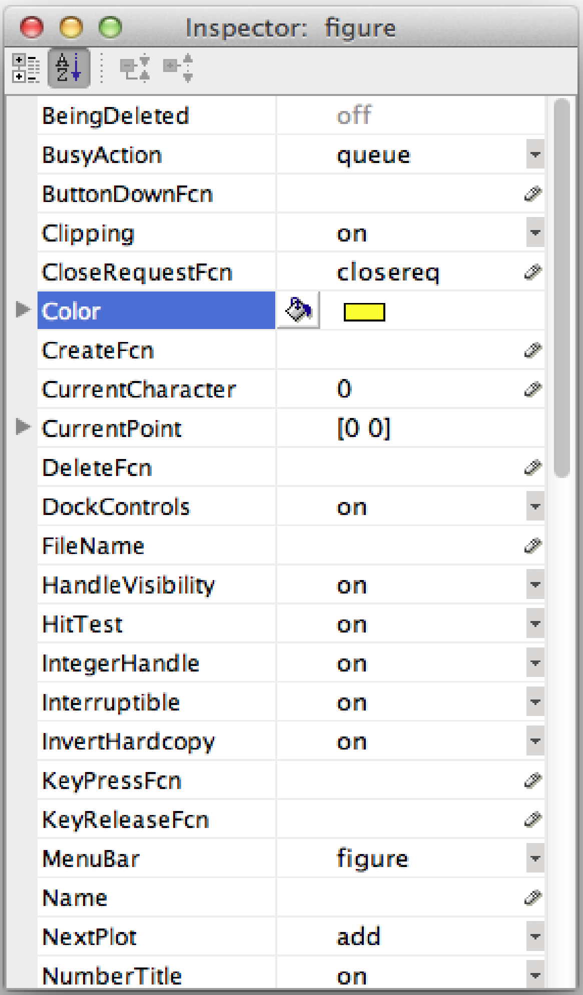


Figure with yellow background, Hf_fig1 has the value 1 if it is the first figure

- `inspect(Hf_fig1)` allows to interactively inspect and modify this figure's properties. If you click on the yellow colour, a colour palette will pop up. Chose red and the background colour will change. Here we see the menu window that pops up when inspect is executed and the result after changing the background colour of figure `Ff_fig1` to red.

- The problem with interactively changing windows' properties is that because you are not doing this in a program, you will have to do this repeatly for each figure each time the program is run.


Inspector: figure

BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
Clipping	on
CloseRequestFcn	closereq
Color	🟨
CreateFcn	
CurrentCharacter	0
CurrentPoint	[0 0]
DeleteFcn	
DockControls	on
FileName	
HandleVisibility	on
HitTest	on
IntegerHandle	on
Interruptible	on
InvertHardcopy	on
KeyPressFcn	
KeyReleaseFcn	
MenuBar	figure
Name	
NextPlot	add
NumberTitle	on

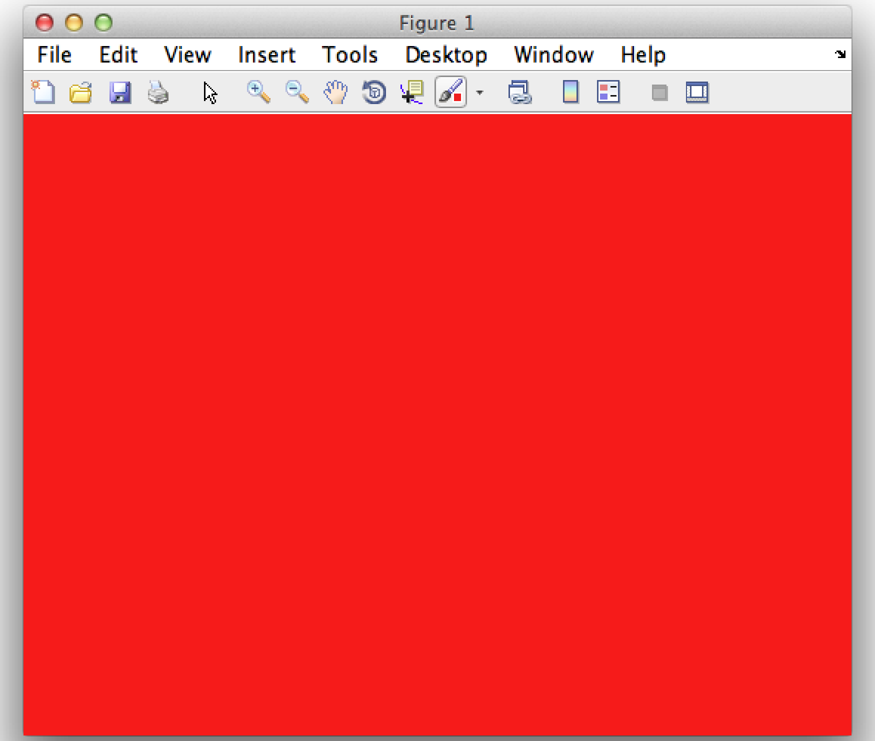


Figure with red background after the use of inspect(Hf_fig1)

Using get/set to Obtain/Change Object Properties

- The **get** and **set** functions can be used to change properties of graphical objects from the command window or in an M file.
- For example, `p_vec=get(Hf_1,'Position')` returns the position vector of the figure having handle `Hf_1` or q=`get(Hl_a,'color')` returns the colour of the object identified by `Hl_a`.
- On the other hand, `set(Hf_1,'position',p_vec)` and `set(Hl_a,'color','red')` set the position of the `Hf_1` to be `p_vec` and the colour of the `Hl_a` object to be red.
- In general, we get can have any number of name/value tuples, for example:

```
set(Hl_a,'color',[1 0 0],'linewidth',2,'linestyle','--')
```

changes of the `Hl_a` object to be red, its line width to be 2 points and its line style to be dashed. [Points is a text formatting term with 72 points being 1 inch.]

- The **get** and **set** functions can be used to provide information about object properties. For example, `set(Hf_fig1,'units')` returns:

```
[inches | centimeters | normalized | points | {pixels} | characters]
```

This shows that there are 6 possible units and `pixels` is the default. Many computer screens are 1024×768 pixels (width by height). Character units are units relative to the width of a character in the default system font. The origin is the lower lefthand point on the screen.

- For the MatLab code:

```
Hf_fig=figure;  % create a figure
Hl_light=light; % add a default light
set(Hl_light)
get(Hl_light)
```

- `set(Hl_light)` lists all the properties (and some possibilities for their values) of this light object while `get(Hl_light)` lists all the properties and their values for this light object.
- Note that a light object only has 21 properties and as such is the smallest object in MatLab! The axis property, on the other hand, has 103 properties! We will **not** be going through the properties of each object type in

this course!!!

- As an example of object handle usage, consider the following MatLab code (in **L11handle_graphics_example.m**):

```
x=-2*pi:pi/40:2*pi;
y=sin(x); % sine of x
% plot sine and save lineseries handle
Hls_sin=plot(x,y)
% change colour (to medium orange)
% and linewidth (to 3)
set(Hls_sin,'color',[1 0.5 0],...
            'linewidth',3);
hold on
```

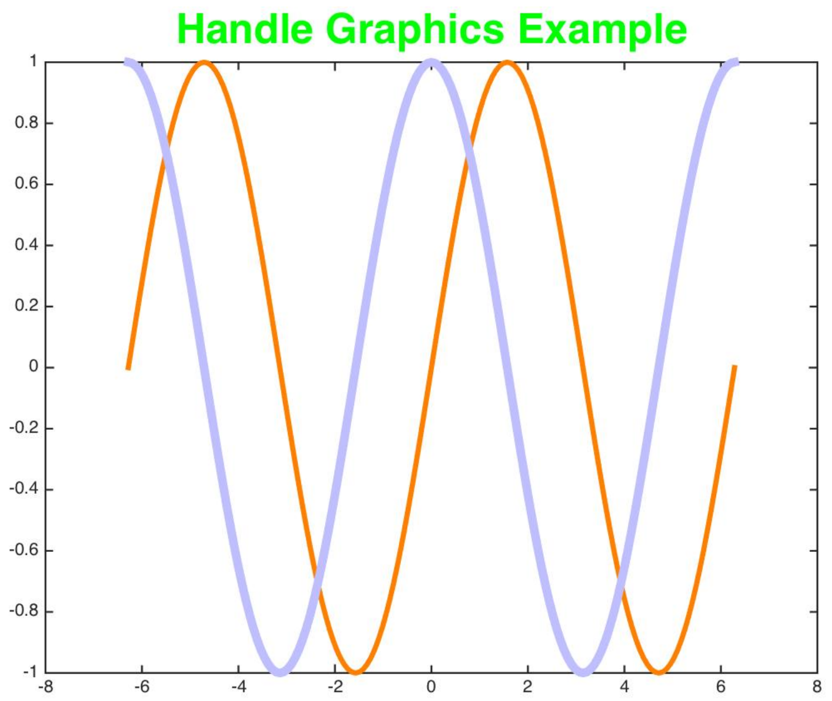
```
pause


z=cos(x); % cosine of x

Hls_cos=plot(x,z);

% colour light blue

set(Hls_cos,'color',[0.75 0.75 1]);

hold off

title('Handle Graphics Example');

pause


% gca: get current axis handle

Ht_title=get(gca,'title')
```

```
set(Ht_title,'fontsize',24,...
             'color','green');
print handle_graphics_example.jpg -djpeg
```

The following is printed:

```
Hls_sin =

  174.0073

Hls_cos =

  175.0068

Ht_title =

  176.0068
```

and the following plot occurs:

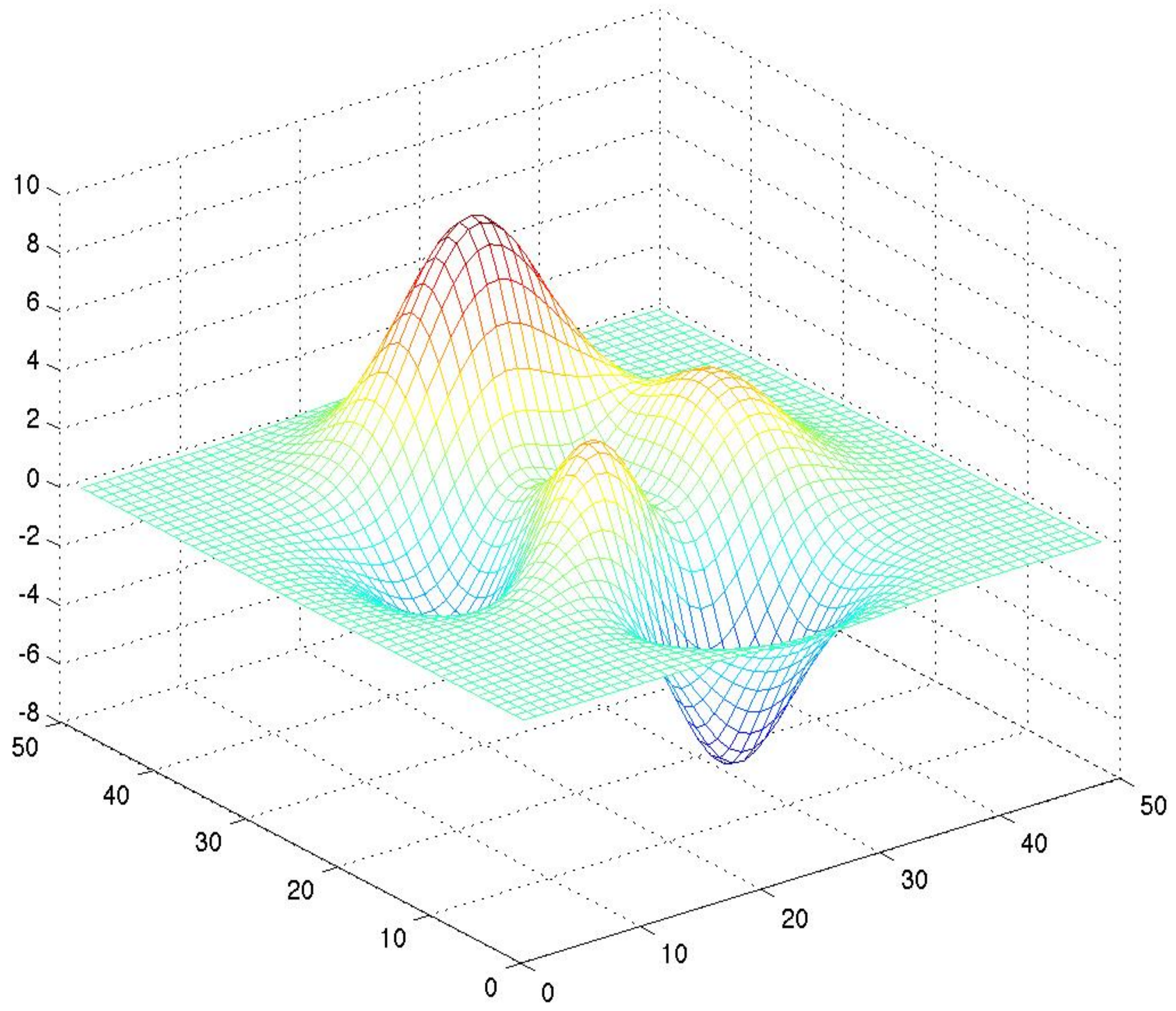


An example of Handle Graphics Usage

- There are many other weird and wonderful things you can do to objects, including copying objects, moving objects, deleting objects and reset objects properties to the defaults (where it makes sense).
- How to find and change properties of more complex objects like `surf` or `mesh`? First, you have to obtain a list of the the object's properties for a particular instance of the object. Use the MatLab code (in **L11mesh_plot_properties.m**):

  ```
  % create standard mesh plot of peaks
  Hm=mesh(peaks);
  print mesh_plot1.jpg -djpeg
  get(Hm)
  ```

  to obtain the plot:



Original mesh plot with all default MatLab property values and the list:

```
          AlphaData: 1
   AlphaDataMapping: 'scaled'
         Annotation: [1x1 hg.Annotation]
              CData: [49x49 double]
       CDataMapping: 'scaled'
        DisplayName: ''
          EdgeAlpha: 1
          EdgeColor: 'flat'
          FaceAlpha: 1
          FaceColor: [1 1 1]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
    MarkerEdgeColor: 'auto'
    MarkerFaceColor: 'none'
         MarkerSize: 6
          MeshStyle: 'both'
              XData: [1x49 double]
              YData: [49x1 double]
              ZData: [49x49 double]
       FaceLighting: 'none'
       EdgeLighting: 'flat'
   BackFaceLighting: 'reverselit'
```

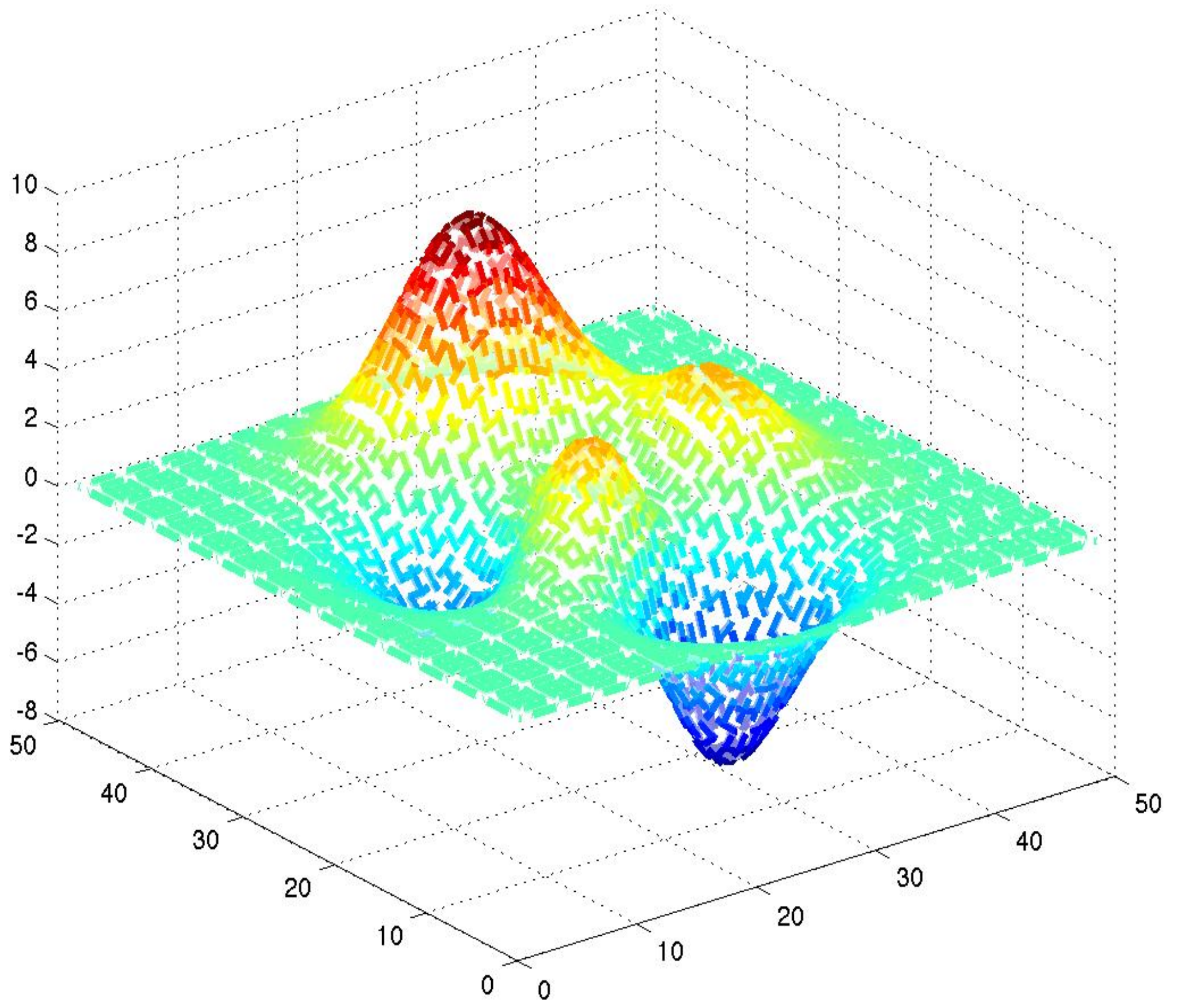
```
           AmbientStrength: 0.3000
           DiffuseStrength: 0.6000
          SpecularStrength: 0.9000
          SpecularExponent: 10
  SpecularColorReflectance: 1
             VertexNormals: [49x49x3 double]
                NormalMode: 'auto'
              BeingDeleted: 'off'
             ButtonDownFcn: []
                  Children: [0x1 double]
                  Clipping: 'on'
                 CreateFcn: []
                 DeleteFcn: []
                BusyAction: 'queue'
          HandleVisibility: 'on'
                   HitTest: 'on'
             Interruptible: 'on'
                  Selected: 'off'
        SelectionHighlight: 'on'
                       Tag: ''
                      Type: 'surface'
             UIContextMenu: []
                  UserData: []
                   Visible: 'on'
       Parent: 173.0018
    XDataMode: 'auto'
  XDataSource: ''
    YDataMode: 'auto'
  YDataSource: ''
    CDataMode: 'auto'
  CDataSource: ''
  ZDataSource: ''
```

You can now change these properties if you know what you are doing!!!

For example:

```
set(Hm,'linewidth',3.0,'linestyle','--',...
        'facealpha',0.5);
print mesh_plot2.jpg -djpeg
```

changes the displayed mesh plot to:



Original mesh plot with changed linewidth from 0.5 to 3, linestyle changed from '-' (solid) to '–' (dashed) and facealpha changed from 1 to 0.5 (some of the grid behind the plot is now visible as can be seen in a higher resolution rendering of this plot using “xv mesh_plot2.jpg”)

Finding Objects

- In order to use handle graphics, you have to know the handles of objects you want to manipulate. MatLab provides some functions to find object handles in situations where you might not know them.
- **gcf** returns the handle of the current figure.
- **gca** returns the handle of the current axes of the current figure.

Default Properties

- MatLab assigns default values to each object when it is created. These are known as the **factory** defaults. You can override these values using **get** and **set** as shown above.

- MatLab allows you to set your own default property values. Use a special property-name string consisting of `'Default'`, followed by the object type and then the property name. For example:

  ```
  set(0,'DefaultFigureColor',[0.5 0.5 0.5])
  ```

  sets the default background colour for all new objects to be medium gray. The root object's handle is always 0 and so all new objects, which are children of root, will have medium gray background.

- Other examples:

  ```
  % larger axes - all figures
  set(0,'DefaultAxesFontSize',14)
  ```

```
% thick axis lines - this figure only
set(gcf,'DefaultAxesLineWidth',2)
% yellow X axis lines and labels
set(gcf,'DefaultAxesXColor','yellow')
% Y axis grid lines - this figure
set(gcf,'DefaultAxesYGrid','on')
% enclose axes - all figures
set(0,'DefaultAxesBox','on')
% dotted linestyle - these axes only
set(gca,'DefaultLineLineStyle',':')
```

- When a default property is changed, only objects created after the change are affected. Existing objects remain unchanged.

- It is always a good idea to restore default values after your code has finished executing. One way to do this is to store the previous settings and restore them after finishing the execution:

  ```
  oldunits=get(0,'DefaultFigureUnits');
  set(0,'DefaultFigureUnits','normalized');
  ```

  The MatLab code to restore these settings:

  ```
  set(0,'DefaultFigureUnits',oldunits);
  ```

- To make MatLab use your user-defined default values all the time, simply include the desired **set** commands in your **startup.m** file. For example, the **L11startup.m** file:

```
% store as startup.m
set(0,'DefaultAxesXGrid','on')
set(0,'DefaultAxesYGrid','on')
set(0,'DefaultAxesZGrid','on')
set(0,'DefaultAxesBox','on')
set(0,'DefaultFigurePaperType','A4')
```

creates all axes with grids, an enclosing box is turned on and the default paper size is set to `A4` (European). The default paper size is `usletter`.

Undocumented or Hidden Properties

- MatLab has some undocumented or hidden properties used by MatLab developers. Some of these can be modified but others are read only.

- Since these properties are undocumented it is a bit “dangerous” to use them. They can be less robust then documented properties and subject to unannounced changes or they could become documented in later versions of MatLab.

- Since they are undocumented, there is no documentation of list of these properties. However, the MatLab code:

  ```
  get(0) % get documented properties
  set(0,'HideUndocumented','off')
  get(0) % get both documented and
         % undocumented properties
  ```

  will produces 2 lists of properties for the root (handle 0): the documented properties and the documented and undocumented properties.

Group Objects

- Group objects are associated with high level graphics functions (for example, the 2D and 3D plots covered earlier, like surfaces or meshes).
- They allow the grouping of core objects, for example, images, lights, lines, patches, rectangles, surfaces or text. So the properties of such a group object are the properties of its object entities plus some additional properties. These additional properties specify properties dependent on the type of plot created. For example, a **bar** plot has **patch** objects as its children.

- Each group object is the parent of one or more core objects.

Annotation Objects

- The “Annotation Axes” in the handle graphics hierarchy is simply a core object axes with its handle visibility turned off. You create such objects interactively using the numerous menu, toolbars, palette, browsers and editors available with a Figure window.

Linking Objects

- In some cases, it is desirable to link identical properties of a number of objects, so that changing this properties for any object changes it for all other linked objects.