

---

---

---



---

---

---


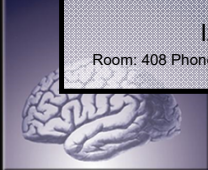
---

---



# Introduction to Programming 2017/18

## Graphic handles



Izhar Bar-Gad  
Room: 408 Phone: 7141 Email: [bargadi@mail.biu.ac.il](mailto:bargadi@mail.biu.ac.il)

---

---

---





---

---

---

---

---



IBG

## Advanced graphics

- Basic graphics in MATLAB are performed using a limited number of functions, allowing access to limited properties of the figures using **procedural programming**.
- Advanced graphics in MATLAB are performed using handles, allowing full access to the object using **object oriented programming**

---

---

---





---

---

---

---

---



IBG

## Objects

- An object is an individual unit of run-time data storage which serves as the basic building block of programs.
- Objects act on each other, as opposed to a traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer.
- Each object is capable of receiving messages, processing data, and sending messages to other objects.
- Each object can be viewed as an independent little machine or actor with a distinct role or responsibility.

(wikipedia)

---

---





---

---

---

---

---

	<b>Week 9 - Graphic handles</b>
	<ul style="list-style-type: none"><li>■ Handles</li><li>■ Properties</li><li>■ Hierarchies</li></ul>
	
	
IBG	

---

---





---

---

---

---

---

	<b>Managing figures – the problem</b>
	<ul style="list-style-type: none"><li>■ Multiple figures coexist simultaneously.</li><li>■ Altering figures is not performed in a sequential manner.</li><li>■ How can we change a specific aspect of a specific figure?</li></ul>
	
	
IBG	

---

---





---

---

---

---

---

	<b>Managing figures – the solution</b>
	<ul style="list-style-type: none"><li>■ The solution is in administrating the figures and their parts using unique IDs – <b>Handles</b>.</li><li>■ The handle provides a unique identifier to the graphical object.</li><li>■ Changing specific aspects of the figure requires accessing the handle in a pre-defined (or agreed) manner – <b>Properties</b>.</li></ul>
	
	
IBG	

---

---

---

---

---

---

---

---



---

---

---

---

---

---

---

---



---

---

---


---

---

---


---

---



## Handles

- The handle is a unique **ID** of the object.
- (Pre 2014B) The handle is a **number** (scalar) which is the ID of a complex object with many properties. The number of the handle has a meaning only as long as the object exists.
- (Post 2014B) The handle is an **object** belonging to a specific class. Allowing both procedural and object-oriented access.

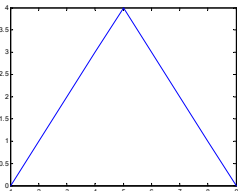



## Figure handle - example

- A *figure* or *plot* command will return a handle to the figure.
- Example
 

```
>> a=[0 1 2 3 4 3 2 1 0]
>> h=plot(a)
```

(<2014B) h =  
       159.0178  
 (≥2014B) h =  
       Line with properties:  
       ...





## Properties

- The handle enables access to an **object** which has many properties.
- Properties constitute of **name & value** pairs.
  - Property names are their identifiers and are always a string.
  - Property values may be of any data type including other handles...
- To enable proper access an object must have predefined names and data types for each property !

---

---

---


---

---

---

---

---

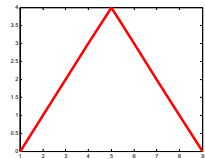


## Accessing properties I

- A property may be read via `propValue = get(h,propName);`
- A property may be changed via `set(h,propName,propValue);`

**Example**

```
>> a=[0 1 2 3 4 3 2 1 0];
>> h = plot(a);
>> set(h,'Color',[1 0 0])
>> set(h,'LineWidth',5)
```



*Comment: colors are defined by [R G B] each in the range 0-1.*

---

---

---


---

---

---

---

---



## Accessing properties II

- `get(h)` with no property name displays all properties.

**Example:**

```
>> get(h)
Color: [0 0 1]
EraseMode: 'normal'
LineStyle: '-'
LineWidth: 0.5000
Marker: 'none'
MarkerSize: 6
MarkerEdgeColor: 'auto'
MarkerFaceColor: 'none'
XData: [1x21 double]
YData: [1x21 double]
ZData: [1x0 double]
BeingDeleted: 'off'
...
```

---

---

---


---

---

---

---

---



## Properties

- Properties are specific to the handle type i.e. a handle to text is going to have different properties from a handle to a line.
- All the properties may be found using the general `get` and explained using the `help` (or `helpdesk`). However, generally properties are not well documented !
- A few of the most popular properties will be covered in the exercise, but since there are a LOT of properties we will only sample them.
- An exception to the "computer oriented" properties is the `UserData` property which is used to store user-specified data.
  - Default: []
  - Available for all objects.
  - Can be used to store any relevant data about the object.

---

---

---


---

---

---

---

---



IBG

### Properties (≥2014B)

- Starting from 2014B handles are **objects**.
- Direct access to properties:

```
>> h = plot([0 1 2 3 4 3 2 1 0]);  
  
>> a = h.LineStyle  
a =  
-  
>> h.lineStyle = ':'
```

---

---

---


---

---

---

---

---



IBG

### Handling multiple figures

- Use the *figure* command with no parameters to open a new figure.  
Syntax: *h=figure*;
- Use the figure command with a handle parameter to access a pre-opened figure.  
Syntax: *figure(h)*

---

---

---


---

---

---

---

---



IBG

### Current graphics

- Handle to the current figure - *gcf*
- Handle to the current axes - *gca*

---

---

---


---

---

---

---

---

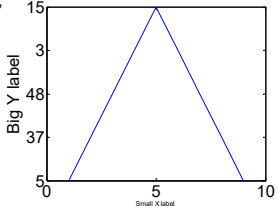


### Current graphics - example

```

>> plot(a);
>> xlabel('Small X label');
>> set(gca,'FontSize',24);
>> set(gca,'YTickLabel',[5 37 48 3 15]);
>> ylabel('Big Y label');

```



---

---

---


---

---

---

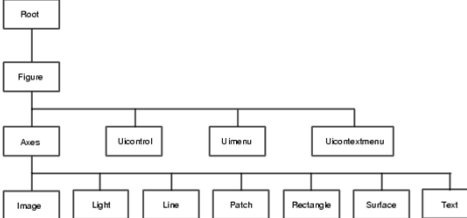
---

---



### Handle hierarchy

- Handles are organized in an hierarchical tree structure.



---

---

---

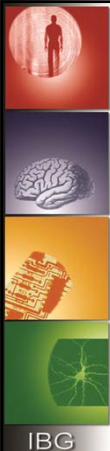
---

---

---

---

---



### Handle hierarchy - example

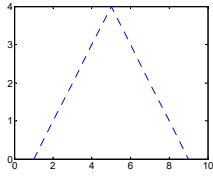
- The hierarchy is based on **children** and **parent** objects.
- The relation is **one to many** → only one parent for each object but many children.
- It is crucial to know which handle you are using or getting

```

% Example:
% Plot returns handle to line
h = plot(a);
p = get(h,'Parent');
set(h,'LineStyle','-');
set(p,'FontSize',16);
nh = get(p,'Children');

```

Note: nh is equal to h !!!



---

---

---


---

---

---

---

---



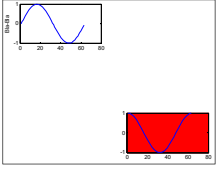
## Handle hierarchy - subfigures

- Subplots are handled as **children** of the figure.
- Example:

```

>> figure;
>> a=[0:1:2*pi];
>> subplot(3,2,1);
>> plot(sin(a));
>> subplot(3,2,6);
>> plot(cos(a));
>> childHandle = get(gcf,'Child');
>> set(childHandle(1),'Color',[1 0 0]);
>> ylabelHandle = get(childHandle(2),'YLabel');
>> set(ylabelHandle,'String','Bla-Bla');

```



---

---

---


---

---

---

---

---



## Object oriented programming

- Objects, handles and properties are part of the bigger picture of object oriented programming.
- Many parts of MATLAB use objects including files, I/O, GUI and most of the toolboxes.
- We will deal more with objects in the late stages of the course.
- Typically languages are either procedural or object oriented. MATLAB started as procedural but currently utilizes a mixed approach.

---

---

---


---

---

---

---

---



## Property editor

- The property editor is a GUI tool for changing many properties interactively.
- The tool is useful for **interactive** programming but not for **batch** executions of code.
- The property editor is context sensitive to the chosen object.