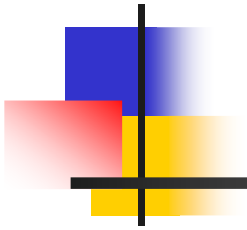
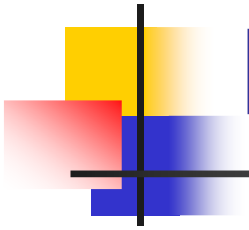
Graphical MATLAB



Lesson 1: Programming a GUI

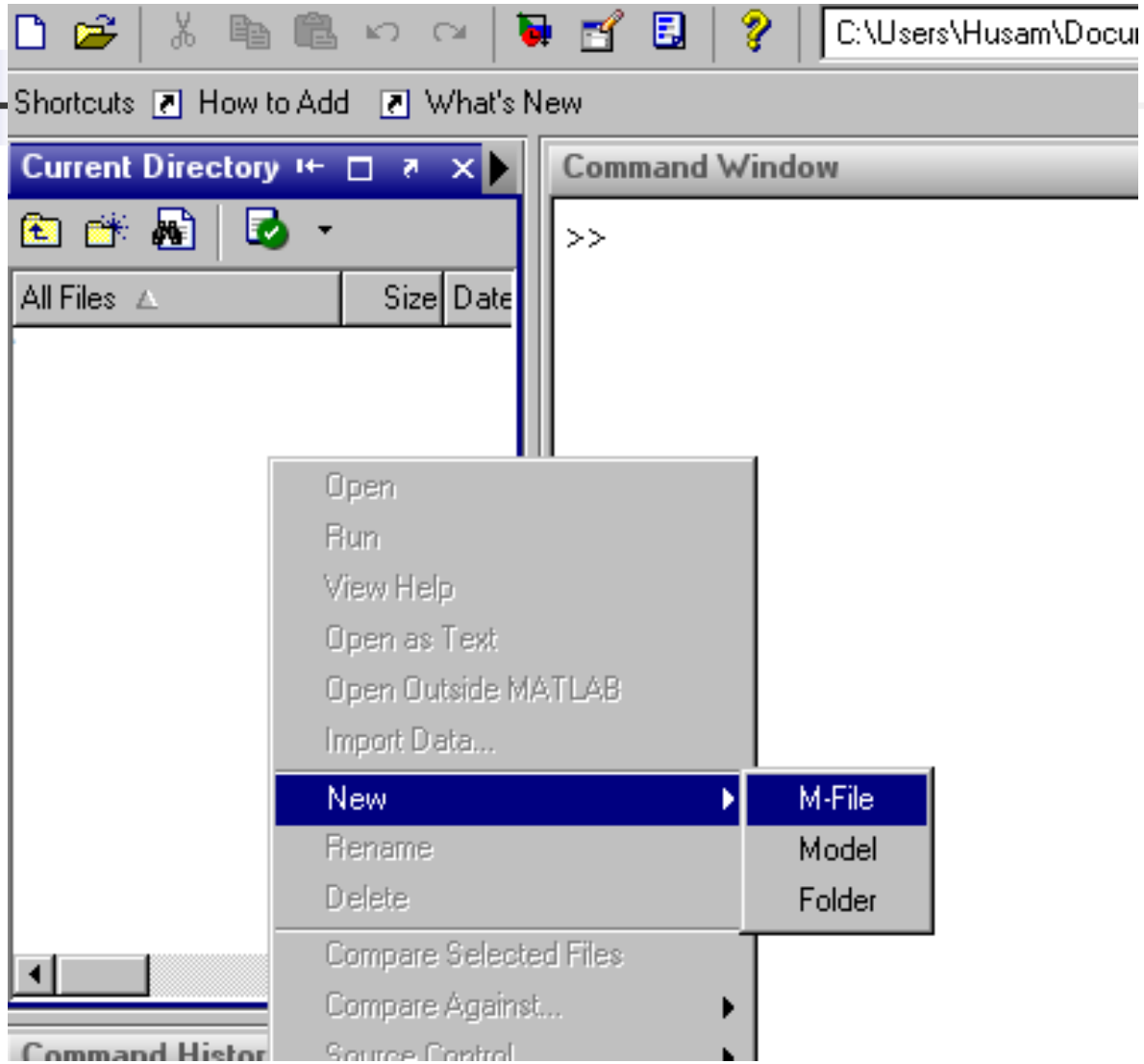
Husam Aldahiyat



Functions

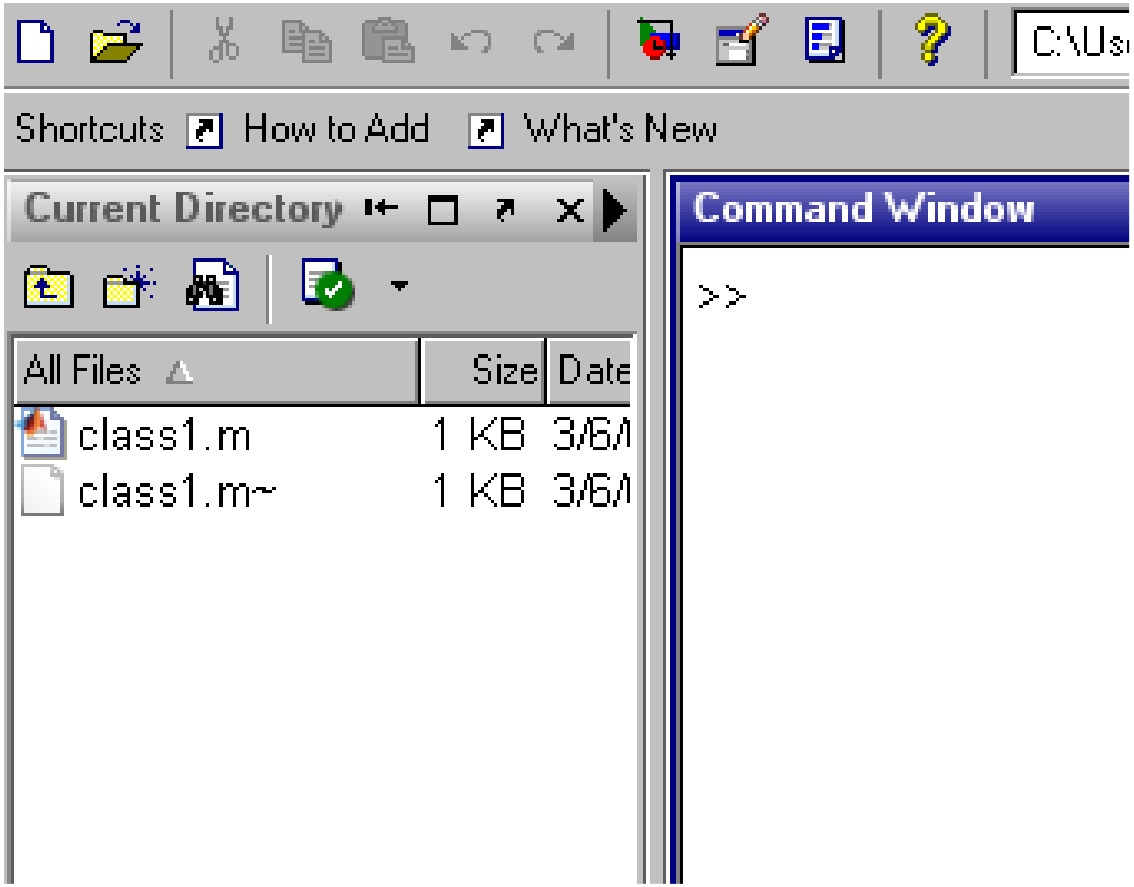
- Each function has inputs and outputs
- For our GUI, we need a function with no inputs or outputs
- This is because the function only calls the GUI

Creating an m File



Naming the m File

- Be sure the m file has the same name as the function



Creating the Function

- Make sure the function name is the same as the m file's.
- No input
- No output

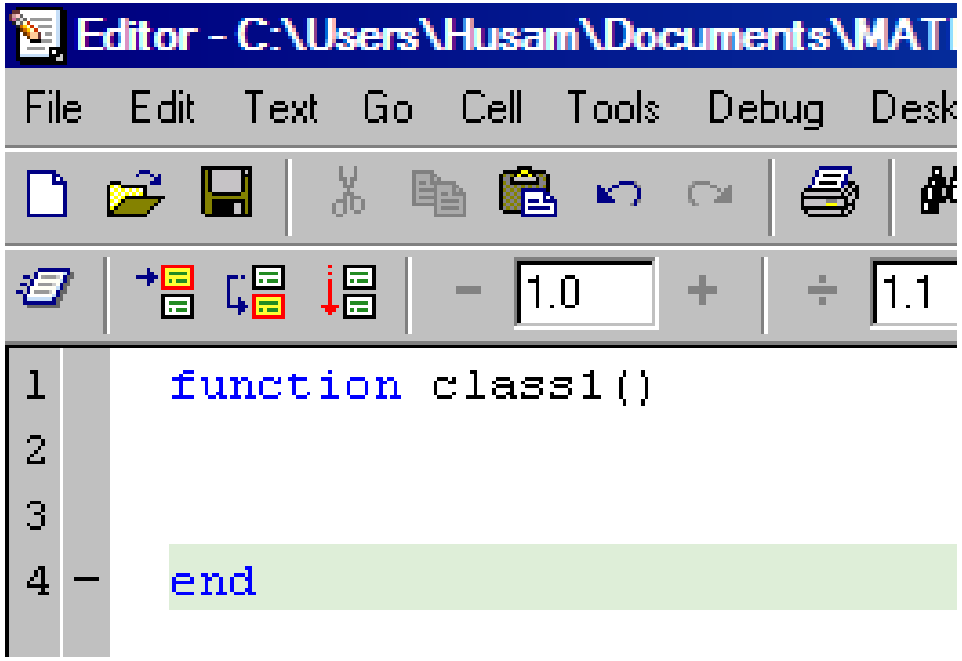
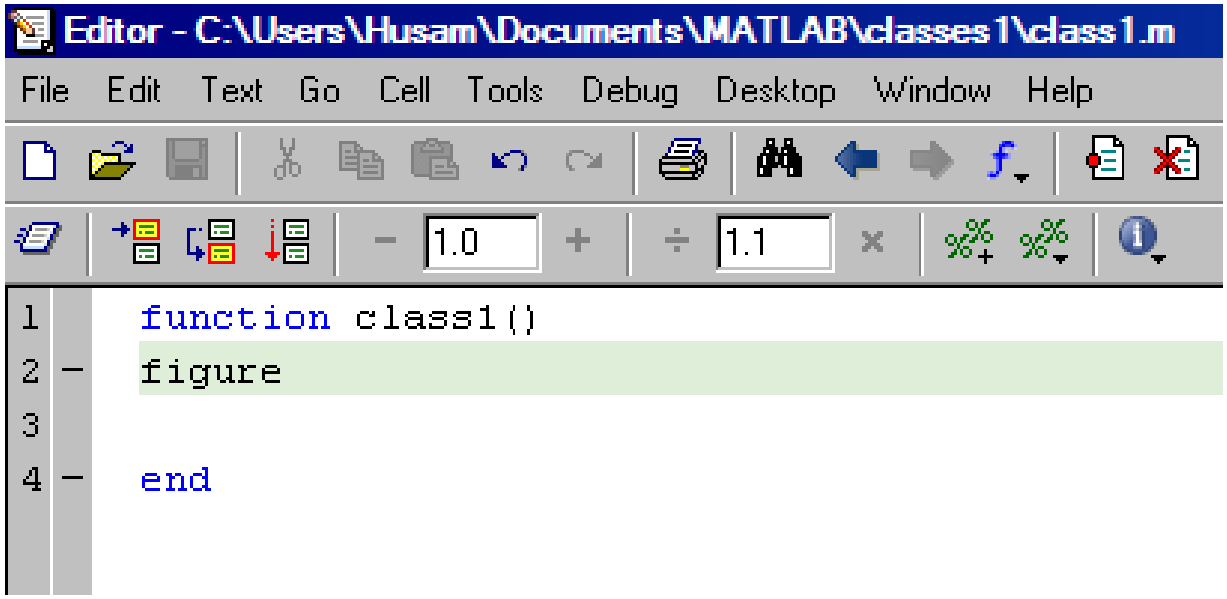


Figure Command

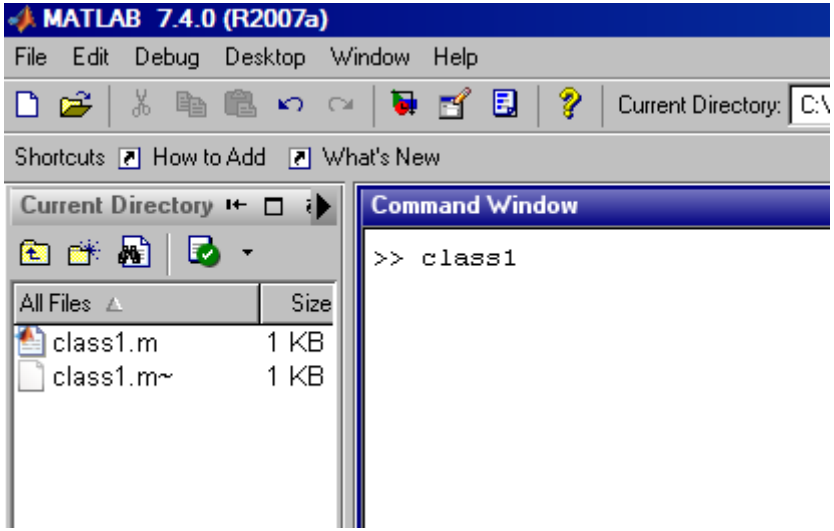
- Creates a FIGURE with default properties



Running the Code

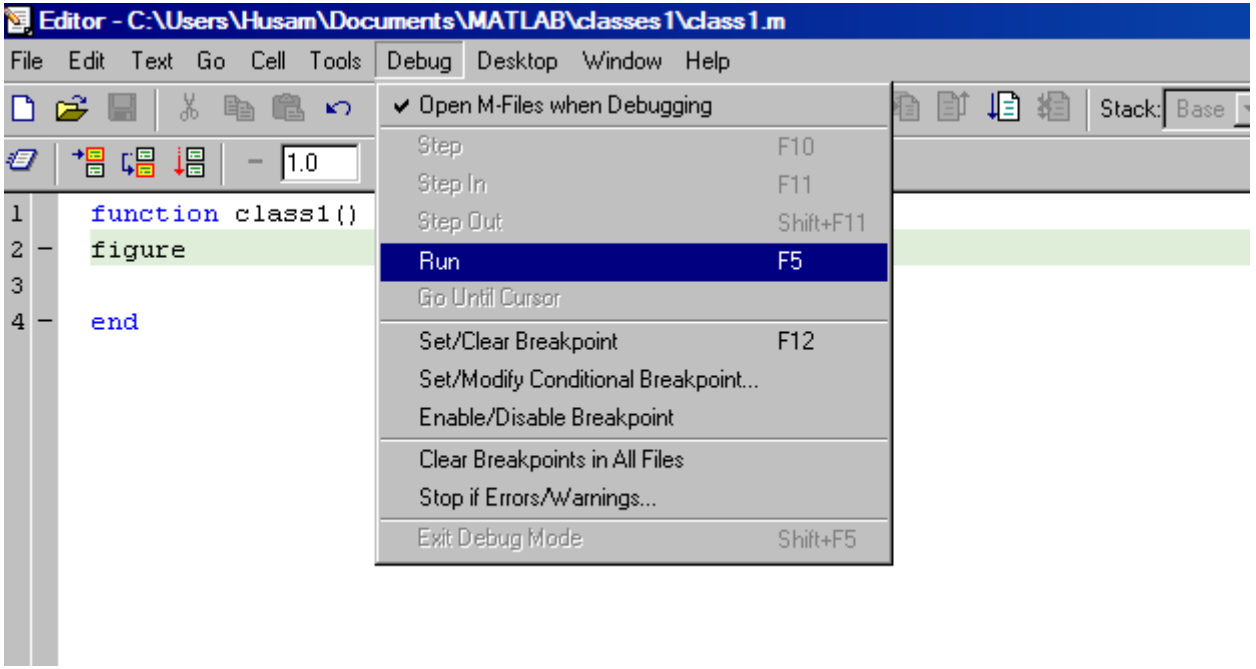
- To run the code, do one of the following:

- 1. Calling it in the main window



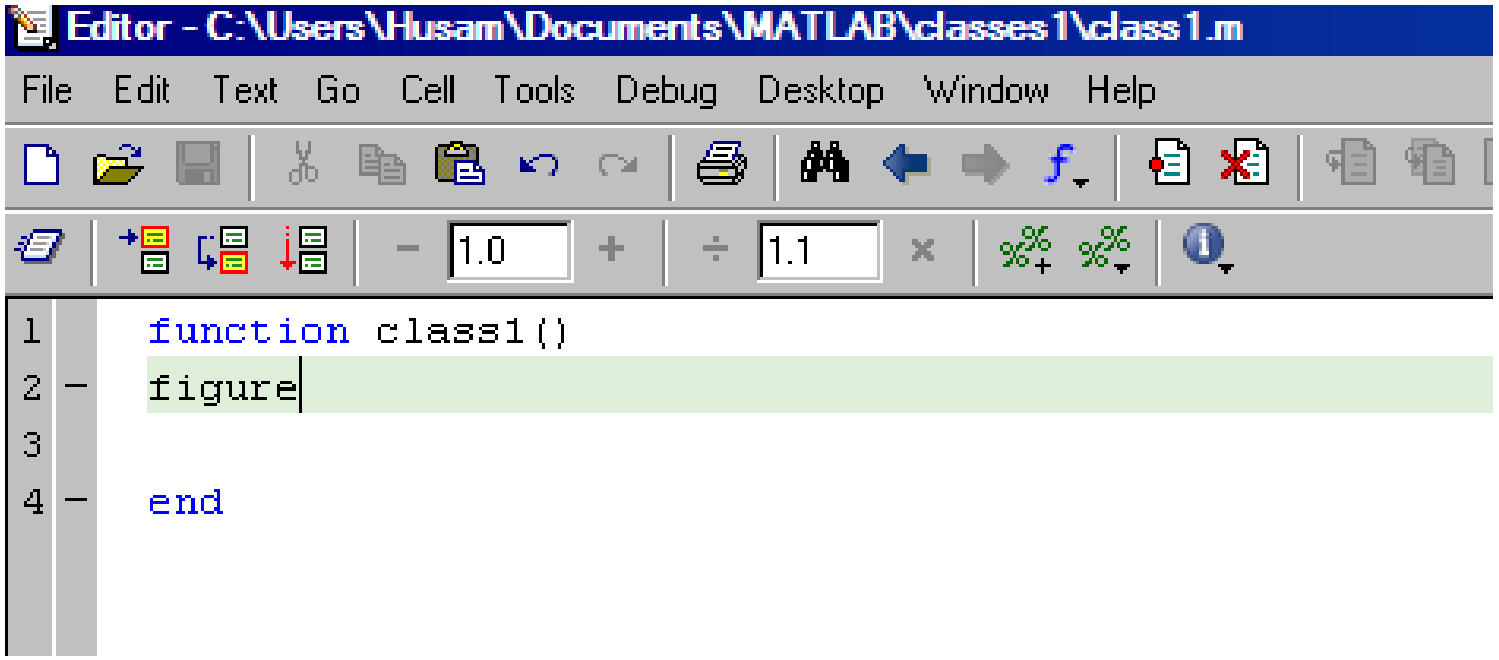
Cont.

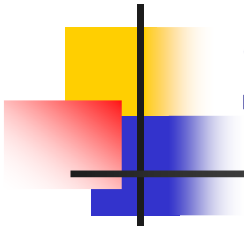
- 2. Go to Debug->Run in editor window



Cont.

- Use the shortcut F5 from editor window





Setting Figure Properties

- Figures have properties and property values.

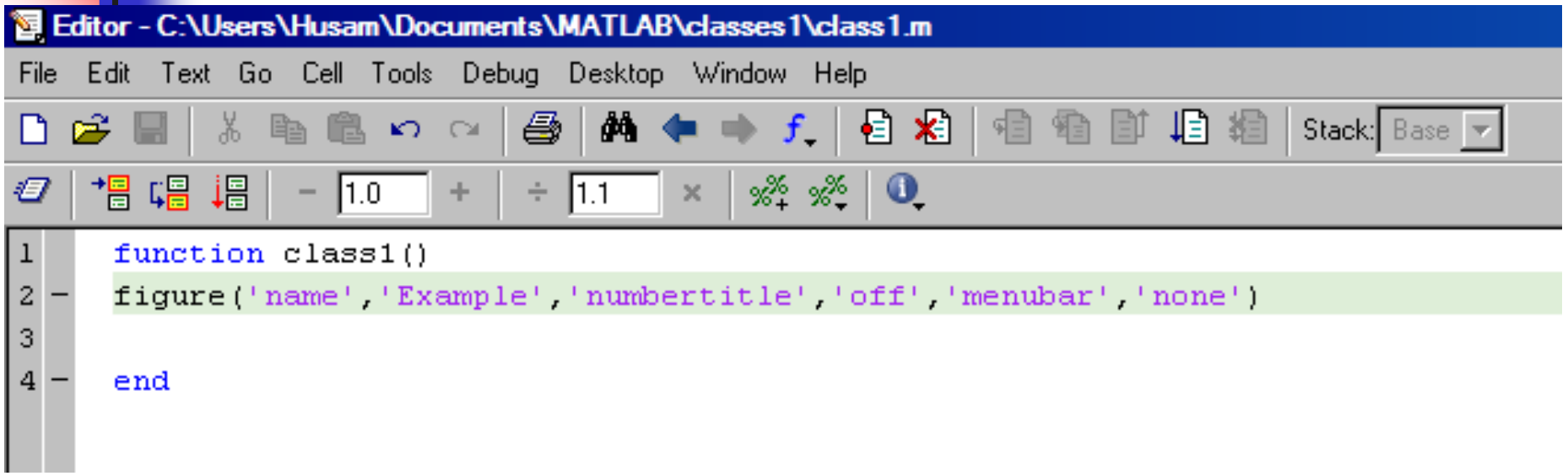
- Syntax for setting figure properties is:

figure('PropertyName',propertyvalue,…)

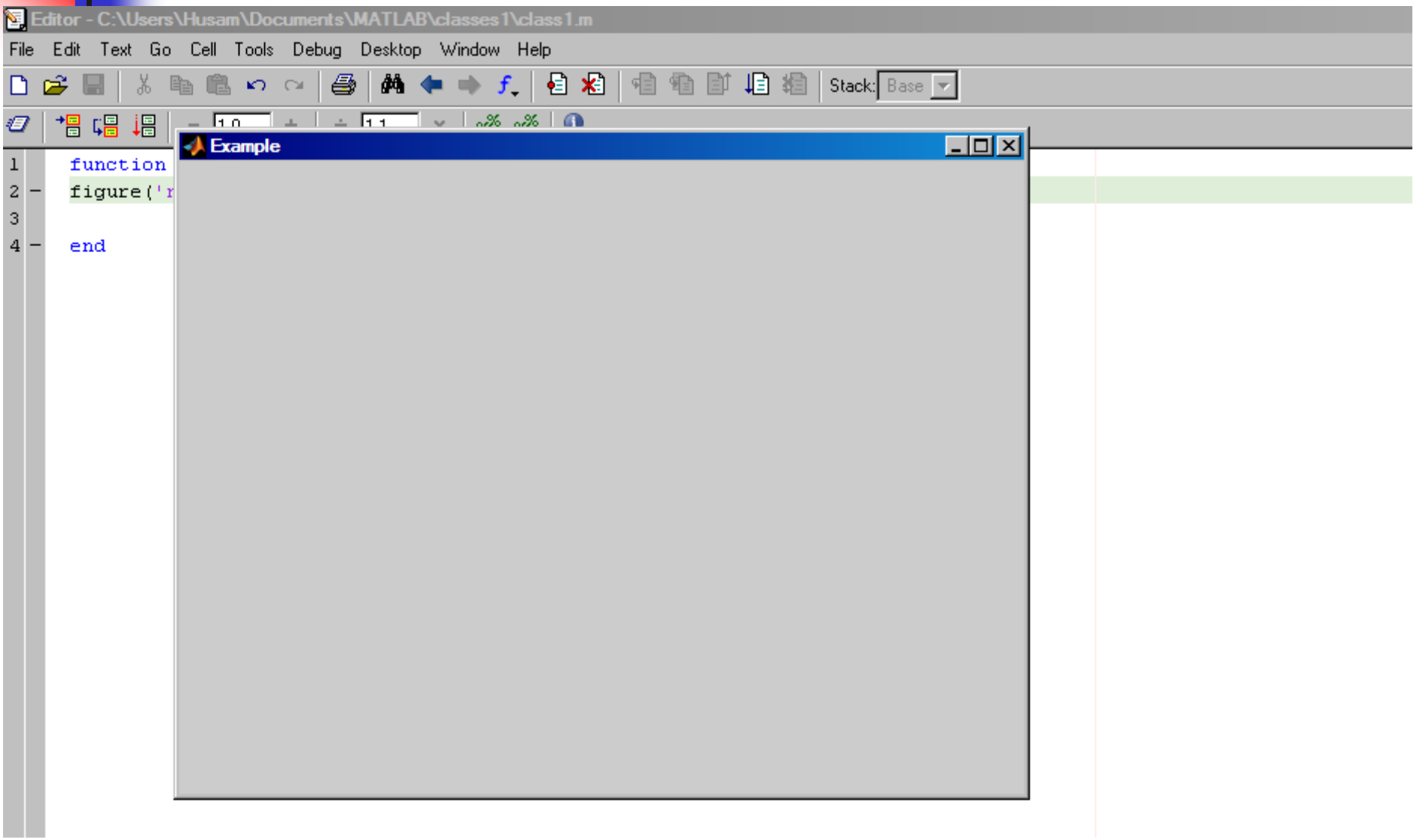
Some Figure Properties

PropertyName	PropertyValue	Example
‘name’	string	‘Example’
‘MenuBar’	‘none’ {‘figure’}	‘none’
‘NumberTitle’	{‘on’} ‘off’	‘on’
‘color’	[R, G, B]	[0.5, 0.9, 0.7]
‘units’	{‘pixels’} ‘normalized’ ‘inches’ ‘centimeters’ ‘points’ ‘characters’	‘inches’
‘position’	[x, y, w, l]	[2, 3, 2, 1]

Setting Figure Properties



Running the Function



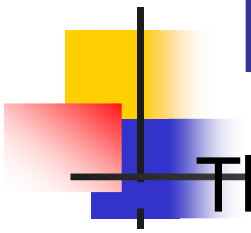


Figure Property: Color

The figure property ‘color’ takes the vector value [R, G, B], where each vector element takes a value in the range [0, 1].

Examples:

- [0, 1, 0] = GREEN
- [1, 1, 0] = YELLOW
- [1, 1, 1] = WHITE
- [0.5 0.5 0.5] = GRAY
- [0.5 0.2 0.75] = PURPLE

Using "..."

- Makes for better readability

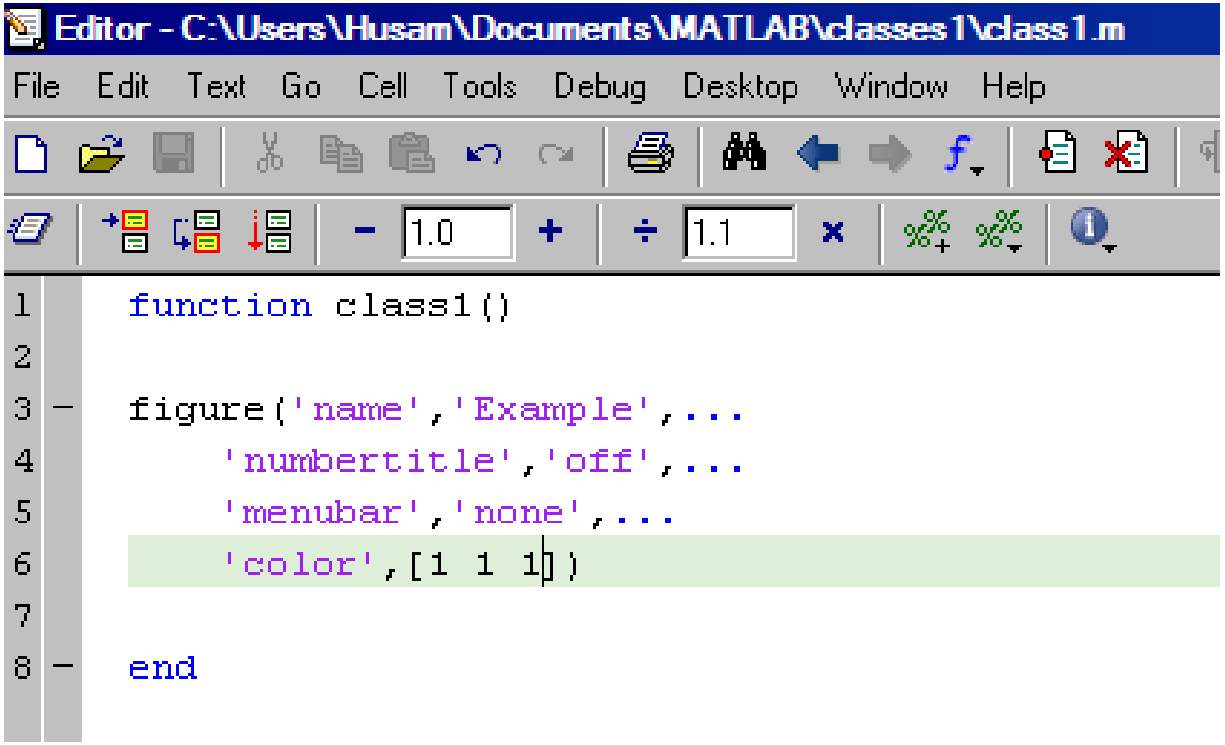
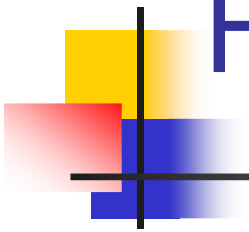
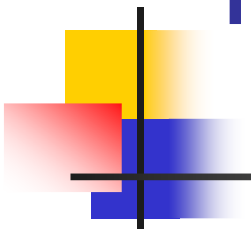


Figure Property: Position



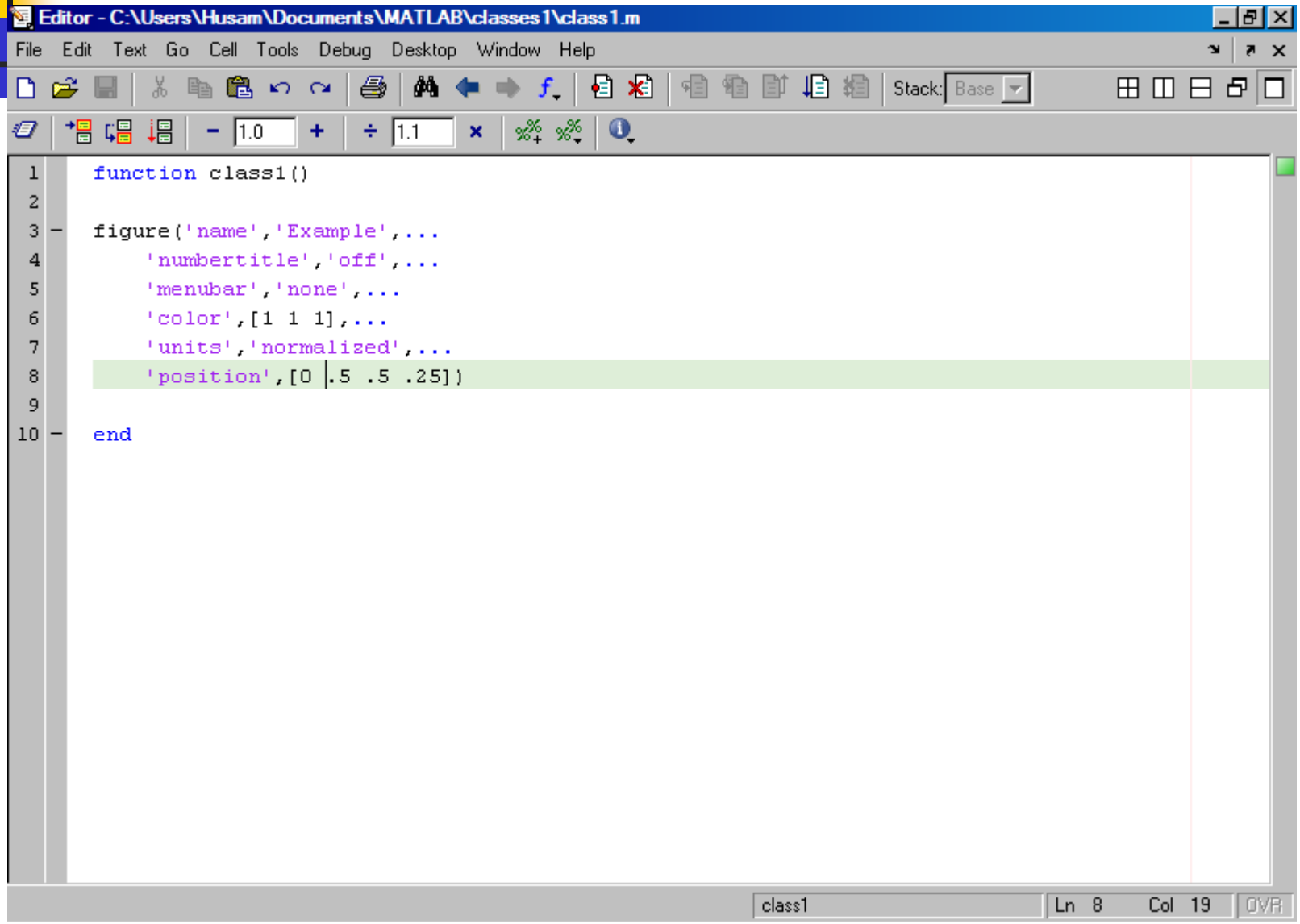
- Takes the vector value [x, y, w, l]
- Each element is a value in the units specified by the figure property ‘units’
- x- Distance from left side of the screen
- y- Distance from bottom side of the screen
- w- Width of figure (horizontally)
- l- Length of figure (vertically)

Figure Property: Units

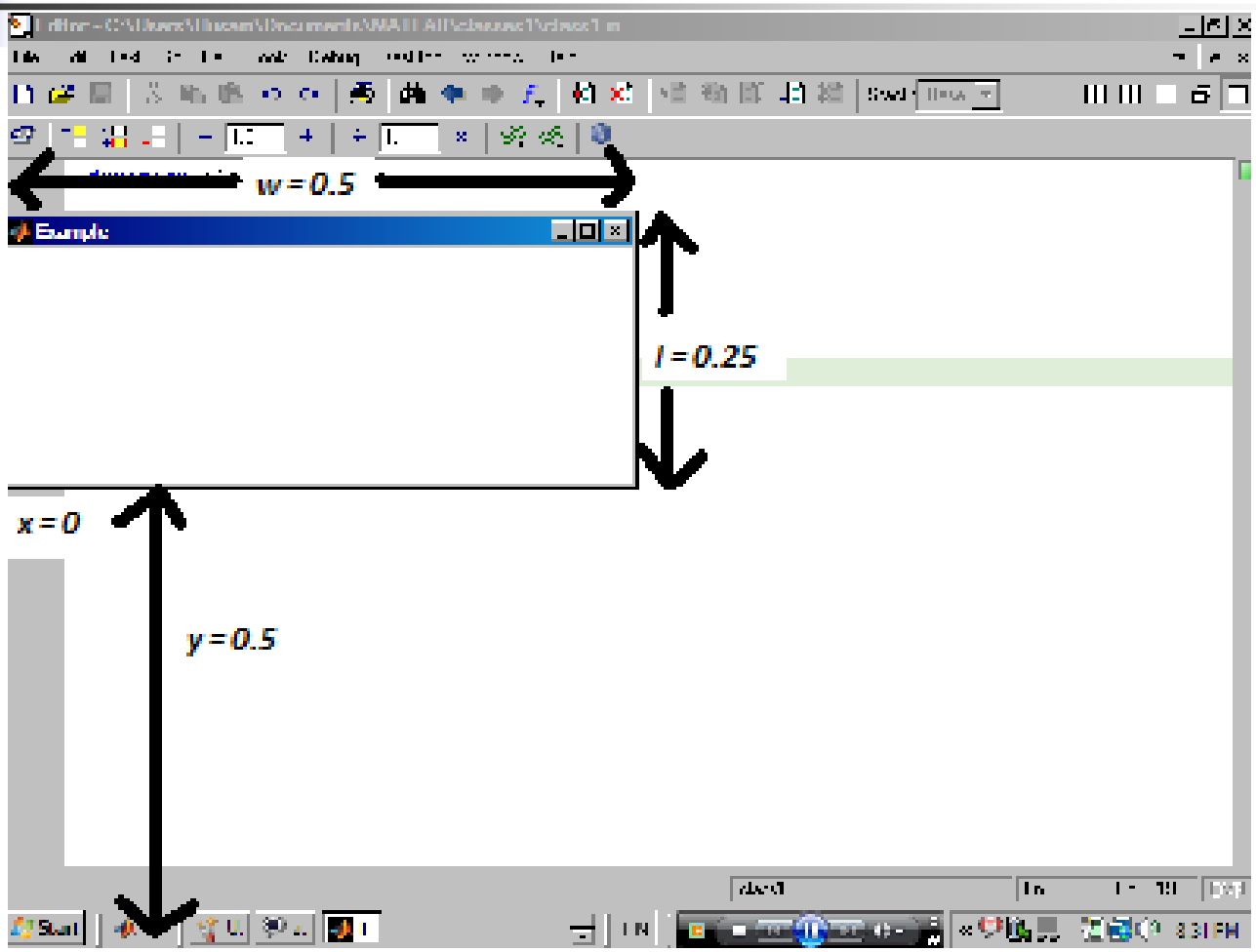


- Safest bet is to use the property value ‘normalized’.
- With ‘normalized’ chosen, each element is relative to the screen and takes the value in the range of [0, 1].
- Bottom left side of the screen is [0, 0].
- Upper right side of the screen is [1, 1].

Applying More Properties



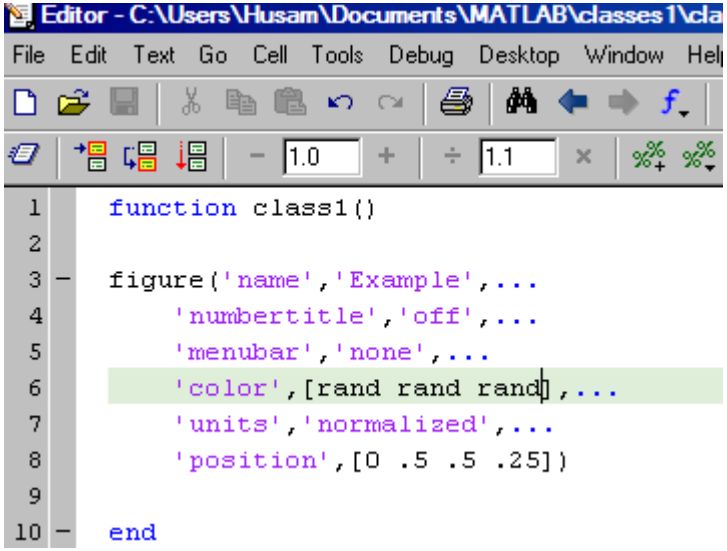
Result



RAND function

- Creates a random number in the range [0, 1].
- Can be used in [R, G, B] vector

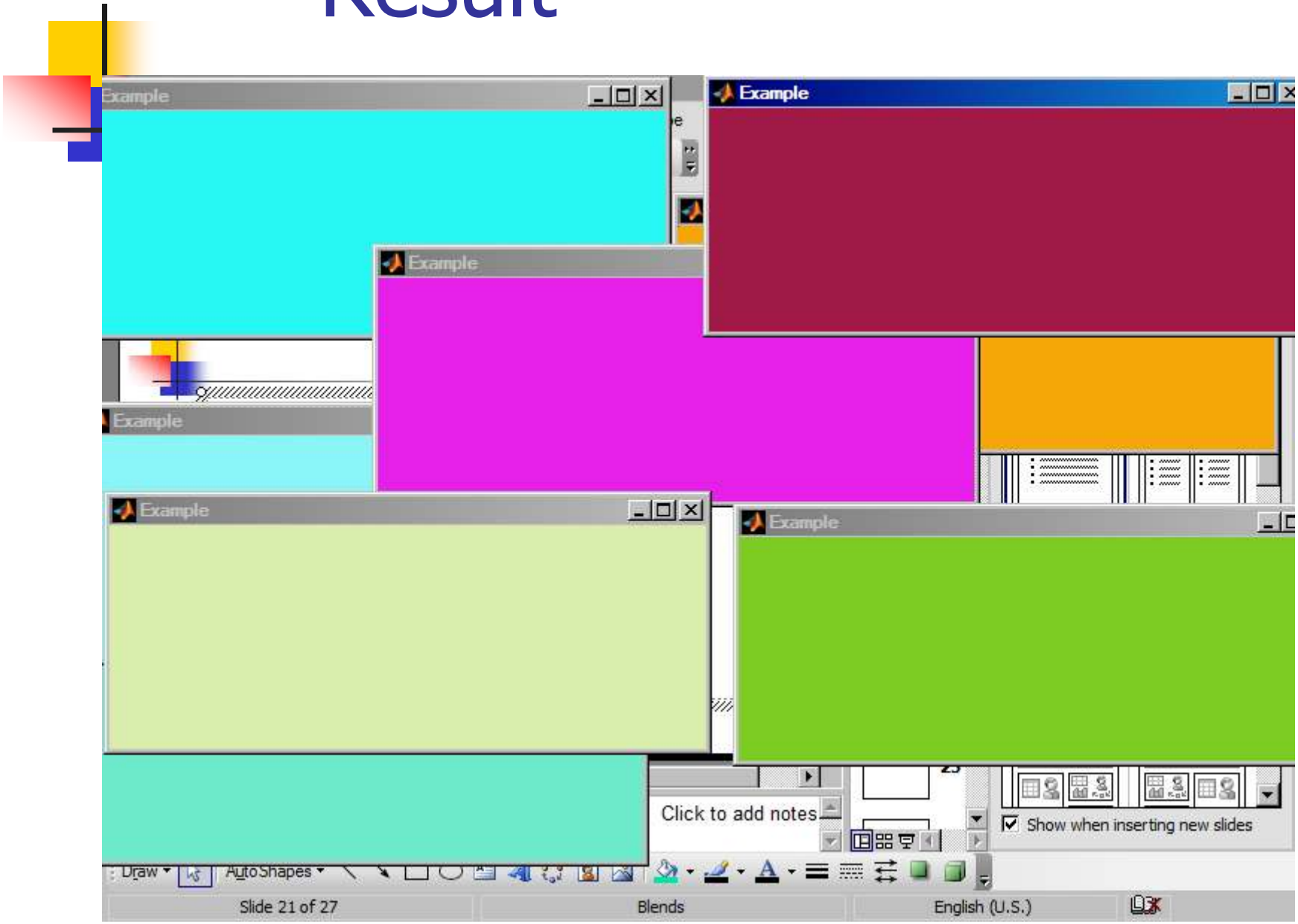
To create a randomly coloured FIGURE.

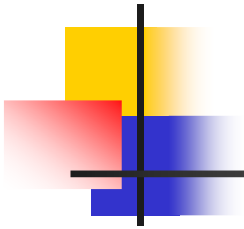


Command Window

```
>> rand

ans =

    0.81472

>>
```

Result

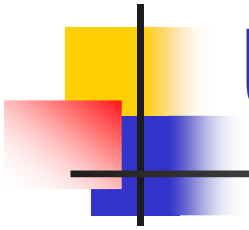

Example
Example
Example
Example
Example
Example
Click to add notes
Show when inserting new slides
Draw
AutoShapes
Slide 21 of 27
Blends
English (U.S.)



UICONTROL Command

- Very important command.
- Creates edit boxes, text boxes, pushbuttons, sliders, popup menus and more.
- Syntax:

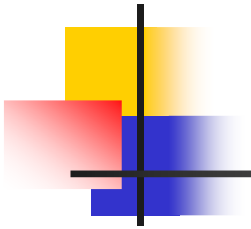
Uicontrol(‘propertyName’,propertyvalue)



UICONTROL Property: ‘Style’

- ‘style’ property specifies the UICONTROL type and can have one of the following as its value:

- pushbutton, togglebutton, radiobutton, checkbox, edit, text, slider, frame, listbox, popupmenu.

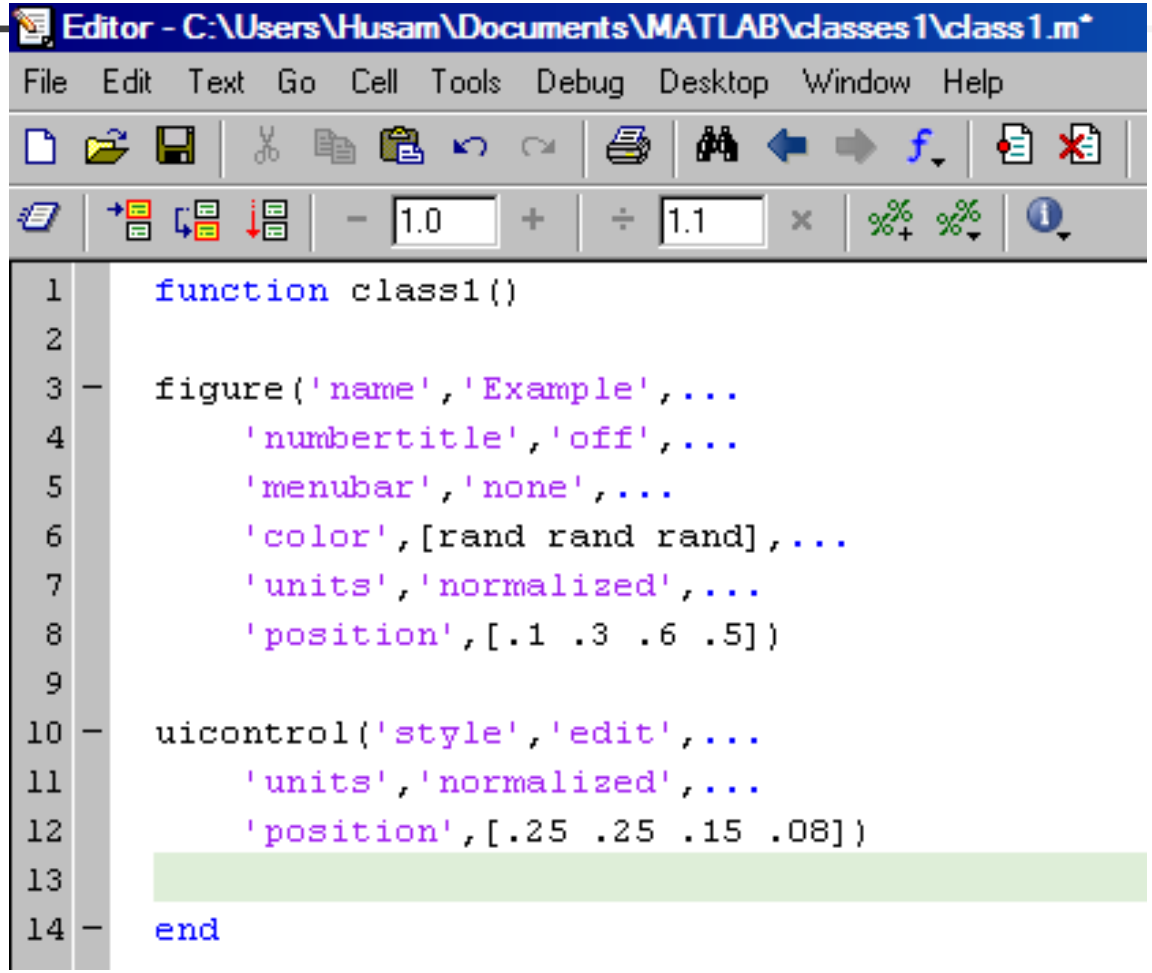


UICONTROL Property: Position

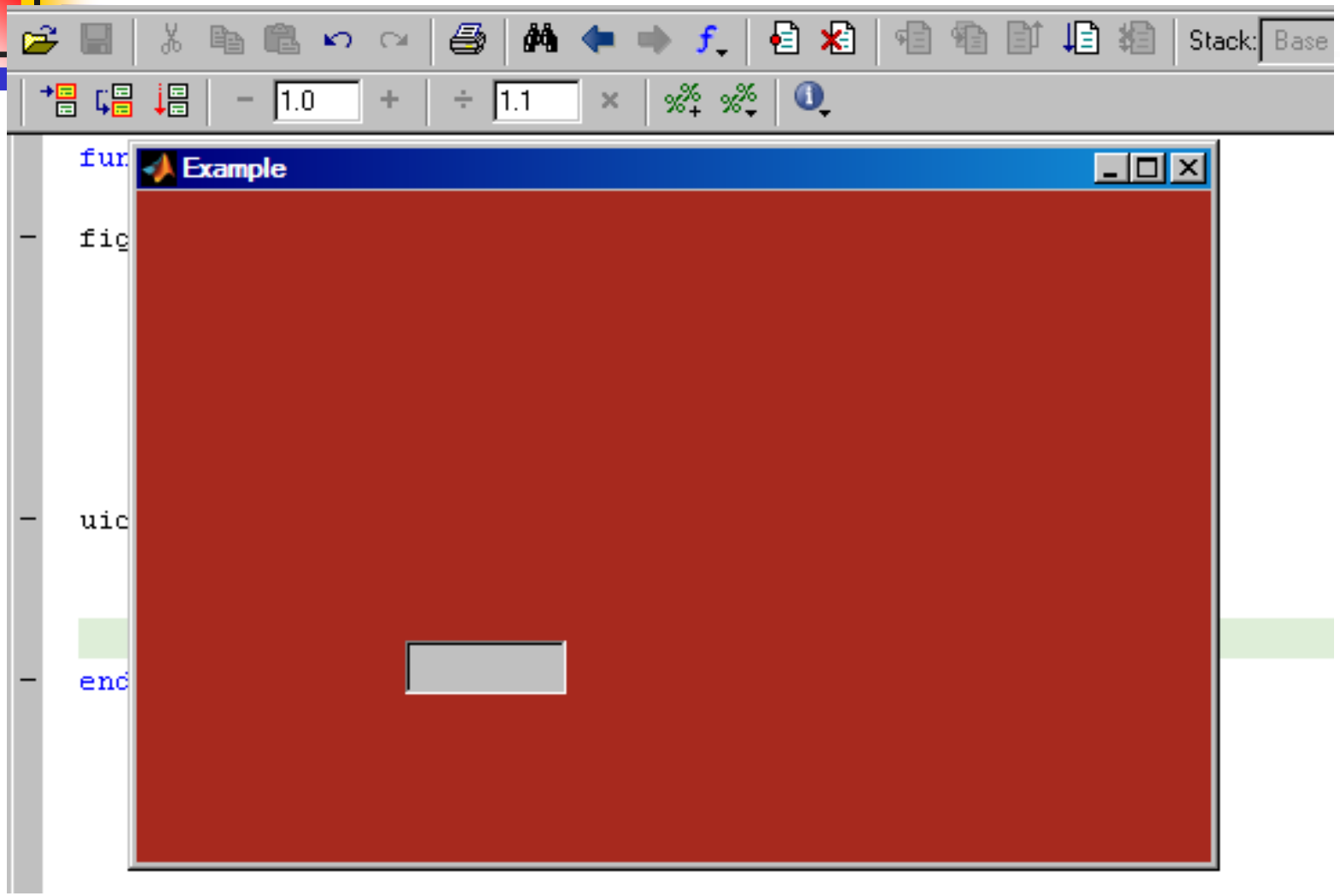
- ‘position’ behaves similarly to the FIGURE property, only here the position is taken relative to the figure window. Likewise, ‘units’ is treated in the same vein.

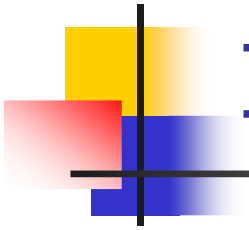
- For example, the point [0, 0] lies on the lower left corner of the figure on which the UICONTROL is laid upon.

Example


Editor - C:\Users\Husam\Documents\MATLAB\classes1\class1.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
1 function class1()
2
3 figure('name','Example',...
4 'numbertitle','off',...
5 'menubar','none',...
6 'color',[rand rand rand],...
7 'units','normalized',...
8 'position',[.1 .3 .6 .5])
9
10 uicontrol('style','edit',...
11 'units','normalized',...
12 'position',[.25 .25 .15 .08])
13
14 end

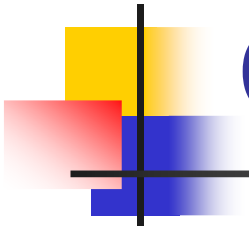
Result





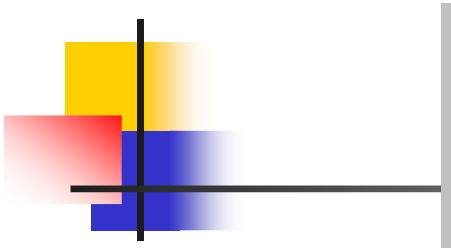
Important UICONTROL Properties

PropertyName	PropertyValue	Example	Explanation
'string'	string	'Enter Number'	Writes string in UICONTROL
'foregroundcolor'	[R, G, B]	[0, 0, 0]	Font colour
'backgroundcolor'	[R, G, B]	[1, 1, 1]	UICONTROL colour
'visible'	{'on'} 'off'	'off'	Sets UICONTROL to visible/invisible
'callback'	String or function handle	@myfunc1	Execute specified function on UICONTROL interaction



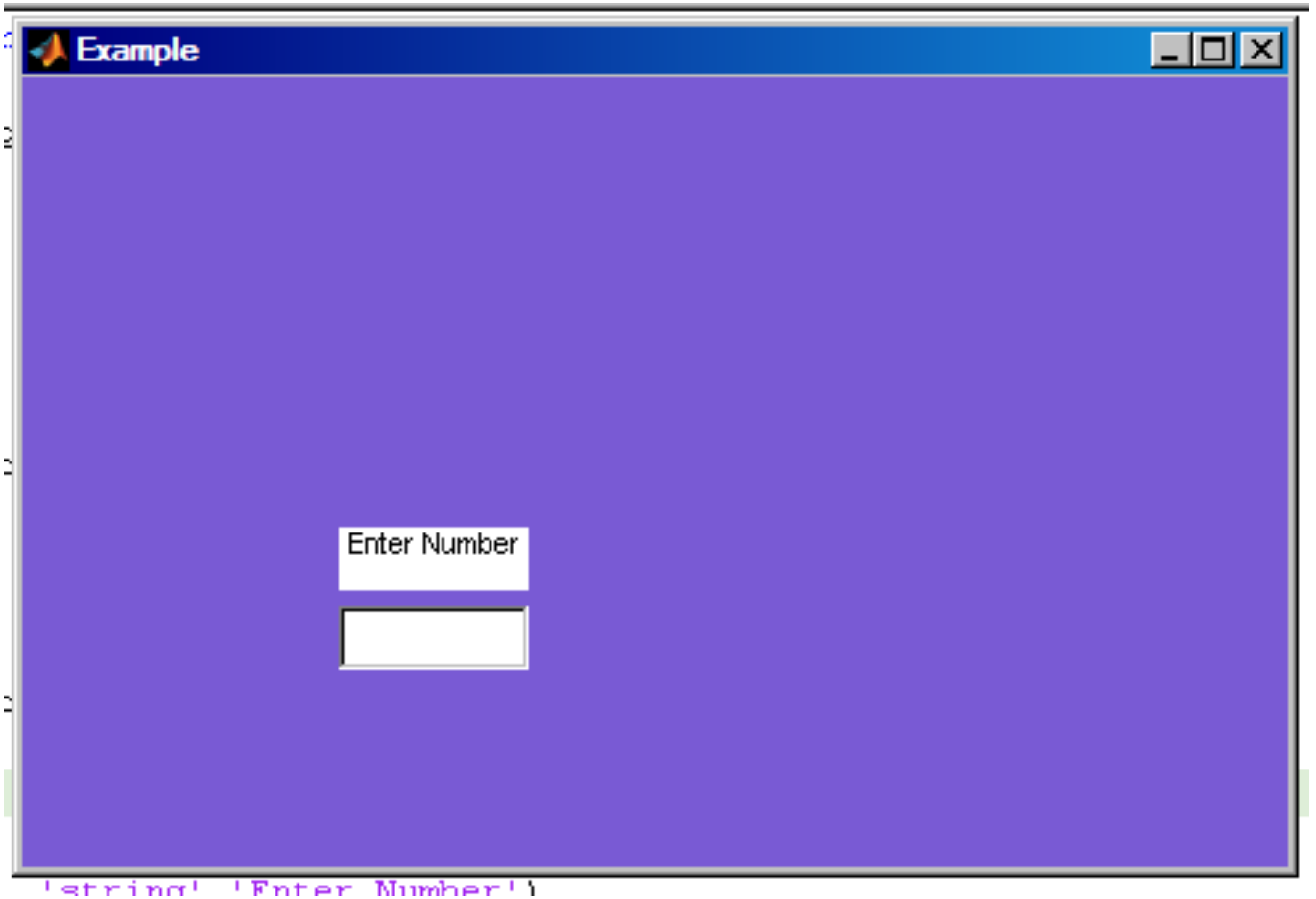
Creating Another UICONTROL

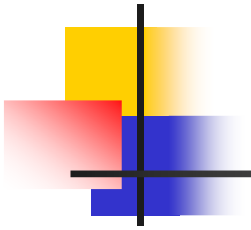
- Let's create another UICONTROL, namely a text box.
- Notice how the (y) element of position is slightly higher than that of the our edit box.



```
function class1()

figure('name','Example',...
    'numbertitle','off',...
    'menubar','none',...
    'color',[rand rand rand],...
    'units','normalized',...
    'position',[.1 .3 .6 .5])

uicontrol('style','edit',...
    'units','normalized',...
    'position',[.25 .25 .15 .08],...
    'backgroundcolor',[1 1 1])

uicontrol('style','text',...
    'units','normalized',...
    'position',[.25 .35 .15 .08],...
    'backgroundcolor',[1 1 1],...
    'string','Enter Number')

end
```

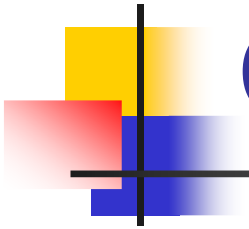
Result





Problem

- Notice how the background colour of the text box is distracting
- It would be better if the ‘backgroundcolor’ property was set to the same as that of the FIGURE.
- However, the FIGURE has a random colour combination, so what is the solution?
- We need to use the command GET.

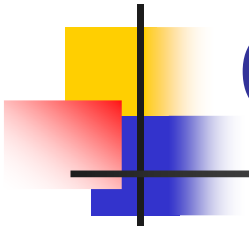


GET Command

- Along with the command SET, essential commands for GUI building

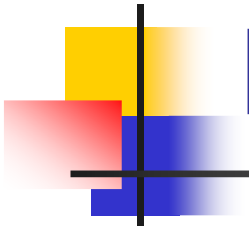
- Syntax:

get(handle,'propertName')



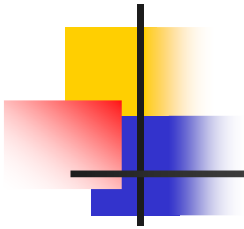
Giving Handles to Objects

- We need to give handles for objects so we could reference them later.
- Not all objects need handles; only those which will be referenced later.
- Nearly anything can be given a handle.



Examples

```
1    function class1()
2
3 -  f=figure('name','Example',...
4        'numbertitle','off',...
5        'menubar','none',...
6        'color',[rand rand rand],...
7        'units','normalized',...
8        'position',[.1 .3 .6 .5]);
9
10 - ed1=uicontrol('style','edit',...
11       'units','normalized',...
12       'position',[.25 .25 .15 .08],...
13       'backgroundcolor',[1 1 1]);
```



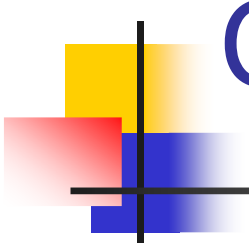
Using GET

- Now that you've given something a handle, you can obtain its properties for later use, as shown.

- First, get the colour of the created FIGURE (with handle "f")

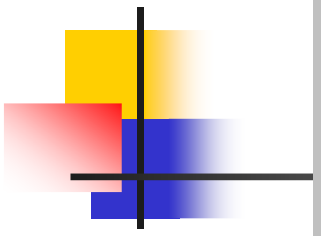
Cont.

```
1     function class1()
2
3 -   f=figure('name','Example',...
4         'numbertitle','off',...
5         'menubar','none',...
6         'color',[rand rand rand],...
7         'units','normalized',...
8         'position',[.1 .3 .6 .5]);
9
10 -  ed1=uicontrol('style','edit',...
11        'units','normalized',...
12        'position',[.25 .25 .15 .08],...
13        'backgroundcolor',[1 1 1]);
14
15 -  col=get(f,'color');
```



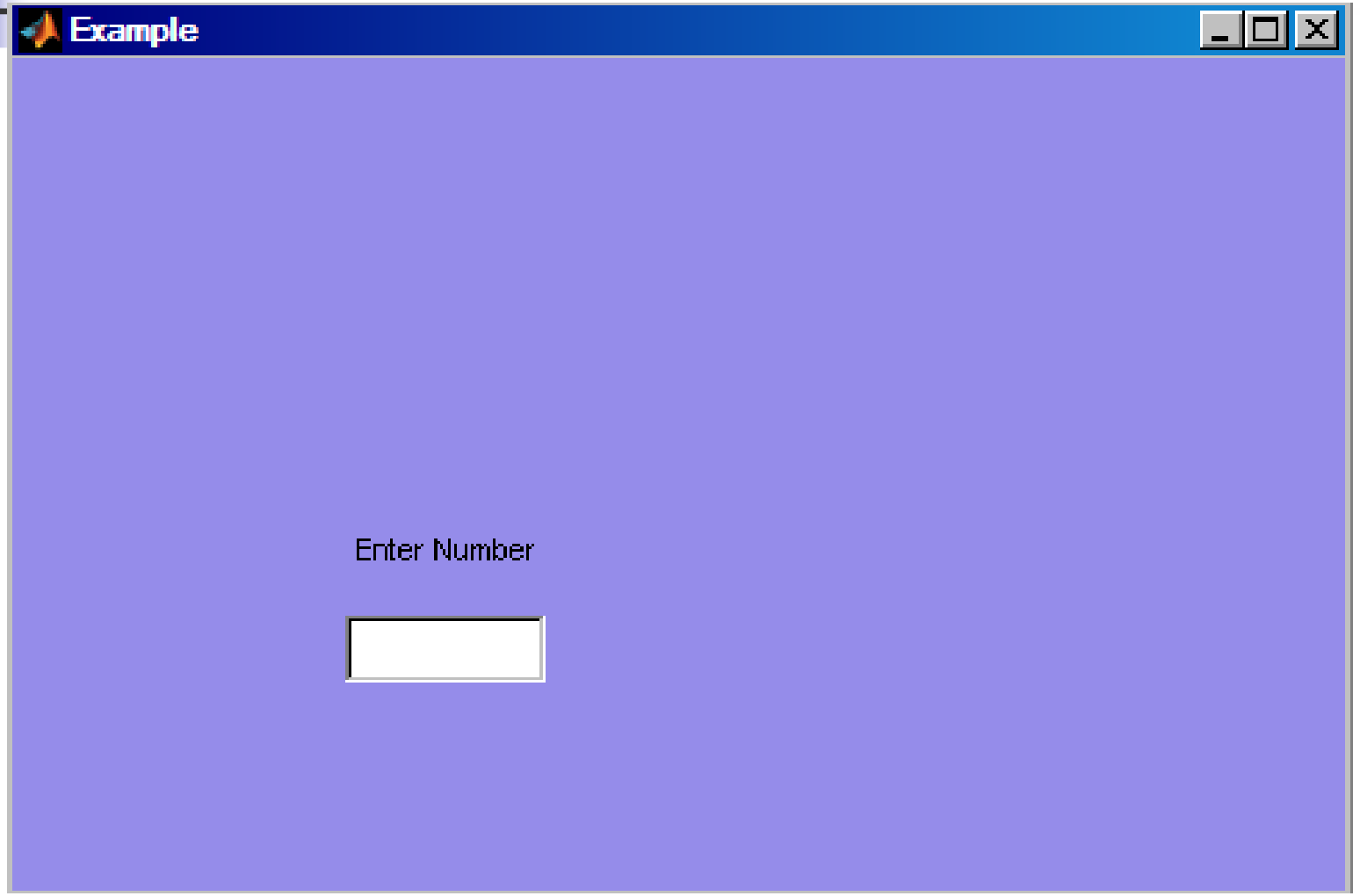
Cont.

- Now that the FIGURE colour is known and stored in the variable “col”, we can use this value as background colour for our text box UICONTROL.

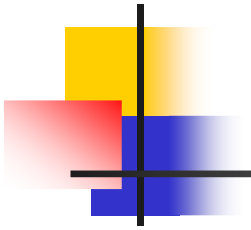


```
function class1()

f=figure('name','Example',...
    'numbertitle','off',...
    'menubar','none',...
    'color',[rand rand rand],...
    'units','normalized',...
    'position',[.1 .3 .6 .5]);

ed1=uicontrol('style','edit',...
    'units','normalized',...
    'position',[.25 .25 .15 .08],...
    'backgroundcolor',[1 1 1]);

col=get(f,'color');

uicontrol('style','text',...
    'units','normalized',...
    'position',[.25 .35 .15 .08],...
    'backgroundcolor',col,...
    'string','Enter Number')
```

Result



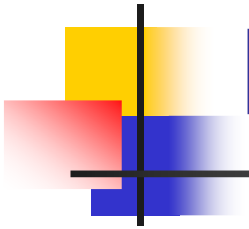
Using the SET Command



- Gives pre-existing objects new values for properties.

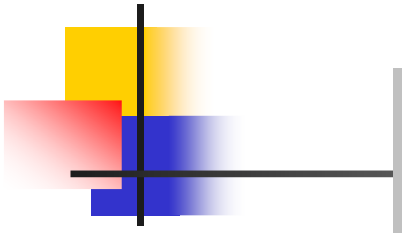
- Syntax:

set(handle,'propertyName',propertyvalue)



Example

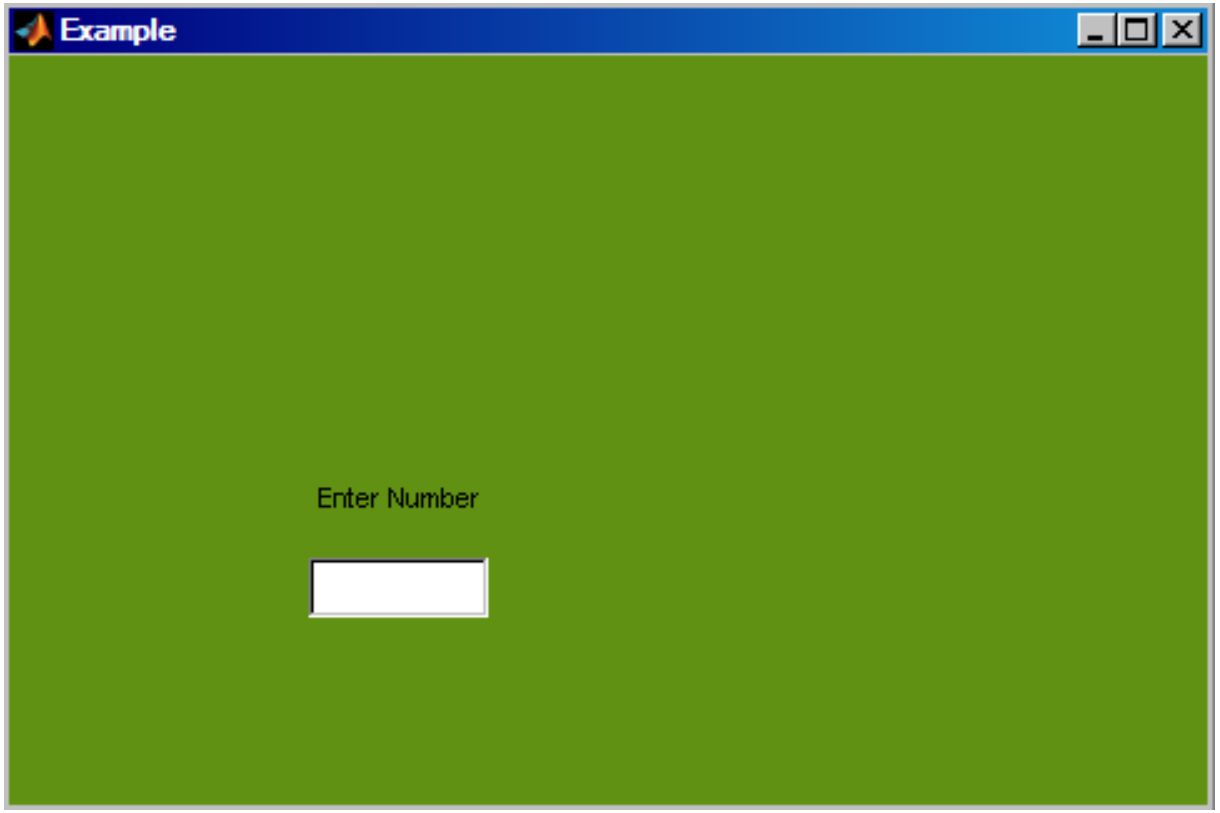
- Change background colour of text box from white to that of the current figure (random).

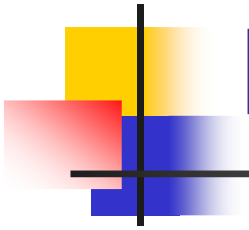


```
1   function class1()
2
3 -   f=figure('name','Example',...
4         'numbertitle','off',...
5         'menubar','none',...
6         'color',[rand rand rand],...
7         'units','normalized',...
8         'position',[.1 .3 .6 .5]);
9
10 -  ht1=uicontrol('style','text',...
11        'units','normalized',...
12        'position',[.25 .35 .15 .08],...
13        'backgroundcolor',[1 1 1],...
14        'string','Enter Number');
15
16 -  col=get(f,'color');
17
18 -  set(ht1,'backgroundcolor',col)
```

Result

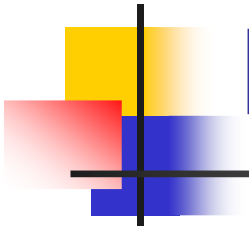
- Same as before.





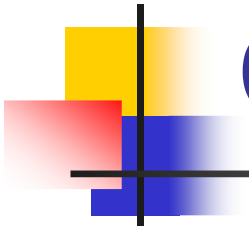
Problem

- FIGURE colour is given a random choice.
- What if resulting random combination produces a figure that is too dark?
- How can we create a random combination for the FIGURE colour, yet maintain that it's still random?



More on RGB Vectors

- Each element in the [R, G, B] vector controls the intensity of its corresponding value (red/green/blue).
- This means for high values in the vector we get brighter colours.
- Thus it is possible to make sure colour combination produces a bright colour if its elements are rather large.



Cont.

- So if we somehow control the RAND command into giving values for a higher range, we would make sure that each time a colour combination is produced, it is on the lighter side.

- For this we use

rand*0.5+0.5

Instead of

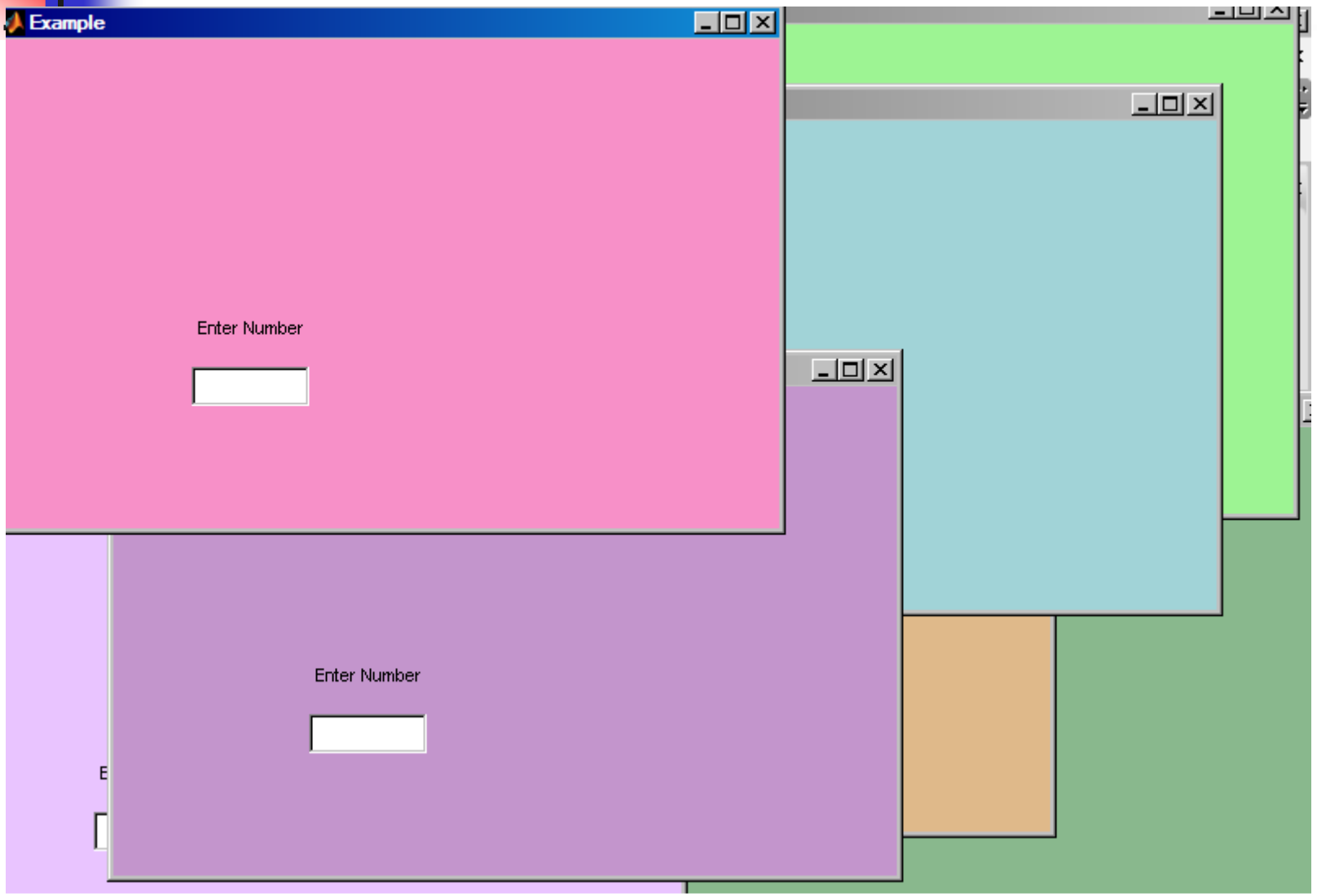
rand

Which gives us values in the range [0.5 1].

Example

```
1     function class1()
2
3 -   f=figure('name','Example',...
4         'numbertitle','off',...
5         'menubar','none',...
6         'color',[rand rand rand].*0.5+0.5,...
7         'units','normalized',...
8         'position',[.1 .3 .6 .5]);
```

Result

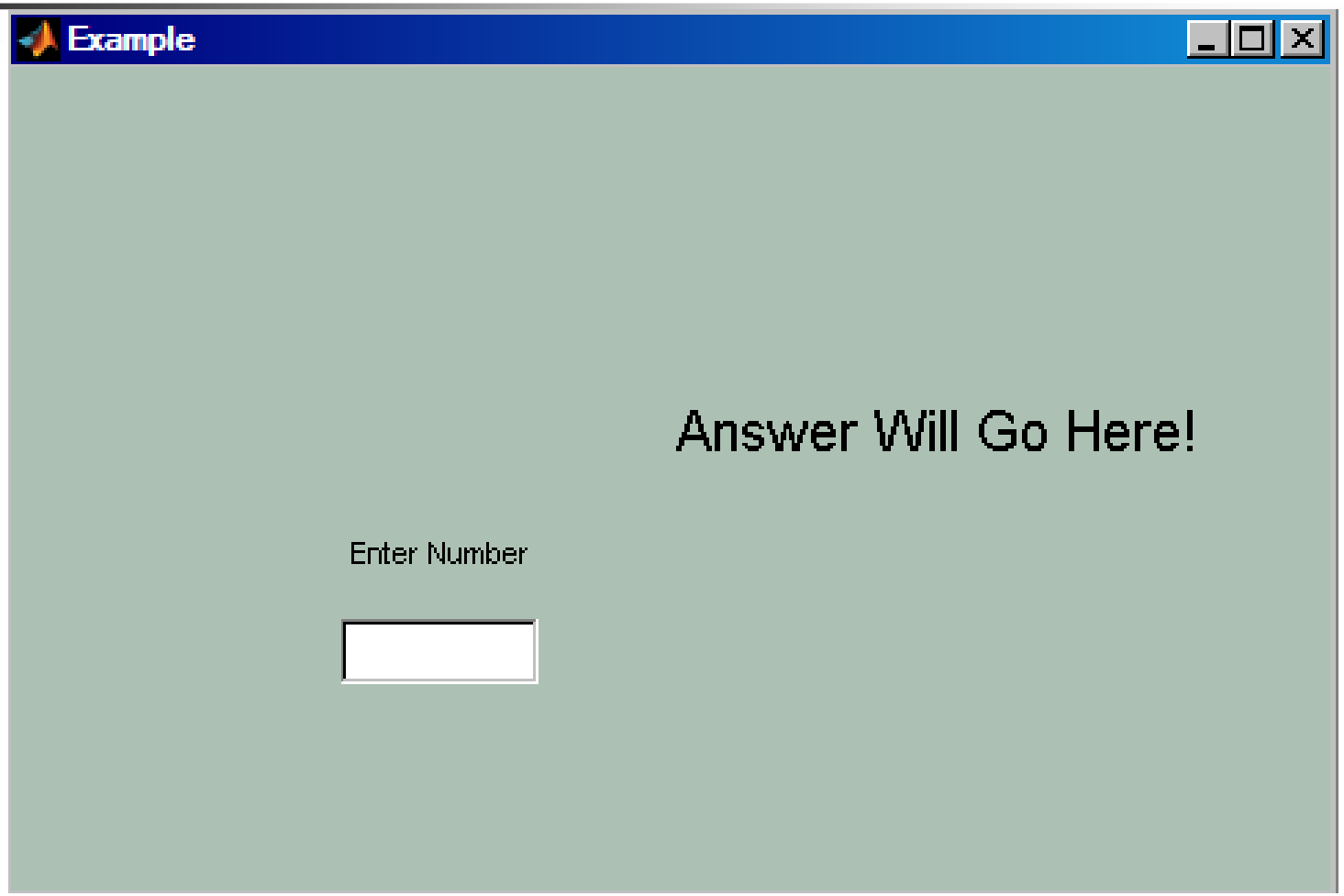


Creating Another Text Box

- Let us create another text box and give it its own handle.

```
ht2=uicontrol('style','text',...
    'units','normalized',...
    'position',[.5 .4 .4 .2],...
    'backgroundcolor',col,...
    'string','Answer Will Go Here!',...
    'fontsize',15);
```

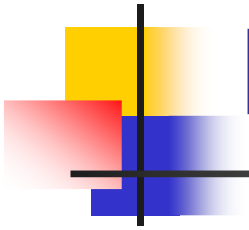
Result



Creating a Pushbutton

- Creating a pushbutton has the same syntax as other UICONTROLs.

```
32 —   uicontrol('style','pushbutton',...
33         'units','normalized',...
34         'position',[.25 .1 .15 .1],...
35         'string','Push Me!',...
36         'backgroundcolor',[1 1 1],...
37         'callback',@go1)
```

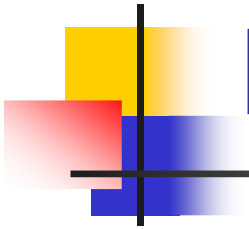


Pushbutton Callback

- Notice how the recently created pushbutton has the ‘callback’ propertyname with the propertyvalue “@go1”.
- This means that whenever the user interacts with the UICONTROL (in our case push the pushbutton), we will execute the function “go1”.

Creating the Callback Function

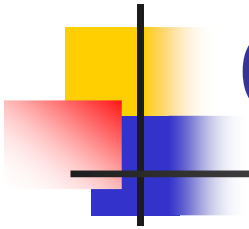
```
uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[.25 .1 .15 .1],...
    'string','Push Me!',...
    'backgroundcolor',[1 1 1],...
    'callback',@go1)

    function go1(varargin)

    end
```



Programming the GUI

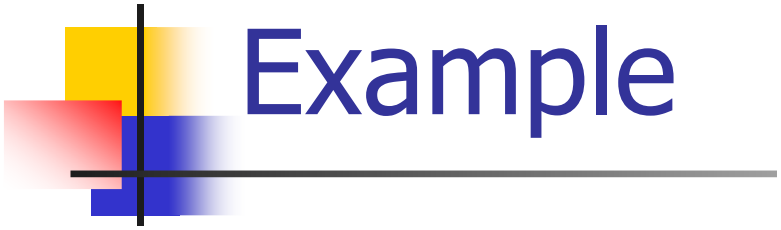
- We want to put lines of code inside the callback function. That way, whenever the user clicks the pushbutton, something happens.

- What happens is the number in the edit box is obtained, an operation is performed on it, then the result is to be displayed on the text box on the right.



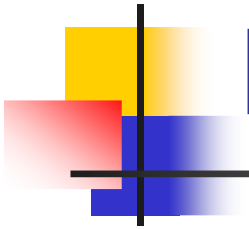
Cont.

- First we want to obtain the value found in the edit box.
- To do this, we need to get the ‘string’ value of the edit box.
- However, the value found here is in character format. To convert it into a numerical value, we use the command STR2NUM.



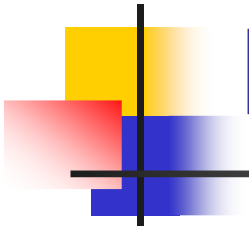
Example

```
ed1=uicontrol('style','edit',...
    'units','normalized',...
    'position',[.25 .25 .15 .08],...
    'backgroundcolor',[1 1 1]);

ht2=uicontrol('style','text',...
    'units','normalized',...
    'position',[.5 .4 .4 .2],...
    'backgroundcolor',col,...
    'string','Answer Will Go Here!',..
    'fontsize',15);

uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[.25 .1 .15 .1],...
    'string','Push Me!',...
    'backgroundcolor',[1 1 1],...
    'callback',@go1)

    function go1(varargin)
        var=get(ed1,'string');
        x=str2num(var);
    end
```



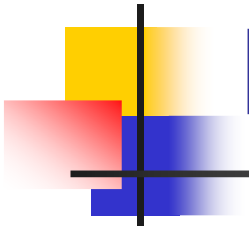
Notes

- The input to the function “go1” is the phrase “varargin”, which stand for variable arguments input.
- This is due to the fact that the function has no inputs to it.
- The command STR2DOUBLE can be used instead of STR2NUM for scalar values.



Programming the Pushbutton

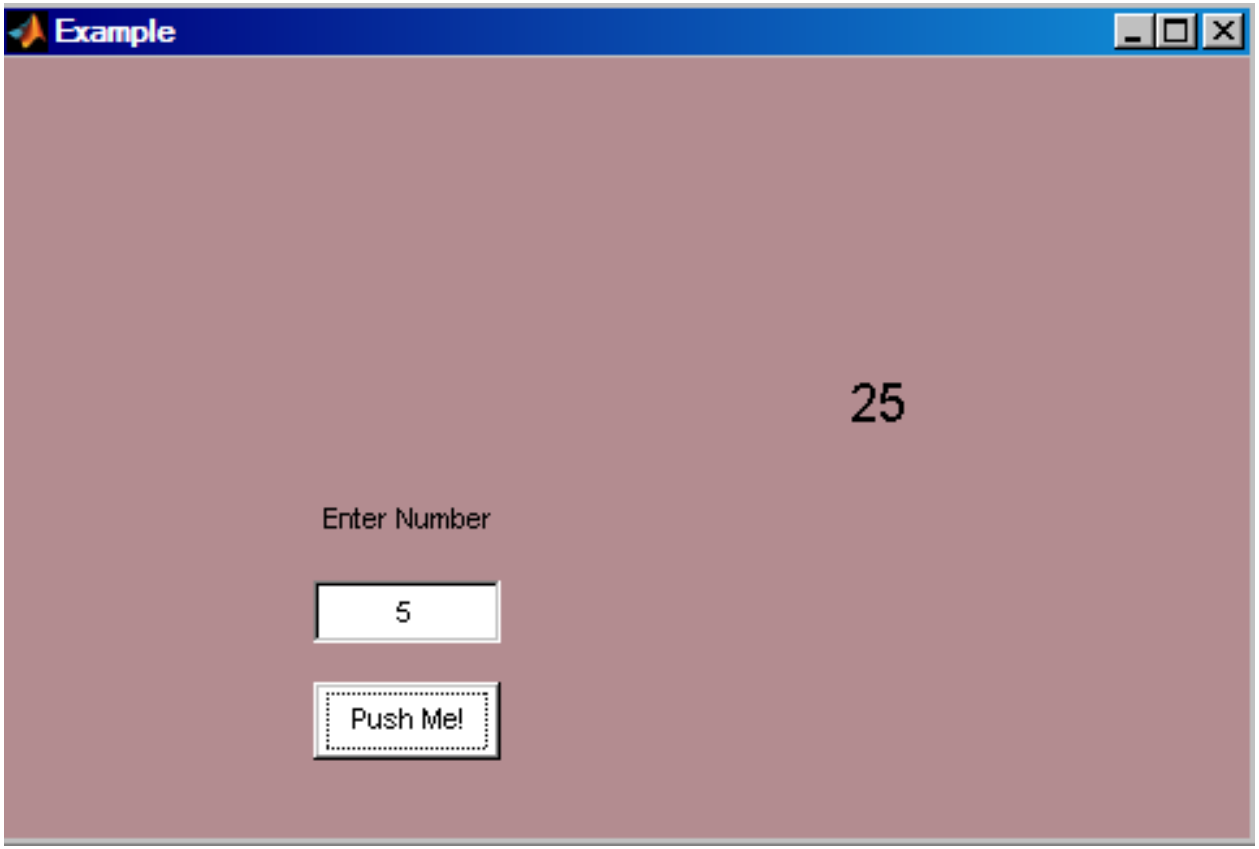
- After obtaining the number in the edit box, we will want to perform an operation on it, say square it.
- After the operation is done, the result should be placed in the text box with handle “ht2”.
- Remember: the numeric value should be converted to character format before being set as string. The command NUM2STR will be used for this purpose.
- The command SET will be used to display the answer.

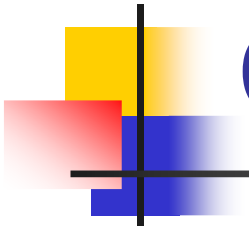


Example

```
function go1(varargin)
    var=get(ed1,'string');
    x=str2num(var);

    y=x^2;
    out=num2str(y);
    set(ht2,'string',out)
end
```

Result

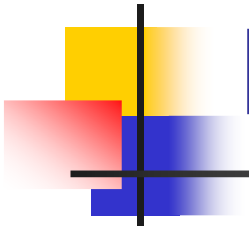




Creating a Popupmenu

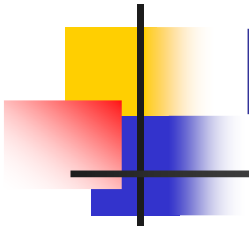
- A popupmenu is a UICONTROL which gives the user a list of possibilities which they can choose from.

- The defining properties of the popupmenu are the properties ‘string’ and ‘value’.



Popupmenu Property: ‘String’

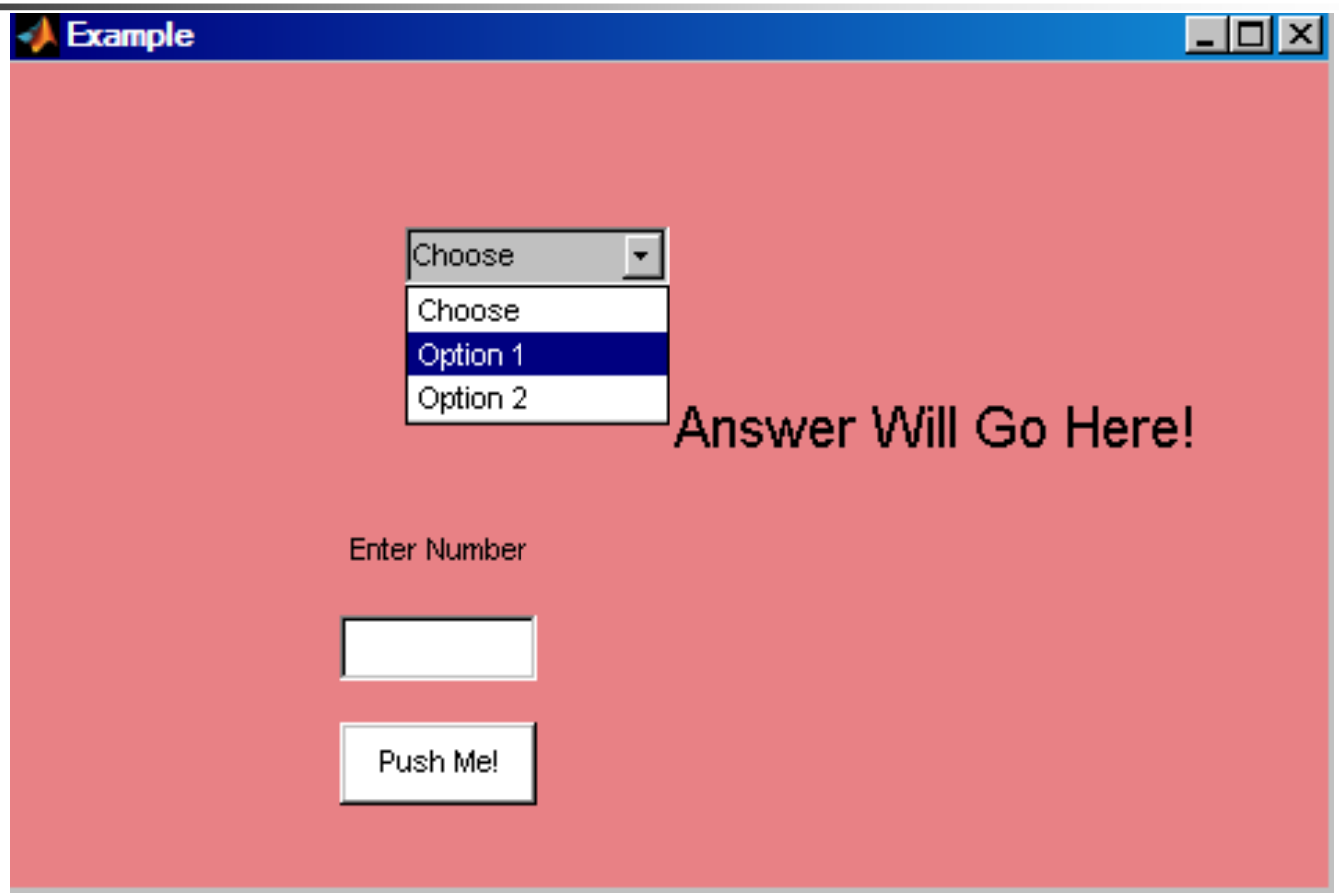
- The ‘string’ property contains a cell matrix with each row containing the string of one value (option).
- Cell matrices automatically stack strings of different lengths together.

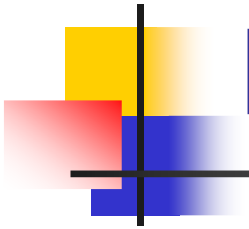


Example

```
ppm=uicontrol('style','popupmenu',...
    'units','normalized',...
    'position',[.3 .75 .2 .05],...
    'string',{'Choose';'Option 1';'Option 2'});
```

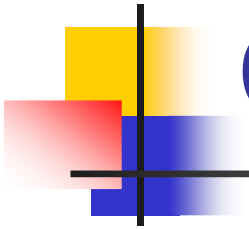
Result


Example
Choose
Choose
Option 1
Option 2
Answer Will Go Here!
Enter Number
Push Me!



Popupmenu Property: Value

- The ‘value’ property is numeric and contains the row of string from the popupmenu which was chosen.
- In our case, we want to edit our pushbutton callback into producing differing results based on the popupmenu options chosen.

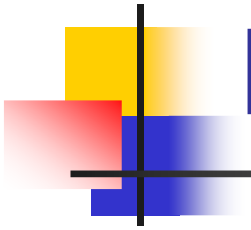


Cont.

- For this, we edit the popupmenu string and have the user chose between squaring and cubing the value in the edit box.

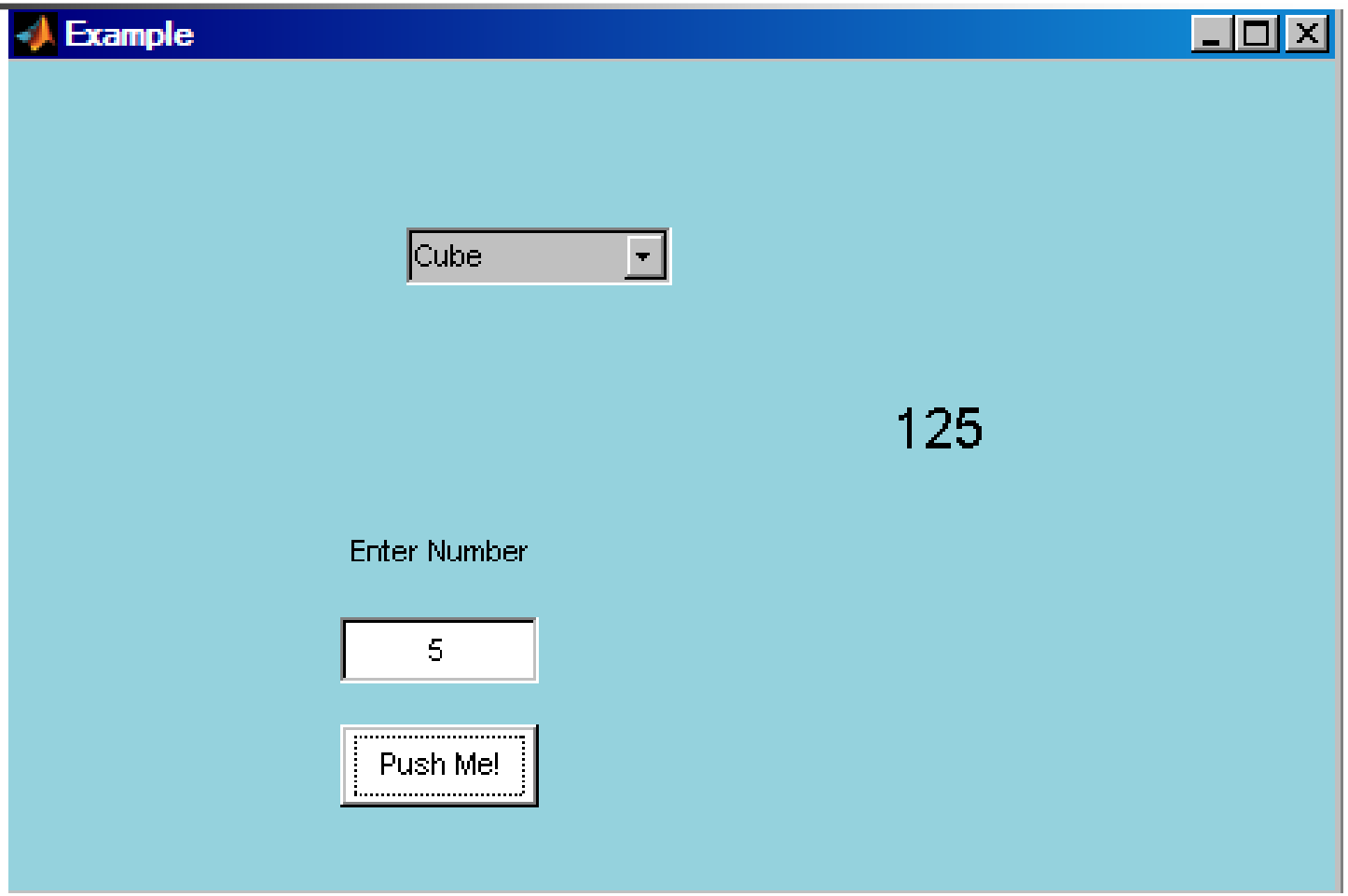
- Thus when the pushbutton is pressed, the popupmenu value is checked and based on it, the operation performed will be different.

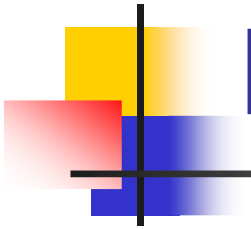
Example



```
39 -  ppm=uicontrol('style','popupmenu',...
40        'units','normalized',...
41        'position',[.3 .75 .2 .05],...
42        'string',{'Choose';'Square';'Cube'});
43
44        function go1(varargin)
45 -          var=get(ed1,'string');
46 -          x=str2num(var);
47
48 -          choice=get(ppm,'value');
49
50 -          if choice==2 % squared
51 -              y=x^2;
52 -          elseif choice==3 % cube
53 -              y=x^3;
54 -          end
55
56 -          out=num2str(y);
57 -          set(ht2,'string',out)
58 -      end
```

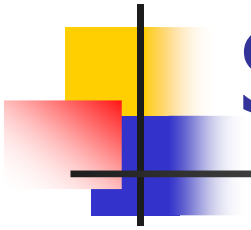
Result





Problem

- Each time the pushbutton is pressed, the value in the text box updates depending on the chosen value from the popupmenu.
- Say we want the answer to update every time the popupmenu is updated, what should we do?



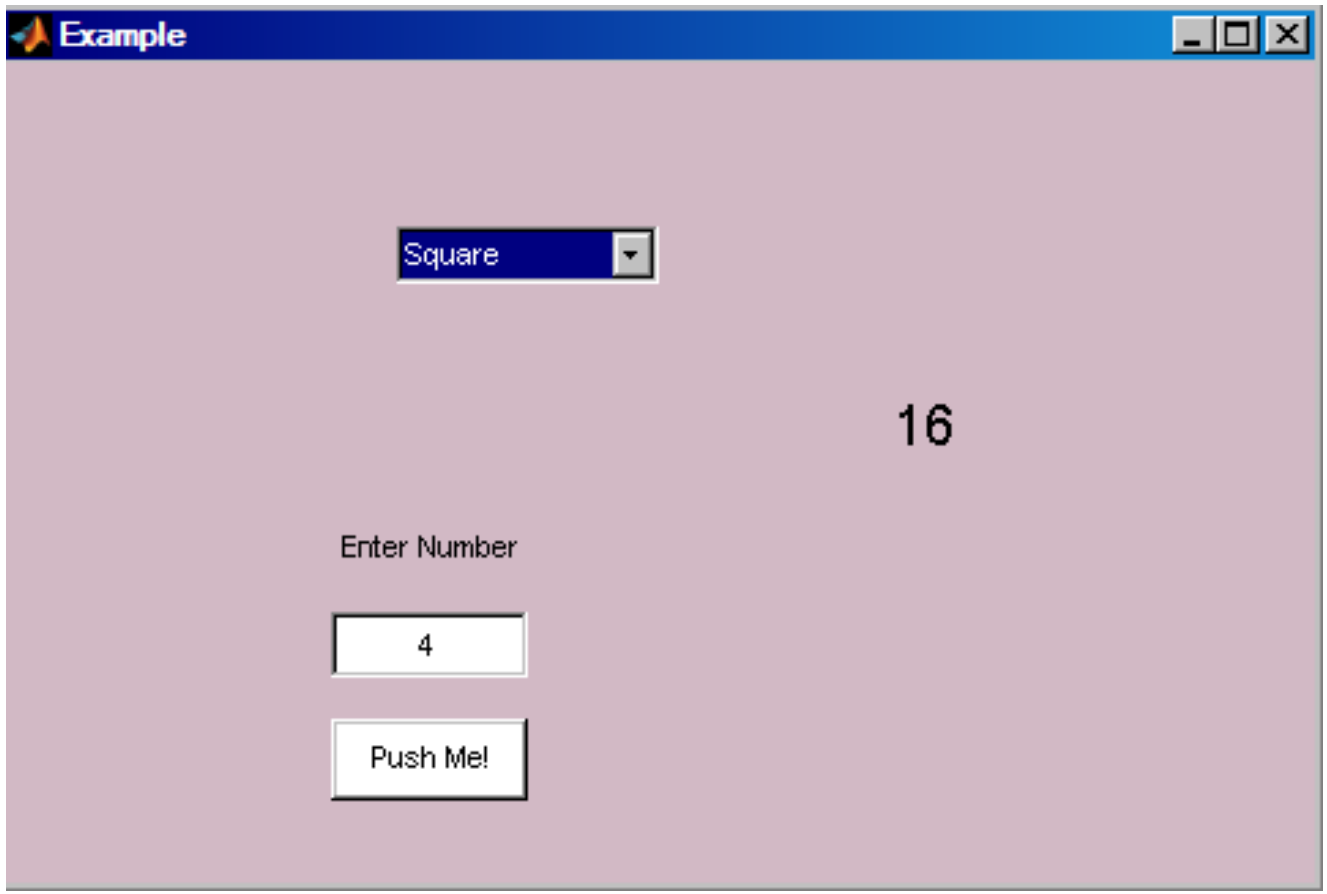
Solution

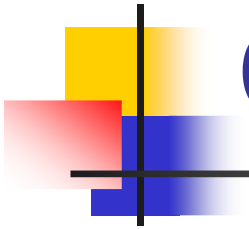
- We want the pushbutton callback to be executed whenever the popupmenu is interacted with.
- This means we should give the popupmenu a callback function.
- Let this be the same as the pushbutton callback function.
- The result is the same callback, but this time being executed when the user either click the pushbutton, or updates the popupmenu.

Example

```
32 -  uicontrol('style','pushbutton',...
33        'units','normalized',...
34        'position',[.25 .1 .15 .1],...
35        'string','Push Me!',...
36        'backgroundcolor',[1 1 1],...
37        'callback',@go1)
38
39 -  ppm=uicontrol('style','popupmenu',...
40        'units','normalized',...
41        'position',[.3 .75 .2 .05],...
42        'string',{'Choose';'Square';'Cube'},...
43        'callback',@go1);
44
45        function go1(varargin)
46 -          var=get(ed1,'string');
47 -          x=str2num(var);
48
```

Result

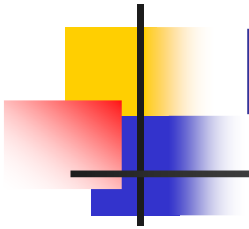




Other UICONTROLs

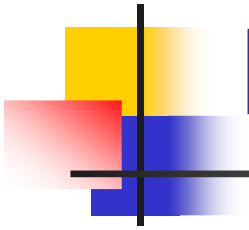
- There exists many other useful UICONTROL styles, but the most important ones have been covered.

- Also to be covered are the listbox, the slider, the checkbox and the radiobutton.



Listbox UICONTROL

- Its function is very similar to the popupmenu.
- In fact it behaves exactly the same in regards to the properties ‘string’ and ‘value’.



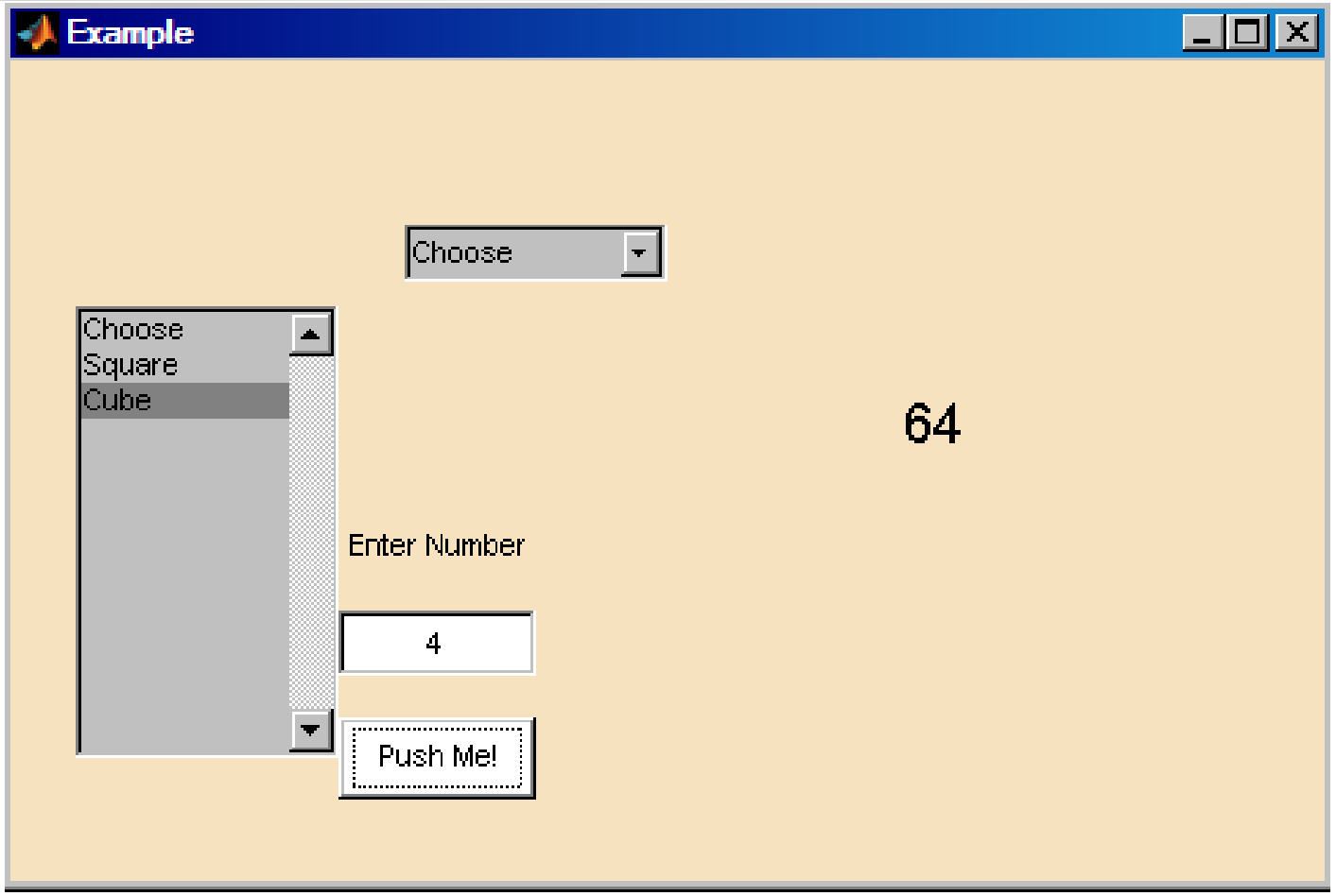
Listbox Value

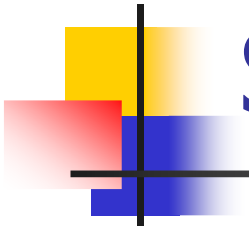
- We want to create a listbox just as we did with the popupmenu.
- Notice how the listbox needs to have a large value for its length.
- In the “go1” callback, the value in the listbox is checked and an operation is performed accordingly.

Example

```
45 -  hlb=uicontrol('style','listbox',...
46        'units','normalized',...
47        'position',[.05 .15 .2 .55],...
48        'string',{'Choose';'Square';'Cube'});
49
50        function go1(varargin)
51 -          var=get(ed1,'string');
52 -          x=str2num(var);
53
54 -          choice=get(hlb,'value');
55
56 -          if choice==2 % squared
57 -              y=x^2;
58 -          elseif choice==3 % cube
59 -              y=x^3;
60 -          end
```

Result

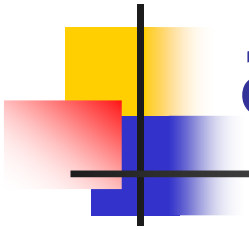




Slider UICONTROL

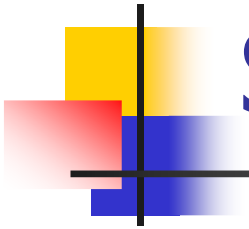
- Sliders can be used in many different ways.
- Important slider properties include MIN, MAX, VALUE and SLIDERSTEP properties.

Slider Properties: MIN, MAX and SLIDERSTEP



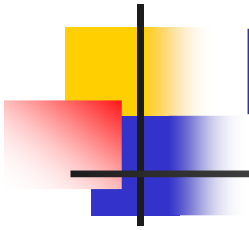
- MIN and MAX properties take the values of the minimum and maximum of the slider, respectively.

- SLIDERSTEP is a two element vector that controls how much the slider VALUE changes with each click.



Slider Property: SLIDERSTEP

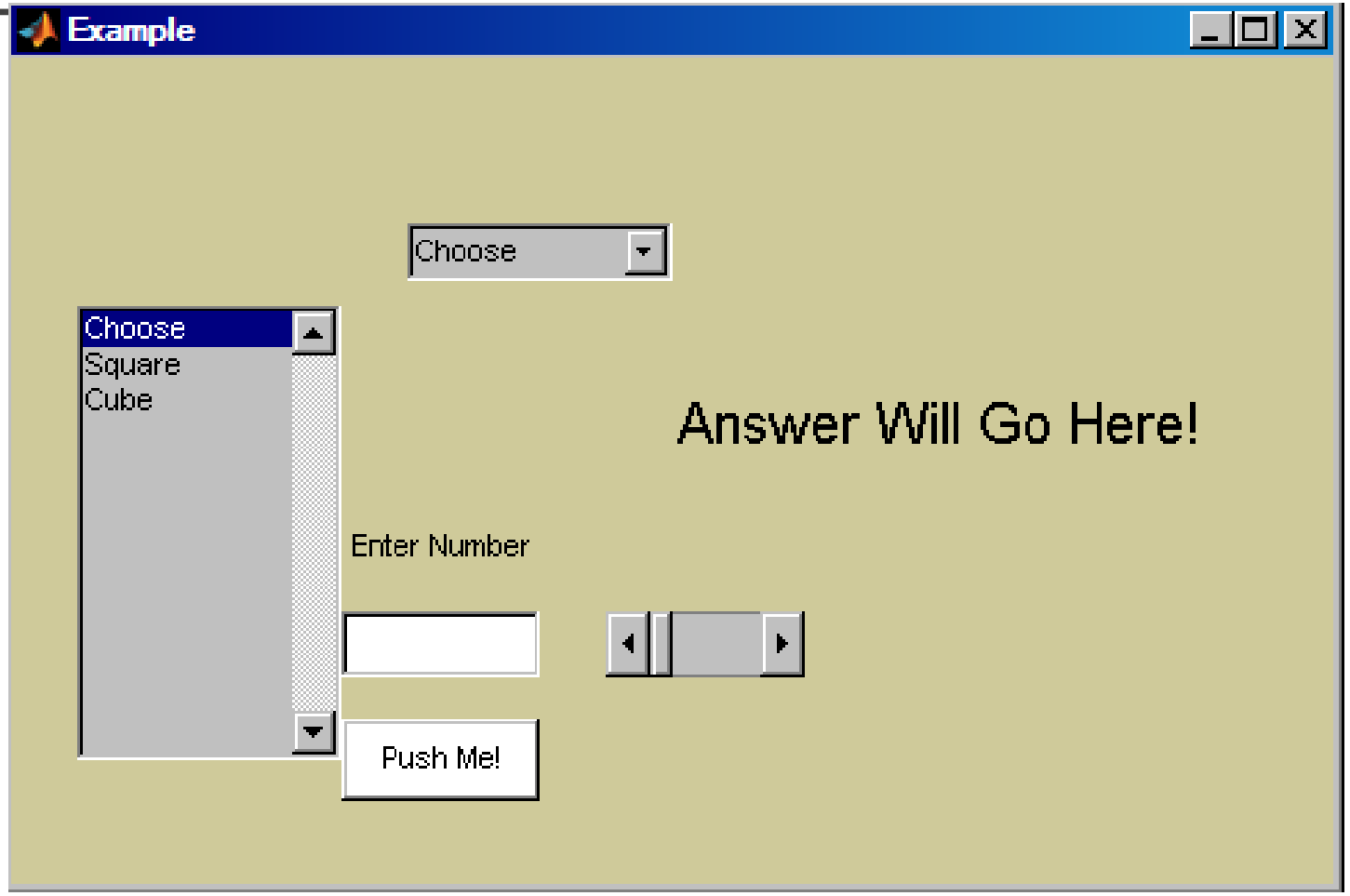
- The first value of SLIDERSTEP controls how much the slider moves when one of the arrows on its far sides is clicked.
- The second element governs the change of slider value when clicking on an empty space in the slider itself.
- Both SLIDERSTEP element are scaled to the MIN to MAX scale, meaning they are to be in the range [0, 1].



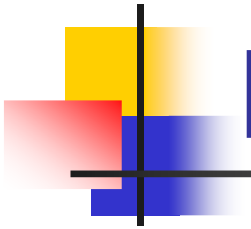
Example

```
hs1=uicontrol('style','slider',...
     'units','normalized',...
     'position',[.45 .25 .15 .08],...
     'min',0,...
     'max',10,...
     'sliderstep',[0.1 0.2]);
```

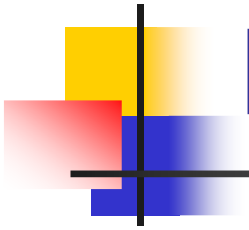
Result


Example
Choose
Choose
Square
Cube
Answer Will Go Here!
Enter Number
Push Me!

Using the slider to its full potential



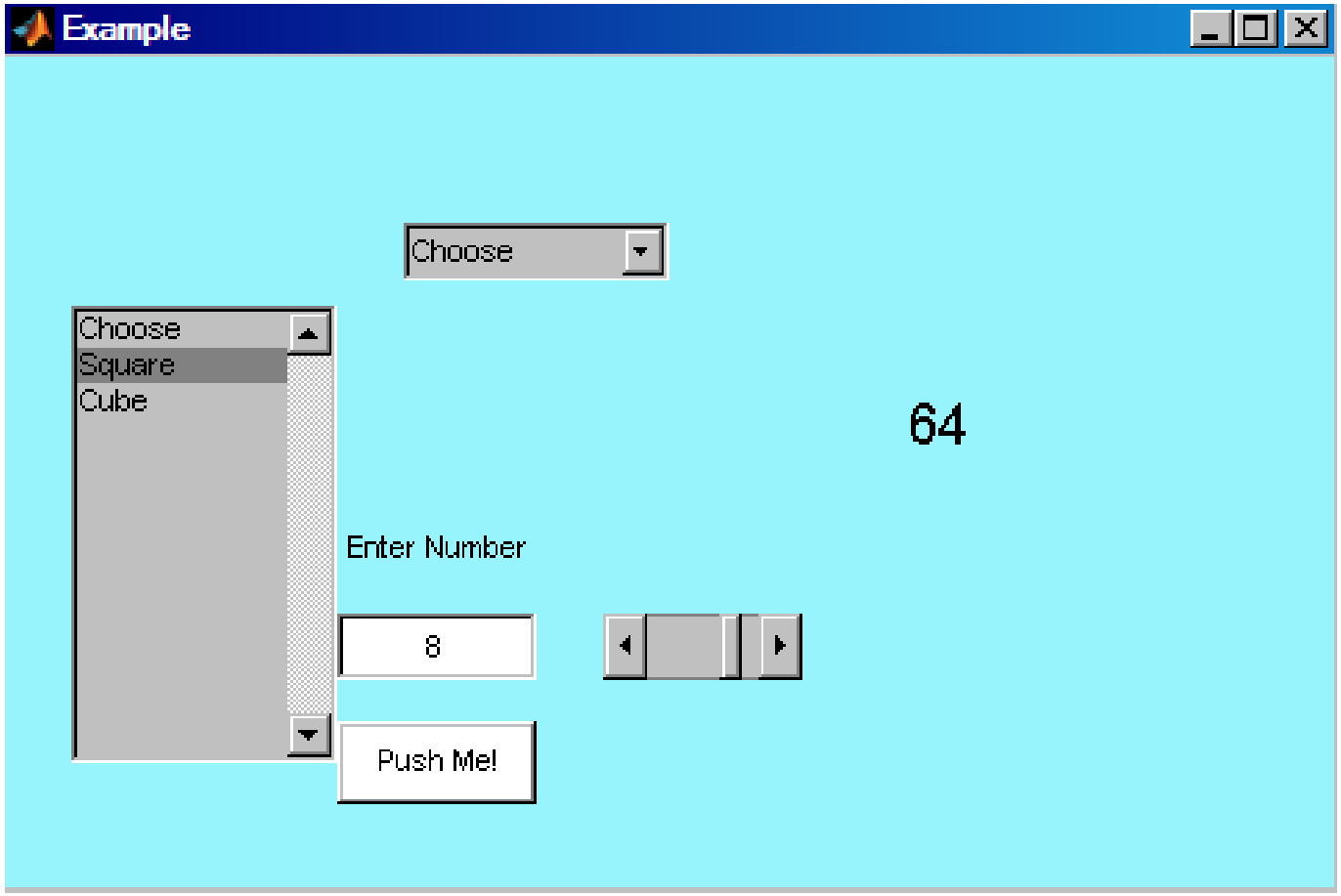
- What we want to do is make the slider control the number to be inputted.
- To do this, we give the slider a callback.
- When interacting with the slider, its value will be placed in the edit box.
- Then the operation specified by the listbox will be performed, and the answer will be displayed in the text box.

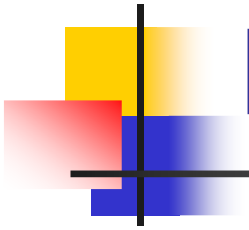


Example

```
50 -  hs1=uicontrol('style','slider',...
51        'units','normalized',...
52        'position',[.45 .25 .15 .08],...
53        'min',0,...
54        'max',10,...
55        'sliderstep',[0.1 0.2],...
56        'callback',@changeedit);
57
58        function changeedit(varargin)
59
60 -          x=get(hs1,'value');
61 -          set(ed1,'string',num2str(x))
62 -          go1
63
64 -       end
```

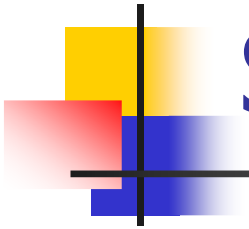
Result





Problem

- What if we want the edit box to control the slider?



Solution

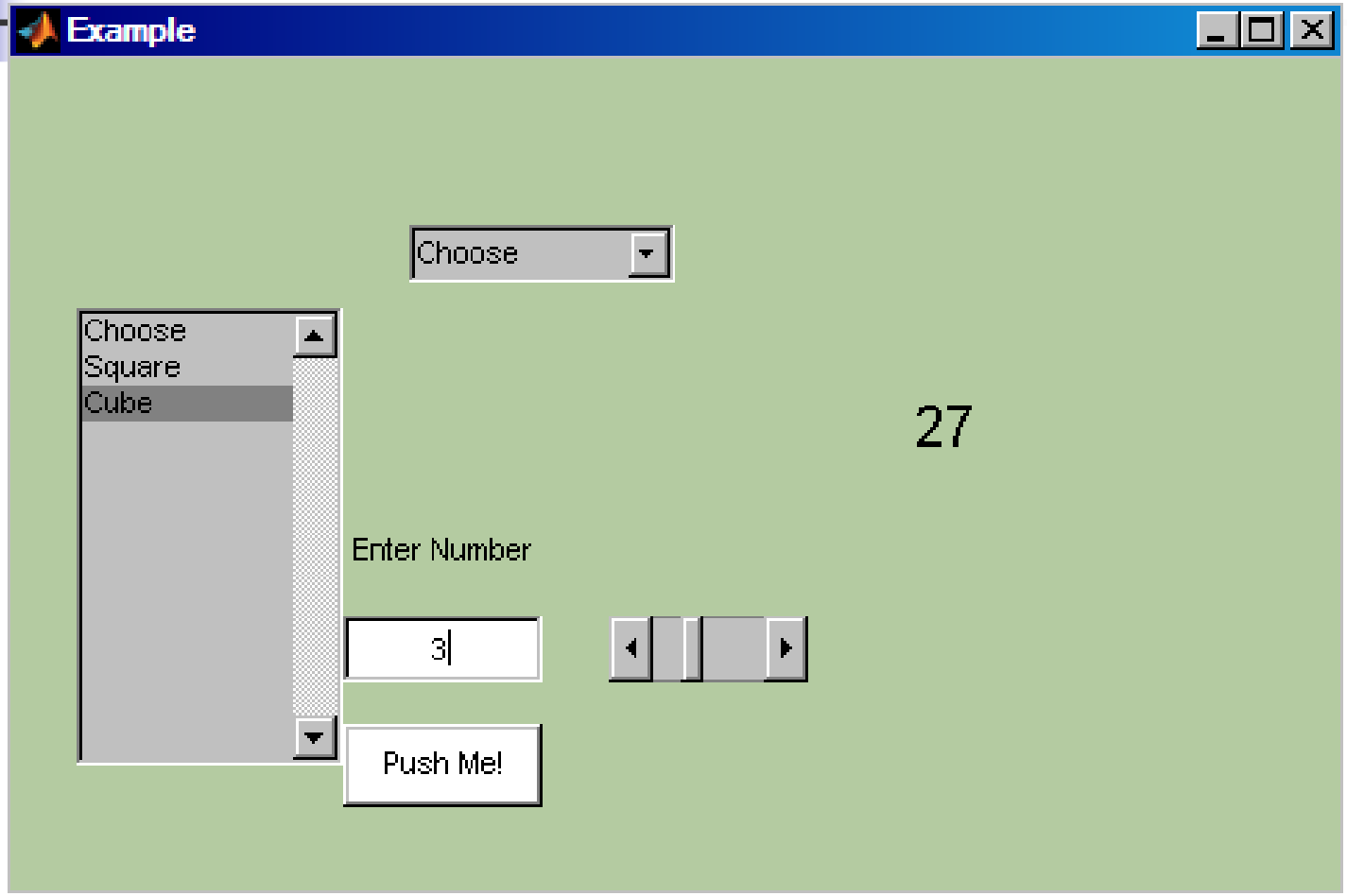
- Just like any other UICONTROL, the edit box can be given a callback as well.

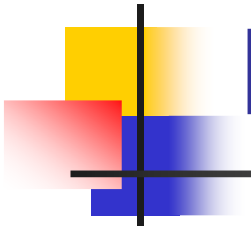
- In the edit box callback, we change the value of the slider accordingly and then apply the main pushbutton callback “go1”.

Example

```
66 -    set(ed1,'callback',@changeslider)
67
68          function changeslider(varargin)
69
70 -            v=get(ed1,'string');
71 -            set(hs1,'value',str2num(v))
72 -            go1
73
74 -          end
```

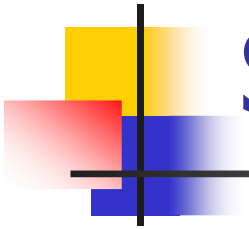
Result





Problem

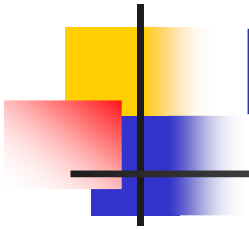
- Remember how we have set the MAX property of the SLIDER to 10?
- Well what if this value is exceeded in the edit box, what should happen?



Solution

- If the value to be set in the slider is higher than its MAX property or lower than it MIN property, then the SLIDER wouldn't be created.

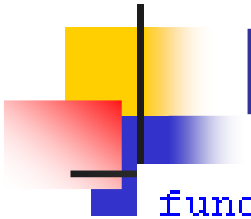
- A solution to this problem is to create an error dialog box whenever this happens, and then proceed to reset the slider.



ERRORDLG Function

- The function ERRORDLG produces an error dialog box with two input parameters.
- The second input is the box title, while the first is the error message itself.
- Syntax:

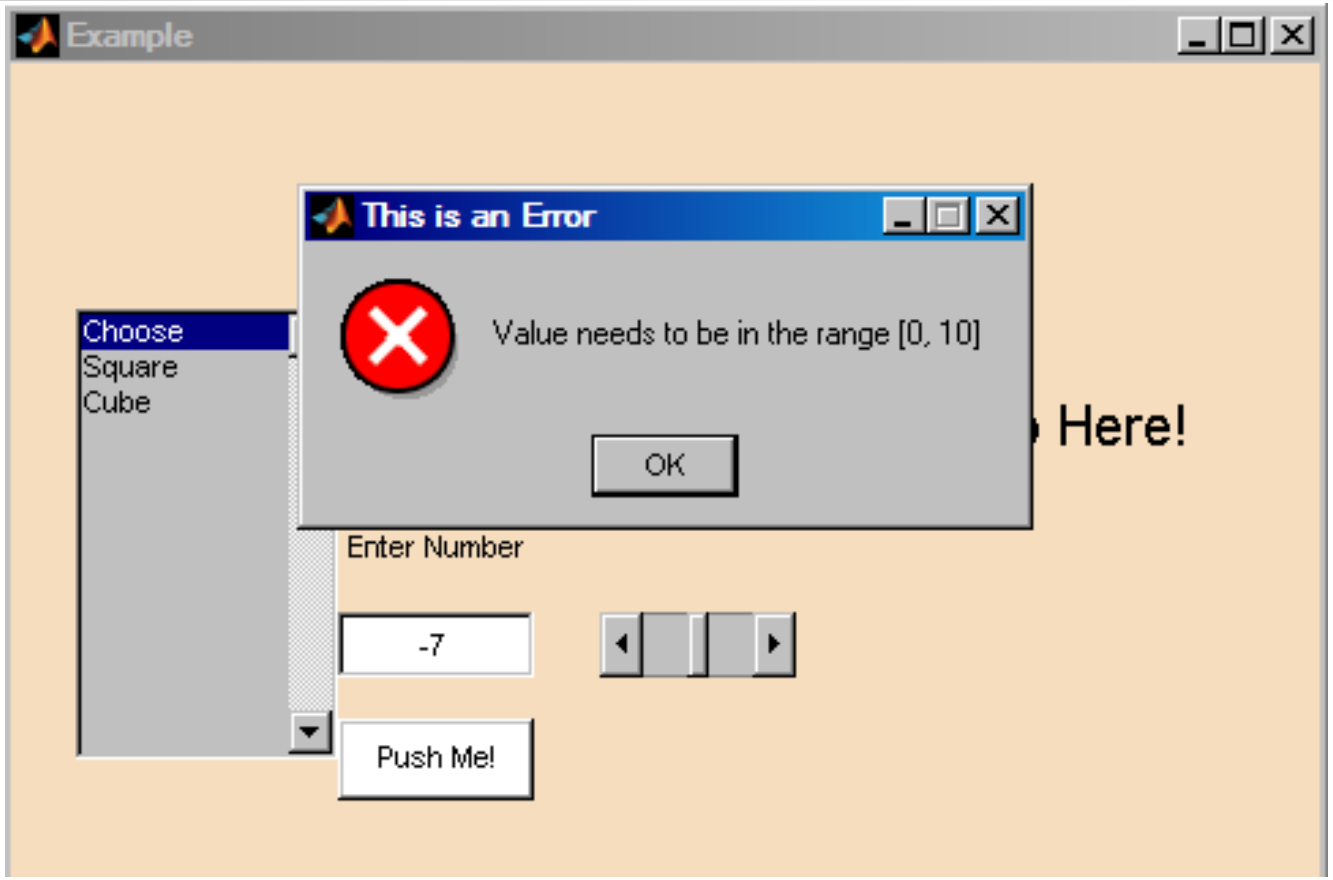
errordlg(title,message)

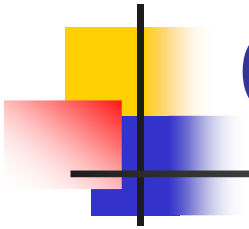


Example

```
function changeslider(varargin)

    v=get(ed1,'string');
    x=str2num(v);

    if x<0 || x>10
        errordlg('Value needs to be in the range [0, 10]',...
            'This is an Error')
        set(hs1,'value',5)
        return
    end

    set(hs1,'value',x)
    go1

end
```

Result


Example
This is an Error
Value needs to be in the range [0, 10]
OK
Choose
Square
Cube
Here!
Enter Number
-7
Push Me!



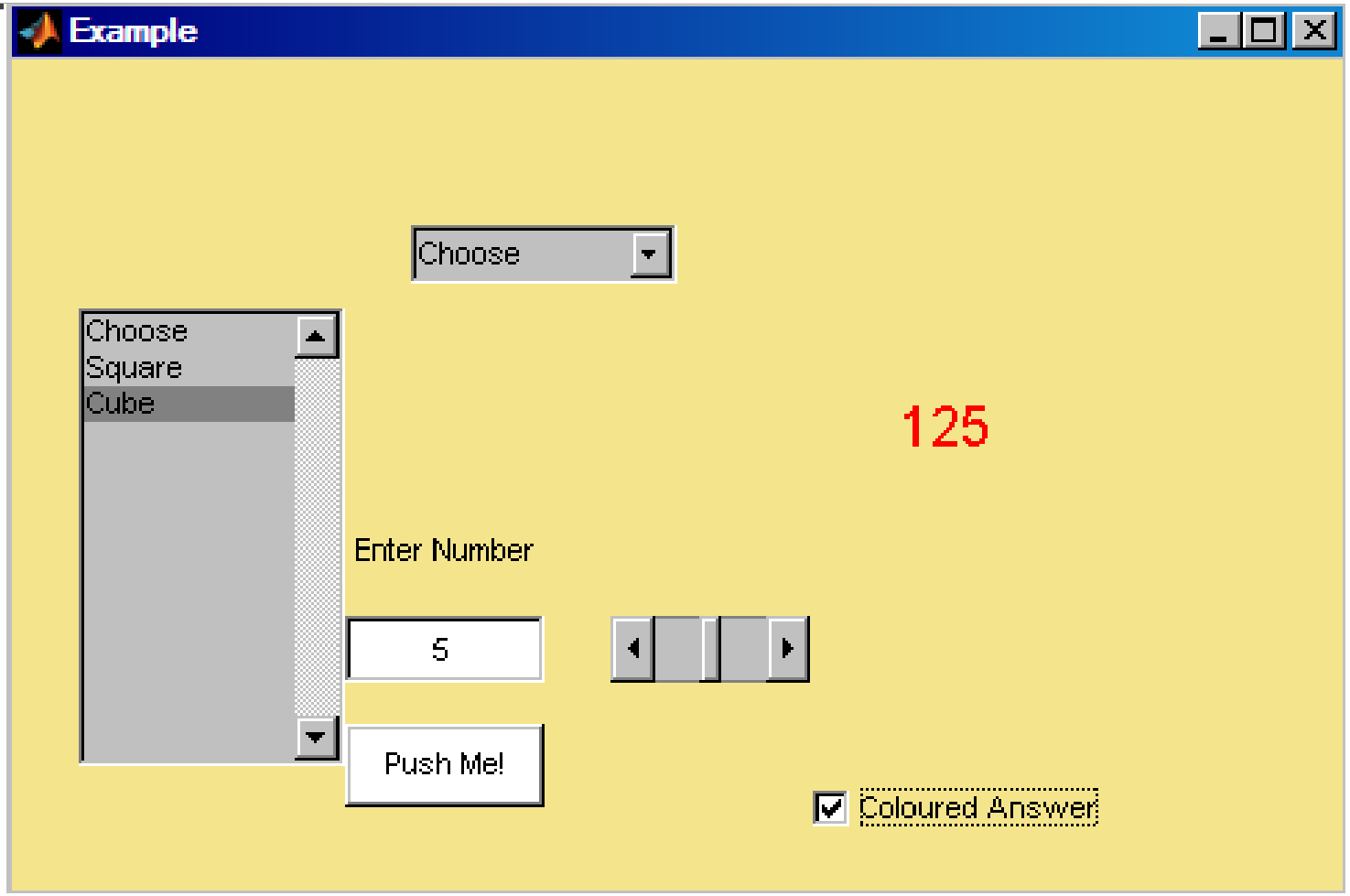
Checkbox UICONTROL

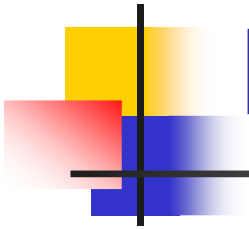
- The checkbox UICONTROL is similar to the listbox and popupmenu in which it has a defining VALUE property.
- The value for each checkbox can either be “1” (i.e. checked) or “0”.
- Another handy property is the ‘string’ property.

Example.

```
102 -   cb=uicontrol('style','checkbox',...
103         'units','normalized',...
104         'position',[.6 .05 .3 .1],...
105         'string','Coloured Answer',...
106         'backgroundcolor',col,...
107         'callback',@chk);
108
109         function chk(varargin)
110 -           val=get(cb,'value');
111 -           if val==1
112 -               set(ht2,'foregroundcolor',[1,0,0])
113 -           else
114 -               set(ht2,'foregroundcolor',[0,0,0])
115 -           end
116 -       end
```

Result


Example
Choose
Choose
Square
Cube
125
Enter Number
5
Push Me!
Coloured Answer



Radiobutton UICONTROL

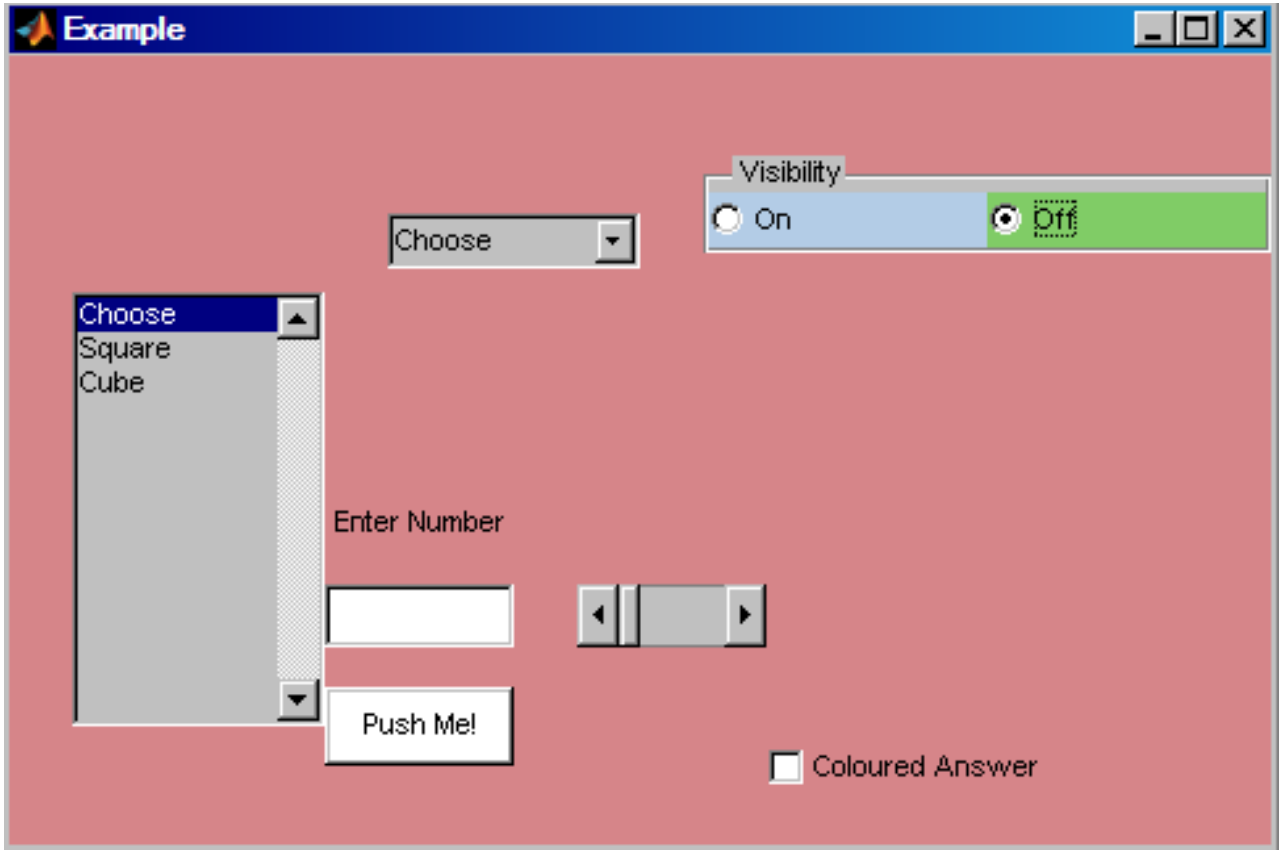
- The radiobutton is very similar to the checkbox.
- You can create a radiobutton group. That way, only one of the buttons in the group can be selected “on” (i.e. has the value of “1”)

Example

```
hr=uibuttongroup('Units','normalized',...
'Position',[.55 .75 .45 .125],...
'SelectionChangeFcn',@grv,... % callback
'title','Visibility');

hb1=uicontrol('Style','Radiobutton',...
    'String','On',...
    'Units','normalized',... the units are normalized
    'Position',[0 0 .5 1],... relative to the uibuttongroup
    'Parent',hr,...
    'backgroundcolor',[.7 .8 .9],...
    'foregroundcolor',[0 0 0]);

hb2=uicontrol('Style','Radiobutton',...
    'String','Off',...
    'Units','normalized',...
    'Position',[0.5 0 .5 1],...
    'Parent',hr,...
    'backgroundcolor',[.5 .8 .4],...
    'foregroundcolor',[0 0 0]);

set(hr,'SelectedObject',hb2) % initial value
```

```
function grv(varargin)
    l=get(hr,'selectedobject');
    if l==hb1
        set(ht2,'visible','on')
    else
        set(ht2,'visible','off')
    end
end
```

Result


Example
Visibility
On
Off
Choose
Choose
Square
Cube
Enter Number
Push Me!
Coloured Answer