Introduction to Graphical User Interface
User Interface (UI)

- Visual part of computer application or operating system through which a user interacts with a computer or a software
- It determines how commands are given to the computer or the program and how information is displayed on the screen
• Can be divided into the following phases:
  – Batch interface
  – Command-line user interface
  – Graphical User Interface

• Depends on the Operating System used
UI – Batch Interface
Punched Card:
Each card represent a line of code. For 100 lines of program codes, need to punch 100 cards!
UI – Batch Interface

• Once the cards were punched, one would drop them in a job queue and wait.
• Operators would feed the deck of punched cards to the computer, perhaps mounting magnetic tapes to supply another dataset or helper software.
• The job would generate a printout, containing final results or (all too often) an abort notice with an attached error log.
• The turnaround time for a single job often spanned entire days. If one were very lucky, it might be hours; real-time response was unheard of.
UI – Command Line

• **Interface** to a computer's OS or an application in which the user responds to a visual prompt by **typing** in a command on a specified line, **receives** a response back from the system and then enters another command, and so forth.

• The MS-DOS Prompt application in a Windows operating system is an example of the provision of a command line interface.
C:\Java\Test\Simple>cabarc -c N test.cab *class

Microsoft (R) Cabinet Tool - Version 1.00.0601 (03/18/97)
Copyright (c) Microsoft Corp 1996-1997. All rights reserved.

Creating new cabinet 'test.cab' with compression 'MSZIP':

-- adding Test.class
-- adding Announcer.class
-- adding TestButton.class
-- adding TestFrame.class
-- adding MyPanel.class
-- adding LabelExample.class
-- adding TextEntryExample.class
-- adding PanelExample.class
-- adding WindowExample.class
-- adding FrameExample.class

Completed successfully

C:\Java\Test\Simple>
UI – Command Line

Command Window

```matlab
>> Ch5a_script
    b
    -------------->
    a
    --------> q
    ....
    vvvvvvvvvvvvvvv
    --------------:--:--:
    ^                 ^
    |                 |
    <--------------- L ------------->
    Please enter 1st distance, a :2
    Please enter 2nd distance, b :5
    Please enter the span, L :7
    Please enter the load, q :20
    Please enter the ratio dx/L :0.01

    Answers:
    Reaction at A is 30.00 kN
    Reaction at B is 30.00 kN
    Max Moment is 82.50 kNm at x = 3.5 m
```

```
    x = 0.000, V = 30.000, M = 0.000
    x = 1.750, V = 30.000, M = 52.500
    x = 3.500, V = 0.000, M = 82.500
    x = 5.250, V = -30.000, M = 52.500
    x = 7.000, V = -30.000, M = 0.000
```
Analysis of Simply Supported Beam

Answer:
- Reaction at A is 64.29 kN
- Reaction at B is 35.71 kN
- Max Moment is 103.32 kNm at x = 3.2 m

Graphical User Interface

Figure 1: SF & BM Diagrams
Graphical User Interface

- Pronounced as "gooey"
- An interface between a user and a computer system that involves the use of a mouse-controlled screen cursor to select options from menus, make choices with buttons, start programs by clicking icons, etc.
- Two of the most popular operating systems, Windows and the Mac OS, are GUI-based.
How does GUI work?

- A GUI provides the user with a familiar environment in which to work.
- It contains pushbuttons, toggle buttons, lists, menus, text boxes and so forth.
- User can concentrate on the purpose of the application instead of the mechanics involved in doing things.
How does GUI work?

- a GUI-based program must be prepared for mouse clicks (or possibly keyboard input) for any GUI element at any time.
- Such inputs are known as events and a program that responds to events is said to be event driven.
GUIs wait for an end user to manipulate a control and then respond to each user action.

Each control, and the GUI itself, has one or more callbacks.

Callbacks: they “call back” to MATLAB to ask it to do things.
How a GUI works?

A particular user action, such as pressing the ‘Calculate’ button, triggers the execution of the button’s callback.

The programmer writes callbacks that define what the components do to handle events.
How does GUI works?

The three principal elements required to create a MATLAB GUI are:

• Components
• Containers
• Callbacks
How does GUI work?

• Components

✓ Each item on a MATLAB GUI is a graphical component
✓ The types of components include graphical controls (pushbuttons, toggle buttons, edit boxes, lists, sliders, etc.), static elements (text boxes), menus, toolbars and axes
How does GUI work?

- **Containers**
  - Components of a GUI must be arranged within a container
  - Is a window on the computer screen
  - The most common container is a figure
  - The other types of containers are panels and button groups
How does GUI work?

• Callbacks

✓ The code executed in response to an event is known as a callback.
✓ There must be a callback to implement the function of each graphical component on the GUI
How a GUI works?

Components:
- Figure
- Containers
- Panel
- Axes
- Static Text
- Push Buttons
- Edit Text
How a GUI work?

```
% --- Executes on button press in btnCalculate.

function btnCalculate_Callback(hObject, eventdata, handles)

% hObject handle to btnCalculate (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Assign input data to input variables
a = str2double(get(handles.txta,'string'));
b = str2double(get(handles.txtb,'string'));
L = str2double(get(handles.txtL,'string'));
q = str2double(get(handles.txtq,'string'));
dx = str2double(get(handles.txtdx,'string'));

% Calculate Reactions
c = b - a;
d = L - b;
Rb = (q*c*(a+c/2))/L;
Ra = q*c - Rb;
```

Callback for ‘Calculate’ Push Button
MATLAB Graphical User Interfaces are created using a tool called **guide**, the GUI Development Environment (*)

This tool allows a programmer to lay out the GUI, selecting and aligning the GUI components to be placed in it.

Once the components are in place, the programmer can edit their properties: name, color, size, font, text to be displayed, and so forth.
Creating & Displaying a GUI

>> guide

Choose: Blank GUI (Default)
Creating & Displaying a GUI
Display the names of the GUI components in the component palette:

a. Select File > Preferences > GUIDE.

b. Select Show names in component palette.

c. Click OK.
The basic steps required to create a MATLAB GUI are as follows:

1. Decide what elements are required for the GUI and what the function of each element will be. Make a rough layout of the components by hand on a piece of paper.

2. Use the MATLAB tool called guide to lay out the components on a figure. The size of the figure and the alignment and spacing of components on the figure can be adjusted using the tools built into guide.
3. Use a MATLAB tool called the Property Inspector (built into guide) to give each component a name (a “tag”) and to set the characteristics of each component, such as its color or the text it displays.

4. Save the figure to a file. When the figure is saved, two files will be created on disk with the same name but different extents. The fig file contains the GUI layout and the components of the GUI; while the M-file contains the code to load the figure as well as skeleton callback functions for each GUI element.
5. Write code to implement the behaviour associated with each callback function
Example 1
Example 1
Example 1

Welcome to SKAA 3413: Computer Programming Course
Example 1

Selamat datang ke kursus SKAA 3413: Pengaturcaraan Komputer

Click Me Klik Saya Exit
Example 1

![Diagram of a GUI layout tool with various components like push buttons, slider, radio button, check box, edit text, static text, pop-up menu, list box, toggle button, table, axes, panel, button group, and ActiveX control.]

- Current Point: [353, 9]
- Position: [64, 49, 437, 101]
Example 1

1. Create 3 Push Buttons and 1 Static Text
2. Double Click the Static Text to open the Property Inspector
3. Change the Tag to txtAns and String to blank
4. Repeat steps 3 & 4 above for the 3 Push Buttons
5. Left Button : Tag = btnClickMe & String = Click Me
6. Middle Button : Tag = btnKlikSaya & String = Klik Saya
7. Right Button : Tag = btnExit & String = Exit
8. Change the Figure properties as follows:
9. Tag = figContoh1 and Name = Contoh1
10. Save As Contoh1
Example 1

2 Files Created: Contoh1.Fig & Contoh1.m

Let's have a look at the Contoh1.m file
Example 1

1. This file contains the main function *Contoh1*, plus sub-functions to specify the behaviour of the active GUI components

2. Sub-functions are:
   i. *Contoh1_OpeningFcn*: Executes just before *Contoh1* is made visible
   ii. *Contoh1_OutputFcn*: Outputs from this function are returned to the command line
   iii. *btnClickMe_Callback*
   iv. *btnKlikSaya_Callback*
   v. *btnExit_Callback*

*Only* edit these 3 Callbacks
Example 1

btnClickMe_Callback:

```matlab
% --- Executes on button press in btnClickMe.
function btnClickMe_Callback(hObject, eventdata, handles)
    % hObject    handle to btnClickMe (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
```

Add the following codes:

```matlab
set(handles.txtAns,'string','...
    'Welcome to SKAA 3413: Computer Programming Course')
```
Example 1

```
function btnKlikSaya_Callback(hObject, eventdata, handles)

% --- Executes on button press in btnKlikSaya.
% hObject    handle to btnKlikSaya (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Add the following codes:

```matlab
set(handles.txtAns,'string',... 'Selamat datang ke kursus SKAA 3413: Pengaturcaraan Komputer')
```
Example 1

**btnExit_Callback**:

```matlab
% --- Executes on button press in btnExit.
function btnExit_Callback(hObject, eventdata, handles)
% hObject    handle to btnExit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Add the following codes:

```matlab
delete(handles.figContoh1)
```

Did you get a similar output as shown in the slide?
• The properties of any object can be examined at any time using the \textit{get} function and modified using the \textit{set} function.

• The most common forms of the \textit{get} function are
  \begin{itemize}
  \item \texttt{value = get(handle,'PropertyName');}
  \item \texttt{value = get(handle)};
  \end{itemize}

• The most common form of the \textit{set} function is
  \begin{itemize}
  \item \texttt{set(handle,'PropertyName1',value1,...);}
  \end{itemize}
In example 1, only the set function was used to change the static text string property during creation time (also known as runtime).

The statement:

```matlab
set(handles.txtAns,'string',... 'Welcome to SKAA 3413: Computer Programming Course'
```

Will change the `txtAns` static text string to a new value i.e. Welcome to .....  

Since the above code was written inside the `btnClickMe` callback function, the above statement will be executed when the `btnClickMe` button is pushed.
Hands-on 1
Type the followings in the command window:

```matlab
>> x = 0:0.1:10;
>> y = x.^2 + 2*x - 3;
>> hdl = plot(x,y);
>> plot_prop = get(hdl);
>> set(hdl,'linestyle','--','Color',[1 0 0])
```
Answers to Hands-on 1

```matlab
>> hdl=plot(x,y)

hdl =
  176.0114

>>
```

Figure 1

```matlab
clc
clear
x=1:0.5:10
y = x.^2 + 2*x - 3;
hdl=plot(x,y)
plot_prop = get(hdl);
set(hdl,'linestyle','--')
clear
x=1:0.5:10
y = x.^2 + 2*x - 3;
clc
```
Answers to Hands-on 1

Change these properties
To: Color – Red
LineStyle – ‘- -’

```
>> plot_prop = get(hdl);
>>
```
Answers to Hands-on 1

Command Window

```matlab
>> plot_prop = get(hdl);
>> set(hdl, 'linestyle', '--', 'Color', [1 0 0])
```

![Figure 1](image-url)
Hands-on 2
(Do example 1)
Example 2
Example 2

1. Change the foreground and background properties of static text

2. Enabling multiline display for static text by changing the max property to > 1
Example 2

1. Open Contoh1.fig and save as Contoh2.fig. Contoh2.m will be created.

2. Change the static text properties Max value to 2

3. Edit the Contoh2.m file. Add the following codes for the btnClickMe callback function. Replace the previous statement with this:

   ```
   ans = sprintf('%s\n%s','Hello,','Welcome to SKAA 3413: Computer Programming Course');
   set(handles.txtAns,...
   'string',ans,...
   'ForegroundColor',[0 0 1],...
   'BackgroundColor',[1 1 1])
   ```
4. Add the following codes for the btnKlikSaya callback function. Replace the previous statement with this:

```matlab
ans = sprintf('%s
%s','Assalamualaikom,',...
'Selamat datang ke kursus SKAA 3413: Pengaturcaraan Komputer');
set(handles.txtAns,...
'string',ans,...
'ForegroundColor',[1 0 0],...
'BackgroundColor',[1 1 0])
```
Hands-on 3
(Do example 2)
Example 3 (Hands-on 4)
Example 3

Section Properties

Data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Answers

- y-bar = ?
- Area = ?
- bx = ?
Example 3

[Image of a computer program interface for calculating section properties. The interface includes fields for inputting values (txtb1, txth1, txtb2, txth2) and buttons for calculate, clear, and exit.]
Example 3

1. Copy U-Section.tif image file to your current folder
2. Create the layout using Guide and save the file as Contoh3
3. Edit the Contoh3.m file. Add the following codes for the `axsGambar_CreateFcn` function
   
   ```
   axes=(hObject)
   imshow('U-Section.tif')
   ```
4. Add the following codes for the `btnCalculate_Callback` function

   ➜ next page
4. Add the following codes for the btnCalculate_Callback function

```matlab
% Assign input data to input variables
b1 = str2double(get(handles.txtb1,'string'));
h1 = str2double(get(handles.txth1,'string'));
b2 = str2double(get(handles.txtb2,'string'));
h2 = str2double(get(handles.txth2,'string'));
Area1 = b1*h1
Area2 = b2*h2
TotalArea = Area1-Area2
```
Example 3

4. Continued

\[ y_1 = \frac{h_1}{2} \]
\[ y_2 = h_1 - \left(\frac{h_2}{2}\right) \]
\[ y_{\text{bar}} = \frac{(\text{Area}_1 * y_1) - (\text{Area}_2 * y_2)}{\text{Total Area}} \]
\[ i_{xx1} = b_1 * h_1^3 / 12 \]
\[ i_{xx2} = b_2 * h_2^3 / 12 \]
\[ i_{xx} = (i_{xx1} + \text{Area}_1 * (y_{\text{bar}} - y_1)^2) - (i_{xx2} - \text{Area}_2 * (y_{\text{bar}} - y_2)^2) \]

```
set(handles.txtybar,'string',ybar);
set(handles.txtarea,'string',TotalArea);
set(handles.txtixx,'string',ixx);
clc;
```
5. Add the following codes for the btnClear_Callback function

%Clear all the input textboxes and reset the output textboxes
set(handles.txth1,'string', '');
set(handles.txtb1,'string', '');
set(handles.txth2,'string', '');
set(handles.txtb2,'string', '');
set(handles.txtybar,'string','?');
set(handles.txtarea,'string','?');
set(handles.txtixx,'string','?');
Example 4
(Hands-on 5)
Example 4

Analysis of Simply Supported Beam

Data
- a = 2 m
- b = 5 m
- L = 7 m
- q = 20 kN/m
- x = 4 m

Answer
- txtAns

Buttons
- btnCalculate
- btnClear
- btnExit
Example 4

1. Copy Beam.tif image file to your current folder.
2. Create the layout using Guide and give suitable tag for the components (edit texts, push buttons, static text and axes)
3. Save the gui as Contoh4 (2 files created Contoh4.fig and Contoh4.m)
4. Edit the Contoh4.m file. Add the following codes for the axsGambar_CreateFcn function
   
   ```matlab
   axes=(hObject)
   imshow('U-Section.tif')
   ```

   *The axsGambar_CreateFcn is not created automatically. Right click the axes, choose view callbacks and click CreateFcn*
Example 4

Analysis of Simply Supported Beam

 axesGambar

Data

\[ a = 2 \text{ m} \]
\[ b = 5 \text{ m} \]
\[ L = 7 \text{ m} \]
\[ q = 20 \text{ kN/m} \]
\[ x = 4 \text{ m} \]
Example 4

```matlab
% --- Executes on button press in btnExit.
function btnExit_Callback(hObject, eventdata, handles)
    % hObject    handle to btnExit (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    delete(handles.figure1)

% --- Executes during object creation, after setting all properties.
function axesGambar_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to axesGambar (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axesGambar
axes = (hObject)
imshow('Beam.tif')
```
5. Add the following codes for the `btnCalculate_Callback` function

```matlab
% Assign input data to input variables
a = str2double(get(handles.txta,'string'));
b = str2double(get(handles.txtb,'string'));
L = str2double(get(handles.txtL,'string'));
q = str2double(get(handles.txtq,'string'));
x = str2double(get(handles.txtx,'string'));
```
5. Continued

% Calculate Reactions

c = b - a;
d = L - b;
Ra = (q*c*(a+c/2))/L;
Rb = q*c - Ra;
5. Continued

% Calculate Moment
if x <= a
    M = Ra *x;
elseif x <= b && x > a
    M = Ra*x - (q*(x - a)^2/2);
else
    M = Rb*(L-x);
end
5. Continued

% Display Calculations Output
Ans1 = 'Reaction at A is %.2f kN\n';
Ans2 = 'Reaction at B is %.2f kN\n';
Ans3 = 'Moment at x = %.1f m is %.2f kNm';
Answer = strcat(Ans1,Ans2,Ans3);
RA = sprintf(Answer,Ra,Rb,x,M);
set(handles.txtAns,'string',RA)
6. Add the following codes for the btnClear_Callback function

```matlab
%Clear all the input edit texts and output static text
set(handles.txta,'string', '');
set(handles.txtb,'string', '');
set(handles.txtL,'string', '');
set(handles.txtq,'string', '');
set(handles.txtx,'string', '');
set(handles.txtAns,'string','');
```
Example 5
(Hands-on 6)
Example 5

1. Use GUIDE to open Contoh4.fig and save as Contoh5.
2. Modify the layout as shown in the next slide and save your gui
3. This exercise will show how to share variables between two push buttons using \textit{GUIDATA} command
4. Similar to example 4, the \textit{Calculate} push button will get the input variables from the gui and calculate the reactions. It will also calculate the shear forces and bending moments at every points on the beam from $x = 0$ to $L$ with steps of $dx*L$
5. The \textit{SFD & BMD} push button will plot the shear force and bending moment diagram. Since variables created in the Calculate push button callback function are local, the values of $x$, shear forces and bending moments need to be shared..
Example 5

Analysis of Simply Supported Beam

Data

- \( a = 2 \) m
- \( b = 5 \) m
- \( L = 7 \) m
- \( q = 20 \) kN/m
- \( dx = 0.01 \) m

Answer

- \( \text{txtAns} \)

Buttons

- btnCalculate
- btnClear
- btnExit
- btnBMD
- txtL
- txta
- txtb
- txtta
- txtdx

Graphical Interface: A simply supported beam with labeled variables and input fields for calculation.
5. The **SFD & BMD** push button will plot the shear force and bending moment diagram. Since variables created in the Calculate push button callback function are local, the values of x, shear forces and bending moments need to be shared between callback functions.

6. What is GUIDATA? Help doc: *GUIDE uses guidata to store and maintain the handles structure. In a GUIDE GUI code file, do not overwrite the handles structure or your GUI will no longer work. If you need to store data other than handles for your GUI, you can add new fields to the handles structure and safely place your data there.*

7. Lets have a look at the codes for the Calculate push button callback function
Example 5

7. Let's have a look at the codes for the Calculate push button callback function

```matlab
% --- Executes on button press in btnCalculate.
function btnCalculate_Callback(hObject, eventdata, handles)
% hObject    handle to btnCalculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Assign input data to input variables
a = str2double(get(handles.txta,'string'));
b = str2double(get(handles.txtb,'string'));
L = str2double(get(handles.txtL,'string'));
q = str2double(get(handles.txtq,'string'));
dx = str2double(get(handles.txtdx,'string'));

% Calculate Reactions

% Rb
Rb = (q*c*(a+c/2))/L;

% Ra
Ra = q*c - Rb;
```
7. Continued...

```matlab
% Calculate Shear Force & Bending Moment
x = 0:dx:L:L;
for i = 1:length(x)
    if x(i) <= a
        M(i) = Ra.*x(i);
        V(i) = Ra;
    elseif x(i) <= b && x(i) > a
        M(i) = Ra.*x(i) - (q.*(x(i) - a)^2/2);
        V(i) = Ra - q.*(x(i)-a);
    else
        M(i) = Rb.*(L-x(i));
        V(i) = Ra - q*c;
    end
end
```
Example 5

7. Continued...

```matlab
[Mmax, loc] = max(M);
xMmax = x(loc);

% Display Calculations Output
Ans1 = 'Reaction at A is %.2f kN\n';
Ans2 = 'Reaction at B is %.2f kN\n';
Ans3 = 'Max Moment is %.2f kNm at x = %.1f m';
Answer = sprintf([Ans1, Ans2, Ans3], R, Mmax, xMmax); 
RA = sprintf(Answer, Ra, Rb, Mmax, xMmax);
set(handles.txtAns, 'string', RA)
```
% Enable the SFD/BMD Button
set(handles.btnBMD,'Enable','On')

% Sharing of Variables x, V & M with btnBMD Callback
handles.x = x;
handles.V = V;
handles.M = M;
guidata(hObject, handles)
Example 5

8. SFD&BMD Push Button Callback Function

```matlab
function btnBMD_Callback(hObject, eventdata, handles)
    % hObject handle to btnBMD (see GCBO)
    % eventdata reserved - to be defined in a future version of MATLAB
    % handles structure with handles and user data (see GUIDATA)

    % Plot SF & BM Diagrams in a new figure
    figure('Name','SF & BM Diagrams')
    subplot(2,1,1);
    area(handles.x,handles.V)
    xlabel('Distance from left support (m)')
    ylabel('SFD (kN)')
    subplot(2,1,2);
    area(handles.x,handles.M)
    ylabel('BMD (kNm)')
    xlabel('Distance from left support (m)')
    set(gca,'YDir','reverse');

    % Enable the Clear and Exit Buttons
    set(handles.btnClear,'Enable','On')
    set(handles.btnExit, 'Enable', 'On')
```