BÖLÜM-V

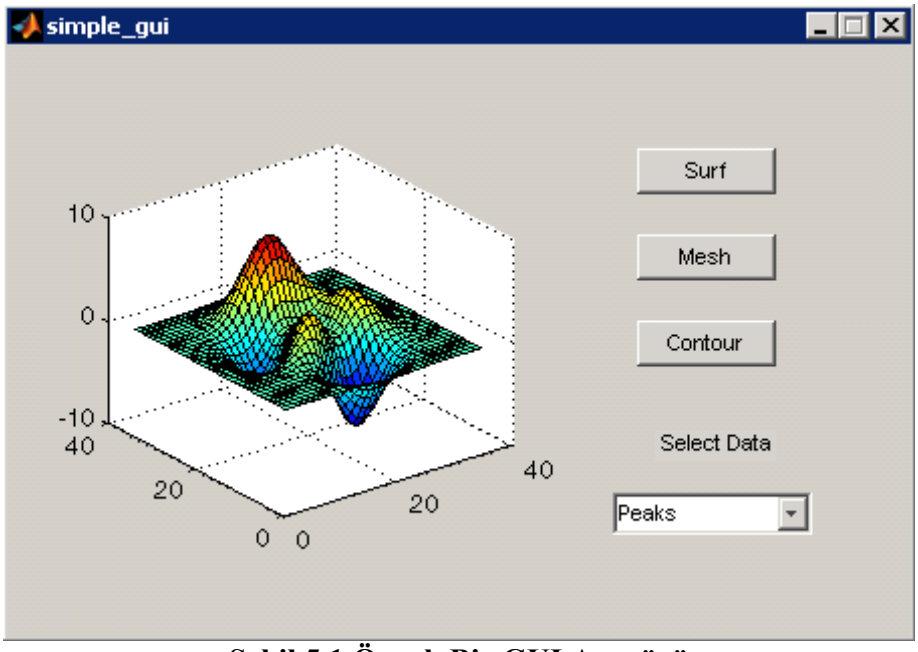
MATLAB'TE GRAFİKSEL KULLANICI ARABİRİMİ (GUI) TABANLI UYGULAMA TASARIMI

5.1 Giriş

İçeriğinde yer alan nesnelerin kullanılması ile kullanıcıya etkileşim sağlayan ve bir işin veya bir programın koşturulmasını sağlayan grafiksel bir program arayüzüdür. Açılımı Graphical User Interface (GUI) dir.

GUI nesneleri menüler, araç çubukları, radio butonlar, liste kutuları veya kaydırıcılar olabilir. Bunların yanında MATLAB GUI ile MATLAB'in sunduğu hesaplama imkânları kullanılarak da data alımı ve grafik çizimi gibi pek çok işlem gerçekleştirilebilir.

Şekil 5.1'de basitçe bir GUI arayüzü görülmektedir.



Şekil 5.1 Örnek Bir GUI Arayüzü

5.2 Grafiksel Kullanıcı Arabirimi (GUI) Nasıl Çalışır?

Her bir nesne (veya komponent) GUI için tanımlanan programlama dosyasında callback diye adlandırılan ayrı alt rutin programlama parçalarına sahiptir. Bu şekilde her bir nesnede oluşan olaylara (örnek olarak bir buton nesnesinin tıklanması ile click event oluşması gibi) GUI o olaya ait callback rutinlerini icra ettirir. Yani, GUI hem bir arayüz hem de bir program çağrılarını icra ettirme mekanizması olarak çalışır.

Yukarıda bahsedilen programlama olay tabanlı programlama diye adlandırılır. Bu tür programlamada her bir olaylara ait alt program parçaları birbirinden bağımsız olarak MATLAB GUI tarafından çalıştırılır.

5.3 Matlab'te GUI Oluşturma Yöntemleri

MATLAB GUI tasarımları iki ayrı yöntem kullanılarak yapılabilir. Bunlar,

- **MATLAB GUIDE aracı kullanılarak,**
- **M-File programlama yöntemi kullanılarak**

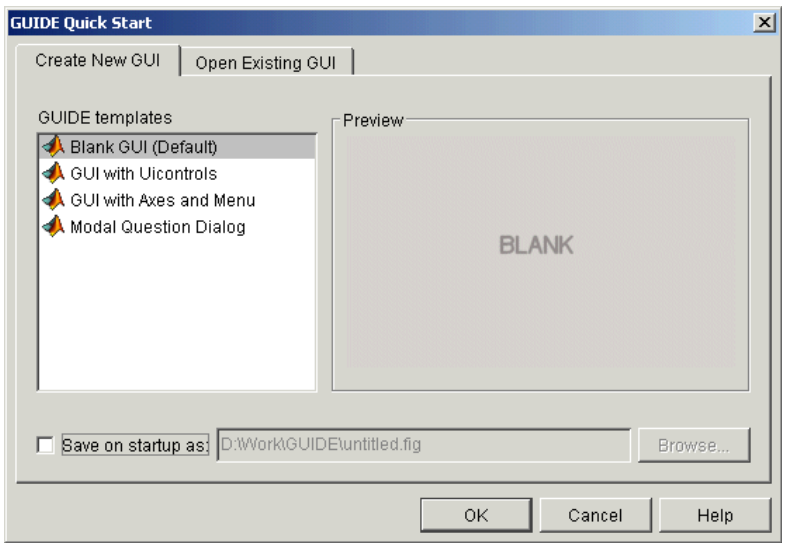
Özellikle GUI tasarımında hızlı arayüzler dizayn etmek ve bu işe ilk başlayan programcılar için MATLAB GUIDE aracının kullanılması büyük bir kolaylık sağlar. Bu aracın kullanılması ile GUI arabirimi kolaylıkla ve yorulmadan sürükle bırak ve açılan pencerelerde özelliklerin değiştirilmesine dayanan bir yöntem kullanılır. Ayrıca, bu yöntemi kullanmanın ileride var olan bir GUI nin düzenlenmesi ve değişiklik yapılması bakımından da çok yararlıdır.

M-File programlama yönteminde tüm GUI tasarımları ve callback program parçalarının yazılması tamami ile programlama kodları kullanılarak yapılır. Burada tasarımcı her şeye hakimdir ve bu teknik uzman bir programlama bilgisi gerektirir. Bu yöntem ile tasarım zamanı uzamasına rağmen programcı her türlü manipülasyonu yapabildiği için programcı açısından çok yararlıdır.

5.4 MATLAB GUIDE Aracı ile GUI Tasarımı Oluşturma

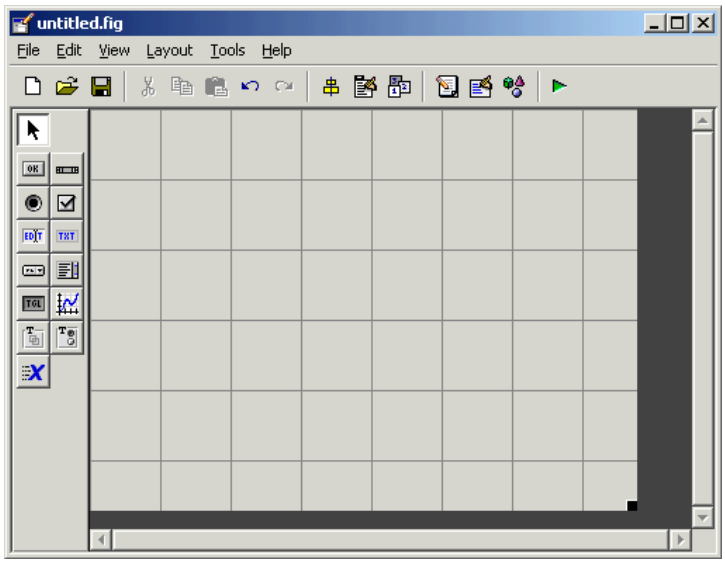
GUIDE matlabin GUI tasarımcılarına sunduğu içerisinde çeşitli araçlar içeren ve kolaylık sağlayan bir grafiksel GUI geliştirme ortamıdır. GUIDE kullanılarak tıkla ve sürükle-burak tekniği ile GUI arayüzüne nesneler (örneğin butonlar, text kutuları, liste kutuları, grafikler v.s.) kolaylıkla eklenebilir. Ayrıca, eklenen nesnelerin hizalanması, tab sırasının değiştirilmesi, görsel ayarlar üzerinde manipülasyonlar yapılması da bu ortamın tasarımcılara sunduğu imkânlardan bazılarıdır.

MATLAB GUIDE aracını tanıyalım. Bu aracını çalıştırmak için ya MATLAB komut satırından GUIDE komutu verilir ya da Start düğmesi tıklanarak MATLAB/GUIDE komutu verilir. Bu adımdan sonra karşımıza Şekil 5.2'deki gibi bir pencere gelir.



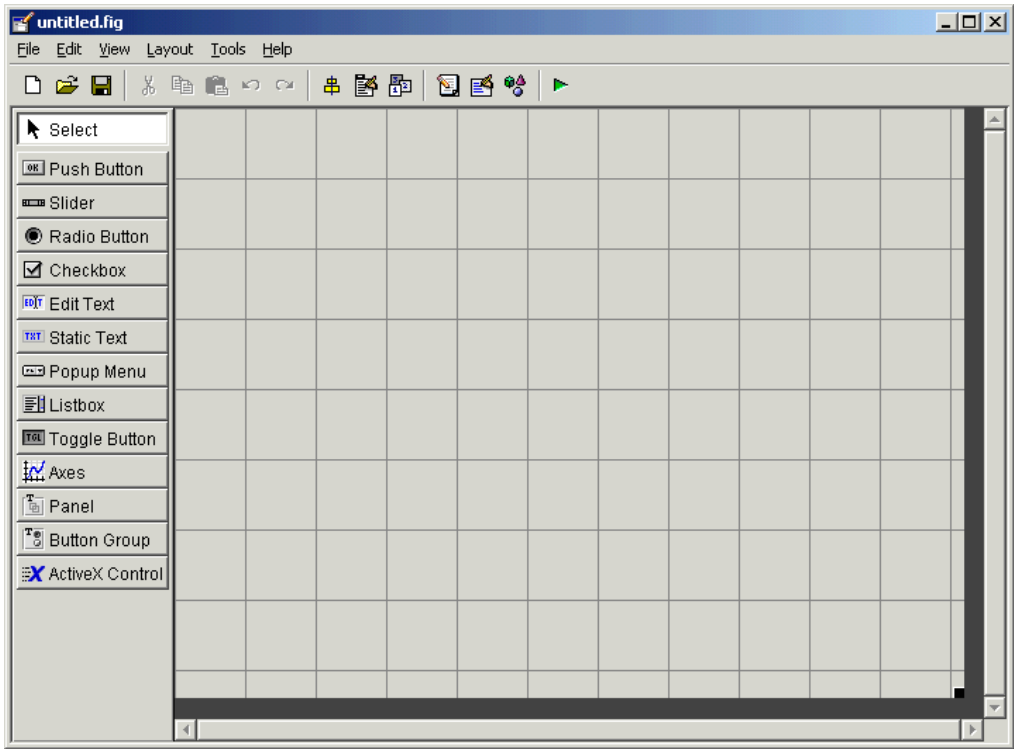
Şekil 5.2

Bu pencereden eğer yeni bir GUI tasarımı yapacak isek Blank GUI seçeneğini seçeriz. Şayet önceden yapılmış bir tasarımı açmak istiyor isek Open Existing GUı sekmesinden sonra istenilen dosyayi seçeriz. Burada yeni bir tasraım oluşturulacağını kabul edelim. Bundan sonra OK düğmesi tıklanılarak Şekil 5.3'teki GUIDE LAYOUT Editor (GUIDE Çalışma Alanı) penceresine ulaşırız.



Şekil 5.3

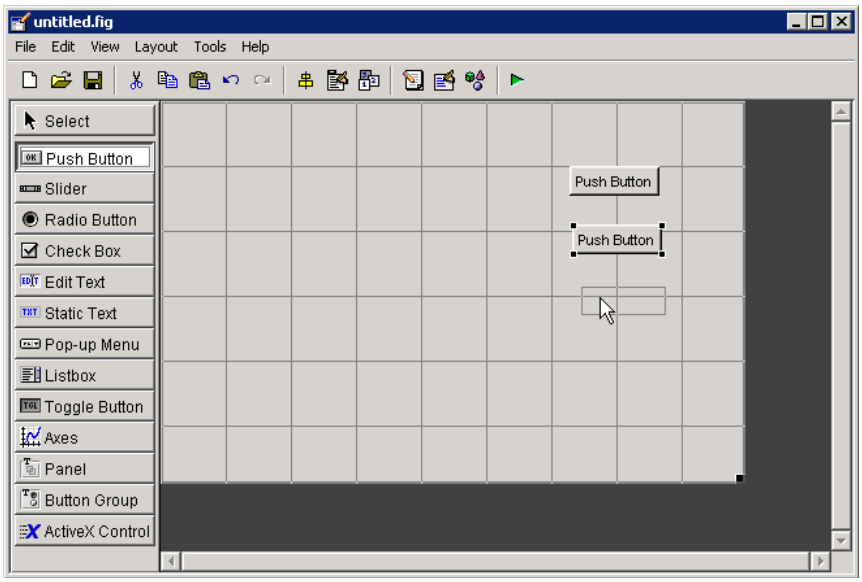
Bu adımdan sonra File/Prefences/Guide yolunu kullanılarak gelen pencereden "Show names in component palette" seçeneğini tiklayip OK düğmesine basalım. Karşımıza Şekil 5.4'teki gibi bir pencere gelecektir.



Şekil 5.4

5.4.1 Komponentleri Çalışma Alanına Ekleme

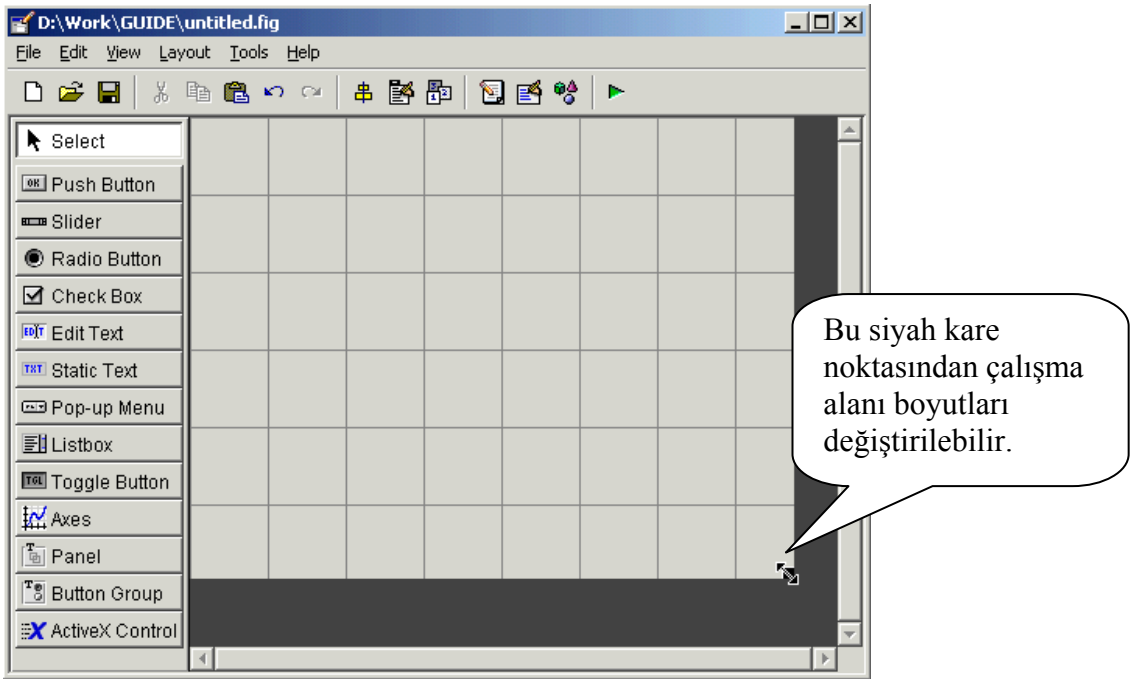
Bunun için sol tarafta bulunan nesne butonlarından istenilen nesneye ait buton tıklanır ve daha sonra çalışma alanında uygun görülen bir noktaya tıklandığında o noktaya ilgili nesne eklenmiş olacaktır. İstenirse çalışma alanındaki bir nesne farenin sol tuşu ile tıklanıp bırakılmadan çalışma alanının herhangi bir yerine sürüklenebilir. Bu durum Şekil 5.5'te de görülmektedir.



Şekil 5.5

5.4.2 Çalışma Alanının Boyutlarını Değiştirmek

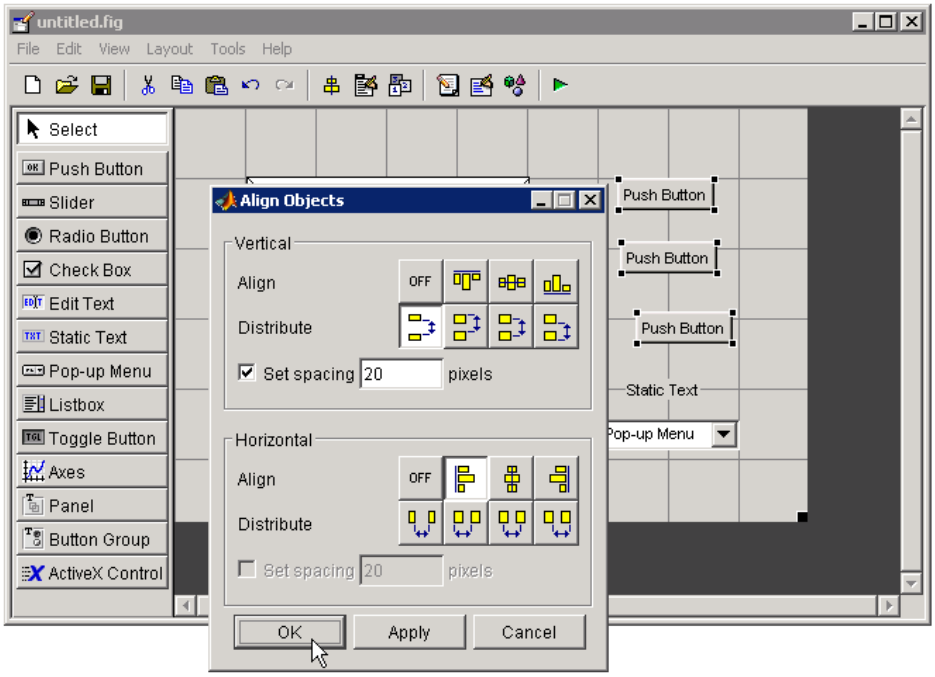
Burada da çalışma alanının sağ alt tarafında bulunan siyah karenin üzerine fare işaretçisi getirilir ve fare işaretçisi konum değiştirdiğinde farenin sol tuşu basılı tutularak çalışma alanı istenilen boyutlarda olacak şekilde düzenleme yapılabilir.



Şekil 5.6

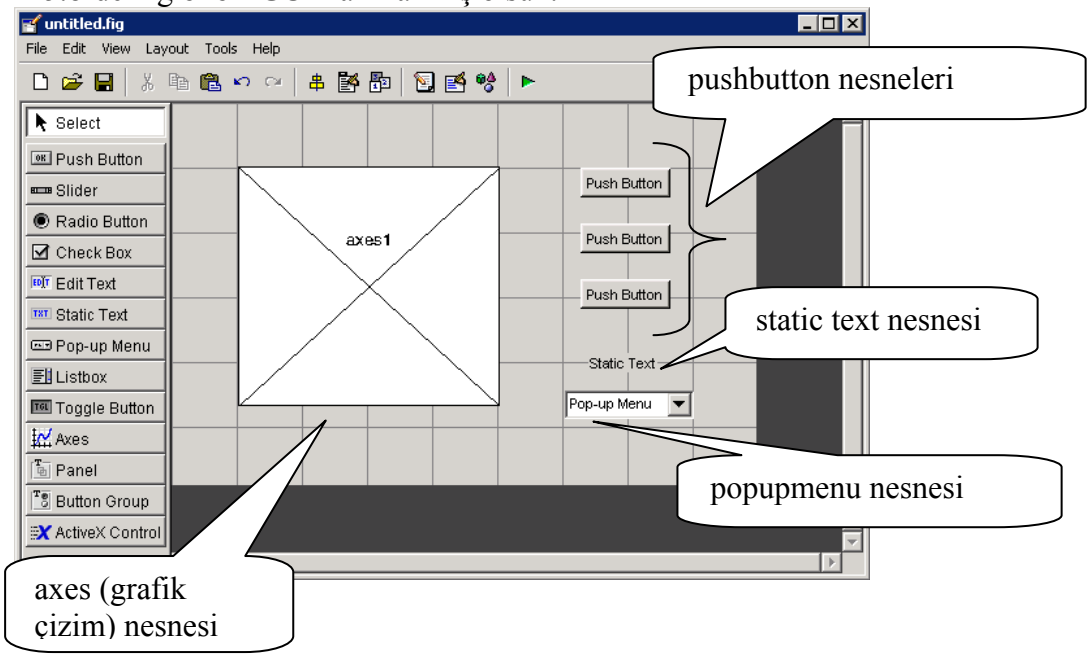
5.4.3 Nesneleri Hizalamak

Bu işlemi yapmak için öncelikle hizalanacak nesneler seçilir. Topluca seçmek için çalışma alanında fare işaretçisini herhangi bir yere tıklayıp sürükleyerek açılan kesik kenarlı pencerenin içinde nesneler kalacak şekilde hareket ettirip, hizalanacak nesneler bu çerçeve içinde kalınca farenin sol tuşunu bırakın. Bu şekilde sadece o çerçeve içinde kalan nesneler seçilmiş olacaktır. Ayrıca, nesneleri Ctrl tuşunu basılı tutarak farenin sol tuşu ile teker teker de seçme imkânı bulunmaktadır. Hizalanacak nesneler seçildikten sonra Tools/Align Objects… yolunu kullanarak Aligment Tool (Hizalama Aracı) penceresini açınız. Şekil 5.7'deki gibi bir ekran ile karşılaşırız. Burada yatay ve dikey hizalamaları kendimize göre butonlardan seçip OK butonuna bastığımız zaman nesnelerimiz hizalanmış olacaktır. Eğer ki hizalama istenilen gibi olmadı ise Ctrl + Z kısayolu ile yapılan işlemler geri alınabilir.



Şekil 5.7

Burada Şekil 5.8'deki gibi bir GUI hazırlanmış olsun.



Şekil 5.8

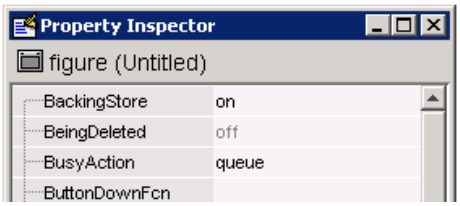
Burada GUI arayüzünde

- Bir adet grafik çizim (axes) nesnesi,
- Bir adet peak, membrane, sinc data setlerini gösteren popup menü,
- Bir adet popup menü başlığı sunan static text nesnesi,
- Üç adet surf, mesh ve contour yazılı buton nesneleri

yer almaktadır.

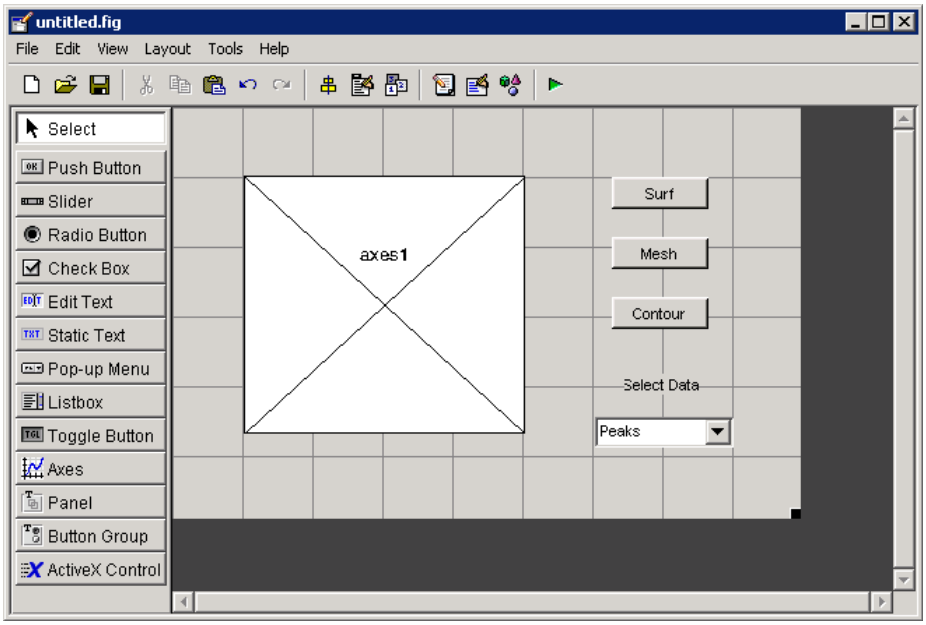
5.4.4 Nesnelere Yazı Ekleme ve Özelliklerini Değiştirme

Nesnelerin özelliklerini değiştirmek istersek ya ilgili nesne farenin sol tuşu ile çift tıklanır ya da ilgili önce seçilip daha sonra View/Property Inspector komutu ile özellikler penceresi



Şekil 5.9

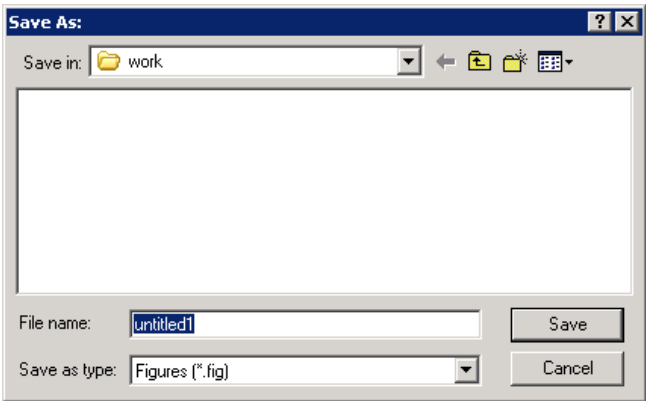
açılır. Buradan örneğimizde eklenen popup menu içeriğine Peaks, Membrane ve Sinc içeriklerini alt alta popup menu nesnesini seçtikten sonra String özelliğine ekleyiniz. Ayrıca, üç adet butonun her birine sırayla Surf, Mesh ve Contour yazıları String özelliklerine eklenmelidir. GUI arayüzü penceresi Şekil 5.10'daki gibi gözükecektir.



Şekil 5.10

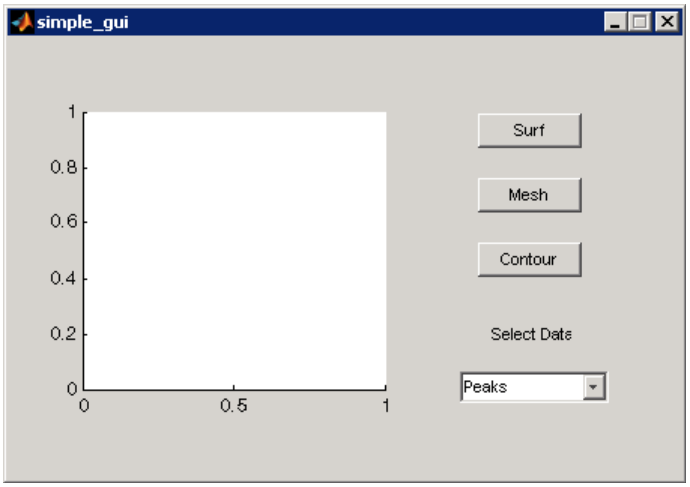
5.4.5 GUI Tasarımını Kaydetme ve Çalıştırma

Bundan sonra bitmiş olan bu GUI arayüzü çalıştırarak görmek için öncelikle Tools/Run yolundan Run (Çalıştır) komutu verilir. Daha sonra gelen pencereden çalışmamnın Run edilebilmesi için kaydedilmesi gerektiğini bildiren bir pencere çıkar Burada Yes butonuna basarız. Bu adımdan sonra MATLAB GUIDE bize tasarımın kaydedileceği dosya ismini



Şekil 5.11

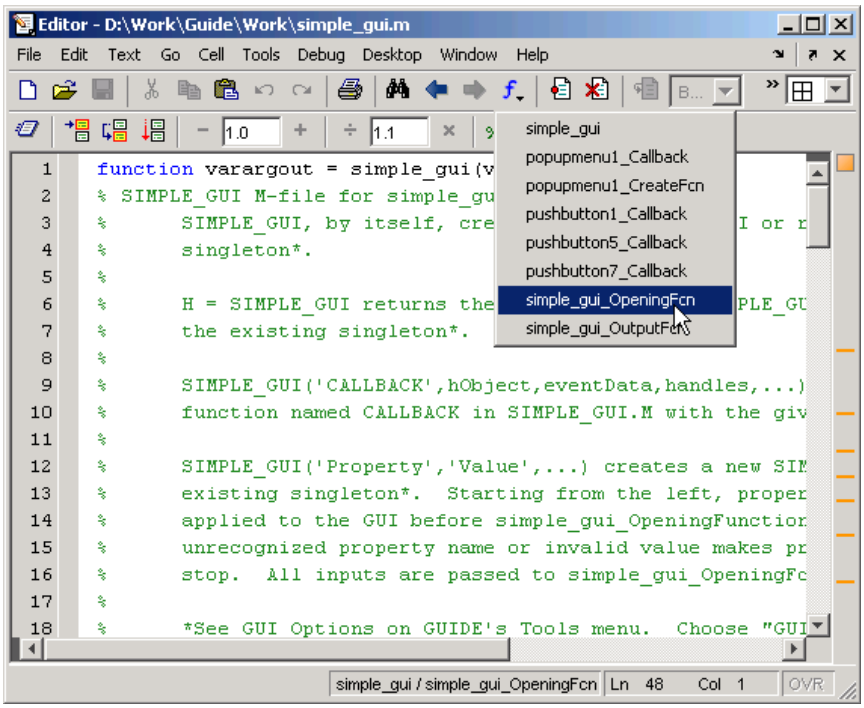
soran bir pencere getirir. Bu pencereden çalışmamıza bir isim vererek tasarımızı kaydetmiş oluruz. Ardından karşımıza Change the MATLAB Directory gibi bir ekran gelirse burada bu ekranı OK tuşuna basarak kapatabilirsiniz. Bu ekran kaydedilen dosya MATLAN tanımlıo dizinler dışında bir yere kaydedilme sözkonusu olduğunda bizi uyarmaktadır. Sonra da GUI tasarımımızın çalışması sonucu gözükecek uygulama penceresi ekranı karşımıza Şekil 5.12'deki gibi bir pencere gelecektir.



Şekil 5.12

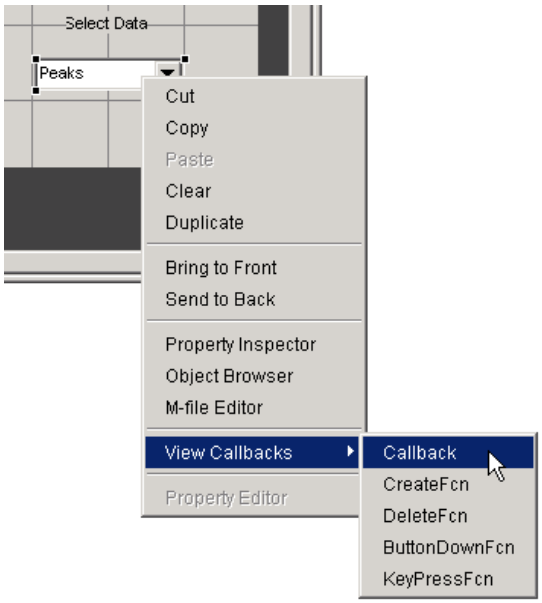
5.5 GUI Arayüzünün Programlanması

Bir GUI arayüzünün programlanması demek o çalışmanın kaydedildiği isimle aynı zamanla oluşturulan .m uzantılı dosya içerisine kodlama satırlarının eklenmesi demektir. Bu dosyanın içine görebilmek, değişiklik yapabilmek için GUIDE çalışma ekranı penceresinden View/M-File Editor komutu işletilebilir. Ardından karşımıza Şekil 5.13'deki gibi bir pencere gelecektir.



Şekil 5.13

Şekil 5.13'deki pencerede hazırlamış olduğumuz GUI tasarımına ait kodlar gözükmektedir. Burada pek çok kodun hazır eklenmiş olduğu görülecektir. Bu kodlar otomatik olarak MATLAB GUIDE tarafından eklenmiştir. Biz burada ilgili butonlara ve liste kutularına ya da istenilen bir nesneye ait callback isimli alt program parçalarına ilgili kodları yazacağız. Bir nesneye ait callback in bulunduğu satıra gitmek için araç çubuğunda yer alan f simgeli butona tıklanır ve açılan listeden ilgili nesneye ait callback in ismi seçilir. Bu durum yukarıdaki pencerede de görülmektedir. Ayrıca, GUIDE çalışma ekranından da direk istenilen bir callback satırına gidilebilir. Bunun için ilgili nesne üzerinde sağ tıklanır ve açılan pencereden View Callbacks menüsünden ilgili callback tıklanması ya da ilgili nesne seçilip View/View Callbacks yolu üzerinden açılan listeden gidilmek istenilen callback tıklanması yeterlidir.



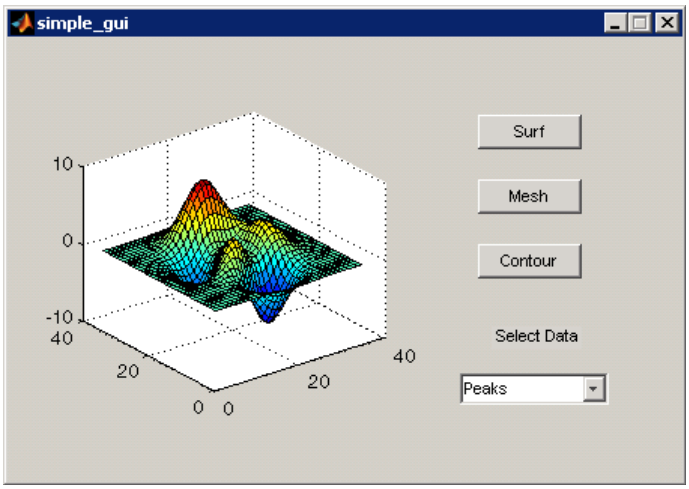
Şekil 5.14

Şimdi GUI arayüzünde yer alan tüm nesneler için View/M-File Editor yolundan kodlama penceresi açılıp, aşağıda yer alan tüm kodlar yazılsın.

```
function varargout = untitled_ilk(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled_ilk_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled_ilk_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function untitled_ilk_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
handles.sinc = sinc;
handles.current_data = handles.peaks;
guidata(hObject, handles);
surf(handles.current_data)

function varargout = untitled_ilk_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
surf(handles.current_data);

function pushbutton2_Callback(hObject, eventdata, handles)
mesh(handles.current_data);

function pushbutton3_Callback(hObject, eventdata, handles)
contour(handles.current_data);

function popupmenu1_Callback(hObject, eventdata, handles)
```

```
str = get(hObject, 'String');
val = get(hObject,'Value');
switch str{val};
case 'Peaks' % User selects peaks.
handles.current_data = handles.peaks;
case 'Membrane' % User selects membrane.
handles.current_data = handles.membrane;
case 'Sinc' % User selects sinc.
handles.current_data = handles.sinc;
end
guidata(hObject,handles)

function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

Burada teker teker nesnelerin üzerinde sağ tıklayıp View Callback ve ilgili Callback satırına gidilip ayrı ayrı da yazılabilirdi. Burada göstermek amaçlı olduğu için kodlar bu şekilde direk verilmiştir. Kodlama satırlarında % işareti ile başlayan satırlar açıklama satırları olup, bu satırlar herhangi bir komut olarak görülmezler sadece açıklama amacı taşırlar. Tüm işlemler tamamladığına göre Tools/Run komutu ile GUI uygulamamızı çalıştırdığımızda Şekil 5.15'teki gibi bir ekran ile karşılaşılır.



Şekil 5.15

Burada yazılan kod parçalarını (callback rutinlerini) kısaca açıklayalım.

```
function varargout = untitled_ilk(varargin)
```

Yukarıdaki function bloğu GUIDE tarafından otomatik olarak oluşturulur. Burada GUI uygulamasına komut satırından gönderilen parametrelerin alınması ve GUI uygulaması çalıştıktan sonra bir fonksiyon olarak dışarıya gönderilecek parametrelerin tanımlanması ile ilgili kod satırları mevcuttur.

function untitled_ilk_OpeningFcn(hObject, eventdata, handles, varargin)

Bu fonsiyon GUI arayüzü ekrana gelmeden (visible olmadan) hemen önce çalıştırılacak kodları içerir. Örneğin böyle bir callback bir GUI uygulaması çalışmadan önce initialization işlemlerinin yapılması ya da bazı GUI nesne özelliklerinin değiştirilmesi istendiğinde kullanılabilir. Ayrıca, varargin giriş parametresi kullanılarak da MATLAB komut satırından girilen parametre değerleri GUI uygulaması içinde kullanılmak üzere bu blokta alınır.

function varargout = untitled_ilk_OutputFcn(hObject, eventdata, handles)

Bu fonksiyon bloğu bir GUI uygulaması hafızadan silinip programı sonlandırılmadan hemen önce (destroy edilmeden önce) çalıştırılacak komutlar içeriri. Ayrıca, komut satırına gönderilecek çıkış parametre değerleri de bu blok tarafından varargout değişkeni kullanılarak işleme konulur.

function pushbutton1_Callback(hObject, eventdata, handles)

Bu callback bloğu pushbutton1 isimli buton (istenilse bu buton ismi butonun Tag özelliğine özellikler penceresinden yeni bir isim verilerek de değiştirilebilir.) ki burada Surf stringine sahip olan buton tıklandığı zaman çalıştırılacak komutları içerir.

function pushbutton2_Callback(hObject, eventdata, handles)

Bu callback bloğu da benzer şekilde pushbutton2 isimli buton ki burada Mesh stringine sahip olan buton tıklandığı zaman çalıştırılacak komutları içerir.

function pushbutton3_Callback(hObject, eventdata, handles)

Bu callback bloğu da benzer şekilde pushbutton3 isimli buton ki burada Contour stringine sahip olan buton tıklandığı zaman çalıştırılacak komutları içerir.

function popupmenu1_Callback(hObject, eventdata, handles)

GUI arayüzüe eklenmiş olan popup_menu nesnesininden herhangi bir eleman tıklanıp seçildiği zaman çalışması istenilen kod parçaları bu callback altında yazılır.

function popupmenu1_CreateFcn(hObject, eventdata, handles)

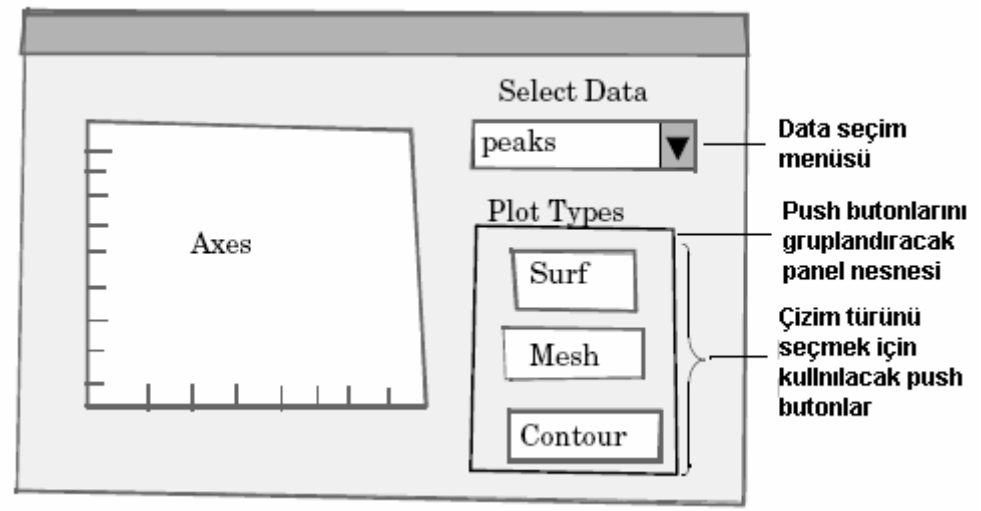
Bu callback GUIDE tarafından otomatik olarak oluşturulmuş olup, popup_menu nesnesi uygulama ekranına gelmeden (visible olmadan) ve de oluşturulmadan önce koşturulacak program satırlarını içerir.

5.6 M-File Programlama Yöntemi Kullanılarak GUI Tasarımı Oluşturma

Burada GUIDE gibi bir tasarım aracı kullanılmaz. Sadece kod satırları yazılarak hem GUI arayüzü hem de bu arayüzün koşturduğu komut satırları aynı dosya içerisinde yazılır. Bu dosyalar .m uzantısına sahiptirler.

Bir GUI arayüzünü bu yöntemle oluşturabilmek için öncelikle tasarım öncesi arayüzün bir planı taslak halinde bir kâğıt üzerine çizilmelidir. Çünkü burada tüm işlemlerin yapılması

muazzam bir çalışma ve ölçümlendirme ile belirlenen nesnelerin uygun yerlere kullanışlı bir GUI arayüzü çıkarmak üzere bir araya gelmesi tamamı ile GUI tasarım ve programcısının yazdığı kodlar ile gerçekleştirilecektir.



Şekil 5.16

Yukarıdaki pencerede görüldüğü üzere bir önceki sayfalarda anlatılan örnek GUI tasarımının taslak görüntüsü görülmektedir. Bu şekilde nesnelerin yerleri tespit edildikten sonra bir GUI uygulaması oluşturulmak üzere programlama yöntemi ile tasarıma geçilebilir.

Şimdi MATLAB komut satırından "edit" komutunu verelim. Karşımıza boş bir m file dosya gelecektir. Genel olarak programlama yolu ile tasarlanılacak GUI uygulaması komut satırları aşağıda belirtilen yapıda olmalıdır. Burada örneğin GUI uygulamamızın adı MYGUI olsun.

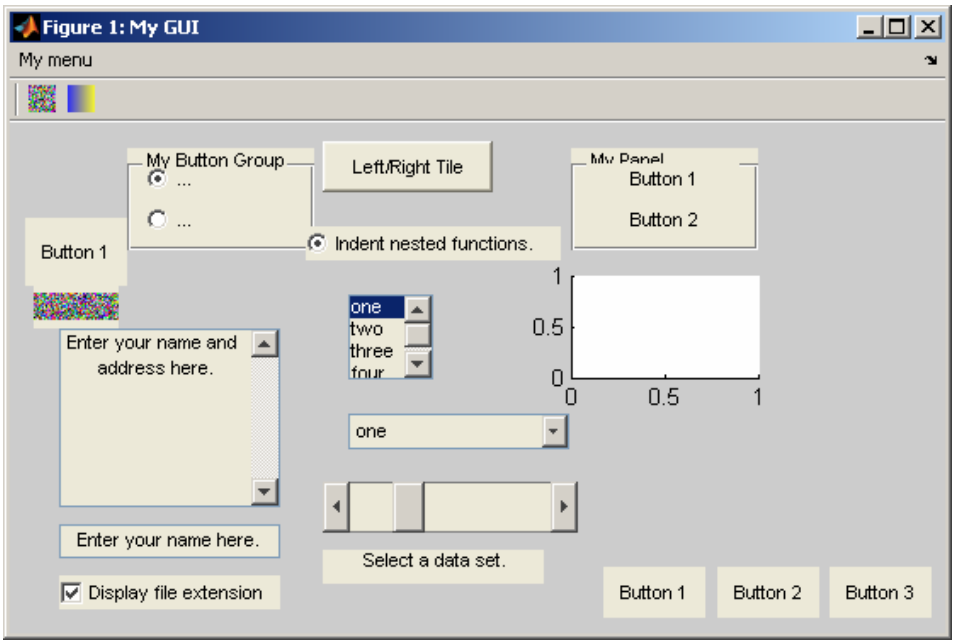
```
function varargout = mygui(varargin) MYGUI uygulaması için mygui.m dosyasi ilk satırı

% MYGUI Kısa bir GUI uyugulaması ile ilgili açıklayıcı bilgi

% Bir adet boş açıklama satırına kadar bu satir ve sonra gelen
% satırlar MATLAB komut satırında GUI uygulamasını
% açıklayıcı ve help komutu ile kullanıcıya sunulan
% yardım satırlarını içerir.

% Burada help satırlarını kod satırlarından ayırmak için bir adet boş açıklama satırı konulur.

% GUI uygulamasının giriş parametre alınması ve ilk önhazırlık işlemleri bloğu

% GUI nesnelerinin oluşturulması ile ilgili satırlar bloğu

% Callback ler öncesi önhazırlık işlemleri bloğu

% MYGUI için gerekli callback fonksiyonları

% MYGUI için kullanılacak fonksiyonlar bloğu
```

end Bu komut fonksiyon bloğunun sonunu belirtmek için konulmuştur.

Yukarıdaki yapıyı oluşturacak şekilde komutlar MYGUI isimli GUI uygulaması için mygui.m isimli dosyaya kaydedilir. Bu GUI uygulamasını çalıştırmak için de MATLAB komut satırından sadece "mygui" komutunun verilmesi yeterlidir. Bu şekilde uygulama penceresi karşımıza gelecektir. Şu aşamada herhangi bir kod yazılmadığı herhangi bir şey olmayacaktır. Ancak, yazılmış olsaydı ilgili GUI penceresi görülecekti.

5.6.1 Programlama Yoluyla Nesnelerin Eklenmesi

Yukarıda bahsedilen mygui.m dosyasının içeriğine aşağıda belirtilen kodları eklediğimizde GUI arayüzümüz şu şekilde görülecektir.



Şekil 5.17

```
function varargout = mygui(varargin)                    mygui icin fonksiyon tanımı

fh = figure('Visible','on','Name','My GUI',...          bu satırlar ekrana belirtilen boyut
'Position',[360,550,550,300]);                          ve konumda figure (GUI yüzeyi) getirme

cbh = uicontrol(fh,'Style','checkbox',...               GUI yüzeyine checkbox nesnesi
'String','Display file extension',...                   ekleme
'Value',1,'Position',[30 15 130 20]);

eth = uicontrol(fh,'Style','edit',...                   GUI yüzeyine edit kutusu ekleme
'String','Enter your name here.',...
'Position',[30 45 130 20]);

eth = uicontrol(fh,'Style','edit',...                   GUI yüzeyine çok satırlı edit kutusu
'String','Enter your name and address here.',...        ekleme (çünkü max-min>1 durumu)
'Max',2,'Min',0,...
'Position',[30 75 130 105]);
```

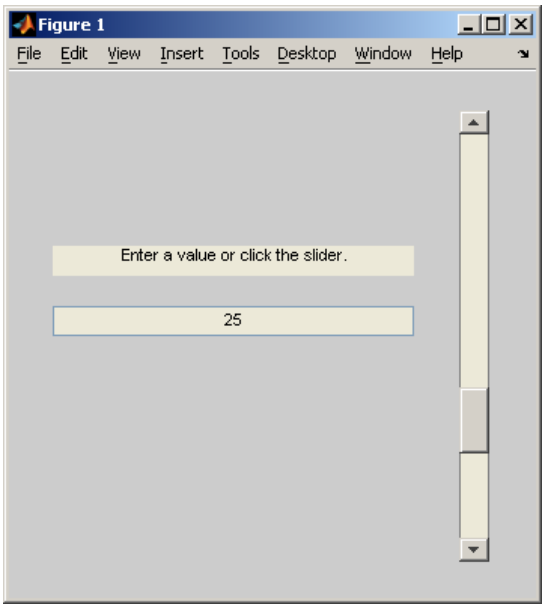
```
lbh = uicontrol(fh,'Style','listbox',...                GUI yüzeyine liste kutusu ekleme
'String',{'one','two','three','four'},...               (elemanlari 'one','two','three','four' )
'Value',1,'Position',[200 150 50 50]);

pmh = uicontrol(fh,'Style','popupmenu',...              GUI yüzeyine popup menü ekleme
'String',{'one','two','three','four'},...               (elemanlari 'one','two','three','four' )
'Value',1,'Position',[200 110 130 20]);

pbh1 = uicontrol(fh,'Style','pushbutton','String','Button 1',...    GUI yüzeyine push buton
'Position',[10 205 60 40]);                                         ekleme

img(:,:,1) = rand(16,64);                               rasgele sayılardan oluşan dizi tanımı
img(:,:,2) = rand(16,64);
img(:,:,3) = rand(16,64);

pbh2 = uicontrol(fh,'Style','pushbutton',...            GUI yüzeyine push buton ekleme
'Position',[15 180 50 25],...
'CData',img);

rbh = uicontrol(fh,'Style','radiobutton',...            GUI yüzeyine radio buton ekleme
'String','Indent nested functions.',...
'Value',1,'Position',[175 220 150 20]);

sh = uicontrol(fh,'Style','slider',...                  GUI yüzeyine kaydırıcı ekleme
'Max',100,'Min',0,'Value',25,...
'SliderStep',[0.05 0.2],...
'Position',[185 60 150 30]);

sth = uicontrol(fh,'Style','text',...                   GUI yüzeyine static text kutusu
'String','Select a data set.',...                       ekleme
'Position',[185 30 130 20]);

tbh = uicontrol(fh,'Style','togglebutton',...           GUI yüzeyine toggle (çift durumlu)
'String','Left/Right Tile',...                          buton ekleme
'Value',0,'Position',[185 260 100 30]);

ph = uipanel('Parent',fh,'Title','My Panel',...         GUI yüzeyine panel ekleme
'Position',[.60 .75 .2 .2]);

pbh3 = uicontrol(ph,'Style','pushbutton','String','Button 1',...    ph paneline push buton
'Units','normalized',...                                            ekleme
'Position',[.1 .55 .8 .3]);
pbh4 = uicontrol(ph,'Style','pushbutton','String','Button 2',...    ph paneline 2. push buton
'Units','normalized',...                                            ekleme
'Position',[.1 .15 .8 .3]);

bgh = uibuttongroup('Parent',fh,'Title','My Button Group',...       GUI yüzeyine buton grubu
'Position',[.125 .75 .2 .2]);                                       ekleme

rbh1 = uicontrol(bgh,'Style','radiobutton','String','Red',...       bgh grubuna radio buton
```

```
'Units','normalized',...                                    ekleme
'Position',[.1 .6 .3 .2]);
rbh2 = uicontrol(bgh,'Style','radiobutton','String','Blue',...   bgh grubuna 2. radio buton
'Units','normalized',...                                    ekleme
'Position',[.1 .2 .3 .2]);

ah = axes('Parent',fh,'Position',[.60 .50 .2 .2]);          GUI yüzeyine grafik çizim
                                                            alanı ekleme

b1 = uicontrol(fh,'Posit',[330 80 60 30],'String','Button 1');   GUI yüzeyine buton 1 koy
b2 = uicontrol(fh,'Posit',[350 50 60 30],'String','Button 2');   GUI yüzeyine buton 2 koy
b3 = uicontrol(fh,'Posit',[310 10 60 30],'String','Button 3');   GUI yüzeyine buton 3 koy

align([b1 b2 b3],'Right','None');                           b1, b2 ve b3 nesnelerini sağa hizala
%align([b1 b2 b3],'Left','Distribute');                     b1, b2 ve b3 nesnelerini sola dağınık hizal.
align([b1 b2 b3],'Center','Fixed',7);                       b1, b2 ve b3 nesnelerini ortala
align([b1 b2 b3],'Fixed',5,'Bottom');                       b1, b2 ve b3 nesnelerini aşağıyı doğru hiz.

set(fh,'MenuBar','figure');                                 standart araç çubuğunun gösterilmesi
set(fh,'MenuBar','none');                                   standart araç çubuğunun gizlenmesi

mh = uimenu(fh,'Label','My menu');                          GUI yüzeyi menüsüne My menu eklenm.
eh1 = uimenu(mh,'Label','Item 1');                          mh menüsüne alt menü tanımlanması
eh2 = uimenu(mh,'Label','Item 2','Checked','on');           mh menüsüne alt menü tanımlanması

set(eh2,'Separator','on');                                  mh2 menu seceneğinin üzerine ayıraç kon.

seh1 = uimenu(eh1,'Label','Choice 1','Accelerator','C',...  eh1'e kisayol (Ctrl+C) tanımı ve
'Enable','off');                                            pasif yapılması
seh2 = uimenu(eh1,'Label','Choice 2','Accelerator','H');    eh2'ye kisayol (Ctrl+H) tanımı

th = uitoolbar(fh);                                         GUI yüzeyine araç çubuğu ekleme

a = [.20:.05:0.95];                                         bu ve alt satırlarla rasgele renkleri
img1(:,:,1) = repmat(a,16,1);                               temsil etmek üzere dizilerin tanımlanması
img1(:,:,2) = repmat(a,16,1);
img1(:,:,3) = repmat(flipdim(a,2),16,1);

pth = uipushtool(th,'CData',img1,...                        th araç çubuğuna push buton ekleme
'TooltipString','My push tool',...
'HandleVisibility','off');                                  handlevisibility komut satırından erişimi ayarlar

img2 = rand(16,16,3);                                       rastgele renk dizisi tanımlama
tth = uitoggletool(th,'CData',img2,'Separator','on',...     araç çubuğuna ayırıcı ekleme
'TooltipString','Your toggle tool',...
'HandleVisibility','off');

oldOrder = allchild(th);                                    bu ve aşağıdaki satırlar ile nesnelerin tab tuşu
newOrder = flipud(oldOrder);                                ile geçiş sırası ayarlanmakta
set(th,'Children',newOrder);
```

delete(tth);	tth handleini tutan nesnenin (burada araç çubuğu) silinmesi
end	fonksiyon sonu

5.6.2 Programlama Yöntemi ile GUI Tasarımında Callback Kullanımı

Şekil 5.18'de görülen penceredeki GUI arayüzü tasarlanmış olsun. Burada amaçlanan kaydırıcının her hareketini text kutusunda göstermek, aynı zamanda text kutusuna 0-100 aralığında değerler girilip enter tuşuna basınca kaydırıcının değerini de o değere set etmektir. Şayet kullanıcı bu aralığın dışında ya da hatalı giriş yaparsa hata sayısını text kutusuna yazdırmaktır.



Şekil 5.18

Böyle bir GUI tasarımı için aşağıdaki kodlar (komut satırından edit komutu ile Editor penceresini açarak) slider_gui.m isimli dosyaya yazılsın ve kaydedilsin.

```
function slider_gui

fh = figure('Position',[250 250 350 350]);

sh = uicontrol(fh,'Style','slider',...
'Max',100,'Min',0,'Value',25,...
'SliderStep',[0.05 0.2],...
'Position',[300 25 20 300],...
'Callback',@slider_callback);

eth = uicontrol(fh,'Style','edit',...
'String',num2str(get(sh,'Value')),...
'Position',[30 175 240 20],...
'Callback',@edittext_callback);

sth = uicontrol(fh,'Style','text',...
```

```
'String',Bir değer girin veya kaydırıcıyı kullanın.',...
'Position',[30 215 240 20]);
number_errors = 0;

function slider_callback(hObject,eventdata)
set(eth,'String',...
num2str(get(hObject,'Value')));
end

function edittext_callback(hObject,eventdata)
val = str2double(get(hObject,'String'));
% text kutusuna girilen değer kaydırıcının min ve max
% değerleri arasında ise kaydırıcının yeni değerini set et.
if isnumeric(val) && length(val) == 1 && ...
val >= get(sh,'Min') && ...
val <= get(sh,'Max')
set(sh,'Value',val);
else
% Hatalı giriş söz konusu ise hata sayacını bir arttır
number_errors = number_errors+1;
set(hObject,'String',...
['Toplam hatalı giriş sayısı = ',...
num2str(number_errors)]);
end

end
```

Buradaki GUI öğreğimizde toplam iki adet callback fonksiyonu kullanılmıştır. “slider_gui” isimli fonksiyon bizim GUI ana uygulamamız için tanımlanmıştır. Burada “slider_callback(hObject,eventdata)” ve “edittext_callback(hObject,eventdata)” isimli callbackler “uicontrol” komutu ile komut satıralrının en başında nesneler GUI yüzeyine eklenirken “Callback” özelliklerine parametre olarak aktarılmıştır. Dolayısıyla hangi nesne ile ilgili bir olay (event) oluşmuş ise ona ait ve o olayla ilgili callcak ler GUI tarafından çalıştırılacaktır. Burada

```
function slider_callback(hObject,eventdata)
```

isimli callback fonksiyonu kaydırıcı nesnemizin değeri değiştirilince icra edilecek komut satırlarını içerir.

```
function edittext_callback(hObject,eventdata)
```

isimli callback fonksiyonu ise texy kutusu içerisine herhangi bir değer girilince (bu sayı da olabilir veya string tipi değerler de olabilir.) ve enter tuşuna basılınca koşturulacak olan komutları içermektedir.

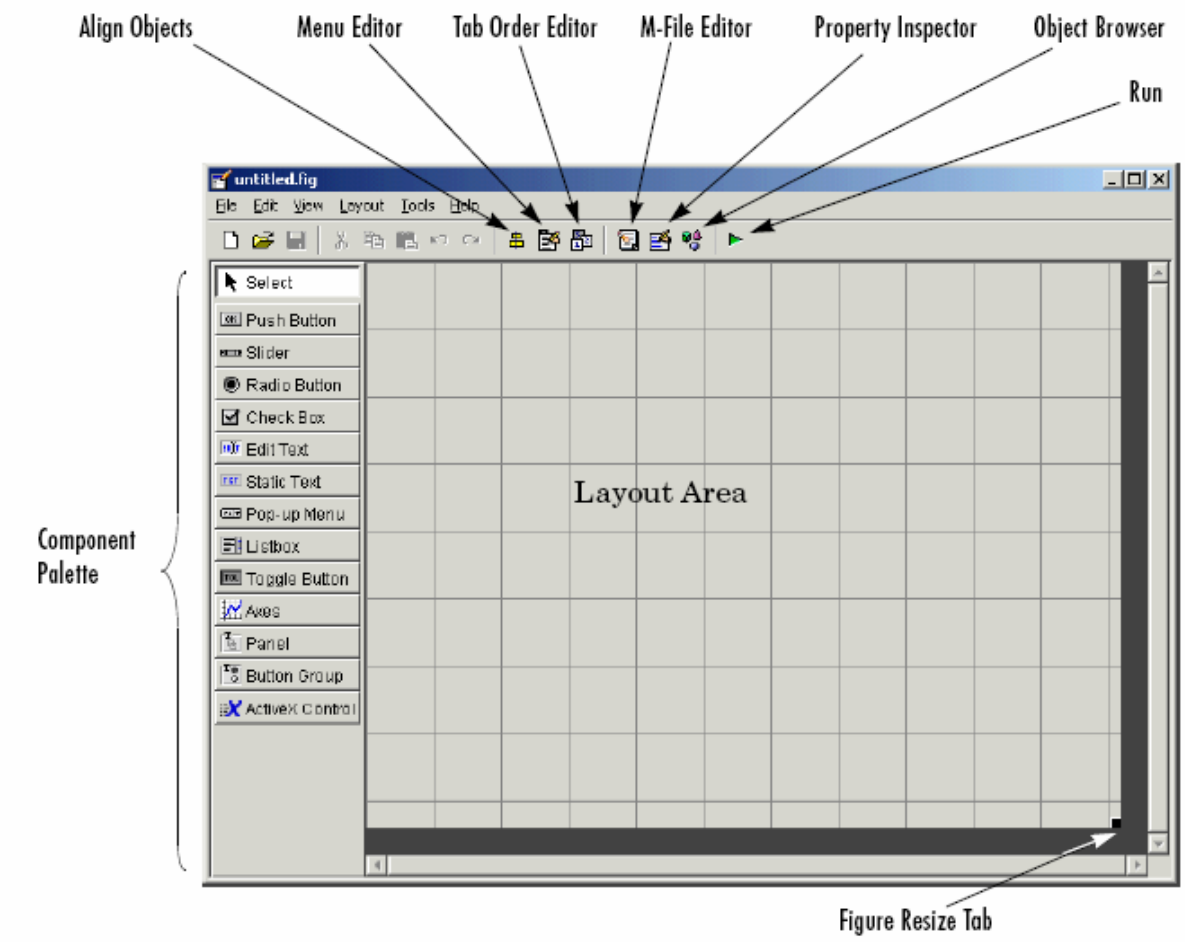
Bu programda özetle text kutusuna girilen değer

```
val = str2double(get(hObject,'String'));
```

komutu ile önce double tipinde sayısal değere çevrilmektedir. Ardından gelen if sorgusu ile girilen değer sayısal ve herhangi bir hata yoksa kaydırıcının yeni değeri set edilmektedir. Fakat girilen değer hatalı ise if sorgusunda bu durum öğrenilmekte ve de else ifadesinden sonra gelen komutlar icra edilmekte olup, burada da "number_errors" isimli genel bir değişken oluşan toplam hata sayısı için sayaç görevi görmekte ve hata durumunda değeri bir arttırılmaktadır. Daha sonra da oluşan hata durumu text kutusunun "string" özelliğinden faydalanılarak kullnıcıya gösterilmektedir.

5.7 GUIDE Aracının İncelenmesi

Bir önceki konularımızda da bu aracı kısaca incelemye çalıştık. Burada GUIDE aracı detaylı olarak incelenecektir. MATLAB komut satırından "guide" komutunu yazdığımızda ve gelen pencereden boş (blank) bir GUI tasarımını seçtiğimizde Şekil 5.19'daki pencere ile karşılaşılır.



Şekil 5.19

Bu ekrandaki araçlar ile ilgili açıklama aşağıda verilmiştir.

5.7.1 Layout Editor

GUIDE çalışma alanı ve penceresidir. Bu ekran ile GUI yüzeyine component paletten seçilen ilgili nesneler eklenebilir ya da diğer araçlar ile program kodlarının yazılması,

nesnelerin GUI yüzeyi üzerinde hizalanması, tab tuşu geçiş sırasının değiştiirlmesi gibi pek çok işlem gerçekleştirilebilir.

5.7.2 Figure Resize Tab

Bu araç GUI çalışma alanının boyutlandırılmasını sağlar. Fare işaretçisi bu alan üzerine getirildiğinde konum değiştirecektir. Bu anda farenin sol tuşu tıklanıp ileri geri hareket ettirilerek GUI yüzey alanının boyutları değiştirilebilir.

5.7.3 Menu Editor

GUI uygulamasına istenilirse File, Edit… v.b. gibi menü içeren programlarda olduğu gibi bir manü eklenmesi ve eklenen menü ile ilgili işlmelerin yapılması bu araç vasıtasıyla sağlanır.

5.7.4 Align Objects

Bu araç sayesinde GUI çalışma alanına eklenilen nesnelerin yatay ya da dikey olarak hizalanması işlemleri gerçekleştirilir.

5.7.5 Tab Order Editor

Tab Order Editor kullanılarak GUI yüzeyindeki nesnelerin birinden diğerine tab tuşu ile geçiş sırası (örneğin bir buton seçili ve aktif iken bir başka butona ya da bir liste kutusuna tab tuşu kullanılarak geçilmesi gibi.) değiştirilebilir.

5.7.6 Property Inspector

Bu pencere sayesinde de GUI uygulamasına eklenen nesnelerin özellikleri değiştirilebilir ya da var olan özelliklerinin ve değerlerinin neler olduğu gözlenilebilir.

5.7.7 Object Browser

Bu araç ile tasarımcı GUI uygulamasına eklemiş olduğu nesnelerin ve isimlerinin neler olduğu genel hali ile bakabilir.

5.7.8 Run

Bu buton yardımı ile de hazırlanmış olan bir GUI uygulaması çalıştırılabilir. Bu şekilde tasarımcı hazırlamış olduğu GUI'yi test etme imkânına sahiptir.

5.7.9 M-File Editor

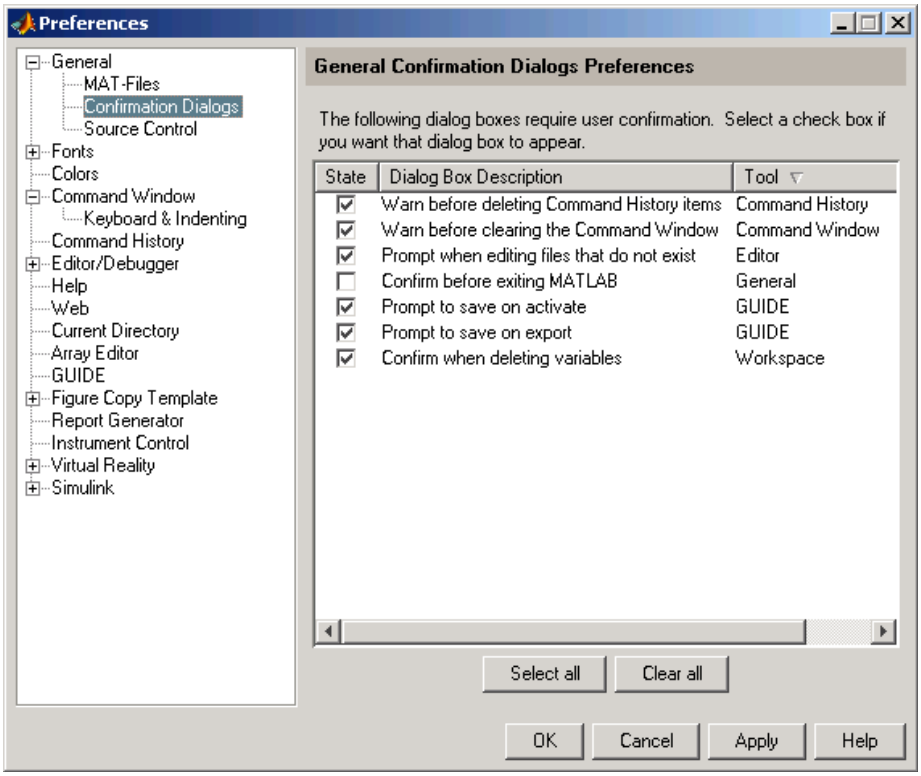
Hazırlanmış olan GUI uygulaması ile ilgili komutları görebilmek ve üzerinde değişiklik yapabilmek için bu araç kullanılır.

5.7.10 GUIDE Tercihleri

Bu tercihleri görebilmek için GUIDE ekranında File menüsünden Preferences komutu çalıştırılır.

5.7.10.1 Doğrulama Seçenekleri

Bu seçeneklere ulaşmak için General/Confirmatin yolu izlenmelidir. Karşımıza Şekil 5.20'deki gibi bir pencere gelecektir.

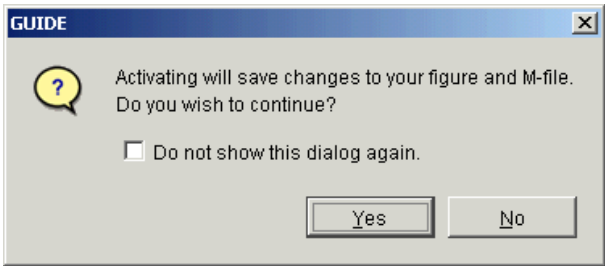


Şekil 5.20

Bu penceredeki seçeneklerden iki tanesi GUIDE ile ilgilidir. Bu seçeneklerin görevleri şu şekildedir:

- **Promp to Save on Activate**

Bu seçenek seçili ise GUIDE bir GUI uygulaması çalıştırılmadan önce onun kaydedilmesi gerektiğini bildiren Şekil 5.21'deki gibi bir pencere ile kullanıcı uyarılır.

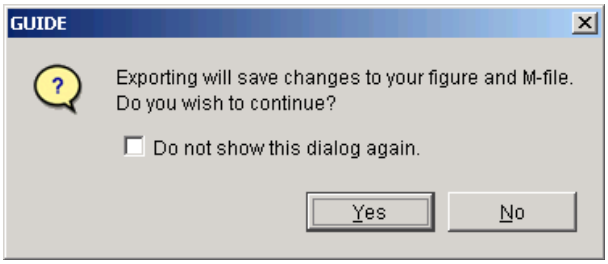


Şekil 5.21

Bu gelen ekranda evet denilerek uygulamanın kaydedişlmesi ve çalıştırılması sağlanılabilir. Eğer ki bu pencerenin her seferinde üzerinde değişiklik yapılan, fakat kaydedilmeyen bir GUI uygulaması olduğunda kullanıcıyı uyarmaması isteniyor ise "Do not show this dialog again." seçeneği işaretlenmelidir.

- **Promp to Save on Export**

Bu seçenek GUIDE çalışma ekranında iken File menüsünden Export komutu verilirse ve tasarlanan GUI uygulaması kaydedilmemiş değişiklikler içeriyor ise kullanıcıya Export işlemi öncesinde var olan değişikliklerin kaydedileceği konusunda Şekil 5.22'deki pencere ile uyarır.

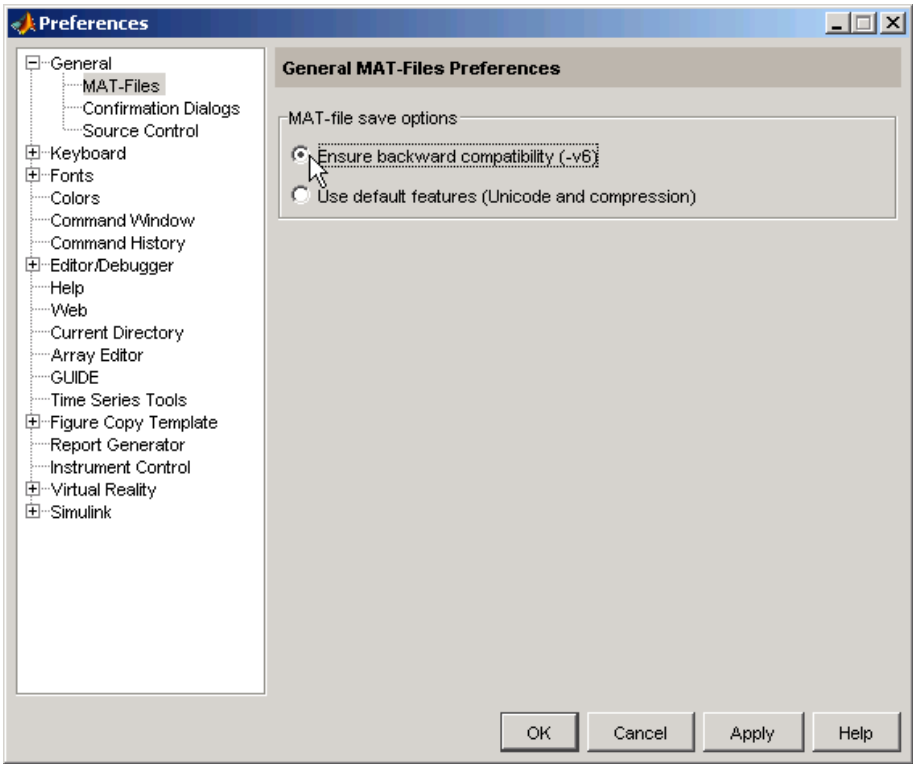


Şekil 5.22

Bu pencerede evet butonuna tıklanılarak işleme devam edilebilir. Eğer ki bu pencerenin sürekli çıkması istenmiyor ise kullanıcı evet demeden önce "Do not show this dialog again." seçeneğini işaretlemelidir.

5.7.10.2 Geriye Uyumluluk Seçeneği

Seçenekler penceresinde General>Mat-Files yolu izlenilerek gelen Şekil 5.23'teki pencereden önceki MATLAB versiyonları ile uyumlu olacak şekilde dosyaların kaydedilme format ayarı değiştirilebilir.



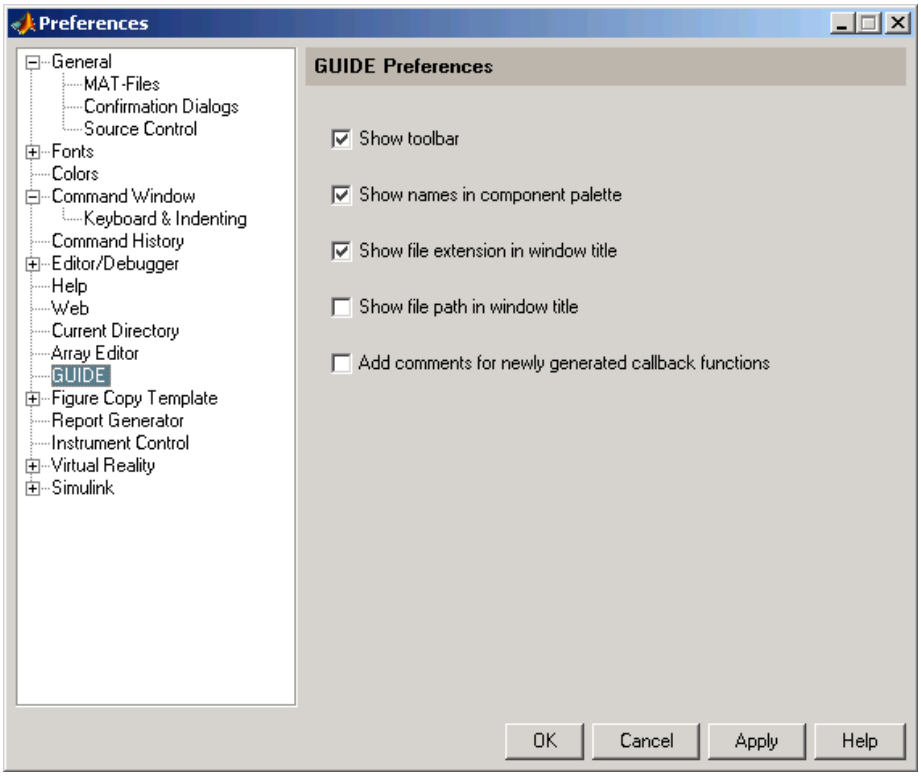
Şekil 5.23

Bu çalışmada kullanılan MATLAB versiyonu 7.0.4'tür. Bu pencere yardımıyla "Ensure backward compatibility (-v6)" seçeneği seçilirse GUIDE kullanılarak hazırlanan GUI

uygulamaları versiyon 6 nolu MATLAB GUIDE uygulamaları ile uyumlu olacaktır. Bunun anlamı hazırlanan her GUI uygulamasının iki adet dosya halinde kaydedilmesi demektir. Bu dosya türleri .m ve .fig uzantılarına sahiptir. Uyumluğu korumak için .m dosyaları GUI tasarımlarının komutlarını içermek üzere kaydedilir. Ayrıca, yine uyumluluk sağlamak için .fig uzantılı dosya formatı kullanılarak hazırlanan dosyada da GUI arayüzünün görsel ayarları ve nesnelerin görünümü ile ilgili bilgileri saklanır.

5.7.10.3 Diğer Tercihler

Preferences ekranında ayrıca GUIDE yolu altında diğer tüm GUIDE seçenekleri yer almaktadır. Karşımıza Şekil 5.24'teki gibi bir pencere gelecektir.

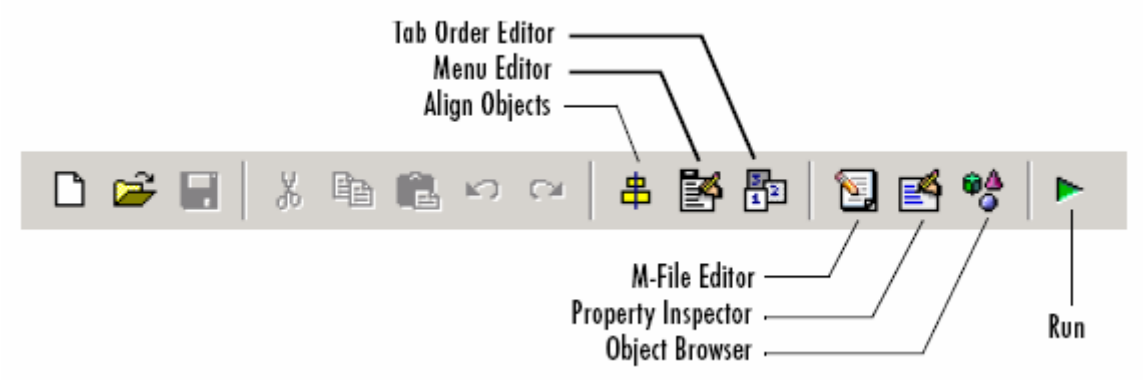


Şekil 5.24

Bu penceredeki seçeneklerin işlevleri şu şekildedir:

5.7.10.3.1 Show Toolbar:

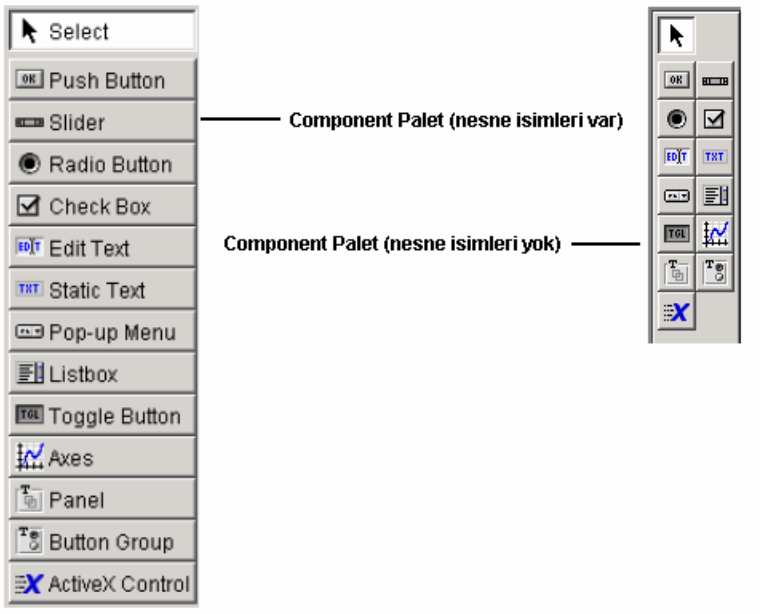
GUIDE ekranında Şekil 5.25'te görülen araç çubuğunu göstermek için bu seçenek seçili işretli olmalıdır.



Şekil 5.25

5.7.10.3.2 Show Names in Component Palette

GUIDE ekranında aşağıda da görüldüğü gibi component paletinde yer alan butonlarda nesnelerin isimlerini göstermek için bu seçenek işaretlenmelidir.



Şekil 5.26

5.7.10.3.3 Show File Extension in Window Title

GUIDE ekranının başlık çubuğunda üzerinde çalışılan GUI uygulamasının dosya isminin yanında .fig uzatısının da gösterilmesi istenirse bu seçenek işaretlenmelidir.

5.7.10.3.4 Show File Path in Window Title

GUIDE ekranının başlık çubuğunda üzerinde çalışılan GUI uygulamasının tüm dosya yolu ile birlikte dosya isminin gösterilmesi istenirse bu seçenek işaretlenmelidir.

5.7.10.3.5 Add Comments for Newly Generated Callback Functions

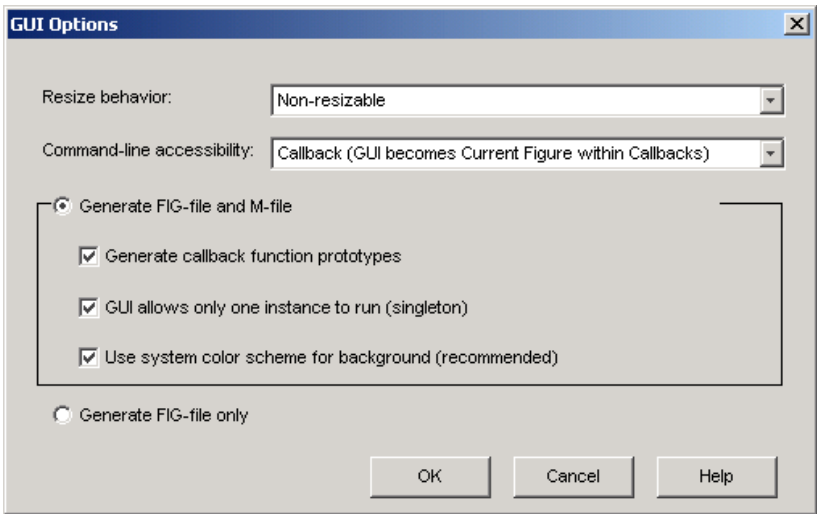
Boş bir GUI uygulaması (untitled bir döküman) ile GUI tasarıma başlandığında .m uzantılı komut satırlarının olduğu dosyaya her bir callback ve bu dosyada yer alan fonksiyonlarla ilgili

otomatik olarak açıklama satırlarının eklenmesi istenirse bu seçenek işaretli olmalıdır. GUI kodlamasında açıklama satırlarının başında % işareti yer alır ve varsayılan olarak M-File Editor'de açıklama satırları yeşil renkte gözükür. Örnek açıklama satırları aşağıda gözükmektedir.

```
% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject handle to figure1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

5.7.11 GUIDE Seçenekleri

Bu seçeneklere GUIDE çalışma penceresinde iken Tools menüsünden GUI Options komutu ile erişilebilir. Karşımıza Şekil 5.27'deki gibi bir pencere gelecektir.



Şekil 5.27

Bu penceredeki seçeneklerin görevleri şu şekildedir.

5.7.11.1 Resize Behavior

Bu seçenek üç farklı durum içerir.

- **Non-resizable**

Kullanıcılar GUI uygulaması penceresinin boyutunu değiştiremezler.

- **Proportional**

Bu seçenek seçildiğinde hem kullanıcı hem tasarımcı GUI yüzeyi ve tüm nesneleri birbiri ile orantılı olarak büyütüp küçültebilirler.

- **Other (Use ResizeFcn)**

Kullanıcılar GUI pencere boyutlarını değiştirseler bile nesnelerin boyutları aynı kalır. İstenirse tasarımcı ResizeFcn callbackini kullanarak pencere boyutlarının değiştirilmesi ile çalışan bu callback i programlayabilir.

5.7.11.2 Command-Line Accessibility

Bu seçenek ile MATLAB komut satırından kullanıcıların GUI figure penceresine erişimi ve kullanmaları engellebilir. Normalde MATLAB gcf komutu ile o an aktif olan figure gösterilebilir. Ancak, bu seçenek sadece GUI figure ekranının callbacklerden erişimine imkân verirse o takdirde aktif gure olarak komut satırından GUI uygulamasının figure ekranına erişilemeyecektir.

5.7.11.3 Generate FIG-File and M-File

Bu seçenek kullanılırsa GUI uygulaması iki dosya halinde kaydedilir ve de bu seçeneğin altındaki görevlere de erişim hakkı kazanmış olur. Alt seçeneklerin işlevleri şu şekildedir.

5.7.11.3.1 Generate Callback Function Prototypes

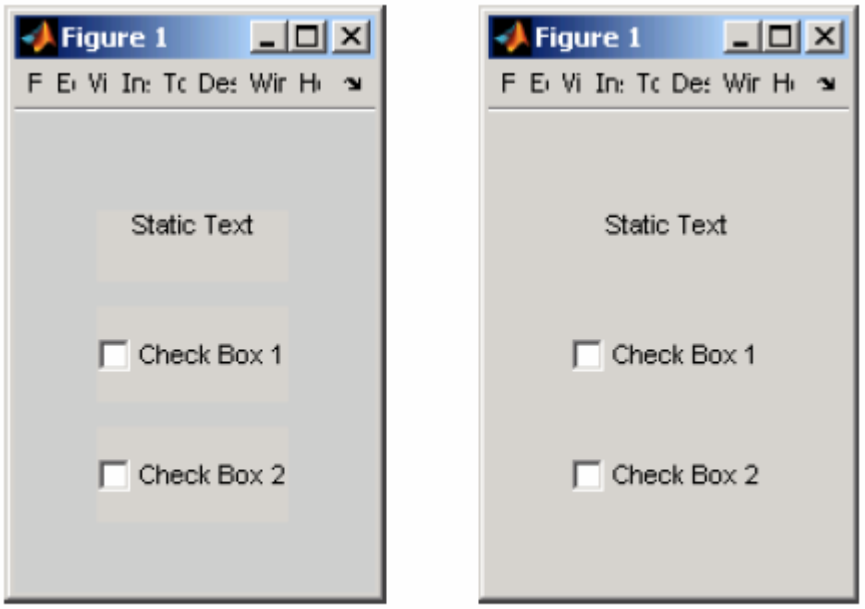
Bu seçenek ile GUI uygulaması ilk oluşturulduğunda (untitled döküman durumu) GUIDE .m uzantılı dosya içeriğine ilk bilinen şablon (template) callbacklere ait fonksiyon tanımlarını içeren komut satırlarını otomatik olarak ekler.

5.7.11.3.2 GUI Allows Only One Instance to Run (Singleton)

Bu seçenek seçilirse GUIDE ekranında tasarlanan GUI uygulaması her run edişinde sadece bir tane GUI penceresi üzerinde çalışacaktır. Ancak, bu seçenek işaretlenmezse her çalıştırmada birden fazla GUI uygulama ekranının çalışmasına izin verilecektir.

5.7.11.3.3 Use System Color Scheme for Background

Bu seçenek seçilirse GUI uygulaması yüzeyi alanının rengi sistemin genel form rengi ile uyumlu olacaktır. Ancak, bu seçenek seçilmezse aşağıda görüldüğü üzere GUI arayüzünde nesne renkleri ile uygulama pencere rengi arasında farklılıklar oluşacaktır.



Şekil 5.28

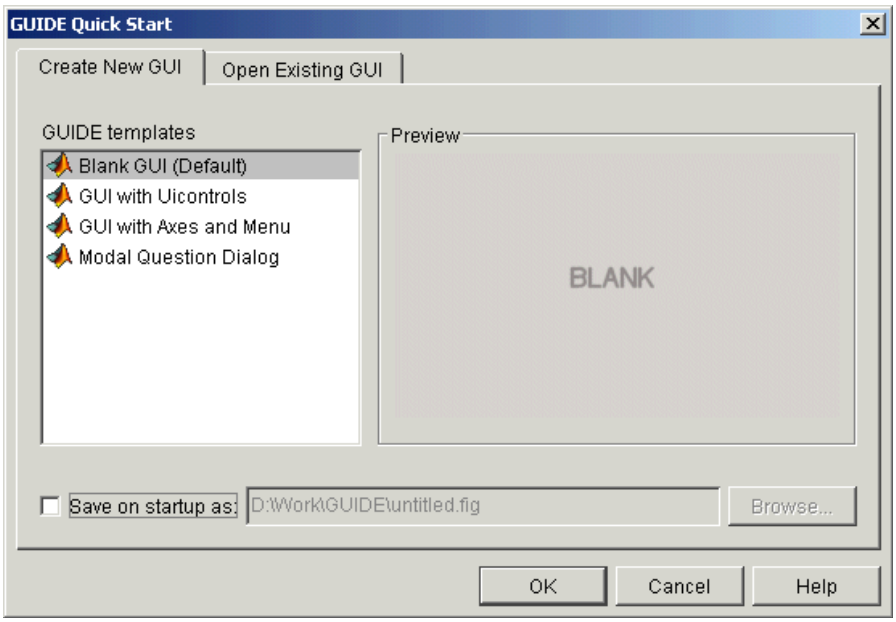
5.7.11.3.4 Generate FIG-File Only

Bu seçenek işaretlenirse GUI tasarımları sadece .fig uzantılı dosya içerisinde hem görünüm ayarlarının, hem de programlama komut satırlarının bulunması sağlanmış olur. Ancak, bu seçeneğin işaretlenmesi ile versiyon 6 GUIDE uygulamaları ile uyumluluk ortadan kalkacak ve de GUI tasarımında programcı işlemlerini yukarıda belirtilen seçenekler olmadan kısıtlı olarak sürdürecektir.

5.7.12 GUIDE Aracında Şablon (Template) Uygulamalar ile Çalışma

MATLAB GUIDE aracı GUI tasarımcılarına ilk başlayanlar için kendi içinde hazır örnek uygulamalar (templates) sunmaktadır. Bu uygulamaların arayüzü ve nesnelerin callbacklerini kullanan komut satırlarını yazma hususunda programcı ve tasarımcıya önbilgi vermesi bakımından çok yararlıdır.

MATLAB komut satırından guide yazıldığında ya daraç çubuğundan GUIDE simgesi tıklandığında karşımıza Şekil 5.29'daki gibi bir pencere gelecektir.



Şekil 5.29

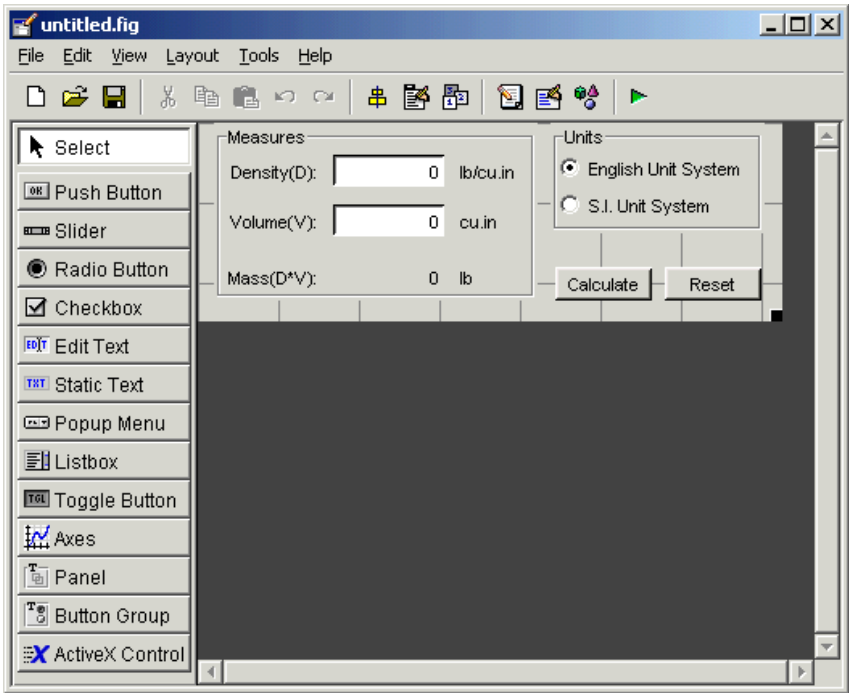
Bu pencereden template olarak karşımıza üç seçenek çıkmaktadır. Bunlar:

- **"GUI with Uicontrols"**
- **"GUI with Axes and Menu"**
- **"Modal Question Dialog"**

Aşağıda bu uygulamalr hakkında ayrıntılı bilgiler verilmiştir.

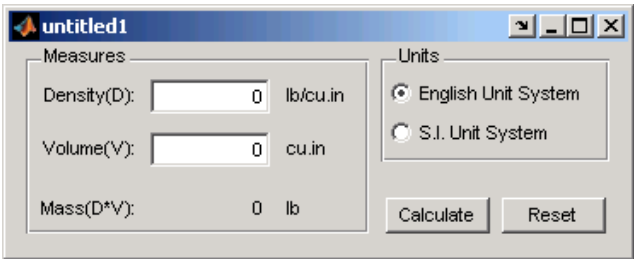
5.7.12.1 "GUI with Uicontrols" Uygulaması

Bu uygulama seçildiğinde Şekil 5.30'da gözüken GUI tasarımı gözükecektir.



Şekil 5.30

Bu uygulamayı araç çubuğundan Run butonuna basarak çalıştırdığımızda da Şekil 5.31'deki gibi bir uygulama arayüzü ekrana gelecektir.

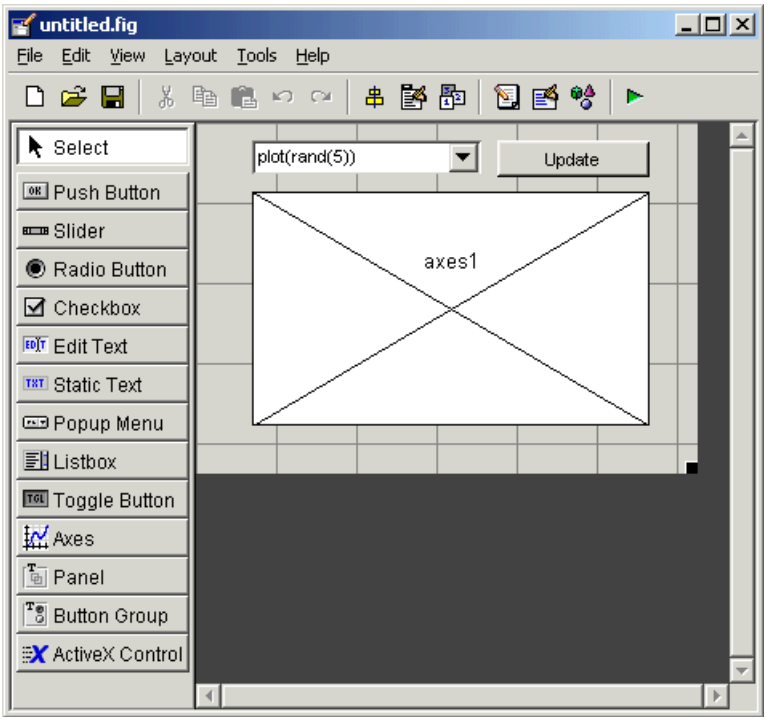


Şekil 5.31

Bu uygulamada amaç seçilen birim sistemine göre yoğunluk ve hacim değerleri girilen bir cismin kütle değerini Yoğunluk*Hacim (D*V) formülünden yola çıkarak hesaplanmasını sağlanmak ve kullanıcıya hesaplanan değeri sunmaktır. Burada push butonların ve text kutuların program kodları ve callback parçalarının kullanımına yönelik olarak programcıya bilgiler verilmektedir.

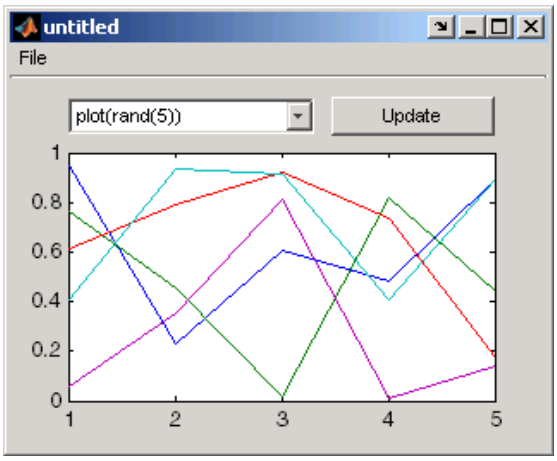
5.7.12.2 "GUI with Axes and Menu" Uygulaması

Bu seçenek seçilerek bir GUI template uygulaması açıldığında karşımıza Şekil 5.32'deki gibi bir tasarım şablonu gelecektir.



Şekil 5.32

Burada da bu GUI uygulamasının çalıştırılması durumunda programcı Şekil 5.33'teki gibi bir arayüz ile karşılaşacaktır.



Şekil 5.33

Bu çalışmada amaçlanan liste kutusundaki seçilen elemanlara göre farklı formülasyonlar kullanılarak update butonuna tıklandığında grafik nesnesi üzerinde bulunan sonuçlara göre çizimin yapılmasını sağlamaktır. Bu uygulamada ile programcıya axes (grafik çizimi) nesnesinin nasıl kullanıldığı ve de liste kutuları ile ilgili callback satırlarının nasıl işletildiği hususunda bilgiler verilmektedir. Ayrıca, bu uygulamada "rand(5)" komutunun kullanımı ve rasgele değerler içeren dizilerin nasıl üretildiği de gösterilmiştir. Bu uygulamanın bir başka faydalı yönü de uygulamanın menü içermesi ve menülerin programlanması konusunda tasarımcıya önbilgi verilmektedir. Menülerin kullanımı ile ilgili olarak

- file = uigetfile('*.fig') komutu ile bir başka .fig dosyanın bir grafik çizim nesnesinde nasıl aktarılacağı,

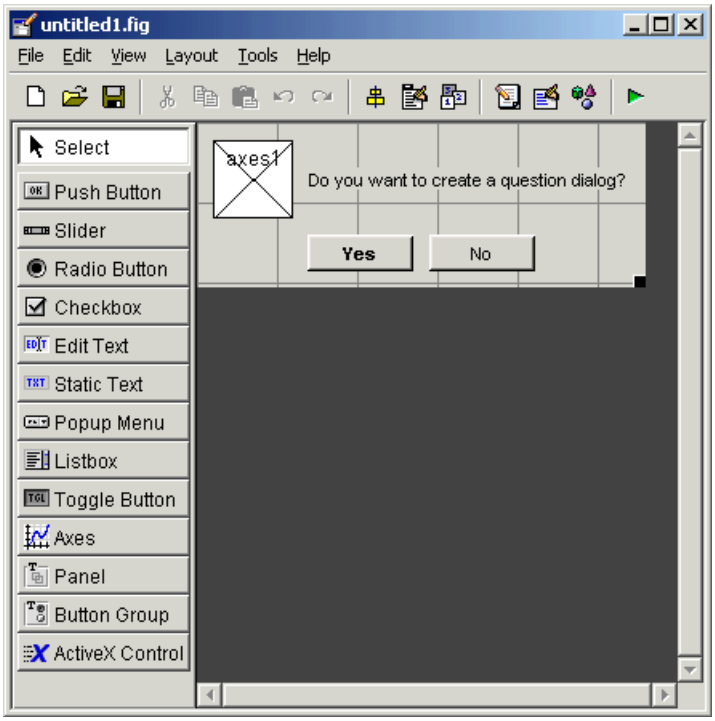
- printdlg(handles.figure1) komutu ile de varolan bir çizimin yazıcıdan nasıl çıktı alınacağı

- close komutu ile aktif bir GUI figure ekranının nasıl kapatılacağı

konularında programcıya programlama teknikleri sunulmaktadır.

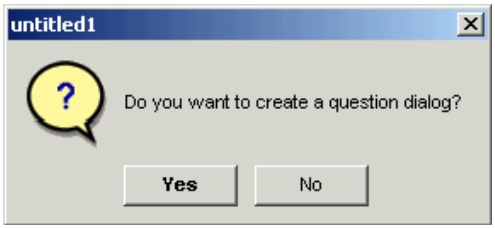
5.7.12.3 "Modal Question Dialog" Uygulaması

Bu seçenek seçildiğinde karşımıza Şekil 5.34'teki gibi bir template GUI uygulaması çıkacaktır.



Şekil 5.34

Bu uygulamayı çalıştırdığımızda da Şekil 5.35'teki gibi bir GUI arayüzü ile karşılaşılır.



Şekil 5.35

Bu GUI uygulamasında amaçlanan kendisine komut satırından verilen dış parametreleri alarak kendi içerisinde yorumlamak ve buna göre oluşturulacak GUI uygulama ekranının içeriğini (burada pencere başlığı ile text kutusunun string değerini) değiştirerek kullanıcıya sunmaktır. Ayrıca, bu uygulama ile programcı şu konularda bilgilendirilmektedir:

- varargin ve varargout komutlarını kullanarak dışarıya bilgi gönderme ve dışarıdan verilen parametreleri yorumlamak

- uiwait(handles.figure1) komutu ile bir uygulama penceresinin aktif konumunun nasıl bloklandığı ve ne iş yaptığı,

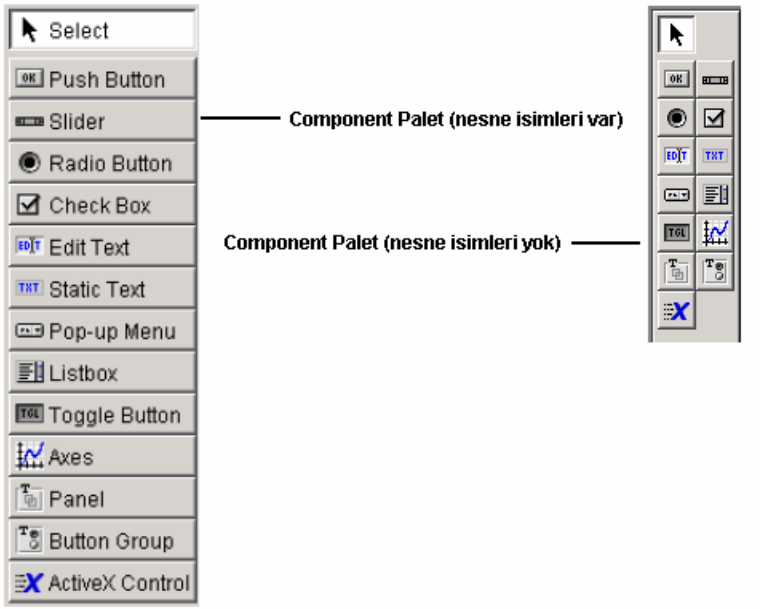
- uiresume(handles.figure1) komutu ile bloklanan bir GUI penceresinin nasıl eski haline getirildiği,

- "modal" programlama tekniği ile tasarlanan bir GUI uygulamasında açılan bir pencerenin arka taraftaki bir başka pencerenin aktif kontrolünü ele geçirmesi için gerekli programlamanın nasıl yapıldığıdır.

Ayrıca, bu uygulama MATLAB GUI tasarımlarında birden fazla form ile nasıl çalışılabileceği konusunda bir ipucu niteliği taşımaktadır. Bu uygulama ile giriş ve çıkış parametreleri kullanan GUI uygulama pencereleri vasıtasıyla kendi içinde birden fazla GUI arayüzü içeren GUI uygulamalarının programlanma tekniği tanıtılmaktadır.

5.7.13 GUI Nesnelerinin Açıklanması

MATLAB GUIDE aracı kullanarak boş (blank) bir GUI çalışma ekranını açtığımızda sol tarafta görülen component panel pek çok nesnenin kullanılabileceği görülmektedir.



Şekil 5.36

Şimdi bu nesnelerin sırasıyla özellikleri ile ilgili bilgiler verilecek ve nasıl programlanacağı gösterilecektir.

5.7.13.1 Push Button:

Normal bir buton özelliği taşımaktadır. Bir buton üzerine tıklanması ile yapılacak komutlar bu buton ile ilgili callback lerin altına yazılır.

5.7.13.2 Toggle Buton:

Çift durumlu bir buton özelliği taşıyan bu nesne ile iki farklı seçenek içeren durumlarda örneğin bu buton basılı ise bir işlemin, bu buton basılmamış ise başka işlemlerin yapılması gerektiği yerlerde tercih edilen bir nesnedir. Buton grubu nesnesi ile beraber kullanımı tavsiye edilir.

5.7.13.3 Radio Buton:

Birden fazla seçeneğin olduğu, ancak seçeneklerden sadece herhangi birinin seçilebileceği hallerde bu nesne kullanılır. Buton grupları ile kullanılması genellikle tercih edilir.

5.7.13.4 Check Box:

Kullanıcıya seçim yapabileceği ve birden fazla şıkkı işaretleyebileceği durumlarda bu nesne kullanılır.

5.7.13.5 Edit Text: 

Bir kullanıcıdan bilgi girişi ya da bir değerin alınması söz konusu olduğunda giriş elemanı olarak sıklıkla kullanılan bir nesnedir.

5.7.13.6 Static Text: 

Kullanıcıya herhangi bir bilgi verme ya da bulunan bir sonuç veya değeri gösterme amacıyla sıklıkla kullanılan bir nesnedir.

5.7.13.7 Slider:

Kullanıcıdan bir giriş değerini kaydırılmak suretiyle kolaylıkla alınmasına imkân veren bir nesnedir.

5.7.13.8 List Box: 

Kullanıcıya bilgi verme amacıyla kullanılabileceği gibi bir değeri listeden seçmek amacıyla da kullanılan sabit bir liste kutusu niteliğinde kullanılan bir nesnedir.

5.7.13.9 Pop-Up Menu: 

Kullanıcıdan alınmak istenilen bilgileri açılan bir listeden seçme özelliği taşıyan bir nesnedir.

5.7.13.10 Axes: 

Yapılan iş ile ilgili grafik çizimlerinin kullanıcıya gösterilmesini sağlayan bir nesnedir.

5.7.13.11 Panel: 

GUI yüzeyi nesnelerinin kullanıcıya daha anlamlı ve güzel gözükmesini sağlayan, ayrıca tasarımcıya GUI dizaynında kolaylık sunan bir nense olup, GUI yüzeyi nesnelerinin gruplanması ve bir arada gösterilmesi amacıyla kullanılır.

5.7.13.12 Button Group: 

Radio veya toggle tipteki buton nesnelerinin bir arada kullanılarak kullanıcının birden fazla seçenekten sadece bir tanesini seçmesini sağlamak amacıyla kullanılan bir nesnedir

5.7.13.13 ActiveX Component: 

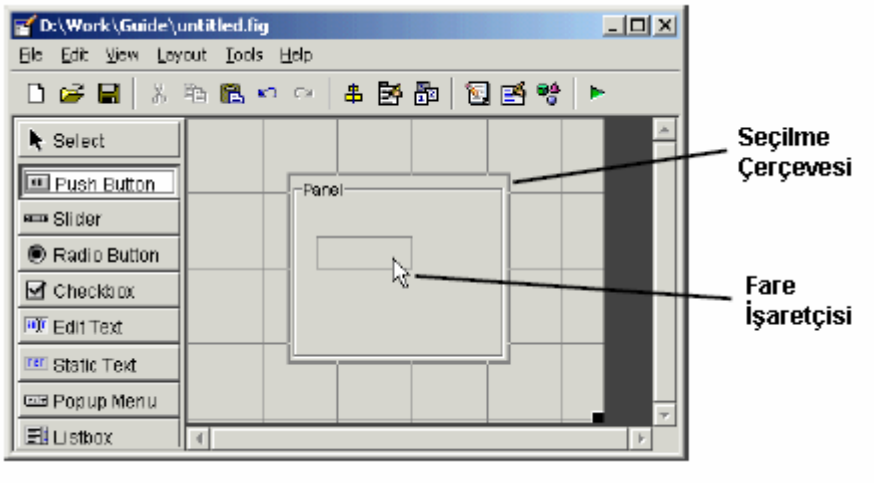
MATLAB GUI tasarımları sadece yukarıda belirtilen nesneler ile sınırlı değildir. Tasarıcı ve programcı ayrıca, ActiveX adı verilen ve değişik alternatifi olan nesnelerin kullanılmasına da imkân verir. Böylece hem tasarımcı hem tasarlanılacak GUI arayüzünün kullanımı bakımından kullanıcıya esneklik sağlanmış olunur.

5.7.14 Nesnelerin GUI Yüzeyine Yerleştirilmesi

Bir nesneyi çalışma alanına eklemek için yapılması gereken sol tarafta yer alan component panelden yerleştirilmek istenilen nesnenin butonunu tıklamak ve GUI yüzeyinde yerleştirilmek istenilen yere tıklamak ya da yerleştirilmek istenilen bölgeyi farenin sol tuşu ile basılı tutarak beliren çerçevenin nesne boyutları olacağını düşünerek yerleştirme işlemi yapılabilir.

5.7.14.1 Bir Nesnenin GUI Yüzeyinde Bulunan Bir Panele ya da Buton Grubuna Yerleştirilmesi

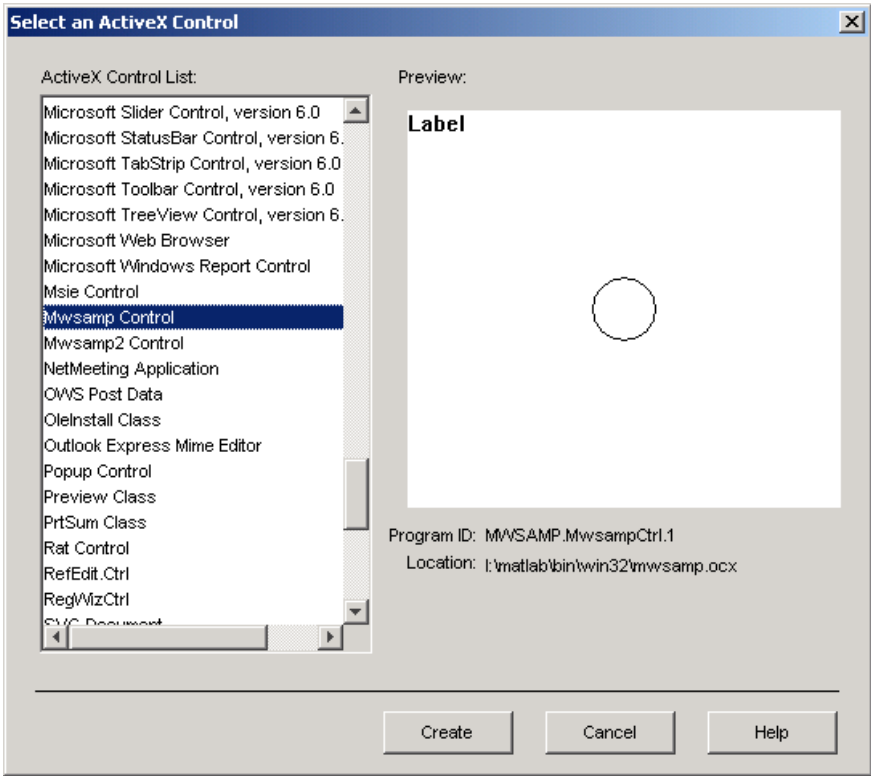
Bir nesne GUI güzeyinde daha önceden yerleştirilmiş olan bir panele ya da buton grubuna yerleştirmek için öncelikle yerleştirilecek nesne component panelden seçilir ve daha sonra fare işaretçisi yerleştirilecek panel ya da buton grubu üzerine götürülür. Bu durumda fare imlecinin üzerinde olduğu bu grup nesnesi bir anda seçili hale gelecektir. Şayet bu işlem ana GUI figure üzerine getirilirse bu sefer grup nesnesinin aktifliği kaybolacak ve figure yüzeyi seçili duruma dönüşecektir. (Bir nesnenin aktif veya seçili olduğu durumu nensnenin kenarında beliren çerçevelerin varlığı ile anlaşılabilir.) Bu durumu GUI yüzeyine yerleştirilmiş olan bir panel nesnesine bir push buton nesnesinin yerleştirilmesi örneği üzerinde Şekil 5.37'de görüldüğü üzere özetlenebilir.



Şekil 5.37

5.7.14.1.1 ActiveX Control Nesnesinin GUI Yüzeyine Yerleştirilmesi

Bir ActiveX nesnesini GUI yüzeyine eklemek için öncelikle component panelden seçilir. Daha sonra GUI alanında yerleştirilmesi düşünülen bir yere farenin sol tuşu ile tıklanır. Bu adımdan sonra karşımıza Şekil 5.38'deki gibi bir pencere gelecektir. Bu pencerede GUI yüzeyine

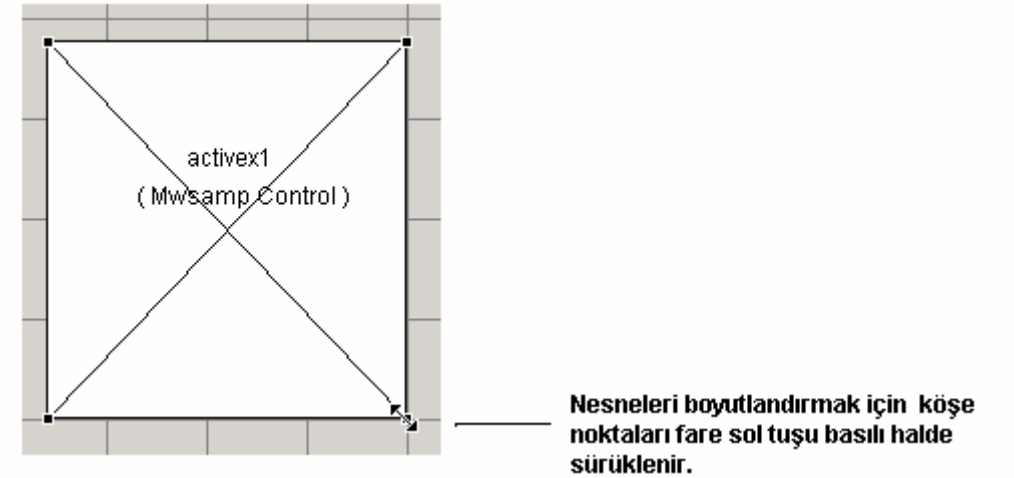


Şekil 5.38

yerleştirilmek istenmek istenilen ActiveX componenti sol taraftaki listeden seçilmeli ve ardından Create butonuna basılmalıdır.

5.7.15 GUI Yüzeyine Eklenen Nesnelerin Boyutlandırılması

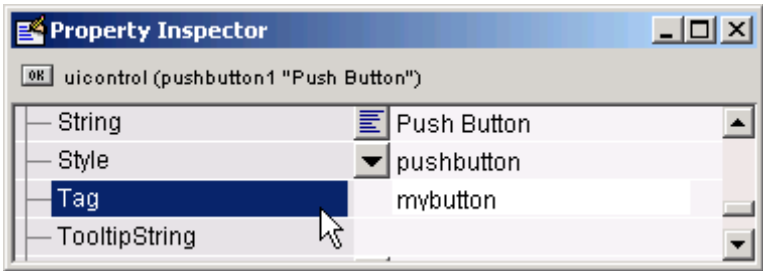
GUI alanına eklenen bir nesnenin boyutlarını değiştirilmek için ilgili nesne öncelikle farenin sol tuşu yardımı ile seçilir. Daha sonra da etrafında beliren köşe noktaları kullanılarak boyutları değiştirilebilir. Bu durum Şekil 5.39'da gösterilmiştir.



Şekil 5.39

5.7.16 Her Nesneye Tanıtıcı Bir İsim Atamak

Bir nesneye tanıtıcı ve o nesneye özel bir isim vermek için öncelikle o nesne farenin sol tuşu ile GUI tasarım yüzeyinde seçilir. Daha sonra View menüsünden Property Inspector komutu verilir. Karşımıza gelen özellikler penceresinden nesnemizin Tag özelliğine istenilen bir isim verilebilir. Ayrıca, Property Inspector penceresini açmak için nesne üzerinde farenin sol tuşu ile çift tıklanılarak da açılabilir. Şekil 5.40'da örnek olarak bir buton nesnesi için bu durum görülmektedir.



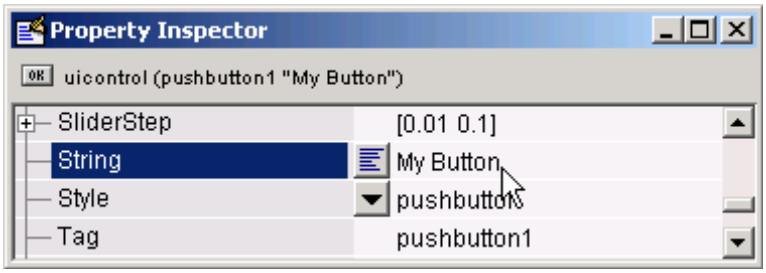
Şekil 5.40

5.7.17 Nesnelere Yazı Eklemek

Bazı nesneler özellikleri gereği kullanıcıya bilgi vermek veya bir seçenek sunmak amacıyla string bilgiler içerirler. Bu nesneler Push Button, Toggle Button, Radio Button, Check Box, Text, List Box, Popup Menu, Panel ve Button Group nesneleridir. Bu nesneler de yapıları gereği değişik özellikler içermektedir ve bu özellikleri üzerinden string değerler atamak mümkün olmaktadır. Bu nesnelere yazı bilgilerinin nasıl verildiği aşağıda sırasıyla açıklanmıştır.

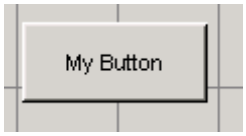
5.7.17.1 Push Button, Toggle Button, Radio Button, Check Box, Text Nesnelerine Yazı Eklenmesi

Bu nesnelere yazı eklemek için ilgili nesne seçilerek özellikler penceresinden String özelliklerine istenilen bir metin bilgisi girilebilir. Ayrıca, bu özellik programlama komut satırlarıyla da değiştirilebilir. Örnek olarak Şekil 5.41'de görülen örnekte bir push butonun üzerindeki yazının değiştirilmesi gözükmektedir. Eğer ki bu nesnelere alt alta olacak şekilde birden fazla satır içeren bilgiler girilmek istenirse bir sonraki başlık altında anlatılan teknik kullanılmalıdır.



Şekil 5.41

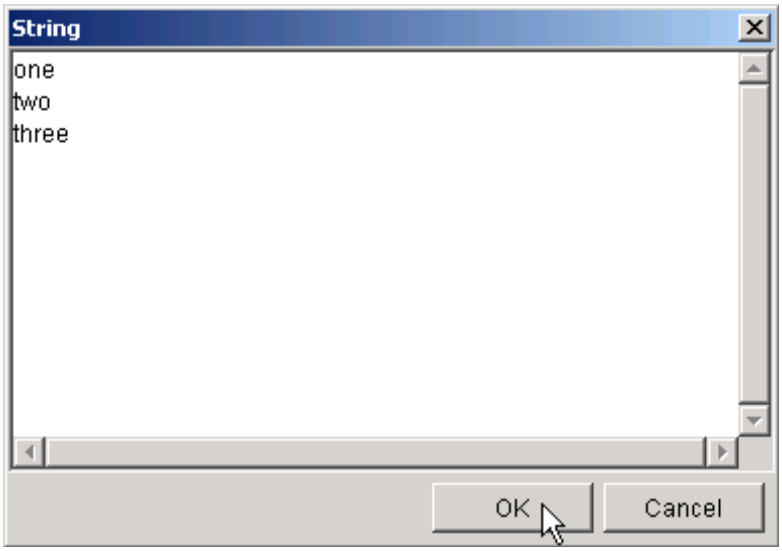
Gerekli değişiklik yapıldığında butonumuzun görüntüsü Şekil 5.42'deki gibi olacaktır.



Şekil 5.42

5.7.17.2 List Box ve Popup Menu Nesnelerine Yazı Eklenmesi

Bu nesneler çoklu string bilgiler içeren liste kutusu tarzı yapılardır. Bu nesnelere string bilgiler eklemek istenirse öncelikle nesne seçilir. Ardından String özelliğinin yanında yer alan butona tıklanır. Karşımıza aşağıdaki gibi bir pencere gelecektir. Bu pencereye gerekli bilgiler girildikten sonra OK butonuna basılır. Böylece string verilerin girilmesi işlemi tamamlanmış olur. İstenirse bu özellik programlama yoluyla da değiştirilebilir. Bu yöntem eğer ki bir önceki başlıkta belirtilen nesnelere birden fazla bilgi girilmesi istendiğinde de bu nesneler için de kullanılabilir.



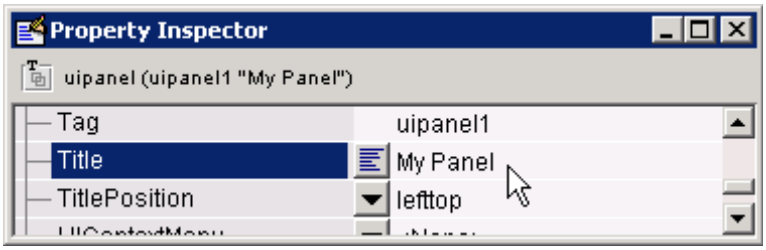
Şekil 5.43

Şekil 5.43'te görülen bilgiler girildiğinde örneğimizdeki popup menü nesnemizin görüntüsü Şekil 5.44'te görüldüğü gibi olacak ve kenarındaki buton tıklandığında da tüm seçenekler kullanıcıya sunulacaktır.

Şekil 5.44

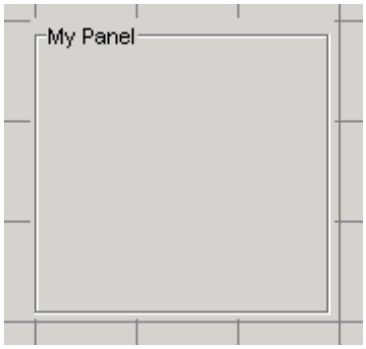
5.7.17.3 Panel ve Buton Group Nesnelerine Başlık Eklenmesi

Bu nesneler içinde yer alan pek çok nesne gruplandırma imkânına sahip olup bu nesnelere başıl eklemek için farklı bir özellik kullanılmaktadır. Bu nesnelere başlık ekmek için nesne seçili iken Property Inspector penceresinden Title özelliğine gerekli bilgi girilmelidir. Bu durumu Şekil 5.45'ten de takip edilebilir.



Şekil 5.45

Örneğimizde bir panel nesnesi kullanılmış olup başlığı My Title olarak değiştirilmiştir. Bu değişiklik yapıldıktan sonra panelimizin GUI çalışma alanındaki görüntüsü Şekil 5.46'daki gibi olacaktır.



Şekil 5.46

5.7.18 GUI Çalışma Alanında Nesneler ile Çalışma

GUI yüzeyindeki nesnelere tasarım ortamında istenildiği şekilde müdahale edilebilir. GUI çalışma alanındaki nesneler üzerinde kopyalama, silme, taşıma, öne getirme, en arkaya gönderme, hizalama, tab tuşu ile seçim sırasının değiştirilmesi, başka bir noktaya taşınması veya boyutlarının değiştirilmesi, cetvel ve ızgara kullanılarak işlemlerin yapılması, GUI uygulamasında ana menü oluşturmak ya da istenilen bir nesne üzerinde context menü oluşturmak, GUI uygulamasına araç çubuğu eklemek, GUI tasarımında kullanılan nesnelerin görülmesi gibi pek çok işlem için GUIDE tasarımcıya pek çok kolaylık sağlamaktadır.

5.7.18.1 Nesnelerin Seçilmesi

GUI yüzeyindeki nesneleri teker teker seçmek için fare işaretçisi ilgili nesne üzerine götürüp farenin sol tuşuna basmak yeterlidir. Ancak, tasarımcı birden fazla nesneyi seçmek istiyor ise o zaman Ctrl ya da Shift tuşu basılı halde iken ilgili nesneleri birer birer tıklamalıdır. Ayrıca, çoklu seçim işlemi için fare işaretçisi GUI yüzeyinin boş bir yerinde farenin sol tuşu basılı tutularak bu halde hareket ettirilir. Bu esnada oluşan pencere içerisinde seçilmesi istenilen nesneler var olduğunda farenin sol tuşu bırakılır. Bu şekilde de topluca seçim işlemi yapılmış olacaktır.

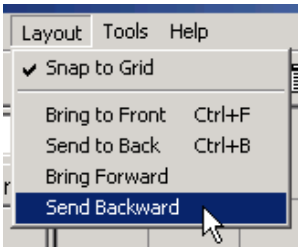
5.7.18.2 Nesneler Üzerinde Kopyalama, Silme, Taşıma, Çoğullama İşlemlerinin Yapılması

GUI yüzeyine eklenene bir nesneden kısa yoldan yeni bir kopya almak istenirse nesne üzerinde farenin sağ tuşu ile tıklanır ve açılan menüden Duplicate (Çoğalt) komutu verilir. Bu şekilde kopya alınan nesnenin tüm görünüm özellikleri aynı yeni bir kopyası oluşturulmuş olacaktır. Ayrıca, bir nesne kopyalama ve yapıştırma tekniği ile de çoğaltılabilir. Bunun için nesne üzerinde farenin sağ tuşu tıklanır açılan menüden de Copy (Kopyala) komutu verilir ya da bu işlem yerine kısaca Ctrl + C tuş kombinasyonu kullanılabilir. Daha sonra GUI yüzeyinin istenilen bir noktasına fare işaretçisi götürülür ve ardından farenin sağ tuşu tıklanır. Açılan menüden Paste (Yapıştır) komutu verilir. Bu işlem kısaca Ctrl + V tuş kombinasyonu ile de yapılabilir.

Bir nesnenin bir noktadan başka bir noktaya taşınması istenirse nesne ya farenin sol tuşu basılı halde istenilen noktaya sürüklenebilir ya da nesne üzerinde farenin sağ tuşu tıklanıp açılan menüden Cut (Kes) komutu verilir. Bu işlem kısaca Ctrl + X tuş kombinasyonu ile de gerçekleştirilebilir. Ardından istenilen noktaya fare işaretçisi götürülür ve sağ tuşu tıklanıp açılan menüden Paste (Yapıştır) komutu verilir.

5.7.18.3 Bir Nesneyi Diğer Nesneler Arasında Öne veya Arkaya Getirme

Bir nesnenin diğer nesnelere göre pozisyonunu değiştirmek için önce nesne seçilir. Daha sonra Layout menüsünden ilgili komut verilir. Bu durum Şekil 5.47'de de gözükmektedir. Buradaki komutların işlevleri şöyledir:



Şekil 5.47

- Bring to Front komutu seçili nesneyi en öne getirir.
- Send to Back komutu seçili nesneyi en arkaya gönderir.
- Bring Forward komutu seçili nesneyi bir adım öne getirir.
- Send Backward komutu seçili nesneyi bir adım arkaya gönderir.

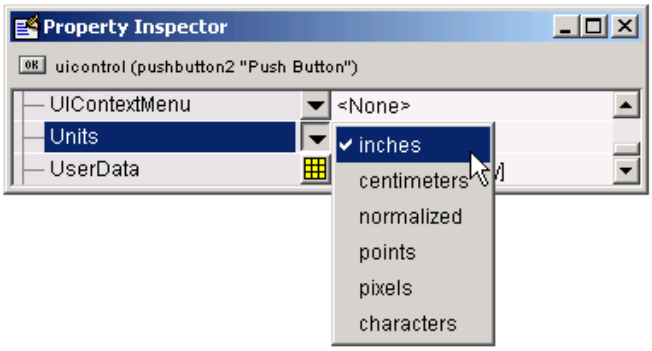
5.7.18.4 Nesnelerin GUI Çalışma Alanında Taşınması

Bir nesneyi GUI yüzeyinde farklı bir noktaya götürmek için üç farklı yöntem vardır. Bunları açıklayalım.

İlk yönteme göre bir nesne fare işaretçisi üzerinde iken farenin sol tuşu ile tıklanır ve bırakılmadan farklı bir noktaya sürüklenir. Bu şekilde nesnenin konumu kolaylıkla değiştirilmiş olacaktır.

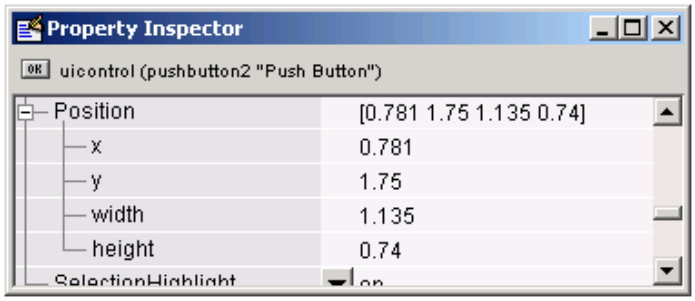
İkinci yönteme göre bir nesne önce farenin sol tuşu ile seçilir. Ardından klavyedeki yer-yön tuşları vasıtasıyla istenilen yönlerde hareket ettirilebilir. Bu yöntem özellikle bir nesne daha hassas olarak başka bir konuma yerleştirilmek istendiğinde kullanılır.

Son olarak da bir nesnenin konumu nesne seçili iken Property Inspector penceresinden Posiiton özellikleri değiştirilir. Bu şekilde de bir nesnenin konumu değiştirilmiş olacaktır. Ancak, standart ölçü birimi olarak GUIDE normalized veya characters ölçü biirmlerini kullanmaktadır. Bunlar yerine öncelikle ölçü birimini inches olarak değiştirilirse bir nesnenin konumlandırmasında tasarımcıya çok büyük kolaylık sağlayacaktır. Bu durum Şekil 5.48'de görülmektedir.



Şekil 5.48

Bu adımdan sonra da Position özelliği kullanılarak ilgili nesnenin konumu değiştirilebilir. Bu özellik hiyerarşik yapılı değişken özelliği göstermekte olup alt parametrelerinin değiştirilmesi için özellikler penceresinde Position özelliğinin yanındaki + işareti tıklanır. Böylece alt parametreler görülebilir. Örnek pencere görüntüsü Şekil 5.49'da verilmiştir. Buradaki bilgiler şu özelliktedirler:



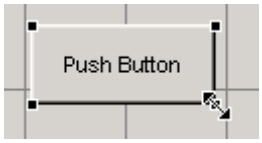
Şekil 5.49

- x bilgisi, bir nesnenin sol kenardan itibaren uzaklığnı gösterir.
- y bilgisi, bir nesnenin alt kenardan itibaren uzaklığını gösterir.
- width bilgisi, bir nesnenin yatay uzunluğunu gösterir.
- height bilgisi, bir nesnenin dikey uzunluğunu gösterir.

Bu yöntemin kullanılması x ve y parametreleri dışında width ve height özellikleri kullanılarak bir nesnenin boyutlarının da değiştirilmesi bakımından tasarımcıya yarar sağlar.

5.7.18.5 Nesnelerin Boyutunu Değiştirmek

Bir nesnenin boyutlarını değiştirmek için nesne önce farenin sol tuşu ile seçilir. Daha sonra nesnenin her bir köşesinde beliren siyah boyutlandırma kutucuklarından istenilen biri üzerine fare işaretçisi götürülür ve üzerinde iken farenin sol tuşu basılı tutulur. Daha sonra bu tuş basılı halde iken işaretçi ileri ve geri hareket ettirilir. Bu şekilde ilgili nesnenin konumu değiştirilmiş olacaktır. Bu durum Şekil 5.50'de de görülmektedir.



Şekil 5.50

Ayrıca, bir nesnenin konumunu değiştirmek için özellikler penceresi de kullanılabilir. Bir nesnenin width ve height özelliklerinin değiştirmek suretiyle nesnemizin boyutları değiştirilmiş olacaktır. Bu durum için bir önceki başlık altında bulunan bilgilere bakılabilir.

5.7.18.6 Nesnelerin Hizalanması

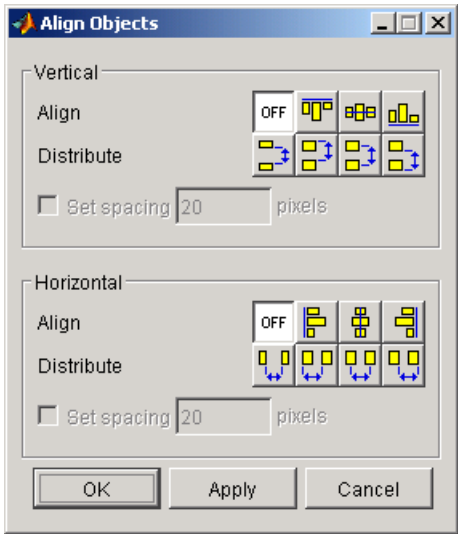
GUIDE bu konuda sunduğu çeşitli ve yararlı araçlar sayesinde tasarımcının işini kolaylaştırmaktadır. Bu araçlar şunlardır:

- Alignment Tool (Hizalama Aracı)
- Property Inspector (Özellikler Penceresi)
- Grid and Rulers (Izgara ve Cetvel)
- Guide Lines (Klavuz Çizgileri)

Aşağıda bu araçlar sırayla açıklanmıştır.

5.7.18.6.1 Alignment Tool (Hizalama Aracı)

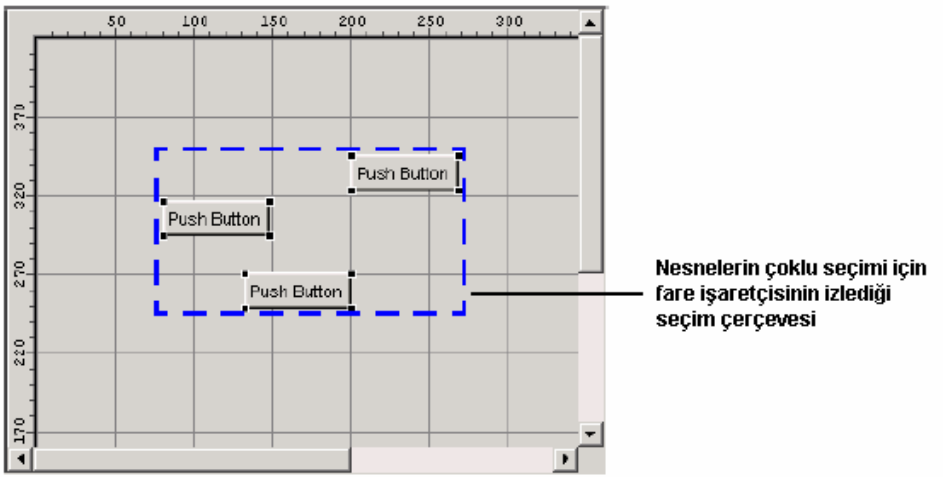
Birden çok nesneyi yatay ve dikey olarak belirlenen referans hattına göre hizlama için bu araç kullanılır. Hizalanacak olan nesneler önce çoklu olarak seçilirler. Ardından Tools menüsünden Align Objects komutu verilir. Karşımıza Şekil 5.51'deki gibi bir pencere gelir.



Şekil 5.51

Bu pencerede Vertical (Dikey) ve Horizontal (Yatay) olmak üzere nesneler hizalanabilir. Eğer ki sadece dikey hizalama yapmak isteniyorsa yatay hizalama seçeneği Off (Kapalı) durumda tutulmalıdır. Ancak isteğe göre hem dikey, hem yatay hizalama aynı anda da yapılabilir. Align seçeneği nesneleri hizalamak için kullanılırken Distribute nesneler arasında referans noktasına göre boşluk bırakmak için kullanılır. Eğer ki Distribute seçeneklerinden herhangi biri seçilirse "set spacing" seçeneği etkin olacaktır ve bu seçenek işaretlenirse yanında belirtilen kutu içerisindeki piksel sayısı kadar nesneler arasında referans noktasına göre boşluk bırakacaktır. Bu durum Şekil 5.52'de örneklendirilmiştir.

Öncelikle hizalandırılacak nesneler toplu olarak seçilir.

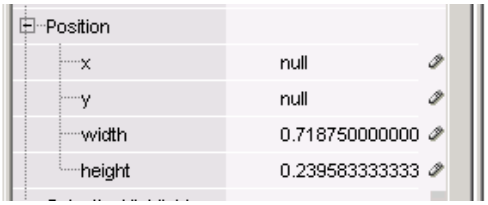


Şekil 5.52

Daha sonra Tools menüsünden hizalama aracı açılır ve gerekli seçimler yapıldığı zaman OK butonuna tıklanır. Böylece nesneler hizalanmış olacaktır. Hizalama isteğiniz gibi olmadı ise herhangi bir anda Ctrl+Z tuş kombinasyonu ile yapılan işlemleri geri alabilirsiniz.

5.7.18.7 Property Inspector (Özellikler Penceresi):

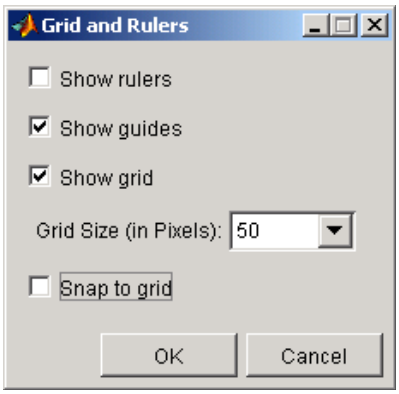
Özellikler penceresi yardımı ile de hizalama yapılabilir. Bunun için önce hizalanacak nesneler toplu olarak seçilir. Daha sonra da View menüsünden Property Inspector penceresi açılır. Gelen pencerede Position özelliği altındaki x ya da y değeri yeni bir değere set edilir. Örneğin, eğer ki nesneler sol kenardan aynı uzaklıkta olacaksa yani dikey olarak hizalanmak isteniyorsa x özelliği aynı (örneğin 2 inches gibi) değere atanır. Benzer şekilde nesneler yatay olarak hizalanmak isteniyorsa y değeri aynı değere ayarlanır. Bu şekilde de nesnelerimiz hizalandırılmış olacaktır. Bu özellikler Şekil 5.53'te de gösterilmiştir.



Şekil 5.53

5.7.18.8 Grid and Rulers (Izgara ve Cetvel)

GUI tasarımında GUIDE aracının tasarımcının nesnelerin hizalanmasında ızgara ve cetvel kullanmasına izin vermesi de tasarımcı için büyük bir kolaylık sağlamaktadır. Bir GUI tasarımında ızgara ve cetvel kullanmak için Tools menüsünden Grid and Rulers komutu verilir. Karşımıza Şekil 5.54'teki gibi bir pencere gelecektir. Buradaki seçenekler şöyledir:



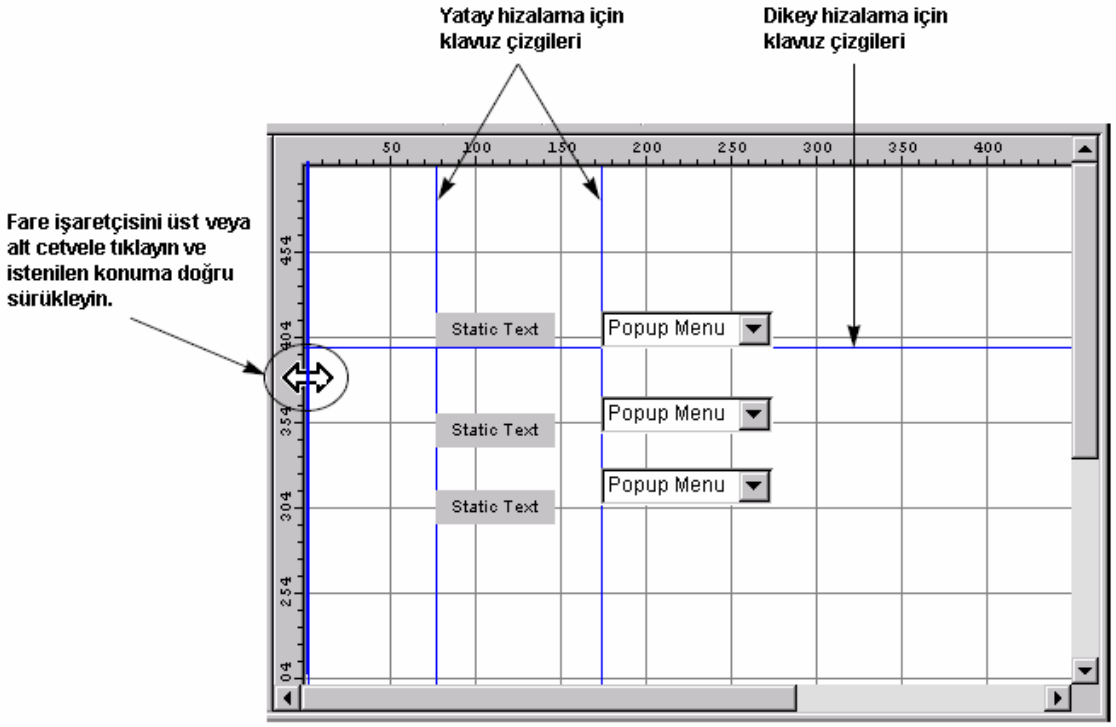
Şekil 5.54

"Show rulers" seçeneği işaretlenirse cetvelleri gösterir.
"Show guides" seçeneği işaretlenirse klavuz çizgileri gözükecektir.
"Show grid" seçeneği işaretlenirse GUI yüzeyinde ızagara belirecektir.
"Snap to grid" seçeneği işaretlenirse nesneler GUI yüzeyinde ızgara hatlarına tutturulacaktır.

İstenirse bu pencerede ızgara aralarının büyüklüğü değişitirlebilir. Bu işlem için Grid Size seçeneği kullanılabilir.

5.7.18.9 Guide Lines (Klavuz Çizgileri)

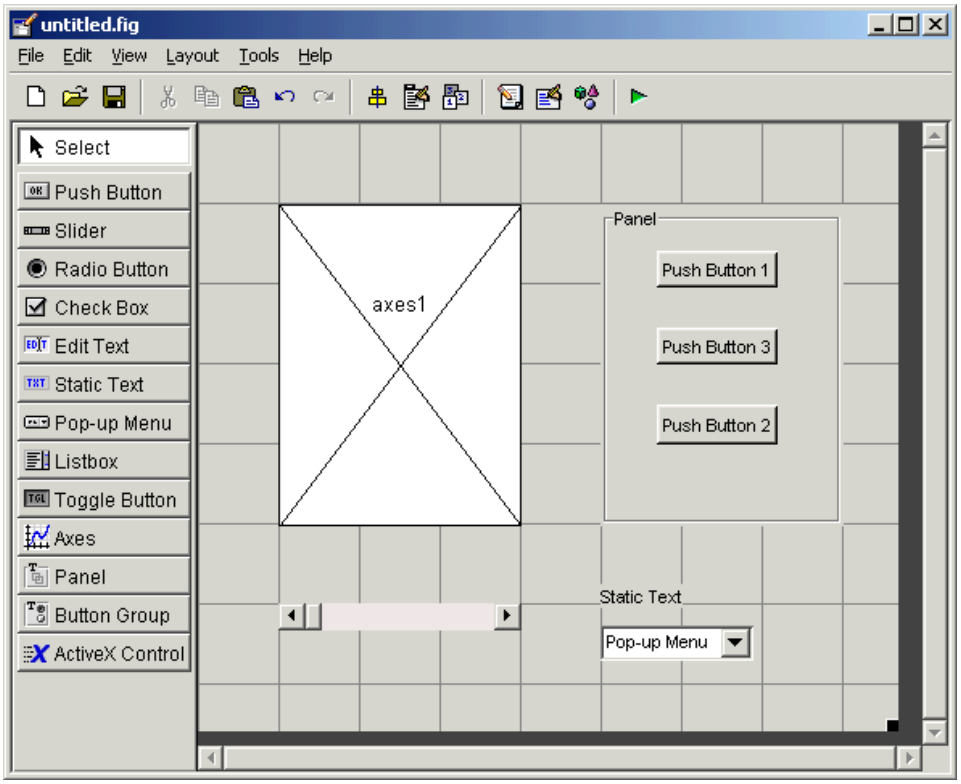
GUI yüzeyi tasarlanırken klavuz çizgileri kullanılarak nesnelerin hizalanması rahatlıkla yapılabilir. Klavuz çizgilerini çıkarmak için tasarımcı önce dikey veya yatay istediği bir cetvel üzerinde ve istediği hizadan başlayarak farenin sol tuşunu basılı tutulur. Bu durumda fare işaretçisi hareket ettirilir. İstenilen hizaya gelindiğinde farenin sol tuşu bırakılır. Böylece GUI çalışma alanında mavi renkte klavuz çizgilerini çıktığı görülecektir. Bu durum Şekil 5.55'de gösterilmiştir.



Şekil 5.55

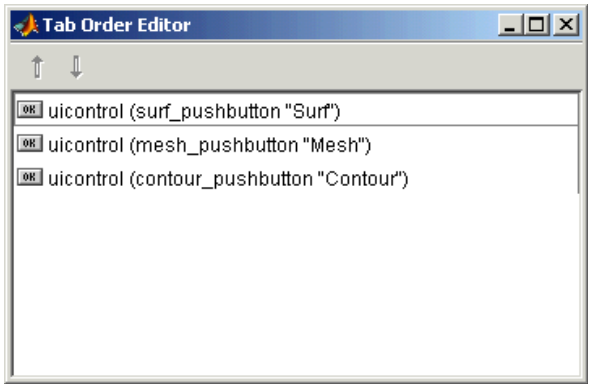
5.7.18.10 Tab Tuşu ile Geçiş Sırasının Ayarlanması

GUI yüzeyinde birden fazla buton veya liste kutuları gibi nesnelerin olduğunu düşünelim. Kullanıcı GUI uygulmasını rahat bir şekilde kullanabilmek için tab tuşu yardımı ile bir nesneden diğerine geçmek isteyebilir. Bu durum için tab sırasının uygun bir sırada değiştirilmesi GUI uygulamamızın kullanım kolaylığını artıracaktır.



Şekil 5.56

Şekil 5.56'da görüldüğü gibi bir GUI arayüzüne sahip olduğu düşünülürse bu arayüzde nesnelerin tab sırasını değiştirmek için öncelikle Tools menüsünden Tab Order editor komutu çalıştırılır.



Şekil 5.57

Daha sonra Şekil 5.57'deki gibi bir pencere ile karşılaşılır. Bu pencereden üst bölümünde yer alan yukarı ve aşağı ok butonları yardımıyla listede görülen nesneler seçilerek tab sıraları değiştirilebilir. Listenin en üstündeki nesneler daha öncelikli tab tuşu geçiş sırasına sahiptirler.

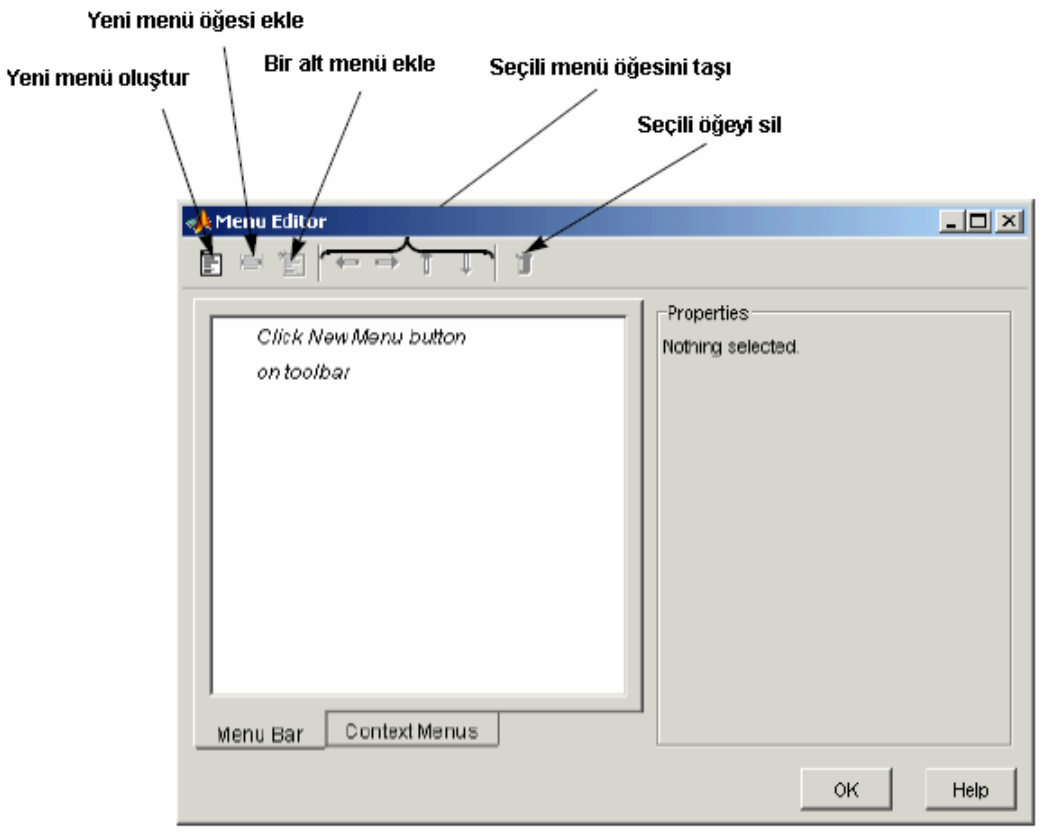
Burada dikkat edilmesi gereken bir husus da Panel, Buton Group ve Axes nesnelerinin tab tuşu sırasında bir etkisi olmaz. Çünkü bu nesneler tab tuşu ile kontrol edilebilecek bir yapıya sahip değillerdir. Ancak, bir panel ya da buton grubu içerisindeki bir nensnenin tab sırası sadece ait olduğu o grup içinde geçerlidir. Bunun için de bu grup nesnelerinden herhangi biri tıklanırsa Tab Order Editor içerisinde o grubun nesneleri gözükecektir.

5.7.18.11 GUI Uygulamasına Menü Eklemek

GUI uygulamarında iki farklı menü eklenebilir. Bunlar dan biri uygulamanın ana ekranında yer alan ve çoğu programlarda yer alan program menüsüdür. Bir diğer menü şekli ise herhangi bir nesne üzerinde farenin sağ tuşu ile tıkladığımızda açılan içerik (context) menüleridir. Aşağıda her iki menü uygulamalarının nasıl yapıldığı ile ilgili ayrıntılı bilgi verilmektedir.

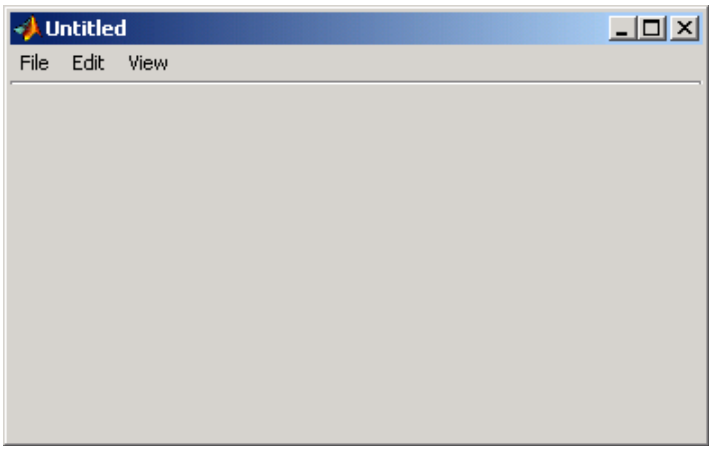
5.7.18.11.1 GUI Uygulamasına Program Menüsü Eklemek

GUI uygulamasına menü eklemek için GUIDE içinde bulunan Menu Editor aracı kullanılır. Bu aracı çalıştırmak için Tools menüsünden Menu Editor komutu verilir. Karşımıza Şekil 5.58'deki gibi bir pencere gelecektir.



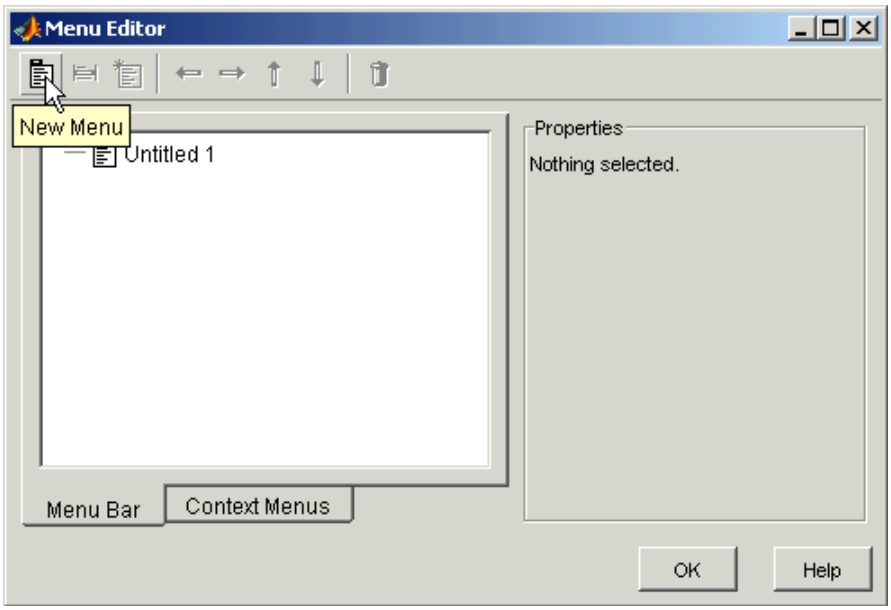
Şekil 5.58

Menu Editor penceresi ile menü ekleme, menüyü yeni bir öğe eklme, varolan bir öğeyi silme, alt menü oluşturma ve pek çok işlem yapılabilir. Örneğin Şekil 5.59'daki gibi bir GUI arayüzü tasarlayacağımızı düşünelim.



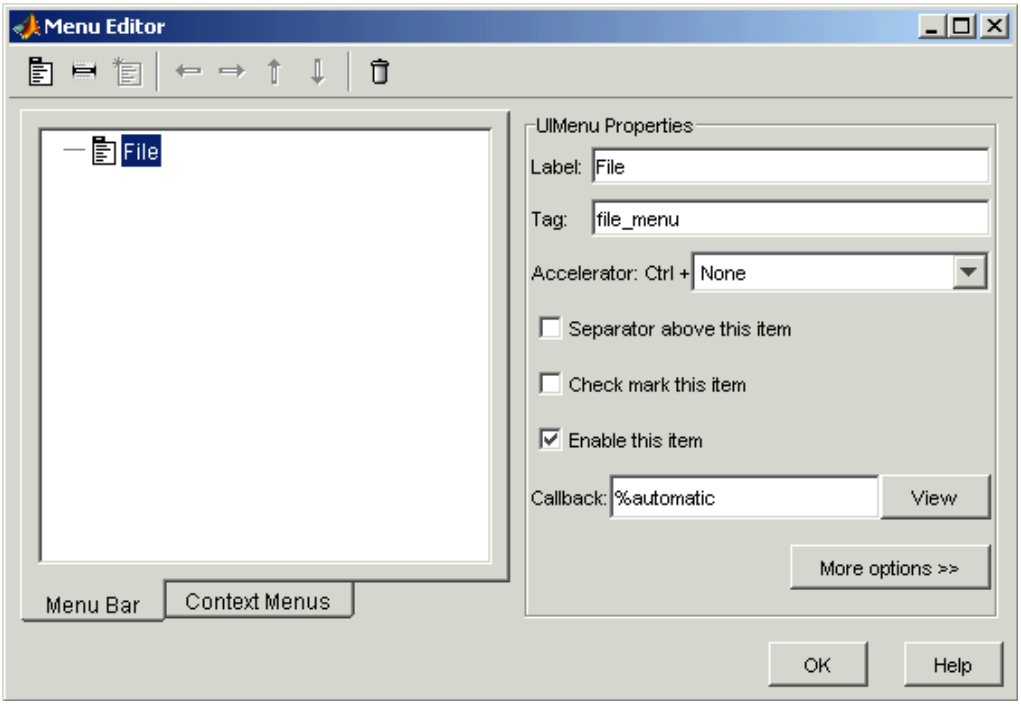
Şekil 5.59

Böyle bir tasarım için öncelikle New Menu tuşu ile yeni bir menü eklenir. Bu durum Şekil 5.60'da gösterilmiştir.



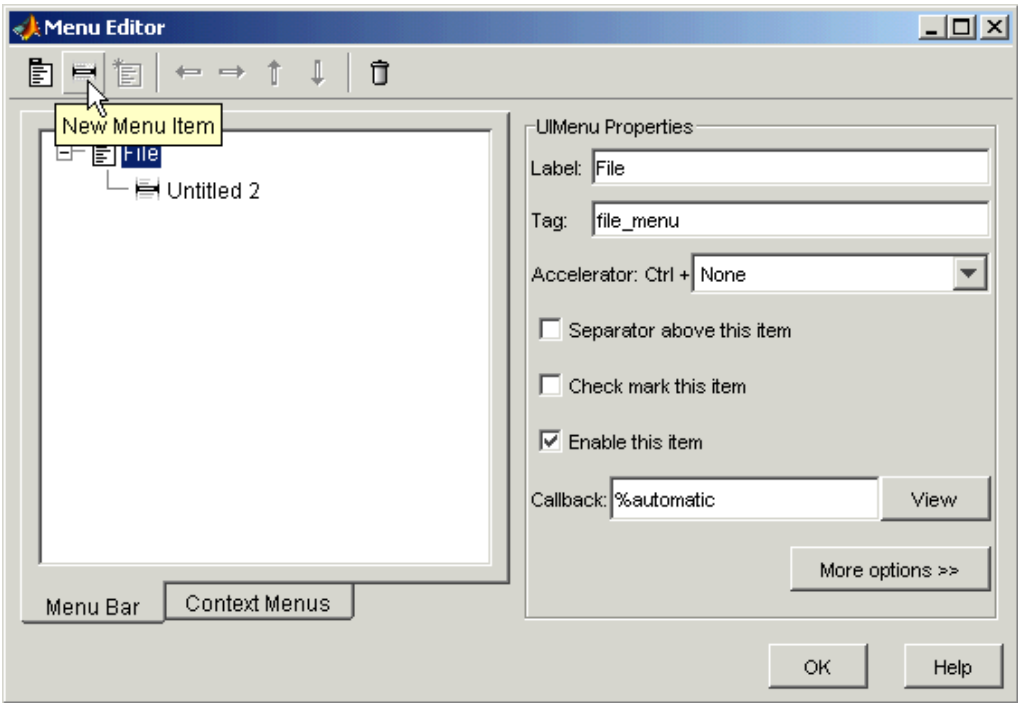
Şekil 5.60

Daha sonra bu menü listeden seçilip Properties (özellikler) panelinden bu menüye "File" etiketi verilir. Bu bilgi Label özelliğine verilir. Tag özelliğine de GUI'yi programlarken kullanılacak tanıtıcı bir isim verilmelidir. Ancak, Tag bilgisinin Label değeri ile aynı olma zorunluluğu yoktur.



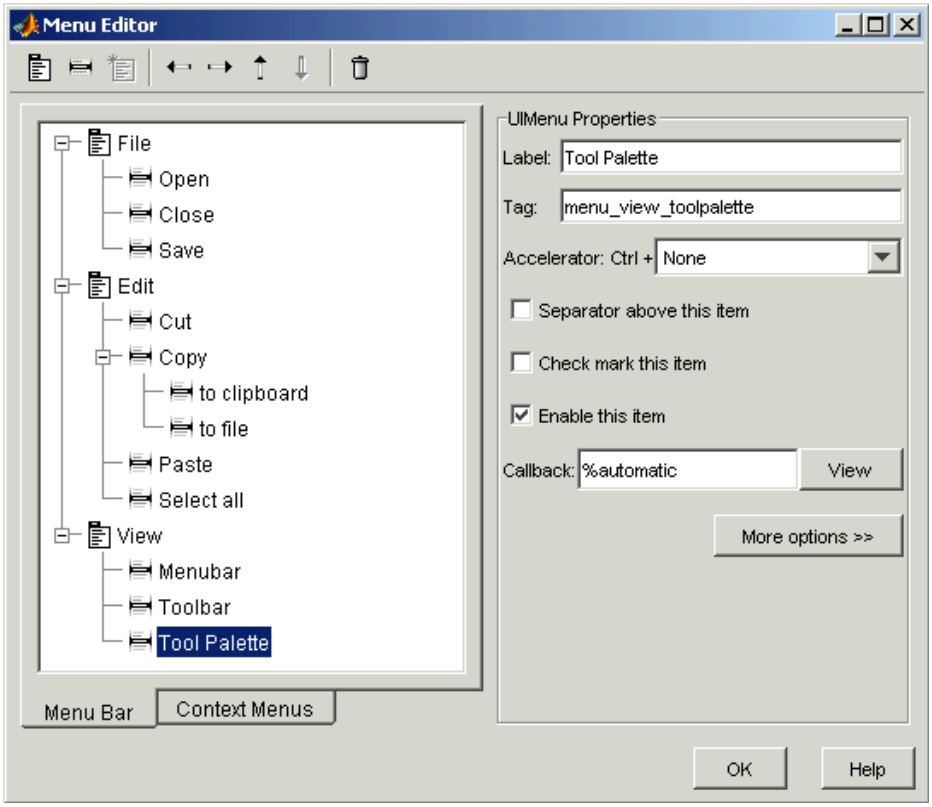
Şekil 5.61

Hazırlanan File menüsüne alt bir öğe eklemek için New Menu Item butonu tıklanır. Böylece yeni öğe File menüsüne eklenmiş olacaktır.



Şekil 5.62

Benzer şekilde bu öğeye de yeni bir etiket (label) verilir. Tüm bu işlemler tamamlandığı zaman Menu Editor penceremiz aşağıdaki gibi gözükecektir.

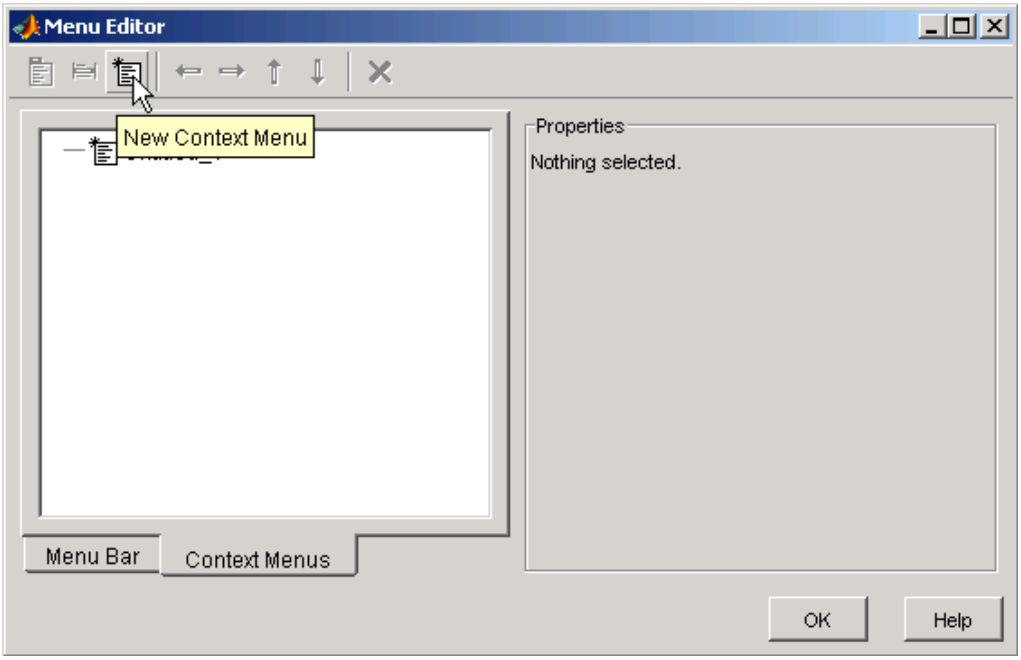


Şekil 5.63

İstenilirse bir menü öğesine Accelerator özelliği kullanılarak bir kısayol tuş kombinasyonu atanabilir. Buradaki herhangi bir menü öğesine tıklandığında çalıştırlacak callback'e gitmek için View butonuna tıklanılabilir. Gelen callback altında girilen komut satırları GUI uygulamasının çalışması durumunda o menü öğesi tıklandığında MATLAB tarafından koşturulacaktır.

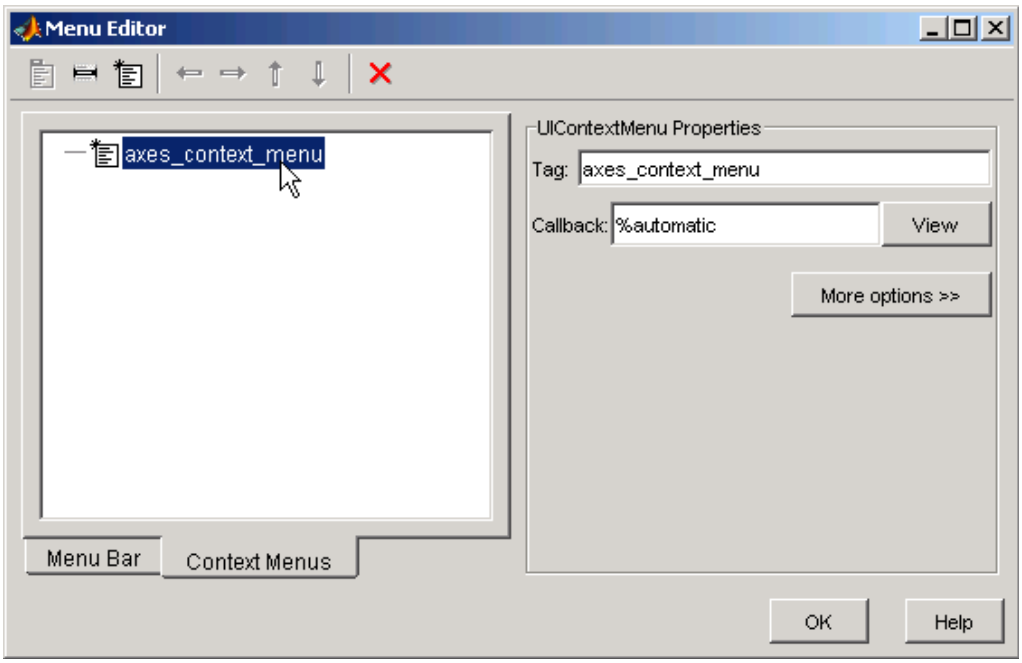
5.7.18.11.2 GUI Uygulamasına Context (İçerik) menüleri Eklemek

Bir GUI uygulamasında kullanıcıya sağlanılacak kolaylıklardan bir de bir nesne üzerinde farenin sağ tuşu ile tıklandığında gözüken context (içerik) menüleridir. Context menü oluşturma işlemleri bir önceki konu başlığı altında incelenen menülü ugyulamalar tasarlama adımlarına çok benzerlik gösterir.



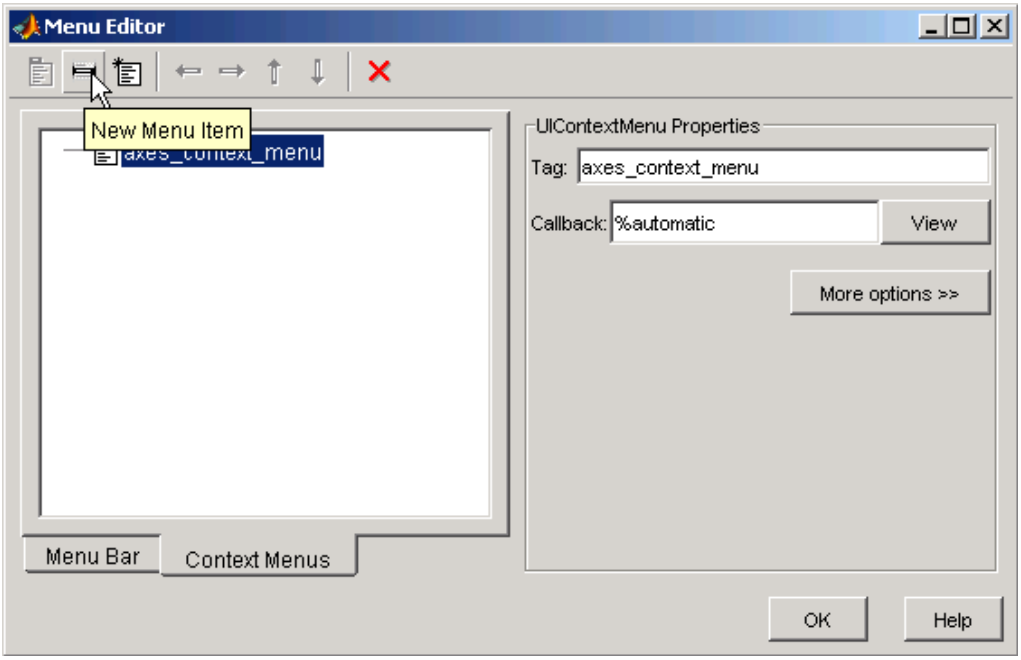
Şekil 5.64

Bir GUI uygulamasına içerik menüsü eklemek için öncelikle yapılması gereken Tools menüsünden Menu Editor komutu vermek ve karşımıza gelen ekrandan Context Menus sekmesine geçmektir. Daha sonra bu pencerenin üst tarafında bulunan New Context Menu butonunu kullanarak yeni bir içerik menüsü oluşturulur. Ardından eklenen menü listeden seçilen sağ tarafta yer alan özellikler kısmından yeni bir Tag ismi verilir. Bu isim kullanıcıya gözükmeyip sadece programlama amacıyla kullanılır.



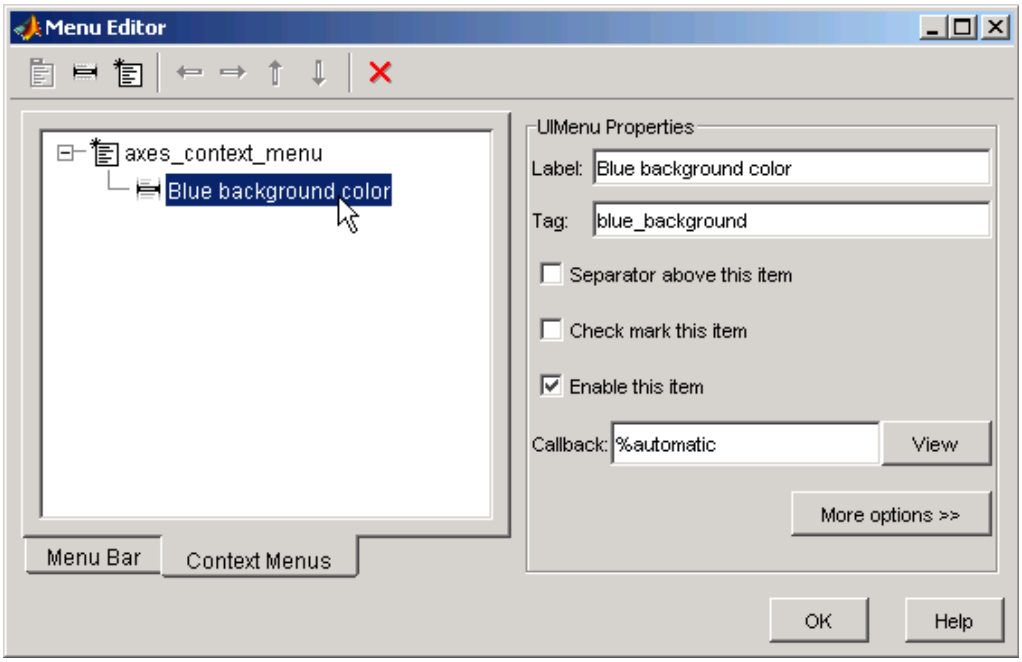
Şekil 5.65

Daha sonra bu içerik menüsüne yeni öğeler eklemek için New Menu Item butonuna basılır.



Şekil 5.66

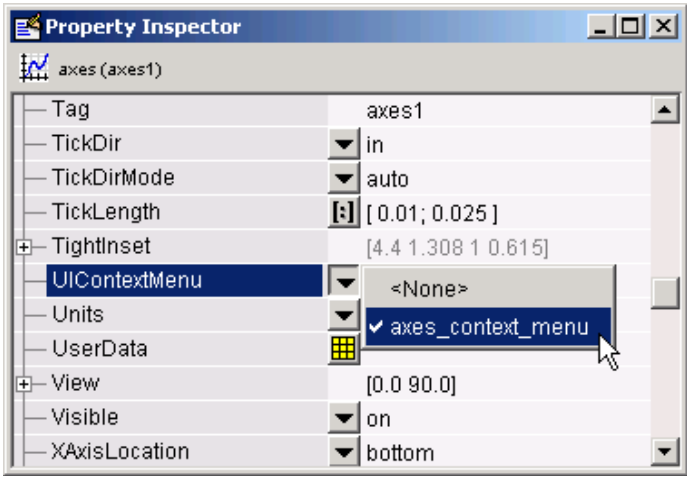
Eklenen bu menü öğesi önce listeden seçilir ve daha sonra da sağ tarafta bulunan özellikler panelinden hem Label hem de Tag bilgileri girilir.



Şekil 5.67

5.7.18.11.2.1 Context (İçerik) menüsünün Bir Nesne ile İlişkilendirilmesi

Yukarıda oluşturulan axes_context_menu isimli içerik menüsünün GUI uygulamasının çalıştırılması esnasında üzerinde farenin sol tuşu ile tıklandığında çıkacak nesne ile ilişkilendirilmesi gerekir. Bunun için ilgili nesne GUI tasarım alanında iken seçilir ve Property Inspector penceresinden UIContextMenu özelliği bu uygulamada oluşturulmuş olan içerik menüsü ile değiştirilir. Bu durum Şekil 5.68'de de görülmektedir.



Şekil 5.68

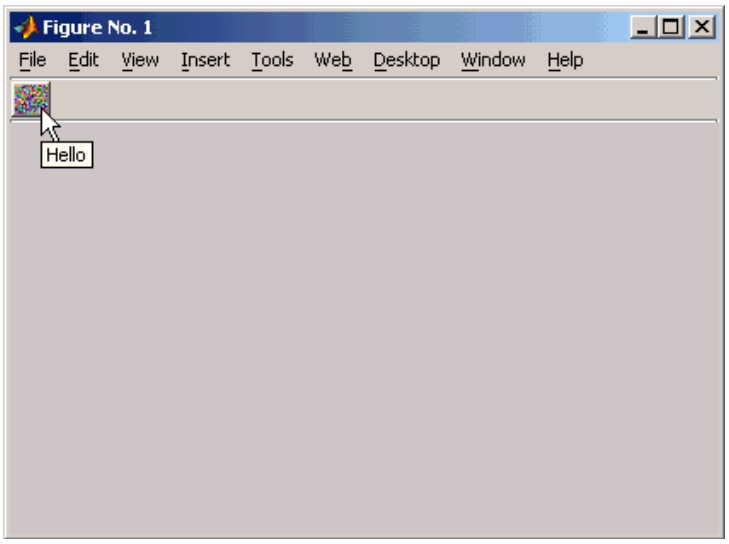
5.7.18.12 GUI Uygulamasına Araç Çubuğu Eklemek

Bir GUI uygulamasına araç çubuğu eklemek kullanıcının o uygulamayı kullanım kolaylığını arttırır. MATLAB GUI'de araç çubuğu ekleme işlemleri görsel olarak değil, komut satırları ile sağlanır. Bunun için GUI uygulamasının OpeningFcn callback bloğu kullnılır. Bu callbck GUI uygulamalarında hazırlık işlemlerinin yapılmasına imkân sağlar. GUI uygulaması daha açılmadan önce koşturulacak komut satırlarını bu blok içermektedir.

Bir GUI uygulamasına araç çubuğu eklemek için o uygulamanın OpeningFcn callback bloğuna şu komut satırları eklenmelidir.

```
ht = uitoolbar(hObject)
a(:,:,1) = rand(20);
a(:,:,2) = rand(20);
a(:,:,3) = rand(20);
htt = uitoggletool(ht,'CData',a,'TooltipString','Hello')
```

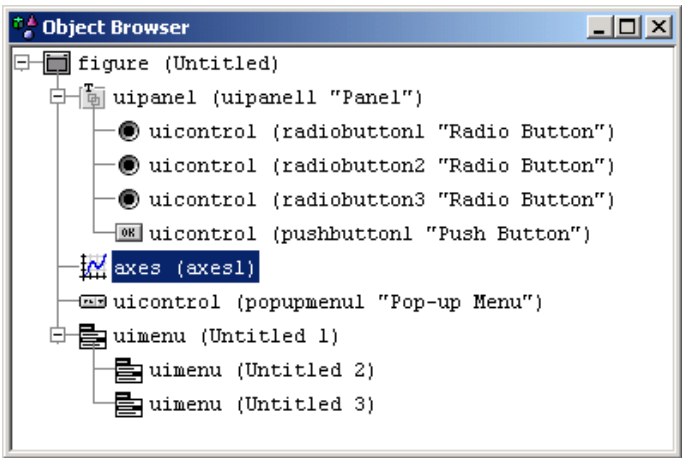
Yukarıdaki satırlar eklendikten sonra bu uygulama çalıştırıldığında Şekil 5.69'daki gibi bir GUI arayüzü ile karşılaşırız. Bu komut satırında uittolbar komutu ile yeni bir araç çubuğunun geçerli GUI figure alanına eklenmesi sağlanmaktadır. Daha sonra "a" isimli diziye eklenilecek bir buton üzerindeki renkleri göstermek üzere rasgele değerler atanmakta olup, uitoggletool komutu ile de bu araç çubuğuna ve rasgele renklerden oluşan bir görüntü ile yeni bir buton eklenmektedir. ToolTipString özelliği fare imleci bu buton üzerine geldiğinde gözükecek açıklama bilgisi için konulmuştur. Ancak, böyle bir açıklam koyma zorunluluğu da yoktur.



Şekil 5.69

5.7.18.13 Nesne Hiyerarşisinin Gösterilmesi

Tasarım esnasında programcı hangi nesneleri ve hangi isimlerle kullandığı genel hataları ile görmek isteyebilir. Bu gibi durumlar için GUIDE tasarımcıya Object Browser aracını sunar. Bu araç View menüsünden çalıştırılabilir. Bu araç Şekil 5.70'te de görülmektedir.



Şekil 5.70

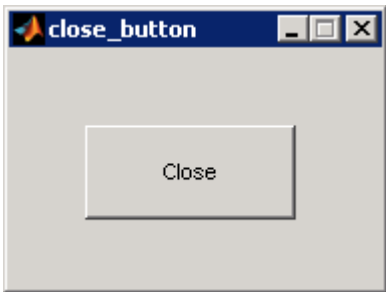
Object Browser aracı ile tasarımcı hangi nesneyi hangi Panel içinde veya hangi isim ile kullandığını kolaylıkla takip edebilir. Örneğin buradaki örnekte GUI figure içinde popup menu (açılır liste kutusu), axes (grafik çizimi), panel ile menü nesnelerinin eklendiği gözükmekte olup, panel içinde de üçü radio olmak üzere toplam dört adet buton kullanıldığı söylenilebilir.

5.7.19 GUI Nesnelerinin Programlanması

Burada her bir GUI nesnesinin programlama esnasında sıklıkla hangi özelliklerinin ve nasıl bir teknik ile kullanıldığı anlatılacaktır.

5.7.19.1 Push Button:

Şekil 5.71'deki gibi bir arayüz için butona tıklandığında GUI uygulamasını kapatacak bir Örnekle bir push buton kullanımını açıklayalım.



Şekil 5.71

Böyle bir GUI çalışma alanında fare işaretçisi push buton üzerinde iken View Callbacks/Callback komutunu verelim ve açılan callback bloğuna aşağıdaki komutları ekleyelim.

```
function pushbutton1_Callback(hObject, eventdata, handles)
display Goodbye
close(handles.figure1);
```

Böylece Close butonu tıklandığında GUI uygulaması sonlandırılmış olacaktır.

Başka bir örnekte de bir push butonun üzerine nasıl resim eklendiğini inceleyelim. Yukarıdaki örnekte Callback bloğuna önceki program satırları yerine aşağıdaki kodları yazalım.

```
a(:,:,1) = rand(16,64);
a(:,:,2) = rand(16,64);
a(:,:,3) = rand(16,64);
set(hObject,'CData',a)
```

Bu GUI uygulamasını Tools menüsünden Run komutunu kullanarak çalıştıralım ya da araç çubuğundan Run düğmesine basalım. Gelen GUI uygulamasında butona tıkladığımızda üzerine rasgele renklerden oluşan bir resimin atandığı görülecektir. Bu durum Şekil 5.72'de de görülmektedir.

Şekil 5.72

5.7.19.2 Toggle Buton:

Bir toggle buton çift durumlu çalışır. Bir kere tıklandığında basılı kalır. Bir daha tıklanırsa basılı kalmayıp eski konumuna geri döner. Böyle bir nesnede geçerli buton konumunu öğrenmek ve kullanabilmek için aşağıdaki komut satırları kullanılmalıdır.

```
function togglebutton1_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
```

```
if button_state == get(hObject,'Max')
% Toggle buton basıldığında yapılacak işlemler
...
elseif button_state == get(hObject,'Min')
% Toggle buton basılmadığı durumda yapılacak işlemler
...
end
```

5.7.19.3 Radio Buton:

Radio buton Buton Group nesnesi ile birlikte kullanıldığında daha etkili sonuçlar alınır. Ancak, kodlama yolu ile de radio butonların konumu kontrol edilebilir. Bir radio butonun basılıp basılmadığının kontrolü için şu kodlar kullanılabilir:

```
if (get(hObject,'Value') == get(hObject,'Max'))
% Radio buton basıldığında yapılacak işlemler
else
% Radio buton basılmadığı durumda yapılacak işlemler
end
```

5.7.19.4 Check Box:

Check Box nesnesinin konum kontrolü de radio butonlarınkine benzer şekildedir.

```
function checkbox1_Callback(hObject, eventdata, handles)
if (get(hObject,'Value') == get(hObject,'Max'))
% Checkbox nesnesi işaretlendiğinde yapılacak işlemler
else
% Checkbox nesnesi işaretlenmediği durumda yapılacak işlemler
end
```

5.7.19.5 Edit Text:

Bilgi girişi amacıyla sıklıkla kullanılan edit text nesnesinin string içerik bilgisini alamak için ilgili komut satırları şöyledir:

```
function edittext1_Callback(hObject, eventdata, handles)
user_string = get(hObject,'String');
% Callback bloğunun devamı ve diğer komutlar
```

Ancak, burada alınan bilgiler string tiptedir ve sayısal olarak kullanılamazlar. Sayısal olarak kullanabilmek için öncelikle edit box içerikleri sayısala dönüştürülmelidir. Daha sonra eğer ki hatalı bir giriş söz konusu ise hata kontrol deyimlerinin kullanılması ile bu durum giderilmelidir. Böyle bir durum için kullanılabilecek komut satırları aşağıda gösterilmiştir.

```
function edittext1_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
```

```
errordlg('Sayısal bir değer girilmelidir!..','Hatali Giris','modal')
return
end
% Callback bloğunun devamı ve diğer komutlar
```

Bu komut satırlarında ayrıca bir hata durumu oluştuğunda errordlg komutu ile kullanıcıya hatalı bir giriş yaptığı uyarı diyalog penceresi gösterilmektedir.

Edit Text nesnesinin callback satırları ancak kullanıcı baksa bir nesneye ya da gUI yüzeyine tıkladığı veya edit nesnesi içinde iken Enter tuşuna bastığı (çoklu giriş kutusu ise Ctrl + Enter tuş kombinasyonu kullanıldığı) zaman icra edilecektir. Aksi takdirde kullanıcı bir değer edit veya text nesnesine girerken bu callback satırları çalışmayacaktır.

5.7.19.6 Static Text:

Edit Text nesnesi ile benzer özelliklere sahiptir. Bu nesnenin edit text ten tek farkı kullanıcıdan bilgi girişi alınamamasıdır, sadece bilgi göstermek amaçlı kullanılır. Kodlama mantığı edit text nesnesi ile aynıdır.

5.7.19.7 Slider:

Bir kaydırıcı (slider) nesnesinin geçerli değerini program yoluyla okumak için gerekli komut satırları şöyle olmalıdır.

```
function slider1_Callback(hObject, eventdata, handles)
slider_value = get(hObject,'Value');
% Callback bloğunun devamı ve diğer komutlar
```

Bir kaydırıcı nesnesinin en küçük ve en büyük değerlerinin de ayarlanması bir programcı için gereklidir. Bunun için bu nesnenin Max ve Min özellikleri kullanılmalıdır.

5.7.19.8 List Box:

List Box nesnelerinin liste tipindeki string içeriğinin kullanılabilmesi için bu nesnelerin Value ve String özellikleri birlikte kullanılır. Kodlama mantığı şu şekilde olacaktır:

```
function listbox1_Callback(hObject, eventdata, handles)
index_selected = get(hObject,'Value');
list = get(hObject,'String');
item_selected = list{index_selected}; % Hücre dizisinden çevirme işlemi
% to string
```

5.7.19.9 Pop-Up Menu:

Popup menü nesnelerinde seçilen bir öğenin hangisi olduğu anlamak için bu nesnelerin Value özelliğinden yararlanılır. Kodlama örneği aşağıda gösterilmiştir.

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
```

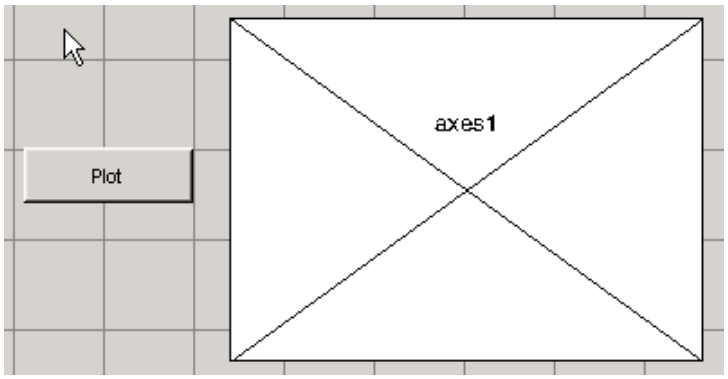
```
switch val
case 1
% Birinci öğe seçili iken yapılacak işlemler
case 2
% İkinci öğe seçili iken yapılacak işlemler
% Callback bloğunun devamı ve diğer komutlar
```

Eğer ki programcı seçilen öğenin stringini öğrenmek isterse şu komut satırları kullanılabilir:

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
string_list = get(hObject,'String');
selected_string = string_list{val}; % Hücre dizisinden çevirme işlemi
% to string
% Callback bloğunun devamı ve diğer komutlar
```

5.7.19.10 Axes:

Grafik çizimlerinin kullanıcıya sunulmasında sıklıkla kullanılan bir nesne olan axes için Şekil 5.73'teki gibi bir GUI arayüzüne sahip olunduğu düşünülmüş olsun.

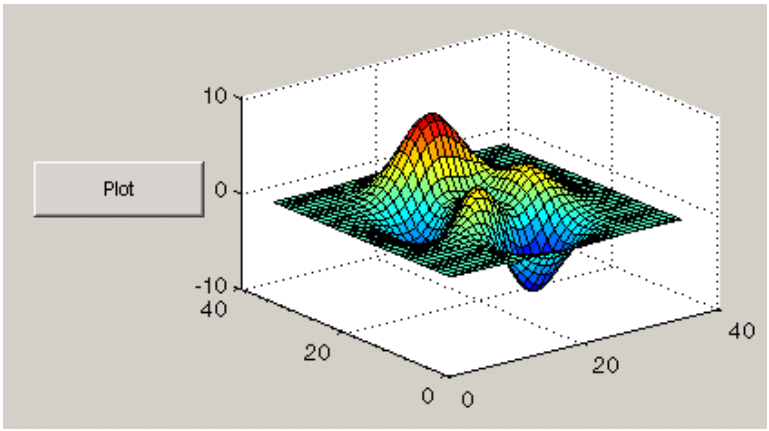


Şekil 5.73

Burada Plot butonunun callback bloğuna şu komut eklensin ve ardından Run komutu ile bu GUI uygulamasını çalıştırılsın.

```
surf(peaks(35));
```

Ekrana gelen GUI uygulama penceresinde Plot düğmesini tıkladığımızda Şekil 5.74'teki gibi bir ekran görüntüsü ile karşılaşılır.

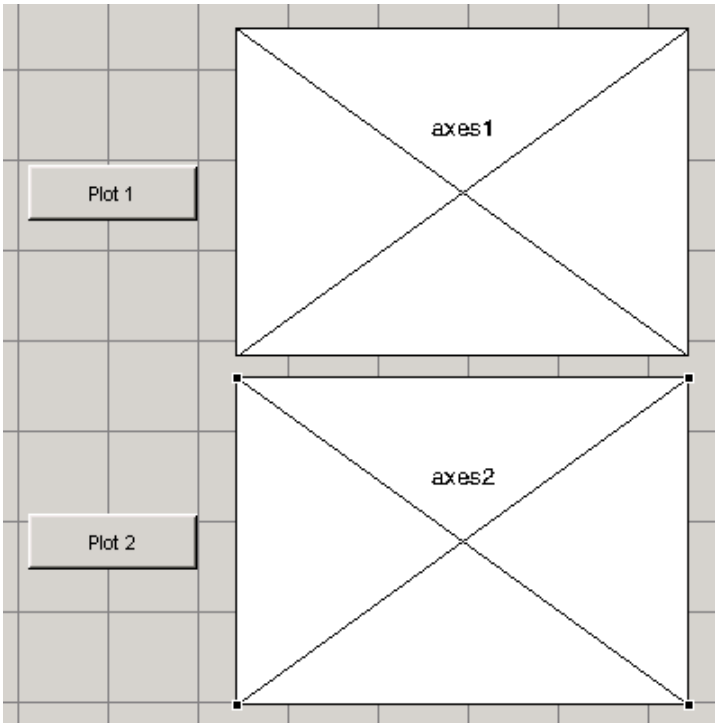


Şekil 5.74

Görüldüğü üzere tek axes nesnesi içeren GUI tasarımlarını programlamak kolay gözükmektedir. Ancak, eğer ki bir GUI arayüzü birden fazla axes nesnesine sahip ise bu takdirde hangi axes nesnesi o an aktif ise o nesne üzerinde çizimimiz gözükecektir. Aktif axes nesnesinin hangisi olacağı şu komutla belirtilir:

```
axes(handles.axes1)
```

Bir sonraki axes nesnesi ile ilgili örneğimizde çoklu axes nesnelerinin programlanmasına ait bilgilere yer verilmiştir. Bu örnek için GUI arayüzünün Şekil 5.75'teki gibi olduğu kabul edilsin. .



Şekil 5.75

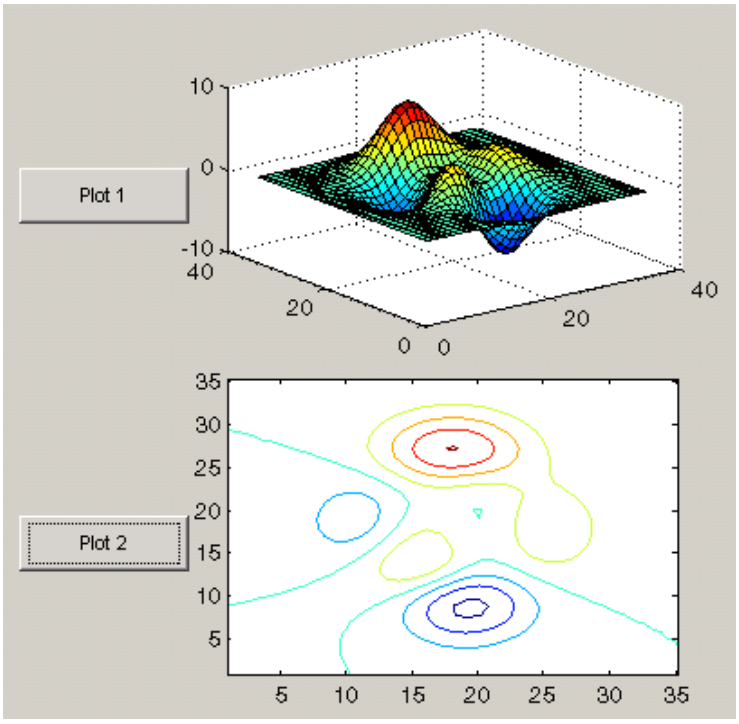
Bu uygulamada Plot 1 butonuna ait callback satırlarına

```
surf(handles.axes1, peaks(35));
```

Plot 2 butonuna ait callback satırlarına da

contour(handles.axes2, peaks(35));

komutları eklensin. Daha sonra da bu GUI uygulamsı çalıştırılsın. Gelen uygulama penceresinde ayrı ayrı Plot1 ve Plot 2 butonlarını tıklansın. Şekil 5.76'daki gibi bir görüntü ile karşılaşılır:



Şekil 5.76

Bu uygulama bize istenilen sonucu vermektedir. Ancak, aynı işlev farklı yöntem kullanılarak da gerçekleştirilebilir.

Plot 1 butonuna ait callback satırlarına önceden yazılan komut satırları yerine

axes(handles.axes1);
surf(peaks(35));

Plot 2 butonuna ait callback satırlarına önceden yazılan komut satırları yerine

axes(handles.axes2);
contour(peaks(35));

satırları yazılsın ve bu uygulama tekrar çalıştırılsın. Butonlar tıklandığında sonuç değişmeyecektir. Ancak, axes() komutunun kullanılması bir programcı olarak daha kolay ve sade bir kodlama anlamı taşıması bakımından genellikle tercih edilir.

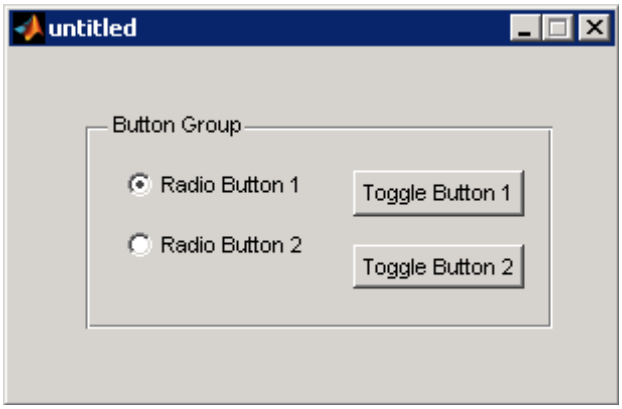
5.7.19.11 Panel:

Bu nesneler programlama amacı taşımayan yapıdadırlar. Panellerin kullanım amacı görsel olarak GUI uygulamasını zengin kılmak ve kullanıcıya yapacağı işlemler ile ilgili kullanımını kolaylaştırmaktır. Ancak, bir GUI uygulamasının koşturulması anında GUI figure alanının

boyutlarının değiştirilmesi söz konusu ise, bu takdirde panel nesnelerinin ResizeFcn metodu kullanılabilir.

5.7.19.12 Button Group:

Örnek olarak Şekil 5.77'de görülen bir GUI arayüzüne sahip olunduğu düşünülsün. Böyle



Şekil 5.77

bir durumda Buton Group nesnesi özelliğinden ötürü bu dört nesne toggle durumlu olarak (yani herhangi bir anda yalnizca biri seçili olabilir şekilde) çalışacaktır. Böyle bir uygulamada hangi buton seçili ise yaptırılmak istenilen programlama işlemleri şöyle olacaktır:

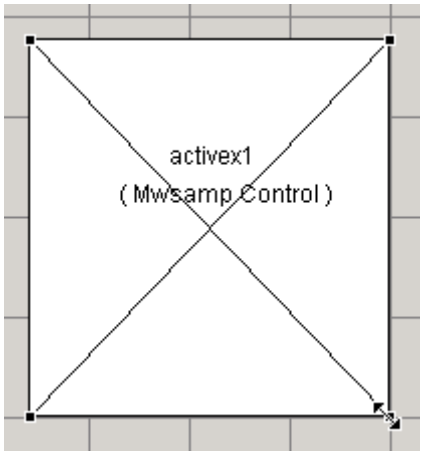
```
function uibuttongroup1_SelectionChangeFcn(hObject,...
eventdata,handles)
switch get(hObject,'Tag') % Seçili nesnenin Tag bilgisini alma
case 'radiobutton1'
% radiobutton1 seçili ise yapılacak işlemler
case 'radiobutton2'
% radiobutton2 seçili ise yapılacak işlemler
case 'togglebutton1'
% togglebutton1 seçili ise yapılacak işlemler
case 'togglebutton2'
% togglebutton2 seçili ise yapılacak işlemler
% Daha fazla sayıda buton varsa koşulların kontrolü böylece devam eder.
otherwise
% Hiçbir buton seçili değilse yapılacak işlemler
end
```

5.7.19.13 ActiveX Component:

Active X nesneleri ile MATLAB GUI uygulamalarının esnekliği arttırılmıştır. Böylece sadece GUI'nin kendi nesneleri ile tasarımcı sınırlı kalmamış olup, pek çok farklı nesneyi de alıp kullanabilir.

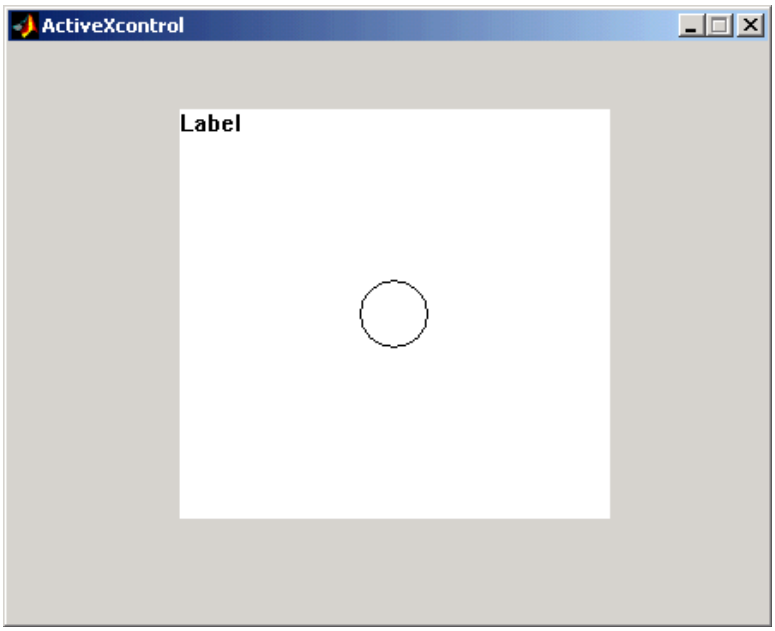
GUI yüzeyine önce bir ActiveX nesnesi ekleyelim ve karşımıza gelen component listesinden "Mwsamp Control" ü tıklayıp Create butonuna basalım. Ekelenen bu ActiveX

nesnesinin köşesinde boyutlarını değiştirmek ve büyütmek mümkündür. GUI çalışma alanındaki görüntü Şekil 5.78'deki gibi olacaktır:



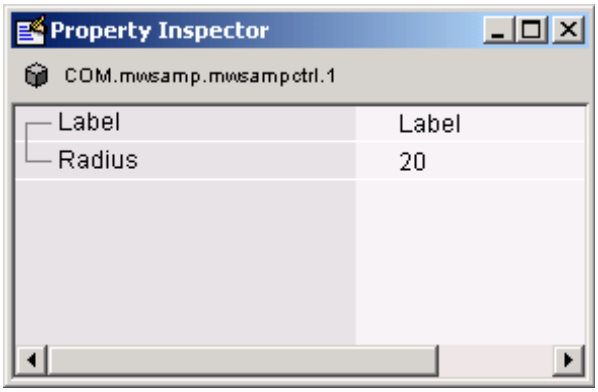
Şekil 5.78

Bu GUI uygulamasını çalıştırıldığında karşılaşılan arayüz ekranı Şekil 5.79'da gösterilmiştir.



Şekil 5.79

Tekrar tasarım ortamına geçilsin, Activex nesnesi seçilip Property Inspector penceresinden özelliklerine bakılsın.



Şekil 5.80

Bu ActiveX nesnesinin iki özelliği vardır (Bir ActiveX nesnesi ile ilgili daha kapsamlı bilgilere üreticinin hazırlanmış olduğu Kullanıcı Kılavuzundan ulaşılabilir.). Bunların işlevi şu şekildedir:

- Label özelliği, kullanıcıya ekranda sunulacak bilgidir.
- Radius özelliği, ekranda gözükecek dairenin yarıçap uzunluğudur.

Şimdi de bu ActiveX nesnesi ile ilgili programlama teknikleri açıklayalım. Öncelikle ActiveX nesnemizin Click Callback satırlarına aşağıdaki komutları ekleyelim.

```
hObject.radius = .9*hObject.radius;
hObject.label = ['Yaricap = ' num2str(hObject.radius)];
refresh(handles.figure1);
```

Böylece fare işaretçisi ile her ActiveX nesnesi üzerinde tıkladığımızda bu callback'teki komutlar icra edilecektir. Burada yapılan her tıklanmada daire yarıçapının artırılıyor olmasıdır. Ayrıca, nesnemizin Label özelliğinden faydalanılarak dairemizin yarıçap değeri ekrana yazdırılmaktadır. Burada unutulmaması gereken nokta bir ActiveX nesnesi ile ilgili özellikler değiştirildiği zaman mutlaka bunların o nesneye uygulanabilmesi ve uygulama arayüzünde yapılan değişikliklerin gözlenebilmesi için geçerli GUI figure yüzeyinin refresh edilmesi gerekliliğidir. Bunbun yanında burada Label özelliği string tipi verileri tuttuğu için yarıçap sayısal değerden num2str(komutu kullanılara string tipi bilgiye dönüştürülmekte ve ardından Label özelliğine atanmaktadır. Aksi takdirde programımızın çalıştırılması sırasında hata ile karşılaşılacaktır.

ActiveX nesnelerinin program yoluyla özelliklerinin değiştirilmesi gerektiğinde direk olarak ActiveX nesnelerinin özellikleri get metodu kullanılarak alınabilir. Bu durumla ilgili olarak şu komut satırları incelenebilir:

```
handles.activex1.radius = ...
get(hObject,'Value')*handles.default_radius;
handles.activex1.label = ...
['Radius = ' num2str(handles.activex1.radius)];
refresh(handles.figure1);
```

5.7.19.13.1 GUI Yüzeyine Eklenen Bir Activex Nesnesinin Tüm Metotları

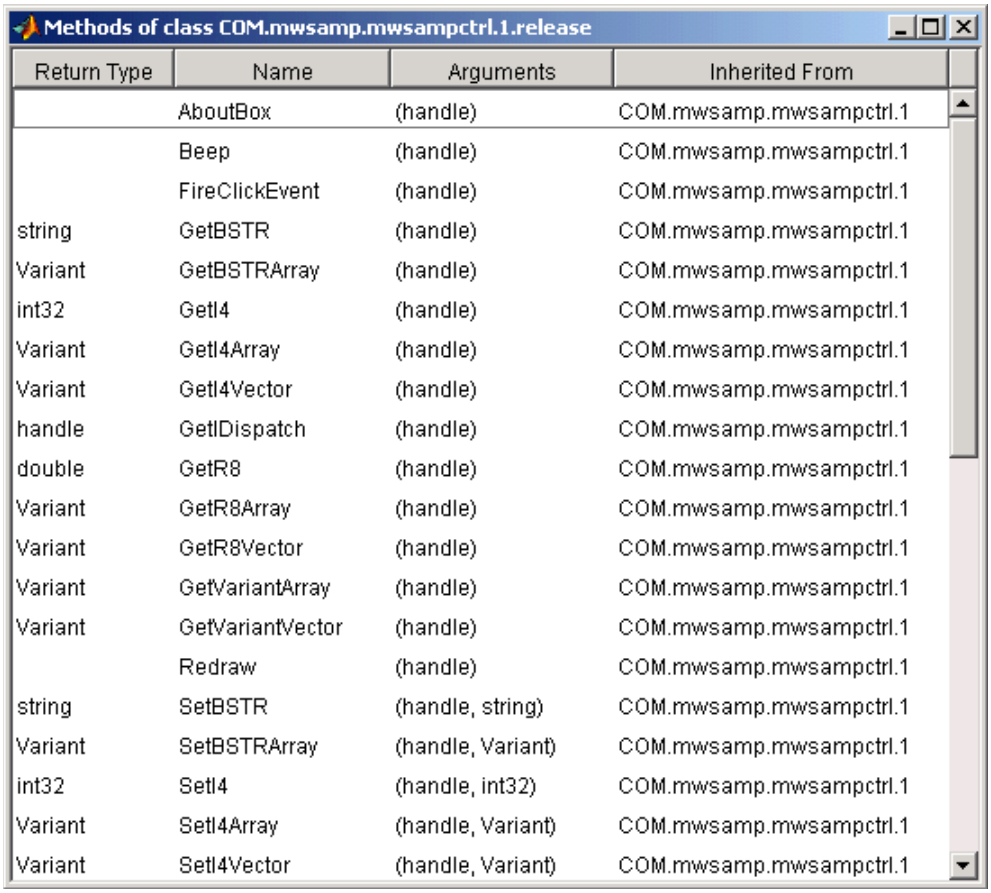
GUI yüzeyine eklenen bir activex nesnesinin tüm metotları

methodsview(hObject)

komutu ile öğrenilebilir. Örneğin bu komutu GUI çalışma alanına eklenen bir ActiveX nesnesinin Click Callback satırlarına eklenmiş olsun. Karşımıza aşağıdakine benzer bir ekran gelecektir. Ayrıca,

methods(hObject)

komutu ile bir nesnenin metotları MATLAB komut satırında da gösterilebilir. Komut çalıştırıldığında gözüken ekran görüntüsü Şekil 5.81'de verilmiştir.



Return Type	Name	Arguments	Inherited From
	AboutBox	(handle)	COM.mwsamp.mwsampctrl.1
	Beep	(handle)	COM.mwsamp.mwsampctrl.1
	FireClickEvent	(handle)	COM.mwsamp.mwsampctrl.1
string	GetBSTR	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetBSTRArray	(handle)	COM.mwsamp.mwsampctrl.1
int32	GetI4	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetI4Array	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetI4Vector	(handle)	COM.mwsamp.mwsampctrl.1
handle	GetIDispatch	(handle)	COM.mwsamp.mwsampctrl.1
double	GetR8	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetR8Array	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetR8Vector	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetVariantArray	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetVariantVector	(handle)	COM.mwsamp.mwsampctrl.1
	Redraw	(handle)	COM.mwsamp.mwsampctrl.1
string	SetBSTR	(handle, string)	COM.mwsamp.mwsampctrl.1
Variant	SetBSTRArray	(handle, Variant)	COM.mwsamp.mwsampctrl.1
int32	SetI4	(handle, int32)	COM.mwsamp.mwsampctrl.1
Variant	SetI4Array	(handle, Variant)	COM.mwsamp.mwsampctrl.1
Variant	SetI4Vector	(handle, Variant)	COM.mwsamp.mwsampctrl.1

Şekil 5.81

5.7.19.13.2 Activex Nesnesi İçeren Bir GUI Uygulamasının Compile Edilmesi

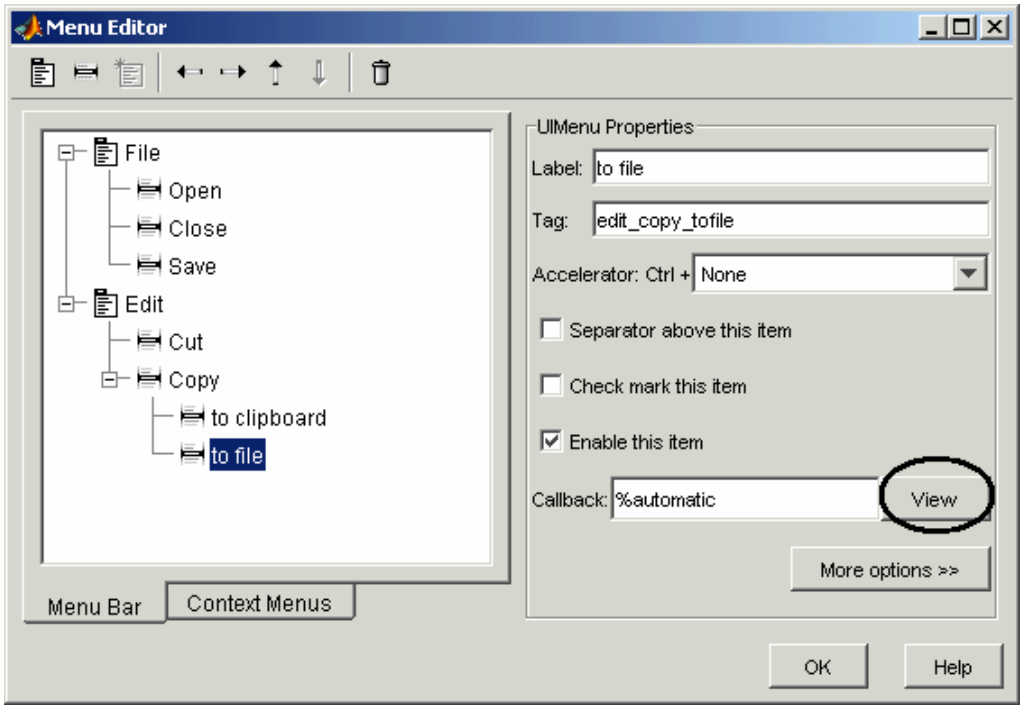
ActiveX içeren bir GUI uygulaması compile edileceği (derleneceği ve bağımsız bir çalıştırılabilir .exe uzantılı dosya edileceği) zaman aşağıdaki gibi bir yazım şekli kullanılmalıdır.

mcc -m mygui -a mygui_activex1

Her bir ActiveX nesnesi için "-a mygui_activex1" yazımı artırılmalıdır. Burada mygui adı ile kaydedilen bir GUI uygulamasının olduğu varsayılmış olup, GUI uygulamalarını derlemek üzere bu komutun uygulanacağı sistem de MATLAB Compiler aracının yüklü olduğu kabul edilmiştir.

5.7.19.14 Menü Öğeleri:

Tasarlanan bir GUI uygulamasına eklenen menünün herhangi bir öğesi tıklandığında istenilen komutların icra edilmesi istenebilir. Bunun için Menu Editor aracında ilgili öğe seçilir ve özellikler panelinden callback satırlarına gitmek için View butonu tıklanır. Bu durum Şekil 5.82'de de görülmektedir.



Şekil 5.82

5.8 GUI Uygulamalarında Callback Türleri

Callbackler önceki konularda da bahsedildiği üzere oluşan herhangi bir olaya bağlı olarak her nesne için ayrı ve olayın türüne göre icra edilen alt program parçalarıdır. Aşağıda sunulan tabloda kullnılan callbacklerin işlevi ve hangi nesneler ile birlikte kullanıldığı beliritilmiştir.

Bir m file dosya yapısı gereği bir GUI uygulaması tasarımında da aynı dosya ismini taşıyan fonksiyon ismi ile başlayan bir m function yapısına sahiptir. Ancak, giriş ve çıkış function varargout = DeneGui (varargin)
M function ilk satırında yer alan (yukarıda da ifade edilen deyimde de görüldüğü üzere) parametrelerinin dinamik olmasına dikkat edilmelidir. Yani, "varargin" deyimi ile giriş parametreleri hücre dizisi formatında birden fazla olabilir. Aynı, şekilde GUI uygulamasının kapatılacağı zaman aynı bir fonksiyon mantığı ile "varargout" dışarıya aktarılacak parametreler bu değişkene aktarılabilir.

Dışarıdan fonksiyon içerisine gönderilen giriş parametreleri ile ilgili bilinmesi gereken iki değişken vardır. Bunlar nargin ve varargin değişkenleridir.

- nargin değişkeni : Fonksiyona (ya da GUI uygulamasına) dışarıdan gönderilen toplam parametre sayısını tutar.

- vargin değişkeni: Fonksiyona gönderilen parametrelerin alınmasını sağlar. Hücre dizisi yapısında olduğundan parametrelerin alınması için dizi indislerinin "{" ve "}" işaretleri arasında yazılması gerekir.

Fonksiyonunun içinden dışarıya GUI uygulaması sonlandırılırken gönderilen çıkış parametreleri ile ilgili bilinmesi gereken iki değişkeni vardır. Bunlar nargout ve varargout değişkenleridir.

- nargout değişkeni: Fonksiyona (ya da GUI uygulamasına) dışarıdan gönderilen toplam parametre sayısını tutar.

- vargout değişkeni: Fonksiyondan dışarıya parametrelerin gönderilmesini sağlar. Hücre dizisi yapısında olduğundan parametrelerin bu değişkene atanması sırasında dizi indislerinin "{" ve "}" işaretleri arasında yazılması gerekir.

Tablo 5.1 GUI Uygulamalarında Callback Türleri

Callback Türü	Tetiklendiği Olay	Kullanıldığı Nesneler	
ButtonDownFcn	Fare göstergeci Figure veya bir nesnenin kenarlarından 5 piksel içerde olduğunda fare butonu tıklandığında oluşur. UI control için Enable özelliğinin true olması gerekmektedir.	axes	figure
		button group	panel
		user interface controls	
Callback	Nesnenin temel olayı. Örneğin bir push button tıklandığında ya da bir menü öğesi seçildiğinde oluşur.	context menu	menu
		user interface controls	
CloseRequestFcn	Figure kapanmadan önce çalıştırılır.	figure	
CreateFcn	Bir nesnenin create edilmesi anında oluşur. Nesne oluşturulduğunda initializing için kullnılabilir. Bu olay nesne create edilince ancak nesne GUI alanında gözükmeden önce icra edilir.	axes	figure
		button group	context menu
		menu	panel
		user interface controls	
DeleteFcn	Bir nesnenin kaldırılması anında oluşur. Herhangi bir nesne veya figure yok edilmeden önce temizlemeye dayalı operasyonlarda kullanılabilir.	axes	figure
		button group	context menu
		menu	panel
		user interface controls	
KeyPressFcn	Fgure veya bir nesne focus (aktif) olduğunda veya klavyeden herhangi bir tuşa basıldığında oluşur.	figure	
		user interface controls	

ResizeFcn	Panel, button group veya figure nesnenelerinin boyutları kuıllanıcı tarafından değiştiirldiğinde oluşur. Ayrıca, bu duurmun gerçekleşmesi için figure'e ait "Resize" özelliği "on" olmalıdır.	button group	figure
		panel	
SelectionChangeFcn	Button grouop nesnesi içinde kullanıcı farklı bir radio veya toggle butonu seçtiğinde bu olay tetiklenir.	button group	
WindowButtonDownFcn	Fare işaretçisi figure penceresi üzerinde iken farenin tuşuna basıldığında oluşur.	figure	
WindowButtonMotionFcn	Fare işaretçisi figure penceresi üzerinde ihareket ettirilirken oluşur.	figure	
WindowButtonUpFcn	Fare tuşu bırakıldığı zaman oluşur.	figure	
Not : User interface controls push button, slider, radio button, check box, editable textbox, static text , listbox ve toggle butonları içeren genel bir tanımlama ismidir. Bazen uicontrols olarak da isimlendirlebilirler.			

5.9 GUI Uygulamalarında Callbackler Arasında Ortak Veri Geçişini Sağlayan Yollar

GUI uygulamalarında bir değişken içeriği birden fazla callback içeriisnde kullanılmak istenebilir. Ya bir GUI callback functionun ürettiği değer başka bir function için giriş verisi olabilir. Konuyu daha geniş anlamıyla anlatılmak istenirse GUI uygulamalrında global değişken kullanım yolları öğretilmeye çalışılmaktadır. Bu durma benzer örnekler çoğaltılabilir. Bu durumu gerçekleştirmek üzere GUI uygulamalrında 6 farklı yöntem vardır.

5.9.1 Handles Yapı Değişkeni Kullanılarak Global Kullanımı

GUI uygulamalrında en sık kullanılan yöntemdir. Bu yiöntemde her callback functiona giriş parametre olarak gönderilen ve GUI uygulamalarında kullanıcı verilerinden ziyade GUI nesneleri ile ilgili handle değerlerini tutmaya yarayan “handles” yapı değişkeninden yararlanılır. Bu yapıyı çok kolaydır. Örnek olarak sistem_cikis_sayisi isminde bir değişkeni her callback içerisinden ortak olarak kullanmak isteğimiz varsayılsın ve içeriği 4 yapılsın. Daha sonra da bu değişkeni, handles yapısı içerisine koymak istediğimizi düşünelim.
Bu işlem için aşağıdaki komut satırları yazılmalıdır.

```
sistem_cikis_sayisi = 4;
Handles . sistem_cikis_sayisi = sistem_cikis_sayisi;
guidata (hObject, handles);
```

burada 1. satır ile bu deyimlerin kullanıldığı callback içinde lokal bir "sistem_cikis_sayisi" isminde değişken oluşturlmuş ve içeriği 4 olarak atanmıştır. Ancak, loakl bu değişkenin değeri ve kendisinini varlığı callback fonksiyonunun icrası tamamlandıkrtan sonra kaybolacaktır. Tekrar aynı callback'e gelindiğinde de önceki değer silinmiş olacaktır. Burada dikkat edilmesi gereken standart bir kalıp şeklinde kullanılan guidata (hObject, handles); komut satırının varlığıdır. Bu satır ile handles yapı değişkeni yeni değerlerle birlikte callback dışına çıkmadan güncellenmekte ve callback dışına çıkıldığında da veya başak acalback çağrıldığında bu yapı içerisinden kullanılmasına imkân verilmektedir.

Örneğin başka bir callback içeirisinde kurulacak bir for döngüsünün toplam sayma adedi bu değişken kadar olmak üzere aşağıda yer alan komut satırları yazılabilir.

```
For i=1:1:handles. sistem_cikis_sayisi
        % döngü içieirinde icra edilmesi düşünülen kom ut satırları
End
```

5.9.2 Global Değişken Tanımlama Deyimi

Herhangi bir callback başında global deyimi ile bir değişken ismi girildiği takdirde eğer daha önce böyle bir değişken yok ise oluşturulur ve başlangıç değeri otomatik olarak 0 (sıfır) değeri atanır. Ancak, eğer daha önceden bu callback icra edilmiş ve global deyimöi ile tranımlanan değişken bellekte bir yere sahip ve değeri var ise bu değerininh bir sonraki callback çağrısında da devam ettirecektir. Bu durumun kullanıma örnek komut satırları aşağıda gösterilmiştir.

```
Function edit1_callback( … )
global sayac;
sayac=sayac+1;
% diğer icra edilecek komutlar
if isequal(sayac,5)
sayac=0;
end
```

Yukarıdaki örnekte sayac değişkeni 0 dahil toplam 6 farklı değer almakta ve 5 olduğunda değeri tekrar sıfırlanmaktadır.

5.9.3 GUI Alanında Visible Ve Enable Özellikleri Off Yapılmış Nesne Kullanılması

Tasarımcı isterse kullanım kolaylığı sağlaması bakımından GUI alanında tasarım aşamasında görünen, fakat GUI icrası sırasında kullanıcılar tarafından görülemeyen ve kontrol edilemeyen ekstra bir nesne kullanabilir. Bunu yapmak için tek yapması gereken bu nesnenin "Enable" ve "Visible" özelliklerinin "off" yapılmasıdır. Bu nesneye ait "Value" veya "String" değerleri kullanılarak global değer kullanımına yönelik programlama yapılabilir.

5.9.4 Load Ve Save Deyimlerinin Kullanılması

Programcı herhangi bir callback içinde kullandığı değişkenleri o hali ile bir mat dosyasına kaydedebilir ve ileride ya da gerekli görüldüğü takdirde tekrar kullanmak üzere çağırabilir. Örnek olarak aşağıdaki komut satırları ele alınabilir.

```
a=5;
b=40;
save sayilar_a_ve_b;
```

Bu komutlar sonucunda a ve b değişkenleri "sayilar_a_ve_b.mat" isimli bir Matlab veri saklama dosyasına kaydedilir. İleride kullanılacağı zaman yapılması gereken çok kolaydır.

```
load sayilar_a_ve_b;
c=a*b;
```

Yukarıdaki komutların icrası ile a ve b değerleri herhangi bir an veya o an için workspace alanına veya callback'in geçici bellek alanına yüklenir. Ancak, değerleri (5 ve 40 sayıları) aynen korunur. Böylece c değişkeninin değeri 200 olacaktır.

5.9.5 Nesnelerin Userdata Özelliğini Kullanmak

Global değişken kullanmanın bir başka ve kolay yolu da her bir GUI nesnesine ait "UserData" özelliğinin kullanılmasıdır. Bu değişkenin içeriği string tipi verileri tutatbildiği gib sayısal tip verileri de direk olarak gönderme ve kullanma imkânı programcıya sunulur. Yani, aralrda num2str veya str2num komutlarının kullanılmasına gerek kalmaz.
Örneğin edit1 nesnesinin Userdata alanına Checkbox1 isimli nesnenin "Value" değerini koyan komut satırları yazılmış olsun.

```
durum = get ( handles.checkbox1 , 'Value' ) ;
set ( handles.edit1 , 'UserData' , durum ) ;
```

Örneğin Edit1'in UserData alanında bulunan bu değeri kendisinin String özelliğine atanmış olsun. Normalde bir string değer sayısal bir değişkene atandığında hata oluşur ve burada da hata oluşması beklenir Çünkü, bilindiği gibi "checkbox" nesnelerinin "Value" özellikleri sayısal tiptedir. Ancak, "edit" nesnelerinin "String" özelikleri sayısal değil, string tipindedirler.

```
userdata_icerigi = get ( handles.edit1,'UserData' );
set ( handles.edit1 , 'String' , userdata_icerigi );
```

Görüldüğü gibi bu komutların icrası sonucu herhangi bir hata oluşmamıştır (sayısal ve string değer dönüşümleri arasında). Yani, UserData özelliği değişkenler arasında uyumluluk sağlamakta ve kendisi otomatik olarak tip dönüşümlerini gerçekleştirmektedir.

5.9.6 Uygulama Datası Yöntemi

En gelişmiş, ancak kullanımı biraz zahmetli olan yöntemdir. Bu yöntem ile herhangi bir GUI nesnesinin varolan özelliklerine sanki handles yapısına yeni bir değişken ekler gibi yeni bir değişken eklenebilirç Eklelnen nesne genellikle figure olduğu için yöntem ismini buradan almaktadır. Örnek olarak toplam_boyut alanının bir herhangi bir nesne callback'inde icrası

sonucu bahsedilen değiken bu callback'e ait nesnenin boyut isminde yeni bir özelliği olarak tanımlanmış olsun. Kullanılacak komut satırları aşağıda verilmiştir.

```
toplam_boyut = 15 ;
setappdata ( hObject, 'boyut' , toplam_boyut ) ;
```

Herhangi bir anda bir nesnenin uygulama datasının alınması gerektiğinde de aşağıdaki komut satırları icra edilmelidir. Örnek olarak yukarıda içeriisnde application data saklanılan nesne edit4 isimli bir nesne olsun. Buna göre ilgili komutlar şu şekilde yazılmalıdır:

```
boyut_degiskeni = getappdata(handles.edit4, 'boyut');
```

5.10 GUI Uygulamalarında Temizleme Komutları

Herhangi bir GUI uygulamsında figure penceresi veya, komut satırı alanı ya da axes (grafik çzim) nesnesinin içeriğinin temizlenmesi gerekebilir. Ayrıca herhangi bir değişkenin workspace den atılarak bellketen temizlenmesi de gerekebilir. Bu durumlar için şu komutlar kullanılmalıdır.

- Figure alanını temizlemek için clf komutu
- Axes nesnesindeki çizimin temizlenmesi için cla komutu,
- Komut satır ekranının temizlenmesi için clc komutu,
- Worspace alanındaki tüm değişikenleri silmek ve bellekten temizlemek için clear all veya kısaca clear komutu
- Workspace alanından örneğin adet_no isimli değişken kaldırılmak istenirse clear adet_no komutu

kullanılmalıdır.

5.11 GUI Uygulamalarında Kullanılan Standart Handle Değişkenleri

GUI uyugulamalrında birden fazla nesne veyua figure alanı ile çalışıldığı düşünülürse bazen yanlış veya istenmeyen nesnelkere kontrollerin kaydığı görülebilier. Bu gibi durumlardan sakınmak için aktif axes veya grafik nesnesi gibi bazı belirli nesnenelr için özel olarak handle numarasını tutmak amacıyla tanımlanmış değişkenler Mtalab tarafından GUI tasarımcılarına sunulmuştur. Ayrıca ince bir bilgi olmasına karşılık bilmekte yarar var. Bir GUI uygulamasına ait figure nesnesinin handle numarası varsayılan olarak GUI handles yapısının içinde yer alan "output" isimli değişkende de tutulmaktadır.

GUI uygulamalarında handle numaralarını öğrenmek amacıyla sıklıkla kullanılan standart değişkenler şunlardır:

gcf : Geçerli figure nesnesinin handle numarasını verir.
gca : Geçerli axes (grafik çizim) nesnesinin handle numrasını verir.
gco : nesne_handle=gco(fig_handle) kullanımı ile GUI alanında en sok tıklanmış ya da en son aktif olan nesnenin handle numarasını verir.
gcbf : figure_bo = gcbf; kullanımı ile hangi figure nesnesine ait bir callback (veya figurein icerdiği bir nesne callback in çalışıyor olabilir) bu figure nesnesine ait handle numarası döner.
gcbo : Aktif olarak hangi nesnenin callback i çalışıyor ise o nesneye ait handle numarası

döner. Nesne_handle = gcbo olarak kullanılabileceği gibi
[nesne_handle, figure_handle] = gcbo şeklinde kullanım ile aktif nesnenin bulunduğu figure handle numrası da elde edilebilir.

5.12 Herhangi Bir GUI Uygulamasının Sonlandırılması

Bir GUI uygulamasını sonladırmak için kullanılacak komut "close" dur. İstenirse close(gcf) ile aktif GUI uygulamsı sonlandırılmak yerine başka açık bir fifure de sonlandırılabilir. Bunun için örneğin figure no su fig_no isimli bir değişkende tutulan figure nesnesinin kapatılması için close(fig_no) komutu icra edilmelidir.

5.13 Yeni Bir Figure Oluşturma Komutu

Bu amaçla "figure" komutu kullanılır. Bu komut eğer bir değişkene atanırsa bu takdirde açılan yeni figure ekranının handle numarası da saklanmış ve korunmuş olacaktır. Örnek olarak fig_handle_no = figure; şeklinde komut yazımı verilebilir.

5.14 MATLAB GUI Uygulamalarında Etkileşim Kutuları Yönetimi

MATLAB GUI ile görsel tasarım hem programcının tasarımı kolay kılmakta, hem de kullanıcı yapacağı işleri görerek ve kolaylıkla birkaç tıklama ile dahi gerçekleştirebilmektedir. GUI uygulamalarının kullanıcılara sunduğu kolaylıklardan biri de kullnıcıların yapılan veya yapılacak işler ilgili uyarılması veya bilgilendirilmesi amacıyla kullanılan etkileşim kutularının kullanılmasıdır. Örneğin bir hata oluştuğunda bu durumu en temel anlamı ile GUI arayüzünde bir nesne kullanarak ve bu nesneye ait bir özelliği (örneğin renk gibi) değiştirerek kullanıcı bilgilendirilebeilir. Ancak, bu yöntem etkin bir uyarma aracı olarak düşünülemez. Bunun yerine az önce bahsedildiği gibi örneğin bir hata penceresi ile hem kullanıcının başka işlem yapması engellenebilir (yani, uygulamanın arka plandaki arayüzü uyarı anında kilitlenebilir), hem de daha gerçekçi ve sade bir diyalog pencere görüntüsü kullanıcının Windows benzeri GUI tabanlı ortamlarda alışkanlıkların dışında tasarım ekranları ile karşlaşmamış olur. Ayrıca, etkileşim kutularının kullanımı ile programıcının da GUI arayüzünde herhangi bir tasarım değişikliğine gitmesine fırsat verilmez, dolayısıyla daha etkin GUI tabanlı uygulama geliştirme imkânı sağlanmış olmaktadır.

Matlab ile hazırlanan GUI uygulamalrında kullanılabilecek etkileşim kutuları Tablo 5.2'de gösterilmiştir.

Tablo 5.2 Matlab ile Tasarlanan GUI Uygulamalarında Kullanılabilecek Etkileşim Kutusu Türleri

Etkileşim Kutusu Türü	Açıklama
errordlg	Hata Penceresi
helpdlg	Yardım Penceresi
inputdlg	Veri Giriş Penceresi
listdlg	Liste Görünüm Penceresi
printdlg	Yazdırma Penceresi
questdlg	Sorgu Penceresi
uigetdir	Klasör Yolu Seçme Penceresi
uigetfile	Dosya Açma Penceresi
uiputfile	Dosya Kaydetme Penceresi

uisetcolor	Renk Seçim Penceresi
uisetfont	Font Seçim Penceresi
warndlg	Uyarı Penceresi
waitbar	Yükleme Çubuğu
pagesetupdlg	Sayfa Yapısı Penceresi
msgbox	Mesaj Kutusu
printpreview	Sayfa Önizleme Penceresi

Aşağıda GUI uygulamalarında çeşitli etkileşim kutularının ne amaçla ve nasıl kullanılacağı ayrıntılı olarak anlatılmıştır.

5.14.1 Hata Penceresi (Error Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında oluşan bir hata hakkında bilgilendirilmesi amacıyla kullanılır. Kullanımı şu şekildedir:

errordlg('Yanlış değer girildi.','Hata Penceresi','modal')

Hata diyalog penceresinin ekran görüntüsü Şekil 5.83'te gösterilmiştir.

Şekil 5.83

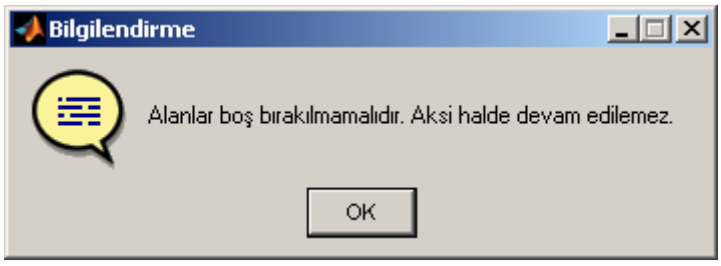
Yukarıdaki kullanımda modal seçeneğinin kullanılması seçimliktir, yani kullanılmasa da olur. Ancak, bu seçenek ile hata penceresine cevap verilmeden uygulamaya ait ilmelerin yapılması engellenmiş olmaktadır. "modal" seçeneği yerine "non-modal" kullanılarak bu özelliğin iptal edilmesi sağlanılabilir.

5.14.2 Yardım Penceresi (Help Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında eksik veya yanlış bilgi girme ya da herhangi bir konuda bilgilendirilme amacıyla kullanılır. Kullanımı şu şekildedir:

helpdlg('Alanlar boş bırakılmamalıdır. Aksi halde devam edilemez.','Yardım Penceresi','modal')

Yardım diyalog penceresinin ekran görüntüsü Şekil 5.84'de gösterilmiştir.



Şekil 5.84

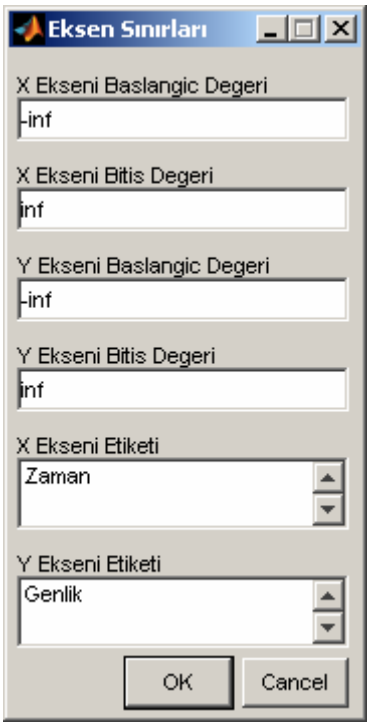
5.14.3 Veri Giriş Penceresi (Input Dialog) :

Bu diyalog penceresi, kullanıcılardan bir veya birden fazla değerin aynı anda alınması amacıyla kullanılır. Kullanımı şu şekildedir:

```
yazi_ifadeleri = { 'X Ekseni Baslangic Degeri', 'X Ekseni Bitis Degeri', ...
    'Y Ekseni Baslangic Degeri', 'Y Ekseni Bitis Degeri' ,'X Ekseni Etiketi', 'Y Ekseni Etiketi'
};
baslik = 'Eksen Sınırları';
satir_adedi = [1 1 1 1 2 2];
varsayilan_degerler = { '-inf', 'inf', '-inf', 'inf' , 'Zaman', 'Genlik'};
diyalog_donus_degeri = inputdlg (yazi_ifadeleri, baslik, satir_adedi, varsayilan_degerler)
```

bu parametrelerden “satir_adedi” parametresi açılacak diyalog penceresinde yer alan her bir text kutusunun sahip olacağı toplam satır adedini gosterir. Örnek kullanımda ekran görüntüsünde X ve Y eksenlerine ait etiket değer alanlarının 2 satırdan oluştuğu görülmektedir. Bu parametre her bir text kutusu için tanımalandığından matris şeklinde bir yapıya sahiptir. Diğer bir parametre olan “varsayilan_degerler” diyalog penceresi ilk görüntülendiğin text kutularının içinde olması gereken başlangıç değerlerini gösterir. “baslik” parametresi diyalog pencerensin caption değeri için ve yazi_iafadeleri her bir text kutusu ile ilgili bilgileri ekranda göstermek için kullanılmıştır. “yazi_ifadeleri” ve “varsayilan_degerler” isimli parametrelerin hücre dizisi şeklinde tanımlanaması gerektiğine dikkat edilmelidir (“ { ” ve “ } ” simgeleri hücre dizisi tanımlamalarında kullanılır.).

Yukarıda verilen örnek kullanıma ait veri giriş diyalog penceresi için ekran görüntüsü Şekil 5.85’te gösterilmiştir.



Şekil 5.85

Bu diyalog penceresinden geri dönüş değeri kaç tane bilgi giriş kutusu varsa, bu uzunlukta dizi şeklinde döner. Örnek olarak "y_ekseni_bitis_degeri" nin elde etmek için

y_ekseni_bitis_degeri = diyalog_donus_degeri (4)

şeklinde kullanım söz konusudur. Ancak, kullanıcı bu diyalog kutusunu "Kapat" (X simgesi) ya da "Cancel" butonlarından biri ile gönderirse boş bir hücre dizisi döner. . Eğer böyle bir durum kontrol edilecekse "isempty" fonksiyonu kullanılabilir. Bu fonksiyon bir değişken içeriği boş ise "true", dolu ise "false" üretir. "isempty(degisken_ismi)" şeklinde kullanıma sahiptir.

5.14.4 Liste Görünüm Penceresi (List Dialog) :

Bu diyalog penceresi ile kullanıcıların karşısına gelen bir listeden bir veya birden fazla liste elemanı seçmesi sağlanılşabilir. Uygulamada genellikle dosya veya dizinlerin listelenmesi ve seçilmesi amacıyla kullanılır. Örnek bir kullanımı şu şekildedir:

```
dosyalar = dir;                                % aktif dizin yolu üzer. yer alan dosyal. list. alma
 dosya_isimleri = {dosyalar.name};             % dosya_isimleri matrisine dosya isiml. atanması
 [s,v] = listdlg('PromptString',Bir dosya seçiniz : ',...      % liste diyalog pencerensin görünt.
            'SelectionMode','single',...
            'ListString',dosya_isimleri)
```

Dosyalar değişkenine "dir" fonksiyonu vaıtası ile her bir elemanı yapı dizisi olan bir dosya listesi gelir. Her bir yapı dizisinde

- name : her bir dosyanın ismini tutmak için,
- date : her bir dosyanın geçerli tarihini tutmak için,
- bytes : her bir dosyanın boyut bilgisini tutmak için,
- isdir : her bir dosyanın bir klasör olup olamdığı tutulur. Değer "true" ise bu öğe bir dizindir.

alanları bulunmaktadır. Örneğin 5. öğenin tarih bilgisi öğrenilmek istenirse
dosyalar(5).date
komut yapısı kullanılmalıdır.

Örnek kullanımda seçim modu olarak "single" seçeneği kullanılmıştır. Yani, liste elemanlarından sadece bir tanesi seçilebilir. Eğer, çoklu seçim isteniliyorsa "multiple" seçeneği kullanılabilir. Çoklu seçimde kullanıcı birden fazla öğeyi seçebilmek için öğrelrin seçimi sırasında klavyenin "Ctrl" tuşunu basılı tutmalıdır.

Liste görünüm diyalog penceresinden geri dönüş değeri olarak sadece seçilen liste elemanlarının numara bilgisi sütun vektor seklinde (burada örnek kullanım için) s değişkenine atanır. Bu atama sırasında v değişkenine daima 1 değeri atanır. Örneğin 15, 25 ve 36 numaralı liste öğreleri seçildiği düşünülürse

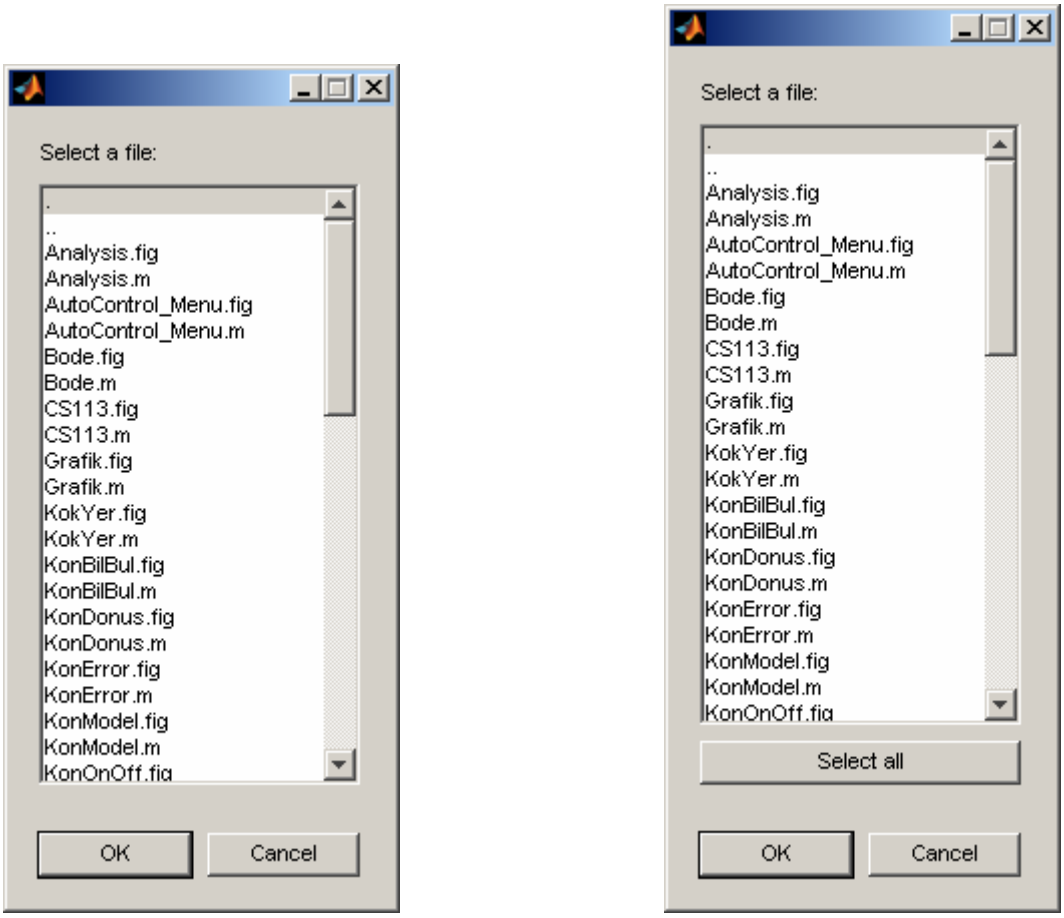
- s değişkeninin içeriği [15 25 36]
- v değişkeninin içeriği 1

olacaktır. Eğer kullanıcı bu diyalog penceresini “Cancel” veya “Kapat” (X simgesi) düğmelerini kullanarak kapatırsa geri dönüş değerleri

- s değişkeninin içeriği [] (yani boş matris)
- v değişkeninin içeriği 0

şeklinde olacaktır. S değişkeninin boş olup olmama durumu “isempty” fonksiyonu ile kontrol edilebilir. Örnek olarak isempty(s) komutu sonucu true (veya 1 değerinde) ise s değişkeninin içeriği boş demektir.

Liste görünüm diyalog penceresine ait “single” ve “multiple” modları için ekran görüntüleri Şekil 5.86’da gösterilmiştir.



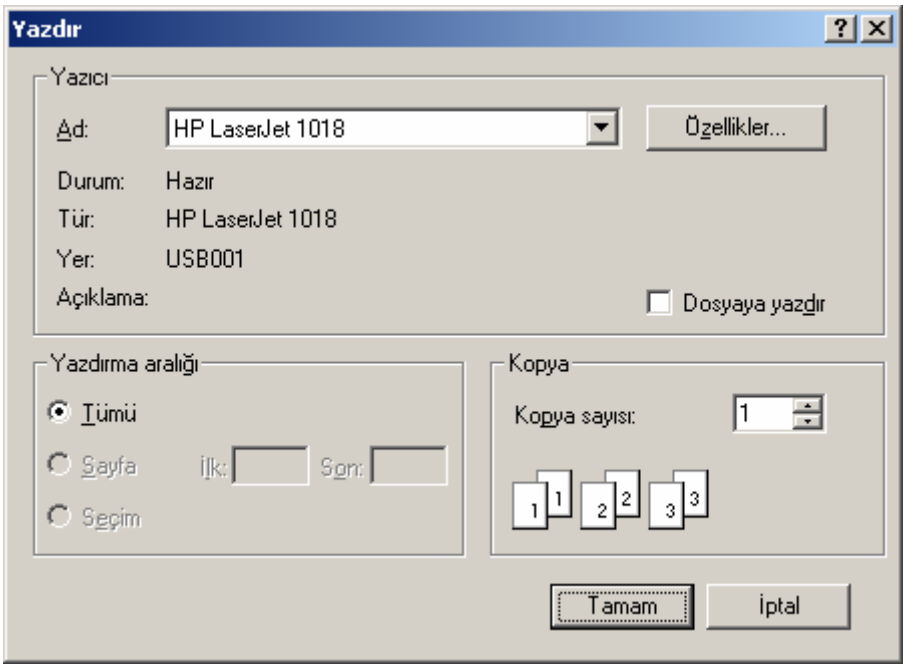
Şekil 5.86

5.14.5 Yazdırma Penceresi (Print Dialog) :

Bu diyalog penceresi, aktif olan figure alanının veya bir grafik çiziminin yazıcıdan direk çıktı alınmasını sağlar. Sayfa konumu diyalog penceresinden farkı dökümanı yazdırmadan önce herhangi bir ayar yapılmamasıdır. Ayrıca, sayfa konumu diyalog penceresinin önceden yapmış olduğu ayarları kullanarak çıktı alınması sağlar. Kullanımı şu şekildedir:

```
printdlg
```

Yazdırma diyalog penceresinin ekran görüntüsü Şekil 5.87’de gösterilmiştir.



Şekil 5.87

Şekil 5.87'deki ekranda kullanıcı "Tamam" butonunu tıkladığında yazdırılmak üzere döküman yazcıya gönderilir ve yazdırılır.

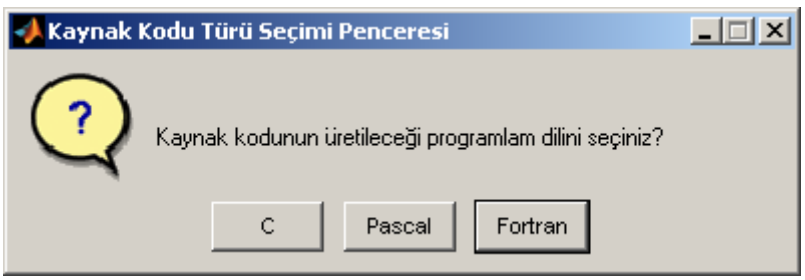
5.14.6 Sorgu Penceresi (Question Dialog) :

Bu diyalog penceresi ile uygulama sırasında kullanılacak verilerin çeşitli seçenekler içerisinden birinin seçilerek kullanıcılardan alınması sağlanır. Kullanımı şu şekildedir:

```
sorgu_sonucu = questdlg ('Kaynak kodunun üretileceği programlam dilini seçiniz?',
                         'Kaynak Kodu Seçimi Penceresi'
                         'C', 'Pascal', 'Fortran', 'Fortran')
```

Yukarıda gösterilen kullanımda son parametre varsayılan olarak seçili olacak seçeneği gösterir. Son parametre bu parametreden bir önceki üç parametreden biri olmalı veya '' şeklinde boş bırakılmalıdır. Eğer kullanıcı diyalog penceresi herhangi bir butona basmadan ve X butonu kullanarak kapatılırsa geri dönüş değeri boş bir dizi şeklindedir, yani '' şeklinde olacaktır. Eğer böyle bir durum kontrol edilecekse "isem pty" fonksiyonu kullanılabilir. Bu fonksiyon bir değişken içeriği boş ise "true", dolu ise "false" üretir. "isempty(degisken_ismi)" şeklinde kullanıma sahiptir.

Sorgu diyalog penceresinin ekran görüntüsü Şekil 5.88'de gösterilmiştir.



Şekil 5.88

Sorgu penceresinden dönen değer basitçe aşağıda gösterilen switch case yapısı ile kontrol edilebilir.

```
switch sorgu_sonucu
        case ‘C’
        case ‘Pascal’
        case ‘Fortran’
end
```

5.14.7 Renk Seçim Penceresi (Color Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında herhangi bir rengi seçmelerine imkân vermek amacıyla kullanılır. Kullanımı şu şekildedir:

```
donus_rengi = uisetcolor
```

Kullanıcı eğer “Cancel” butonu ile bu diyalog penceresi kapatırsa, yani bu diyalog penceresinden herhangi bir renk seçilmez ise “donus_rengi” değişkenine 0 (sıfır) değeri atanır. Örneğin kullanıcı kırmızı rengi seçerse “donus_rengi” değişkeninin içeriği [1 0 0] şkelinde olacaktır. Burada 3 boyutlu sütun vektör yapılı bir matris değişkeni şeklinde değerler döner. Bu matrisin sırasıyla elemanları kırmızı (red, R), yeşil (green, Y) ve mavi (blue, B) renklerinin değerlerini verir. Ancak, bu RGB değerlerinin her biri 0-255 yerine 0-1 arası değerler alır. Uygualamda 1 ile 255 arası oranlama şeklinde gerçek desimal formatta RGB değeri elde edilebilir.

Renk seçim diyalog penceresinin ekran görüntüsü Şekil 5.89’da gösterilmiştir.

Şekil 5.89

5.14.8 Font Seçim Penceresi (Font Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında herhangi bir fontu (yazı tipini) seçmelerine imkân vermek amacıyla kullanılır. Kullanımı şu şekildedir:

secilen _font_bilgisi = uisetfont

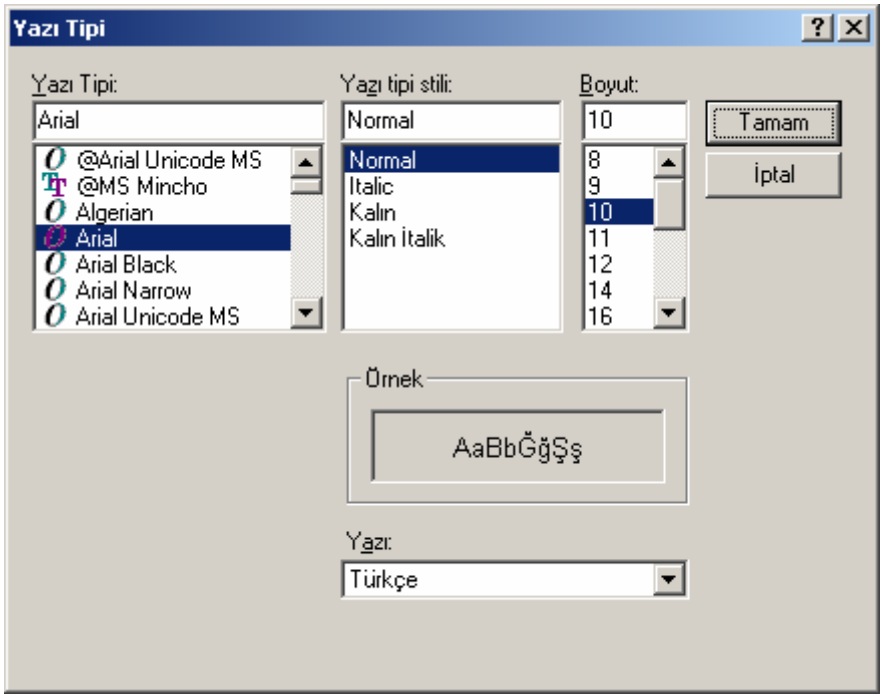
Kullanıcı eğer diyalog penceresi "İptal" butonunu kullanarak kapatmışsa geri dönüş değeri 0 (sıfır) olur. Eğer kullanıcı bir fontu seçip "Tamam" butonuna tıklayarak font seçim penceresini kapatmış ise "secilen_font_bilgisi" değişkenine

- Fontname : font ismi string bilgisi (örneğin 'Arial')
- FontUnits : font birimi string bilgisi (örneğin 'points')
- FontSize : font boyutu sayısal bilgisi (örneğin 10)
- FontWeight : font ağırlığı string bilgisi ('normal' veya 'bold' olabilir.)
- FontAngle : font açısı string bilgisi ('normal' veya 'italic' olabilir.)

alanları olan bir yapı değişkeni döner. Örneğin seçilen fontun boyutunu kullanmak için şu şekilde bir komut yapısı kullanılmalıdır:

font_boyutu = secilen_font_bilgisi. FontSize % "FontSize" ismindeki büyük ve küçük
% harflere dikkat edilmelidir.

Font seçim diyalog penceresinin seçilmiş bir font ile birlikte ekran görüntüsü Şekil 5.90'da gösterilmiştir.



Şekil 5.90

5.14.9 Uyarı Penceresi (Warn Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında eksik veya yanlış bilgi girme ya da herhangi bir konuda bilgilendirilme amacıyla kullanılır. Kullanımı şu şekildedir:

warndlg(‘Sistem kaynakları kritik seviyededir.’, ‘Tehlike’, ‘modal’)

Uyarı diyalog penceresinin ekran görüntüsü Şekil 5.91’de gösterilmiştir.

Şekil 5.91

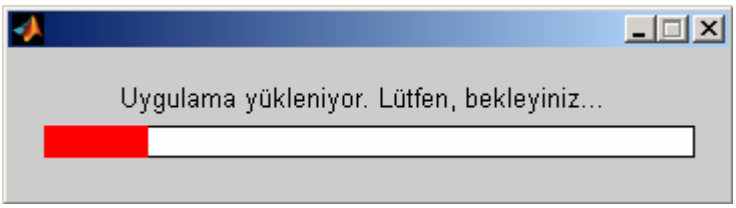
Modal parametresi ile kullanıcının bu pencereye cevap vermeden işlemlerine devam etmesi engellenmiş olmaktadır. Bu seçenek belirtilmez veya ‘non-modal’ seçeneği kullanılırsa modal özelliği devre dışı bırakılmış olur.

5.14.10 Yükleme Çubuğu (waitbar) :

Yükleme çubuğunu programcı bir uygulamanın başında kullanarak kullanıcıya uygulamanın yüklenmekte olduğunu ve ne kadarının yüklendiği görsel bir şekilde sunabilir. Kullanımı şu şekildedir:

```
yukleme_cubugu = waitbar ( 0 , ‘Uygulama yükleniyor. Lütfen, bekleyiniz…’ ) ;
for i=1:1:100              % bir üstteki satır ile waitbar nesnesi oluşturuluyor.
        % uygulamanın yüklenmesi sırasında yer alan işlemler
        waitbar ( i / 100 , yukleme_cubugu ) ; % oluşturulmuş waitbar nesnesinin günc.
end
close (yukleme_cubugu)                    % oluştur. olan waitbar nesnesin. silinmesi
```

Yükleme çubuğunun ekran görüntüsü Şekil 5.92’de gösterilmiştir.



Şekil 5.92

5.14.11 Klasör Yolu Penceresi (UIGetDir Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında bir dizin yolunu seçmeleri amacıyla kullanılır. Kullanım şekilleri çeşitlidir. Bunlar aşağıda gösterilmiştir.

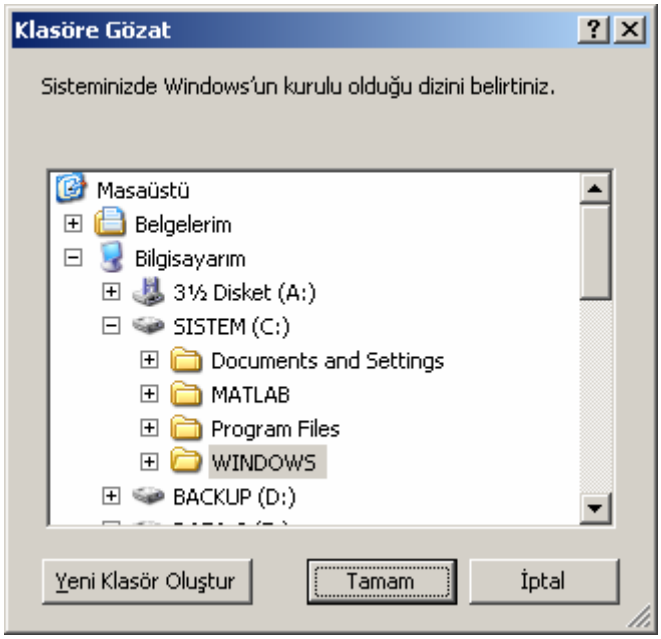
```
klasor_ismi = uigetdir
klasor_ismi = uigetdir (‘baslangic_dizin_yolu’)
klasor_ismi = uigetdir (‘baslangic_dizin_yolu’,’goruntulenecek_mesaj_stringi’)
```

Kullanımlarda “baslangic_dizin_yolu’” parameresi diyalog penceresi açıldığında ilk görüntülenecek klasörün seçilmesini ve son parametre de bu pencere ekstra görüntülenmesi istenilen mesajın çıkmasını sağlar.

Örnek olarak aşağıdaki kullanım ile bir diyalog kutusu ekrana gösterilsin.

klasor_ismi = uigetdir (‘C:\Windows’ , ‘Sisteminizde Windows’un kurulu olduğu dizini belirtiniz.’)

Örnek kullanıma ait klasör yolu diyalog penceresinin ekran görüntüsü Şekil 5.93’te gösterilmiştir.



Şekil 5.93

Bu diyalog kutusu iptal düğmesi tıklanarak kapatılırsa 0 sayısı (false cevabı) üretir. Eğer tamam butonu tıklanırsa seçilmiş olan klasöre ait tam yolu (directory path’i) geri dönüş değeri olarak gönderilir.

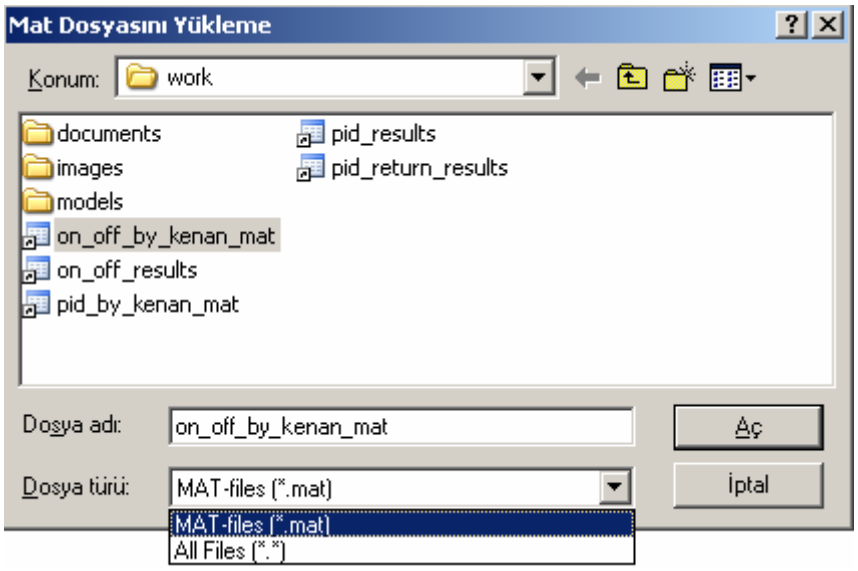
5.14.12 Dosya Açma Penceresi (UIGetFile Dialog) :

Bu diyalog penceresi ile kullanıcıdan bilgisayarda yer alan ve belirlenmiş türde dosyaların seçilerek açılması ve seçilen dosyaya ait path tanımının GUI uygulamasına aktarılması amacıyla kullanılır. Kullanımı şu şekildedir:

[dosya_ismi, dosya_yolu] = uigetfile (‘veriler.mat’ , ‘Mat Dosyasını Yükleme’)

Bu komut yapısında ekrana gelecek dosya açma diyalog penceresinin başlığı “Mat Dosyasını Yükleme” ve varsayılan dosya uzantısı olarak “.mat” olan dosyaların seçilmesi sağlanılacaktır. Bu örneğe ilişkin ekran görüntüsünde de görüldüğü üzere kullanıcı isterse *.* formatlı olmak üzere bu diyalog penceresinde dosya türünü “Tüm Dosyalar” olarak seçebilir.

Örnek olarak verilen komut yapısının işletilmesi sonucu dosya açma diyalog penceresinin ekran görüntüsü Şekil 5.94’teki gibi olacaktır.



Şekil 5.94

Kullanıcı örnek olarak verilen komutun sonucunda açılan diyalog penceresini "İptal" butonuan basarak ya da herhangi bir dosya seçmeden kapatırsa "dosya_ismi" ve "dosya_yolu" değişkenlerine 0 (sıfır) (yani mantıksal false) değeri atanır. Ancak, kullanıcı bu pencere yardımıyla bir dosyayı seçer ve "Aç" butonunu tıklayarak pencereyi kapatırsa "dosya_ismi" değişkenine seçilen dosyanın ismi ve "dosya_yolu" değişkenine bu dosyaya ait path tanımı (dizin yolu) string tipte olarak atanacaktır. Örnek olarak ekran görüntüsünde verilen "on_off_by_kenan_mat.mat" dosyası seçilmiş olduğu düşünülürse

- "dosya_ismi" değişkeninin içeriği 'on_off_by_kenan_mat.mat' string bilgisi
- "dosya_yolu" değişkeninin içeriği 'C:\MATLAB\work\' string bilgisi

olacaktır. Bu örnekte dosya açma ve Workspace'a değerlerin yüklenmesi ile desteklenirse aşağıdaki deyimler kullanılabilir.

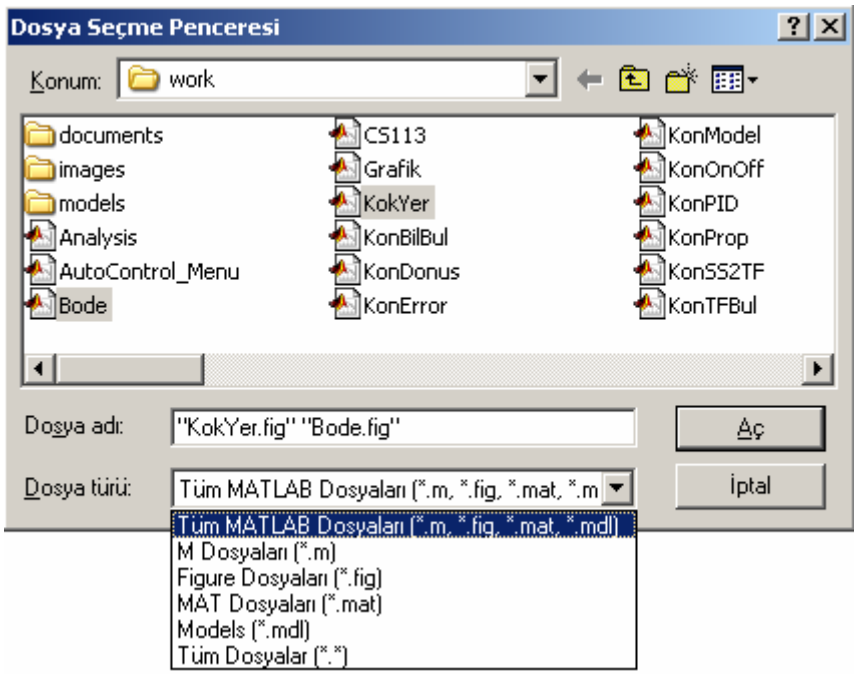
```
[dosya_ismi, dosya_yolu] = uigetfile ( 'veriler.mat' , 'Mat Dosyasını Yükleme' ) ;
if isequal ( dosya_ismi , 0)
        load ( [ dosya_yolu dosya_adi ] );
end
```

Ayrıca, kullanıcının seçmesi istenilen dosya türleri çok çeşitli ise (örneğin resim dosyaları gibi) bu durumda birden fazla dosya formatı olacak şekilde dosya açma diyalog penceresi için filtre formatlar tanımlanabilir. Çoklu dosya formatı tanımlamakla ilgili aşağıdaki örnek incelenebilir. Ayrıca, bu örnekte çoklu dosya seçimi ve yönetilmesi de gösterilmiştir.

```
[dosya_ismi, dosya_yolu, secilen_filtre_no] = uigetfile( ...
   {'*.m;*.fig;*.mat;*.mdl', 'Tüm MATLAB Dosyaları (*.m, *.fig, *.mat, *.mdl)';
   '*.m',  'M Dosyaları (*.m)'; ...
   '*.fig','Figure Dosyaları (*.fig)'; ...
   '*.mat','MAT Dosyaları (*.mat)'; ...
   '*.mdl','Models (*.mdl)'; ...
   '*.*',  'Tüm Dosyalar (*.*)'}, ...
   'Dosya Seçme Penceresi', ...
```

'MultiSelect', 'on')

Yukarıda verilen komut satırında “secilen_filtre_no” geri dönüş parametre değişkenine kullanıcı dosya açma diyalog penceresinin Dosya türü listesinden hangi filtreyi seçmiş ise bu liste öğesinin değeri numarası atanır. Seçilen öğe listenin ilk elemanı ise bu değişkenin içeriği 1 olacaktır. Yukarıdaki komut satırlarının işletilmesi ile Şekil 5.95’te yer alan ekran görüntüsü elde edilir.



Şekil 5.95

Bir dosya açma penceresi varsayılan olarak tek dosya seçmek üzere açılır. Birden fazla dosyanın seçildiği durumlar için ise “uigetfile” komutunun son iki parametresi ‘MultiSelect’,’on ’ string bilgileri olmalıdır. Bu özellik aktif edildiği takdirde kullanıcı klavyenin “Ctrl” tuşu basılı halde birden fazla dosyayı seçebilir. Örneğin ekran görüntüsünde yer alan “Bode.fig” ve “KokYer.fig” dosyalrı seçilmiş ve “Aç” butonuna basılmış olur. Bu durumda geri dönüş parametrelerin içerikleri şöyle olacaktır:

- “dosya_ismi” değişkeninin içeriği [‘Bode.fig’ ‘KokYer.fig’]
- “dosya_yolu” değişkeninin içeriği ‘C:\MATLAB\work\’

Burada örneğin 2. dosya ismi şu komut satırıyla elde edilir:

secilen_ikinci_dosya_ismi = dosya_ismi(2)

Eğer herhangi bir dosya seçilmezse her iki geri dönüş parametresine 0 (sıfır) değeri atanır.

5.14.13 Dosya Kaydetme Penceresi (UIPutFile Dialog) :

Bu diyalog penceresi ile kullanıcıdan bilgisayara kaydedilecek bir dosyanın yeri ve isminin belirlenmesi amacıyla kullanılır. Dosya aç iletişim kutusu ile aynı özelliklere sahiptir. (Dosya açma iletişim kutusu hakkında daha ayrıntılı bilgi için bir önceki konu başlığında bakılabilir.) Ancak, bu diyalog penceresi ile dosya açma diyalog penceresi arasındaki fark

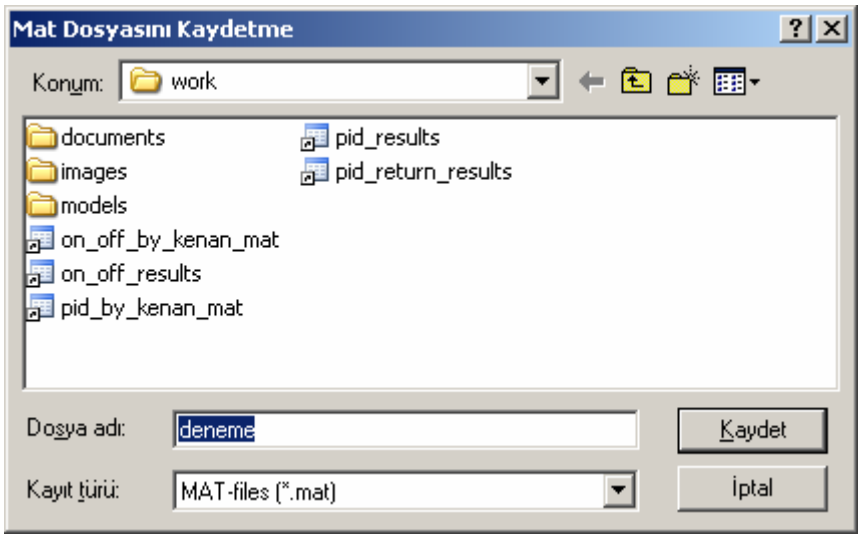
uigetfile komutu yerine uiputfile komutunun kullanılması ve dosya kaydetme kavramları içerisinde “multiselect” (yani çoklu seçim) gibi bir seçeneğin olmamasıdır. Kullanımı şu şekildedir:

[dosya_ismi, dosya_yolu] = uiputfile (‘*.mat’ , ‘Mat Dosyasını Kaydetme’)

Bu komut yapısında ekrana gelecek dosya açma diyalog penceresinin başlığı “Mat Dosyasını Yükleme” ve varsayılan dosya uzantısı olarak “.mat” olacak şekilde dosyanın kaydedilmesi sağlanır. İstenirse ‘*.mat’ parametresi boş bırakılabilir, yani bu parametre ‘’ şeklinde tanımlanabilir. Varsayılan olarak buraya girilen bir dosya ismi dosya kaydetme penceresi açıldığında görülecektir. Örneğin aşağıdaki komut çalıştırılmış olsun. Bu komuta ait ekran görüntüsü de aşağıda sunulmuştur.

[dosya_ismi, dosya_yolu] = uiputfile (‘deneme.mat’ , ‘Mat Dosyasını Kaydetme’)

Örnek olarak verilen komut yapısının işletilmesi sonucu dosya açma diyalog penceresinin ekran görüntüsü Şekil 5.96’daki gibi olacaktır.



Şekil 5.96

Yukarıdaki pencerede “deneme” ve “MAT-Files” filtresi ile “kaydet” butonu tıklandığında geri dönüş parametreleri

- “dosya_ismi” değişkeni içinde ‘deneme.mat’ string bilgisi,
- “dosya_yolu” değişkeni içinde ‘C:\MATLAB\work\’ string bilgisi

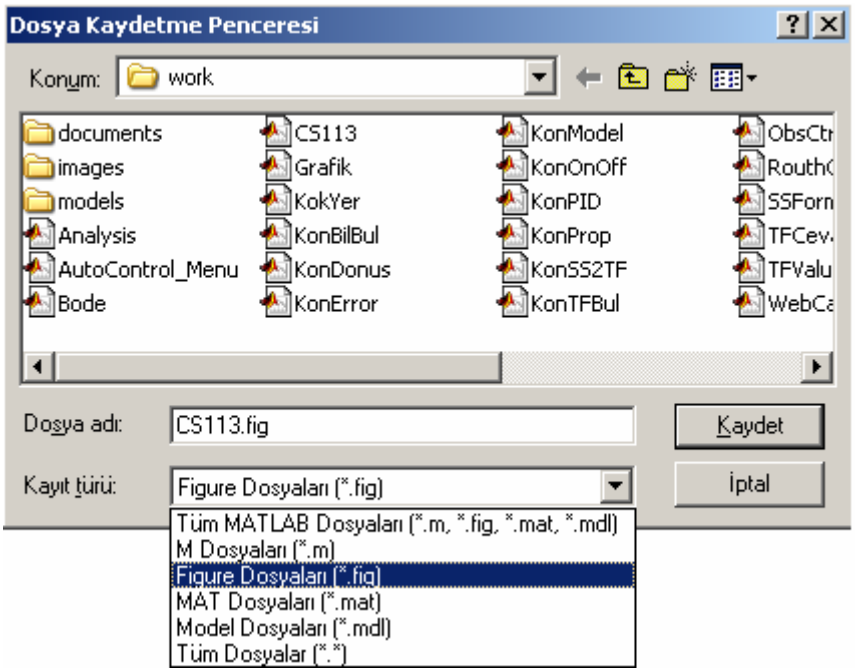
şeklinde olacaktır.

Aşağıdaki örnek birden fazla dosya formatının nasıl tanımlandığı, yani filtrelemenin nasıl yapıldığı konusunda bilgi vermektedir.

```
[dosya_ismi, dosya_yolu, secilen_filtre_no] = uiputfile( ...
   {'*.m;*.fig;*.mat;*.mdl', 'Tüm MATLAB Dosyaları (*.m, *.fig, *.mat, *.mdl)';
    '*.m',  'M Dosyaları (*.m)'; ...
    '*.fig','Figure Dosyaları (*.fig)'; ...
    '*.mat','MAT Dosyaları (*.mat)'; ...
```

'*.mdl','Model Dosyaları (*.mdl)'; ...
'*.*', 'Tüm Dosyalar (*.*)'}, ...
'Dosya Kaydetme Penceresi')

Yukarıda verilen komutun çalıştırılması sonucu Şekil 5.97'deki ekran görüntüsü gelecektir.



Şekil 5.97

Şekil 5.97'deki ekranda CS113.fig dosya ismi verilmiş ve filtre tipi olarak listenin 3. sırasında yer alan "Figure Dosyaları" işaretlenmiştir. Kullanıcı "Kaydet" butonunu tıkladığında geri dönüş parametreleri şöyle olacaktır:

- "dosya_ismi" değişkeni içinde 'CS113.fig' string bilgisi,
- "dosya_yolu" değişkeni içinde 'C:\MATLAB\work\' string bilgisi
- "secilen_filtre_no" değişkeni içinde 3 sayısal bilgisi

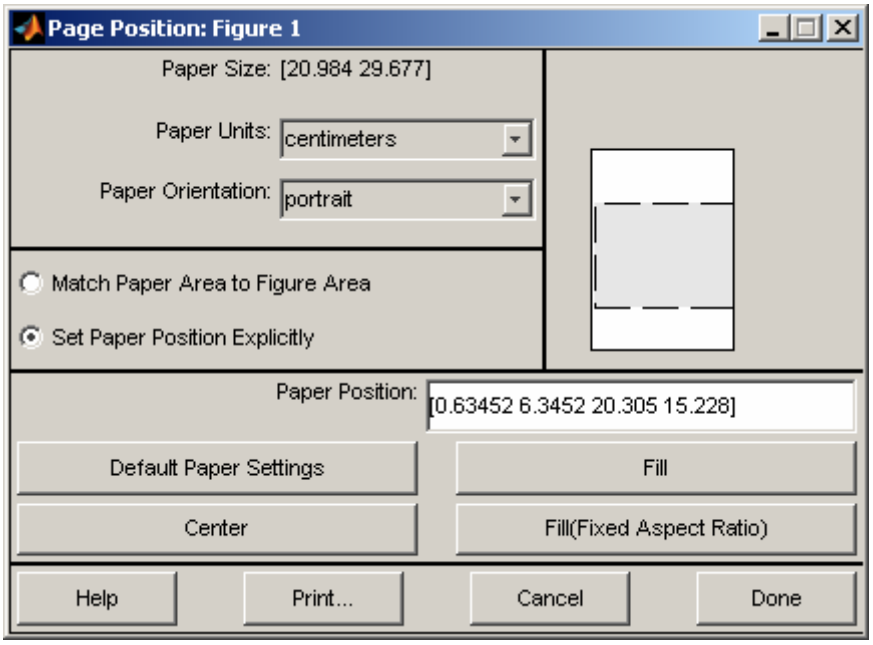
Burada seçilen filtre türlerinden 3. sıradaki filtre seçildiği için "secilen_filtre_no" değişkenine 3 sayısal değeri atanmıştır.

5.14.14 Sayfa Yapısı Penceresi (Page Dialog) :

Bu diyalog penceresi, yazıcıdan çıktı alınacak döküman ile ilgili ayarların yapılması nı ve ayrıca aktif olan figure alanının veya bir grafik çiziminin yazıcıdan çıktı alınmasını sağlar. Kullanımı şu şekildedir:

pagedlg

Sayfa yapısı diyalog penceresinin ekran görüntüsü Şekil 5.98'de gösterilmiştir.



Şekil 5.98

Kullanıcı bu diyalog penceresinde sayfa ile ilgili gerekli ayarlamaları yaptıktan sonra "Print" butonunu kullanarak aktif figure alanı veya grafik çiziminin yazıcıdan çıktı alabilir.

5.14.15 Mesaj Kutusu (MessageBox Dialog) :

Bu diyalog penceresi kullanılarak kullanıcılara herhangi bir mesaj istenilen bir resim ya da ikon dosyası ile birlikte gösterilebilir. Çok çeşitli kullanımlara sahip bir diyalog penceresi olup, aşağıda bu kullanım çeşitleri gösterilmiştir.

```
msgbox (mesaj)                % Şekil  …...ya bakınız.
msgbox (mesaj, baslik)        % Şekil  …...ya bakınız.
msgbox (mesaj, baslik, ikon) % Şekil  …...ya bakınız.
msgbox (mesaj, baslik, 'custom', resim_data_degiskeni, resim_colormap_degiskeni)
msgbox (mesaj, ……., olusturulma_modu)
```

Bu kullanımlardaki parametrelerin tipi ve görevleri şu şekildedir:

- mesaj : Mesaj kutusunda gözükecek string bilgi
- baslik : Mesaj kutusunun pencere başlığında gözükmesi istenilen string bilgi
- ikon : 'none', 'error', 'help', 'warn' veya 'custom' değerlerinden biri olabilir. Varsayılan değeri
  - 'none' olup, Matlab'in kendi içinde kullandığı hazır standart ikonlkarın gösterilmesi
  - bu parametre ile saplanır. Bu paramerenin kullanımı ile ilgili örnek ekran
  - görüntğlerine aşağıdan bakılabilir.
- resim_data_degiskeni : import edilen bir resim dosyasının veri (data) dizis. tutan değiş.
- resim_colormap_degiskeni : import edilen bir resim dosyasının colormap dizis. tutan değiş.
- olusturulma_modu : Bu parametre msgbox komutuna gönderilecek son

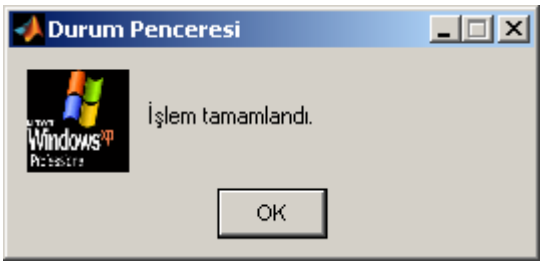
parametre bilgisi olmalıdır. ‘modal’ veya ‘non-modal’ string bilgilerinden biri olabilir. Varsayılan değer ‘non-modal’ seçeneğidir. Eğer modal seçilirse kullanıcı mesaj kutusuna cevap vermeden arka plandaki uygulamaya dönemz, yani arkapalnın kilitlenmesi sağlanır. ‘non-modal’ seçeneği ise bu durumun tam tersidir.

msgbox (mesaj, baslik, ‘custom’, resim_data_degiskeni, resim_colormap_degiskeni)

Yukarıdaki kullnım şekli için için öncelikle bir resmin “imread” komutu ile Matlab’in “Workspace” alanına yüklenmesi ve bu fonksiyonun geri dönüş parametrelerinin daha sonra kullanılması gereklidir. Örnek kullanım için aşağıdaki komut satırları incelenebilir.

```
[resim_data_degiskeni, resim_colormap_degiskeni] = imread (‘C:\Windows\winnt256.bmp’);
mesaj = ‘İşlem tamamlandı.’;
baslik = ‘Durum Penceresi’ ;
msgbox (mesaj, baslik, ‘custom’, resim_data_degiskeni, resim_colormap_degiskeni)
```

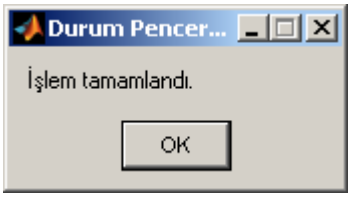
Yukarıda verilen komut satırıçalıştırıldığında Şekil 5.99’daki gibi bir ekran görüntüsü oluşacaktır.



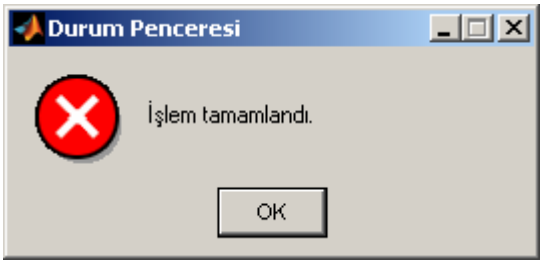
Şekil 5.99

Çeşitli mesaj kutularına ait ekran görüntüleri Şekil 5.100, Şekil 5.101, Şekil 5.102, Şekil 5.103 ve Şekil 5.104’te verilmiştir.

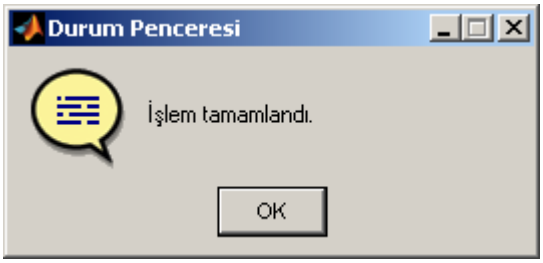
Şekil 5.100



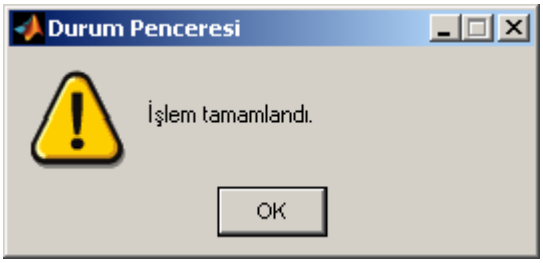
Şekil 5.101



Şekil 5.102



Şekil 5.103



Şekil 5.104

5.14.16 Sayfa Önizleme Penceresi (PrintPreview Dialog) :

Bu diyalog penceresi ile kullanıcılar yazıcıya aktif figure alanını veya aktif axes (grafik çizim nesnesi) içeriğini göndermeden önce sayfanın önizlemesini görebilirler. Ayrıca, sayfa ile ilgili ayarlamaları yapabilir ve sayfanın yazıcıdan çıktısını alabilir. Kullanımı şu şekildedir:

```
printpreview
```

Sayfa önizleme diyalog penceresinin örnek bir uygulama için ekran görüntüsü Şekil 5.105'te gösterilmiştir.