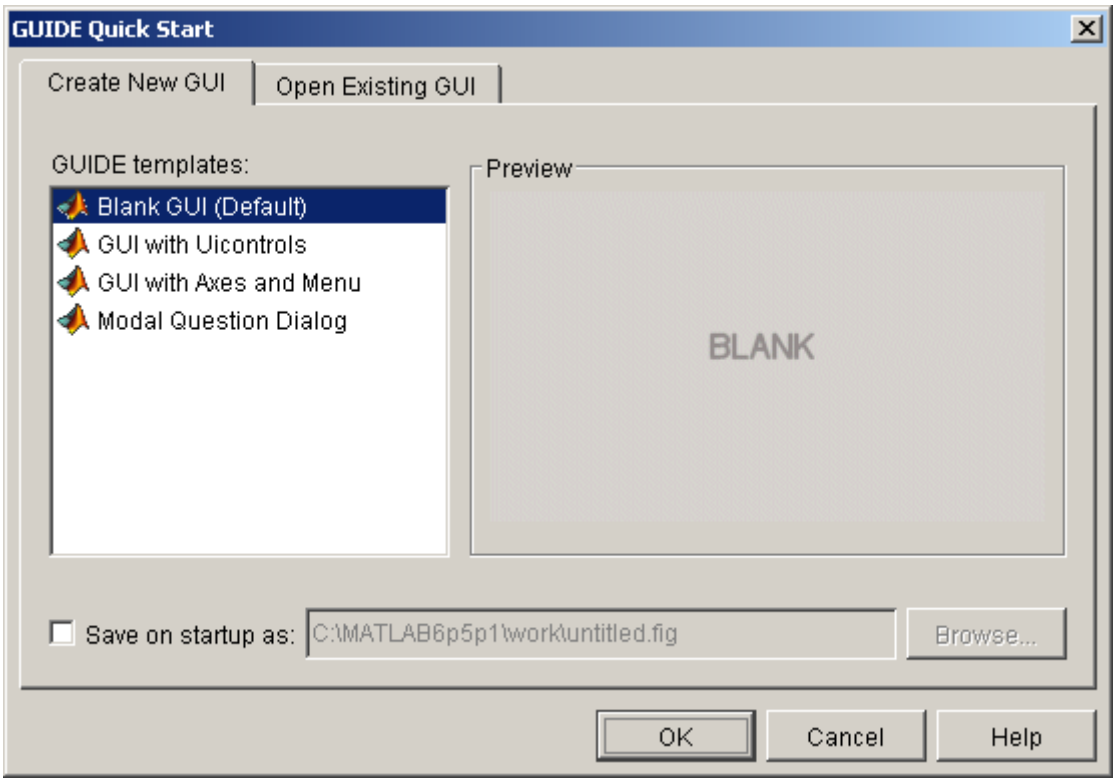
Wstęp do GUI w Matlabie.

GUI (Graphical User Interface) jest systemem ułatwiającym pracę w Matlabie. Umożliwia on użytkownikowi automatyczne wydawanie wielu poleceń (np. poprzez naciśnięcie odpowiedniego klawisza), a także ułatwia pracę osobom, które słabo znają Matlaba i jego komendy. Projektowanie GUI jest także pierwszym krokiem do tworzenia wolno stojących aplikacji.

W Matlabie do projektowania GUI służy program GUIDE (Graphical User Interface Development Enviroment). Uruchamia się go z Matlaba wpisując w oknie „Command Window" komende

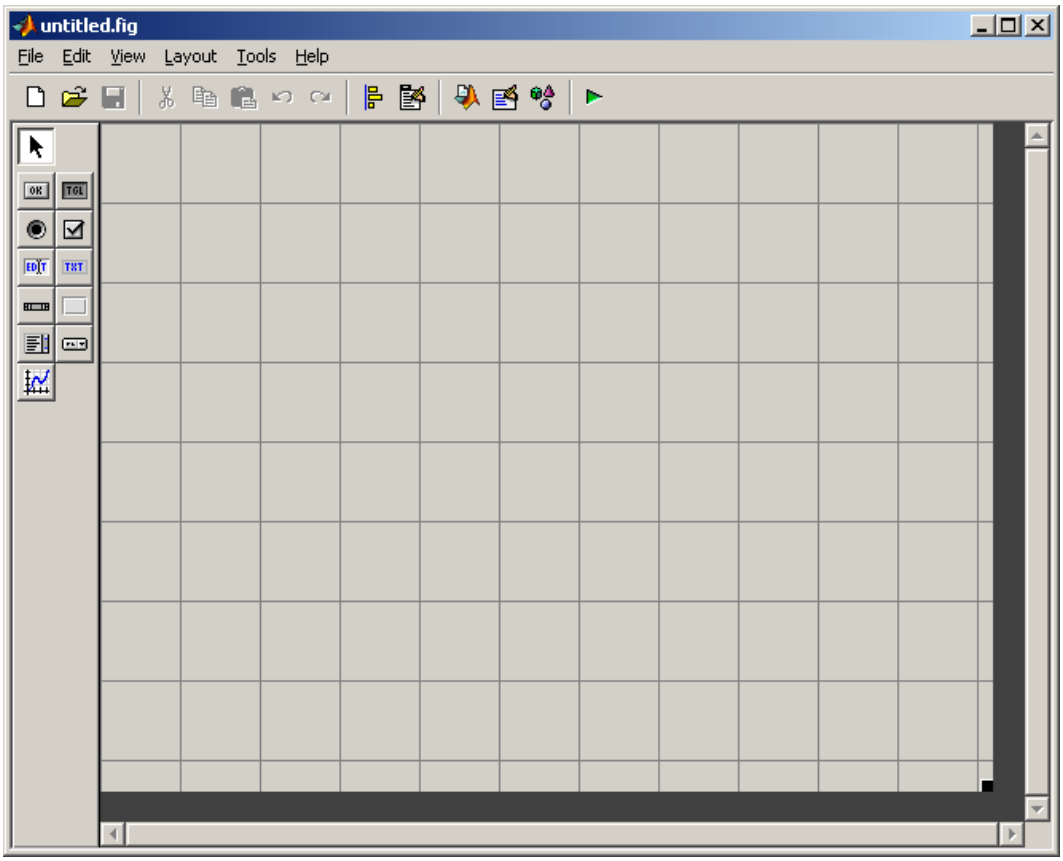
guide

Na ekranie powinno pojawić się następujące okno



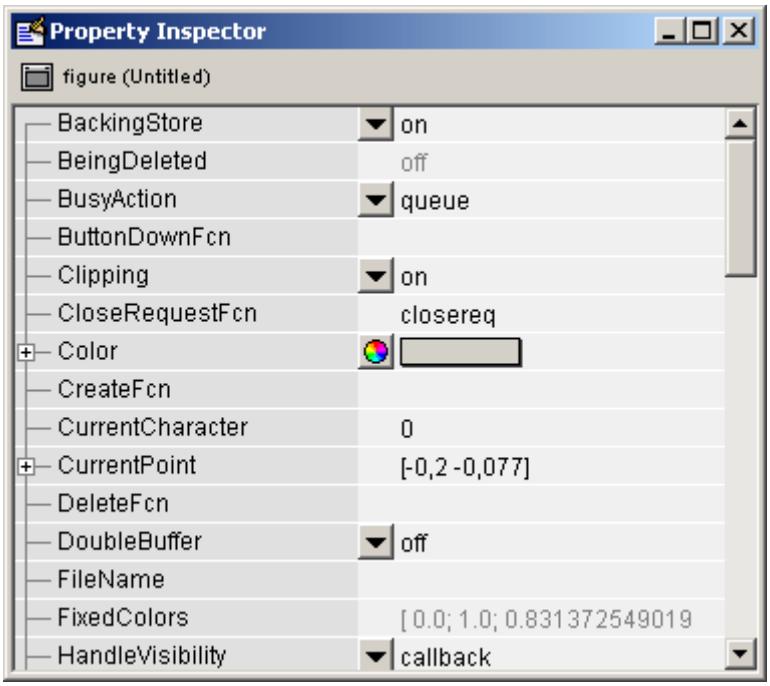
W celu stworzenia tego GUI wybieramy opcję „Blank GUI (Default)" i klikamy OK.

W nowo otworzonym oknie możemy tworzyć już wygląd naszej aplikacji

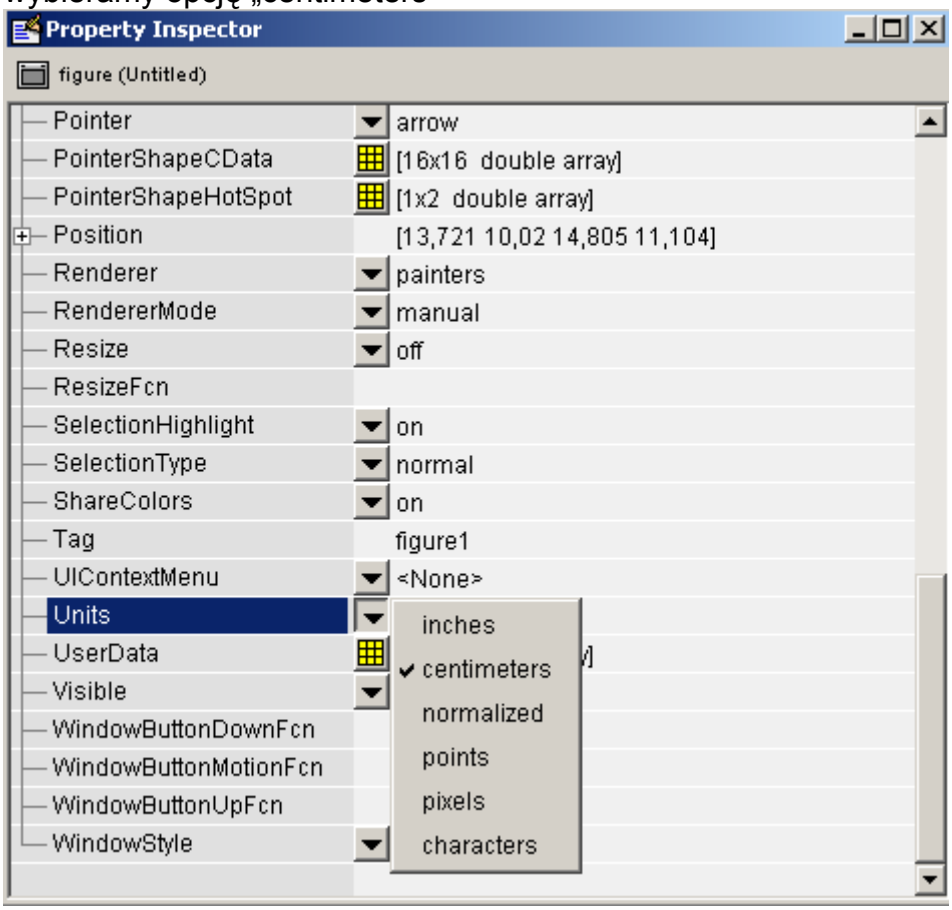


W centrum (część zakratkowana) znajduje się obszar wyświetlany po uruchomienu aplikacji.
Klikając dwukrotnie w dowolnym miejscu pojawia się okno „Property Inspector”. (Inne możliwości uruchomienia to:

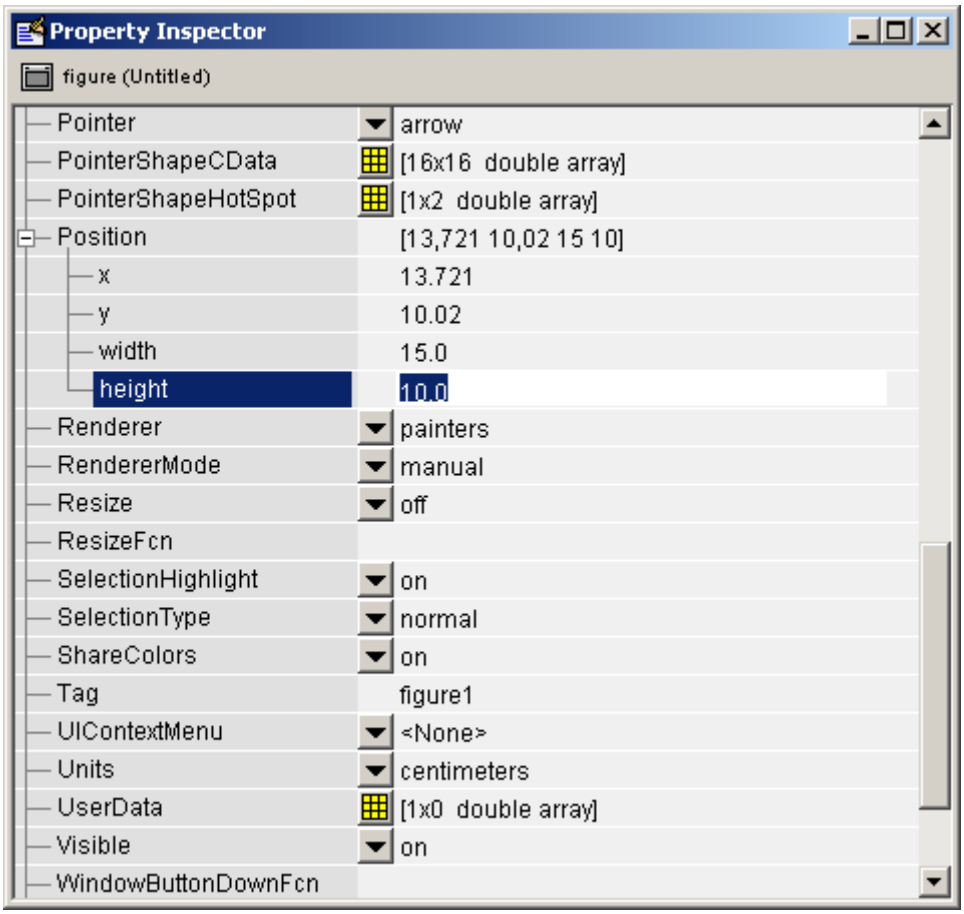
- kliknięcie prawym przyciskiem myszy i z rozwijanego menu wybranie opcji „Property Inspector”
- Z menu „View” wybranie opcji „Property Inspector”)



Każdy obiekt wprowadzony w GUIDE ma własnego „Property Inspector”, w którym można ustawić podstawowe informacje. W przykładzie ustalimy wielkość otwieranego okna na 10 X 15 cm. W tym celu zjeżdżamy w dół do zakładki „Units” i wybieramy opcję „centimeters”

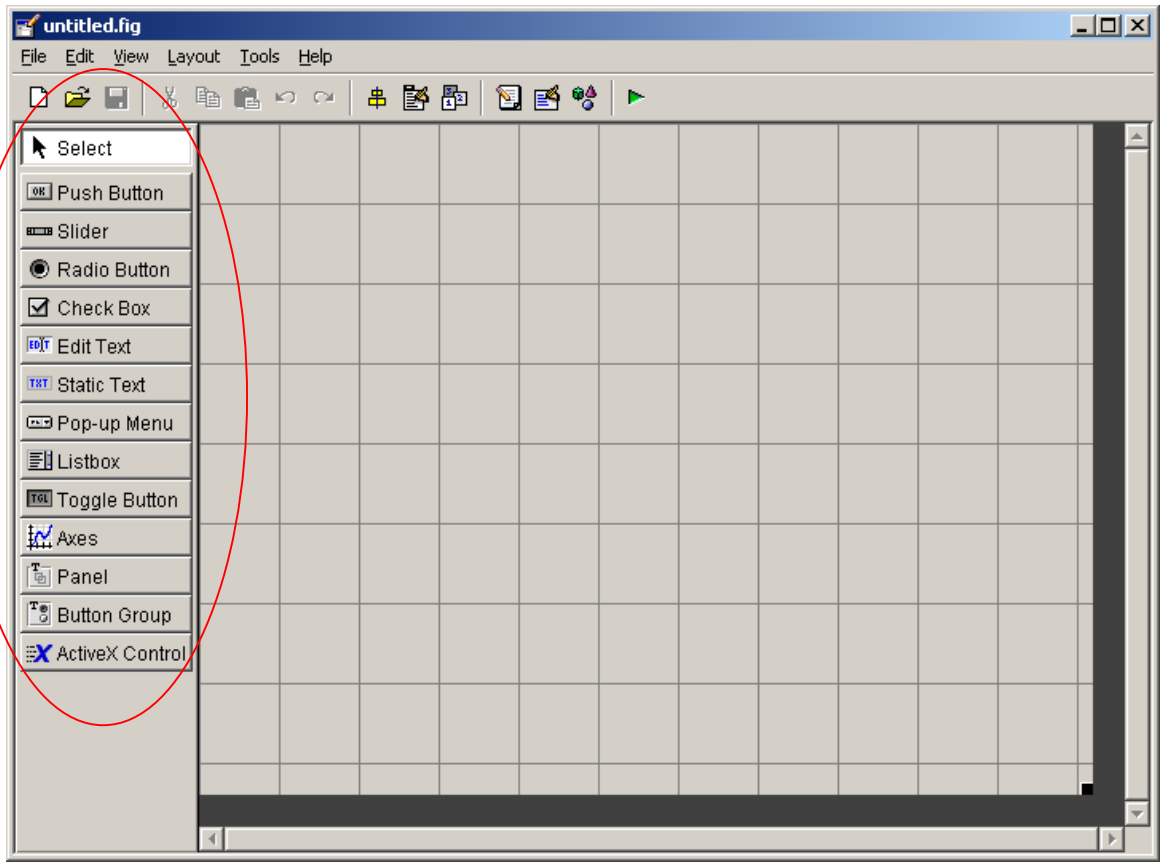


Następnie naciskamy „+” przy zakładce „Position” i wybieramy zmieniamy „width” na 15, a „height” na 10. (Zatwierdzamy przyciskiem „Enter”)



Następnie zamykamy okno „Property Inspector”.

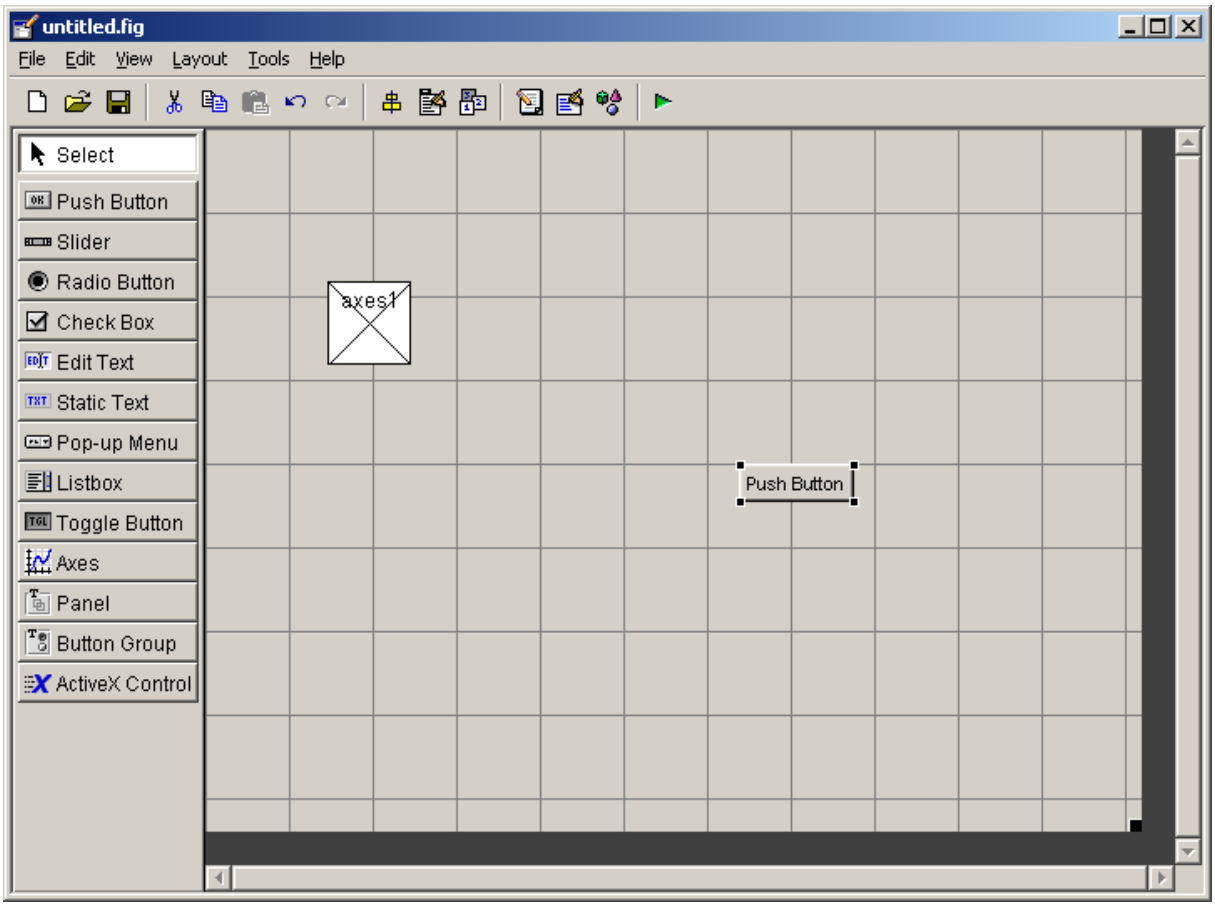
Z lewej strony okna GUIDE znajduje się menu pozwalające wybrać obiekty używane w GUI. Są one bardzo podobno do obiektów używanych przy projektowaniu stron HTML w JavaScript



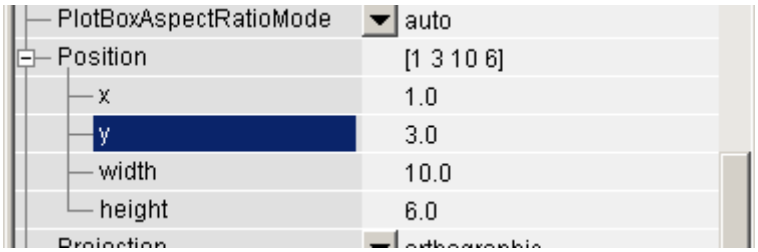
Są to kolejno

- Push Button – przycisk (dzwonkowy – jego naciśnięcie powoduje określoną akcję)
- Slider – suwak
- Radio Button – wybór jednej z opcji w grupie
- Checkbox - włącznik
- Edit Text – tekst edytowalny (w trakcie działania GUI)
- Static Text – tekst (nie zmieniany w trakcie działania GUI)
- Popup Menu – lista rozwijana
- Listbox
- Toggle Button – przełącznik – znajduję się w dwóch położeniach: wciśnięty bądź nie
- Axes – pole wykresu
- Panel – ramka (element dekoracyjny, oraz „nadrzędny”- może sterować obiektami leżącymi w nim; reagować na zmiany wewnątrz)
- Button Group – b. podobne do Panelu
- ActiveX Control – inne funkcje. Np. zegar Windowsowy, etc.

Na początek wstawmy wykres i przycisk (Push button)



Otwórzmy „property inspector” wykresu. Przestawmy opcję Units na centymetry, a następnie wielkość wykresu na 10x6 cm, pozycje x=1, y=3



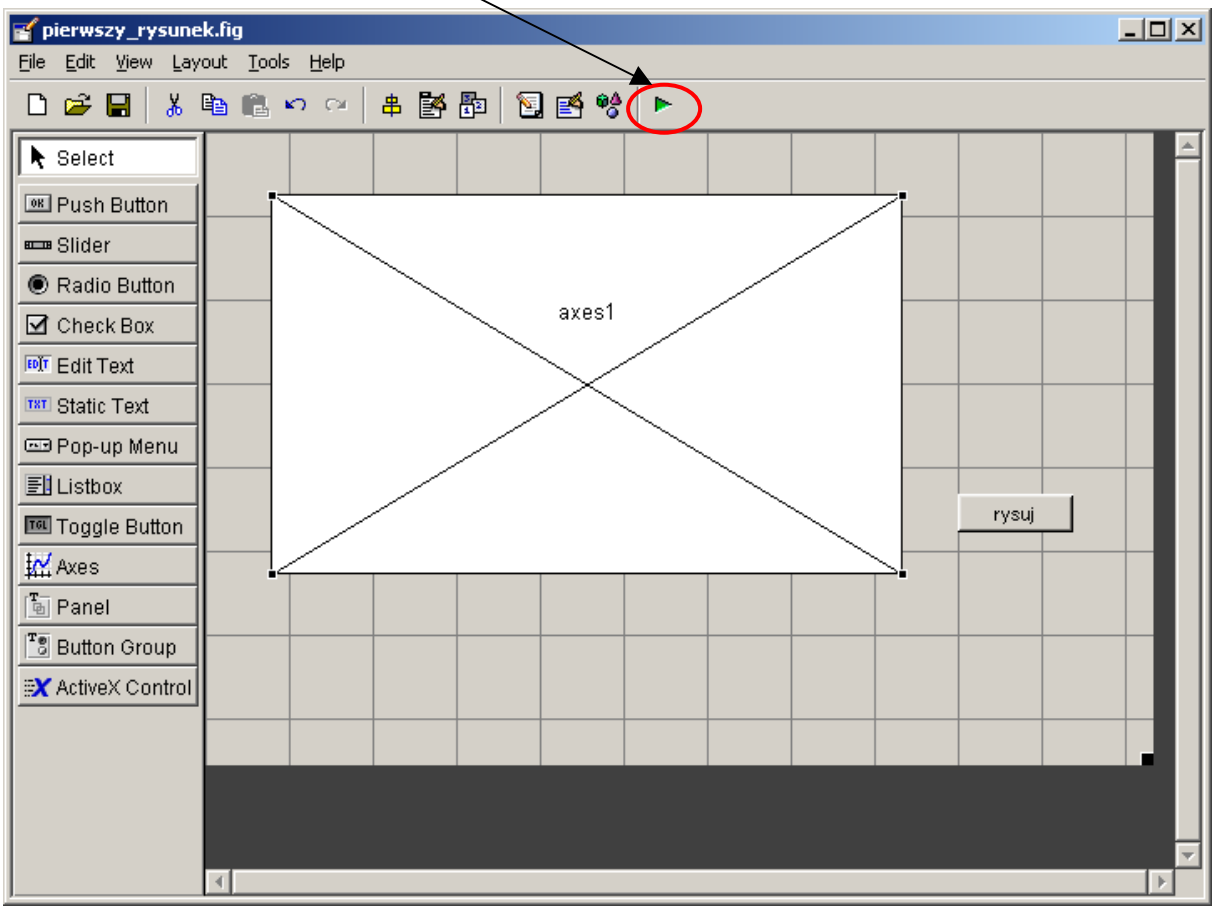
Dzięki temu wykres znajdzie się w prawym górnym rogu – punkt (0,0) znajduje się w lewym dolnym rogu.

„Property inspector” wykresu ma kilkadziesiąt funkcji. Większości z nich właściwie nigdy nie będziemy używać.

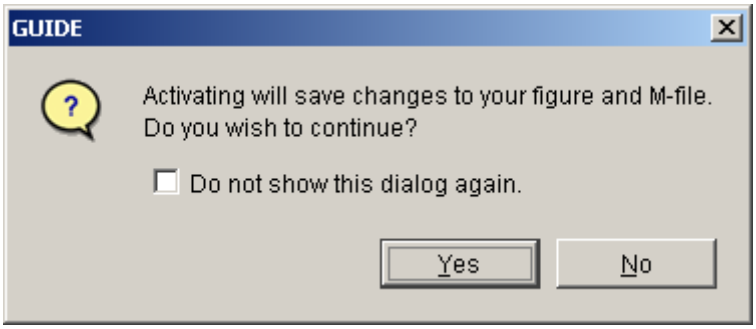
Otwórzmy „P. I.” przycisku.
Najpierw zmieńmy właściwość „String” na „rysuj”. Dzięki temu po włączeniu GUI nasz przycisk będzie miał tą właśnie nazwę.

Uwaga: Właściwość string może zawierać polskie znaki np. „zmień”. Trzeba jednak uważać przy wprowadzaniu, gdyż Matlab przed każdym polskim znakiem dodaje kwadracik („zmie□ń”). Przy większej liczbie polskich usuwanie kwadracików jest dość uciążliwe. Pojawianie się kwadracików jest nowym „udoskonaleniem” pojawiającym się chyba po raz pierwszy w Matlabie 7.0.4.

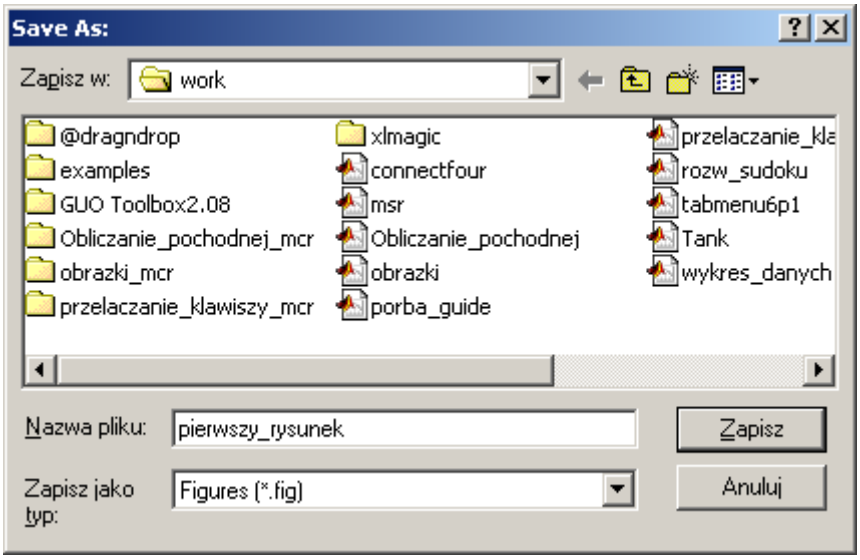
Spróbujmy teraz włączyć nasz rysunek. Możemy to zrobić na 3 sposoby „ctrl+t”, menu Tools->run, lub zielona strzałeczka w prawym górnym rogu



Matlab zapyta się, czy chcemy zapisać zmiany. Bez zapisania zmian nie da się uruchomić GUI.

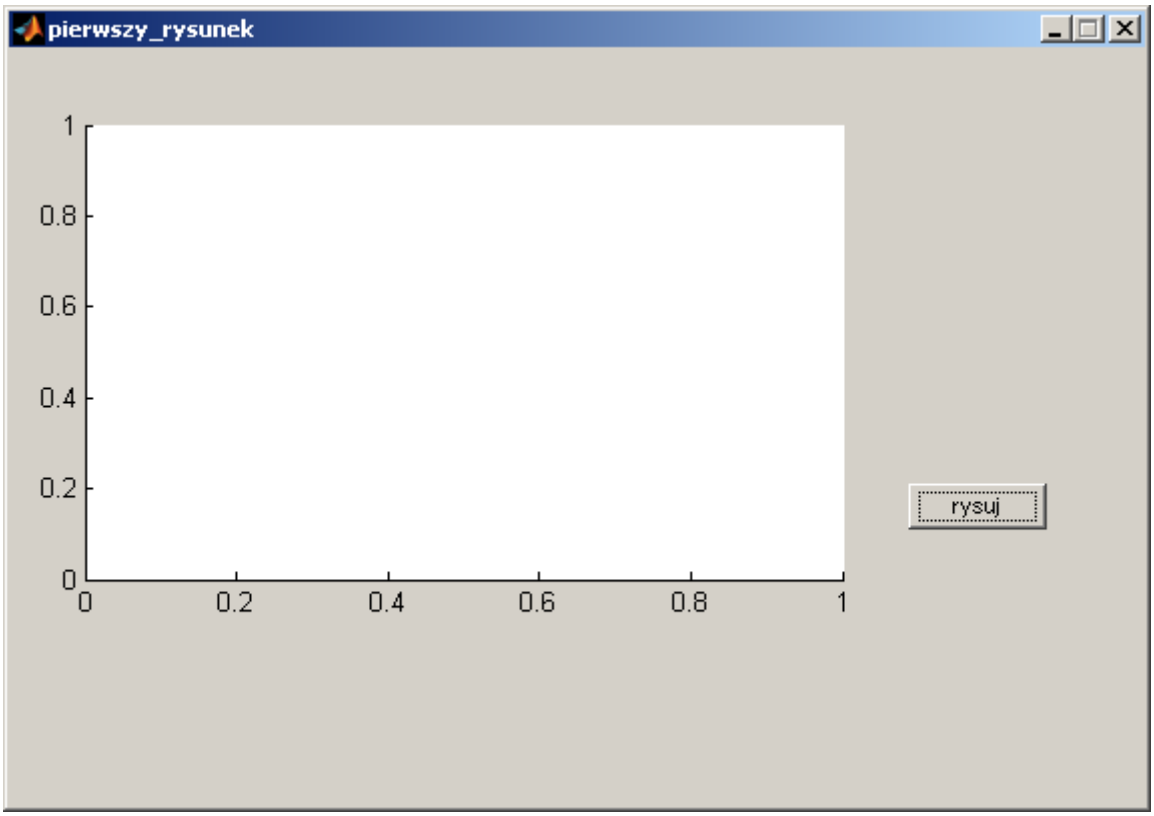


Po kliknięciu na Yes przechodzimy do domyślnego katalogu roboczego. Jeśli nic żeśmy nie zmieniali będzie to „C:\Program Files\Matlab704\work”. Zapiszmy nasz plik jako „pierwszy_guide.fig”



Po zapisaniu wygeneruje się nam automatycznie m-file, o tej samej nazwie co plik fig. Wspólnie tworzą one informacje dla Matlaba jak ma wyglądać GUI.

Powinno pokazać się nam coś takiego:

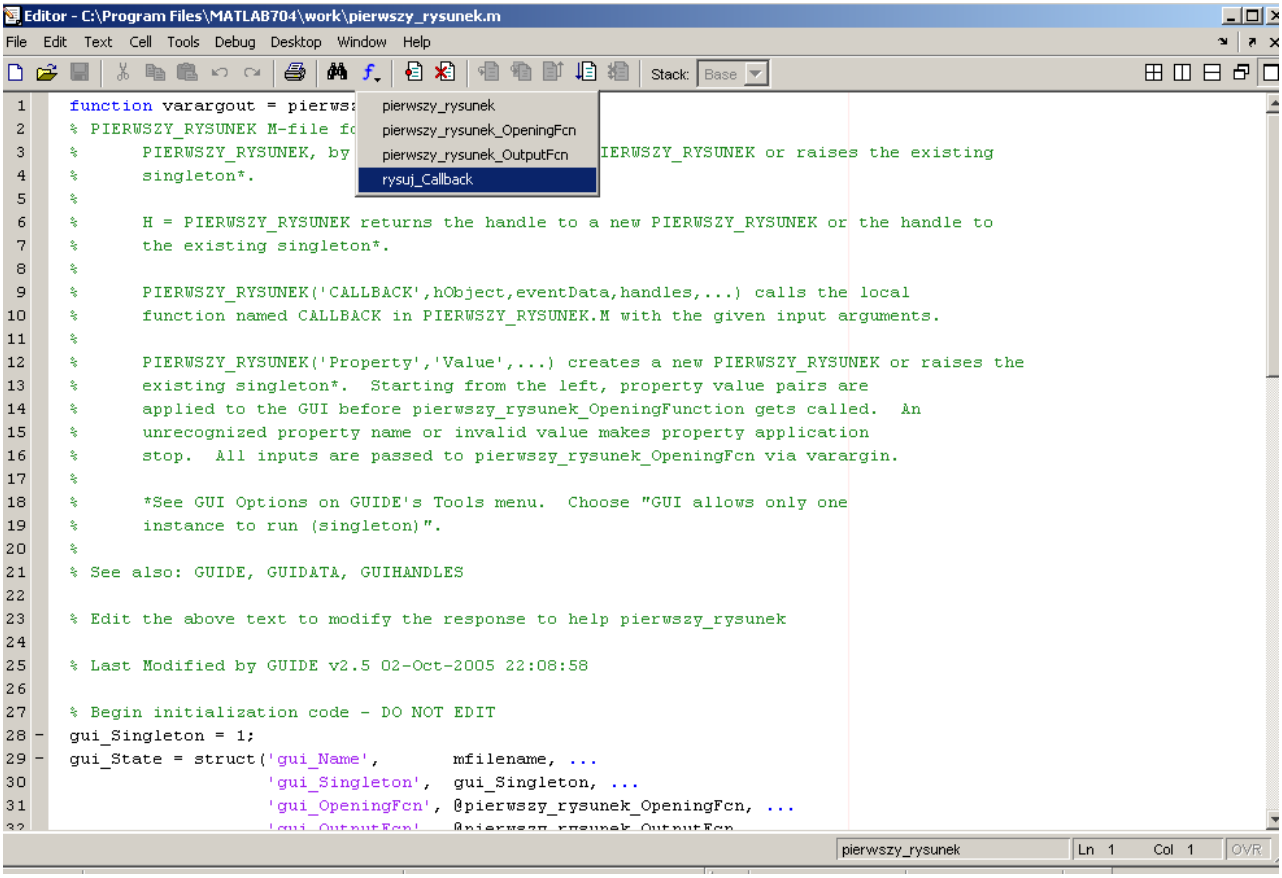


Przycisk już „działa” – można go wciskać. Nie powoduje on jednak żadnej zmiany.

Dodajmy teraz funkcjonalność tego guzika – rysowanie obiektu peaks(12).

Najpierw zmieńmy właściwość „Tag” przycisku „rysuj” z „pushbutton1” na „rysuj”.

Zapiszmy zmiany i przejdźmy do m-filu.



Dzięki zmianie właściwości tag, możemy łatwo przejść do miejsca definiowania funkcjonalności naszego przycisku. Tu może zysk czasu i wygody nie jest jeszcze wielki (mamy, póki co, tylko jeden przycisk), ale przy tworzeniu GUI z kilkoma przyciskami jest to wielka wygoda.

Pod funkcją rysuj_Callback – działającą po włączeniu przycisku, wpisujemy

```
peaks(12);
```

Zapisujemy i uruchamiamy.

Po wciśnięciu przycisku na wykresie powinien pojawić się rysunek – tu uwaga: ustawienie wykresu nie jest przypadkowe. Możemy nim sterować odpowiednimi właściwościami wykresu w „Property inspector”.

Zwróćmy uwagę, że wpisanie bez średnika spowoduje wyświetlenie wartości w głównym oknie matlaba.

Tak więc strasznie na około udało nam się stworzyć, to co moglibyśmy zrobić samą komendą peaks(12); ☺.

Przekazywanie wartości między obiektami

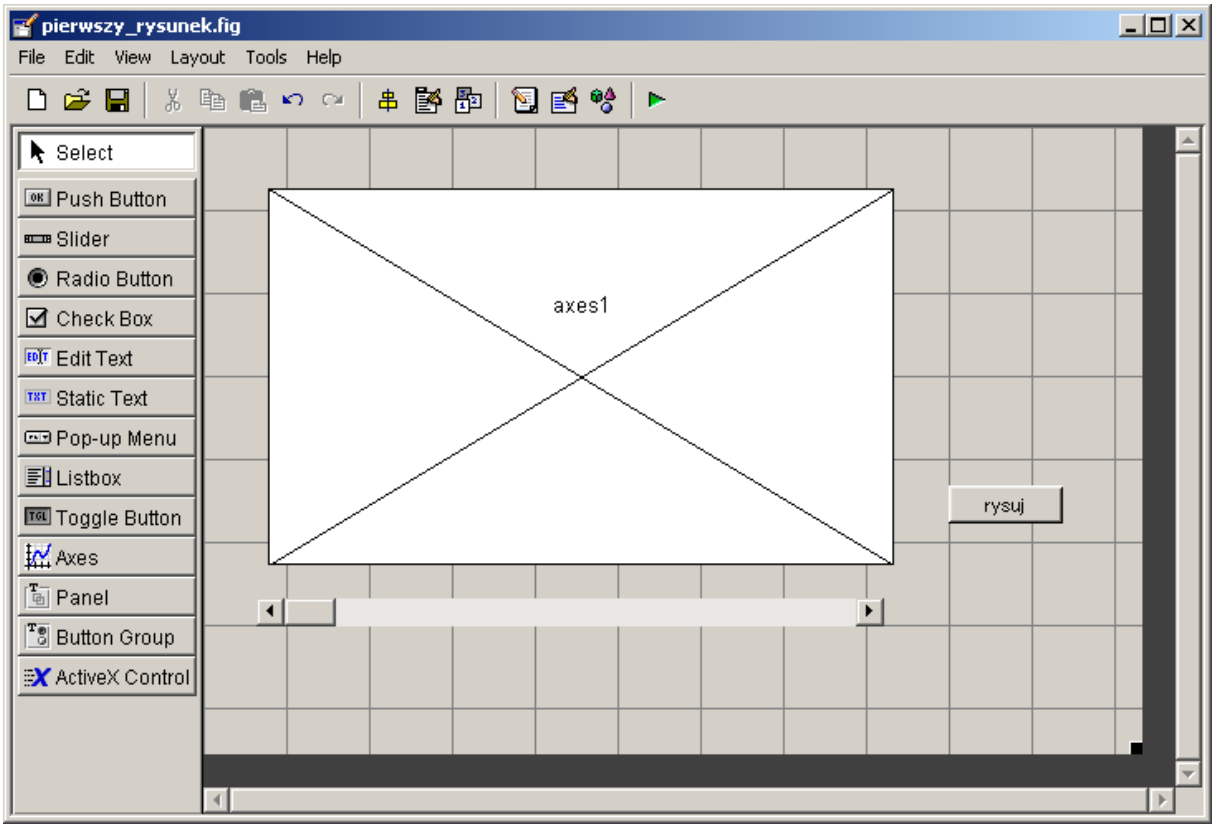
Dodajmy teraz coś „mądrzejszego”. Zróbmy suwak, który umożliwia nam ustalenie wartości dla peaks.

Najpierw wstawmy suwak poziomo. Uwaga:

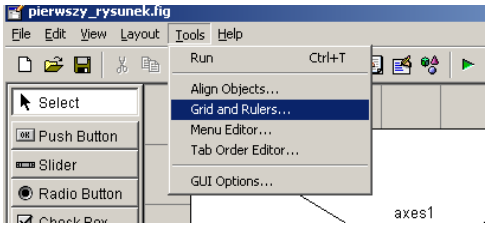
- jeśli weźmiemy z panelu obiekt suwak i wkleimy go na scenę, będzie pionowy. Możemy go następnie przeskalować

- możemy po wzięciu z panelu od razu ustawić wielkość suwaka

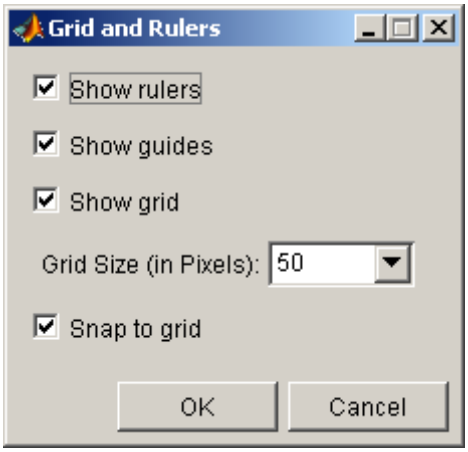
Ustawmy wielkość suwaka na 0.45x10 cm (pamiętajmy o zmianie jednostek - Units)



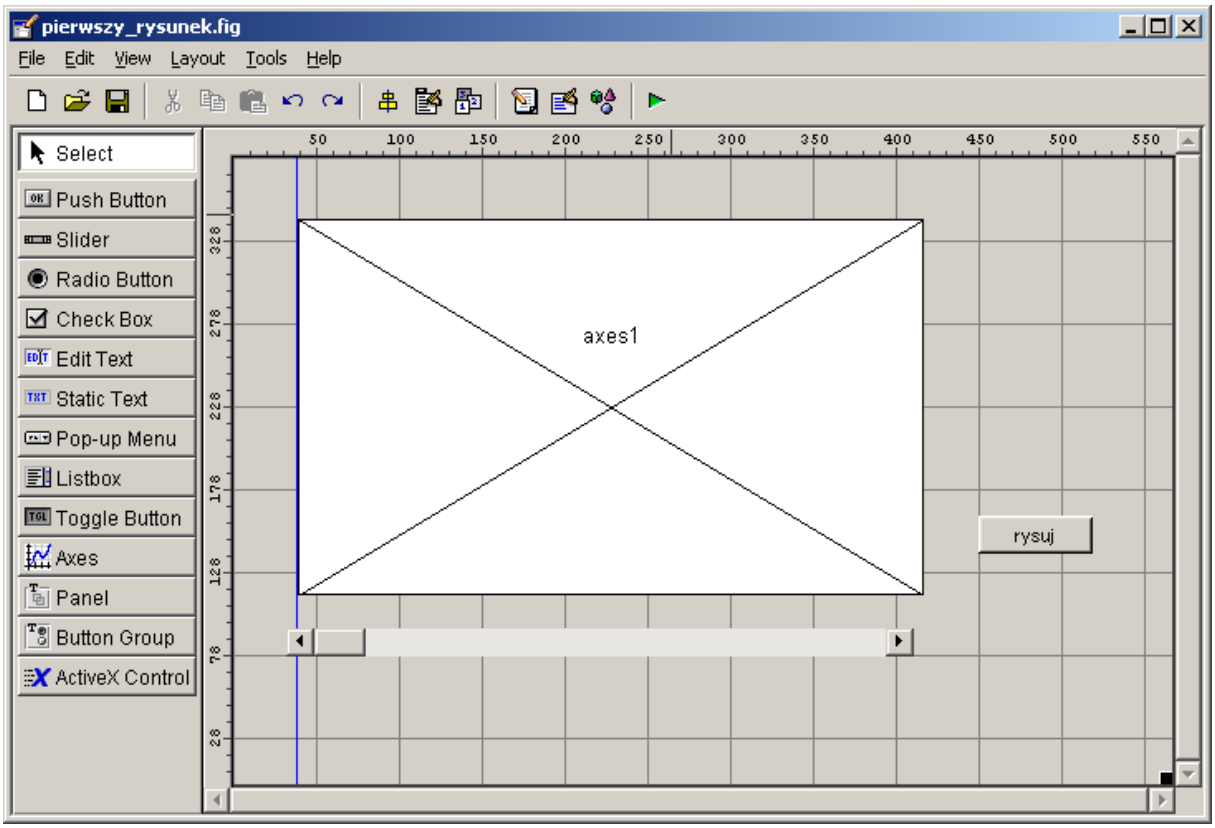
Ustawmy teraz w linii wykres i linijkę.
Z menu Tools wybierzmy „Grid and Rulers”



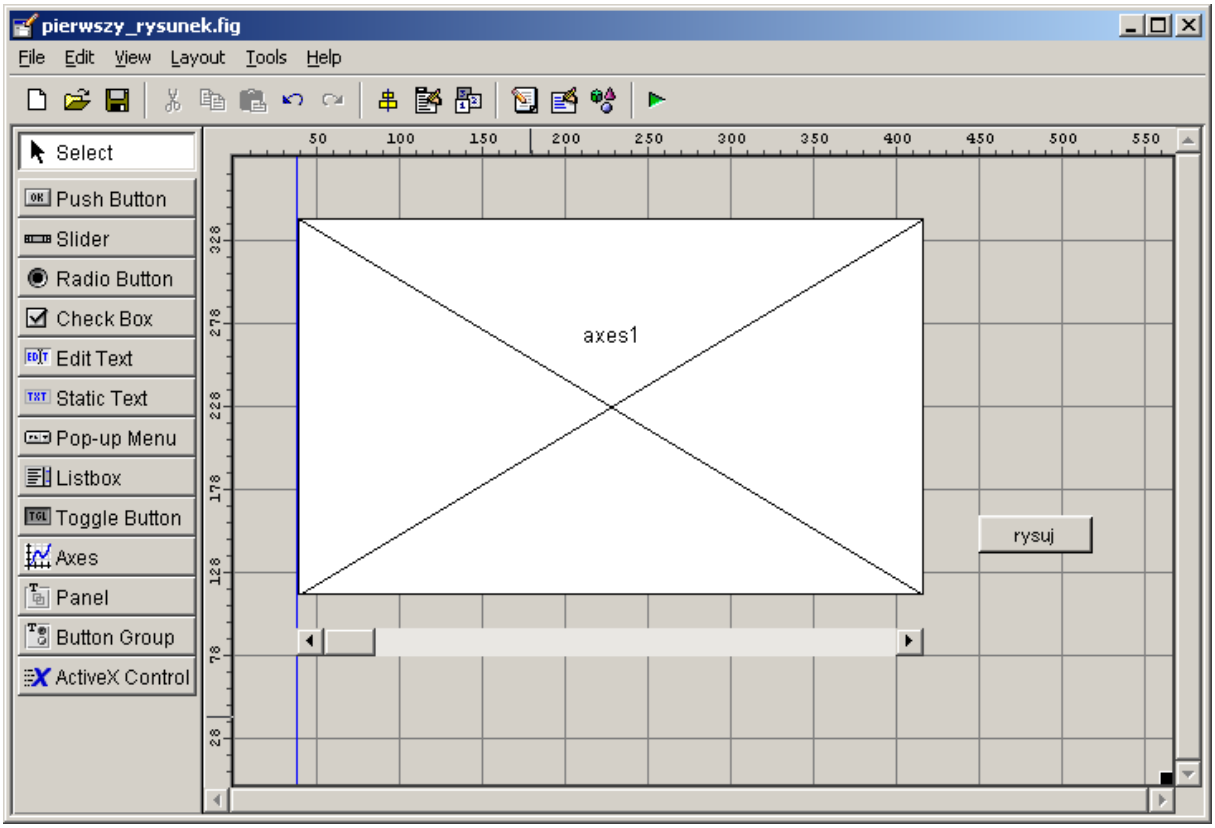
W otwartym oknie zaznaczmy wszystkie obiekty



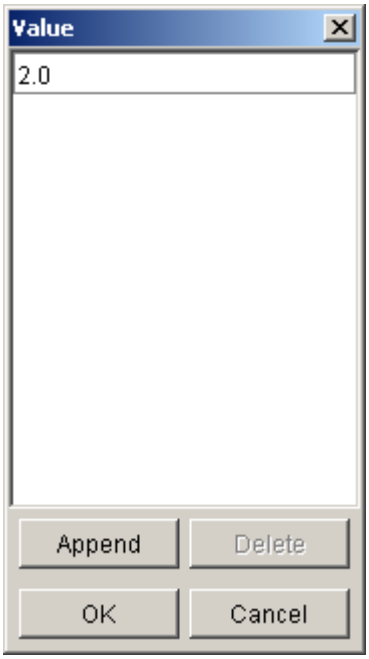
Teraz z lewego boku możemy wyciągnąć linię. Dociągnijmy ją do lewego boku wykresu. Jest to prowadnica, do której możemy ustawiać inne obiekty – działa tak jak np. w Corelu.



Możemy teraz dociągnąć suwak do tej linii.



Zmieńmy teraz w „Property Inspector” właściwość Tag linii na suwak. Zmieńmy też właściwości min i max na 2 i 50. Dzięki temu wartość na tym suwaku będzie mogła przyjmować wartości w tym właśnie zakresie. Domyślnie są to wartości od 0 do 1. Musimy też zmienić wartość Value. Jest to wartość, którą przyjmuje suwak po uruchomieniu GUI. Domyślnie jest to 0. Ponieważ zero jest poza zakresem wartości naszego suwaka, musimy go zmienić. Ustawmy najmniejszą możliwą wartość, tzn. 2.



Przejdźmy do m-filu, do funkcji suwak_Callback.
Funkcja
get(hObject,'Value')

zwraca nam aktualną wartość suwaka (w formacie double). Chcemy zapamiętać ją i odczytać w momencie wciśnięcia przycisku rysuj.

Wszelkie zapamiętywanie wartości odbywa się przy pomocy funkcji „handles."

Zapiszmy

```
handles.suwak=get(hObject,'Value');
```

Dzięki temu w strukturze handles pod nazwą suwak będzie zapisana wartość. Następnie musimy zapisać (odświeżyć) strukturę handles. Odbywa się to za pomocą komendy

```
% Update handles structure
guidata(hObject, handles);
```

WSZELKIE ZMIANY WARTOŚCI ZMIENNYCH MUSZĄ BYĆ ZAPISYWANE JAKO HANDLES I ODŚWIEŻANE KOMENDĄ guidata(hObject, handles); !!!!!!!

Nasz kod powinien wyglądać mniej więcej tak

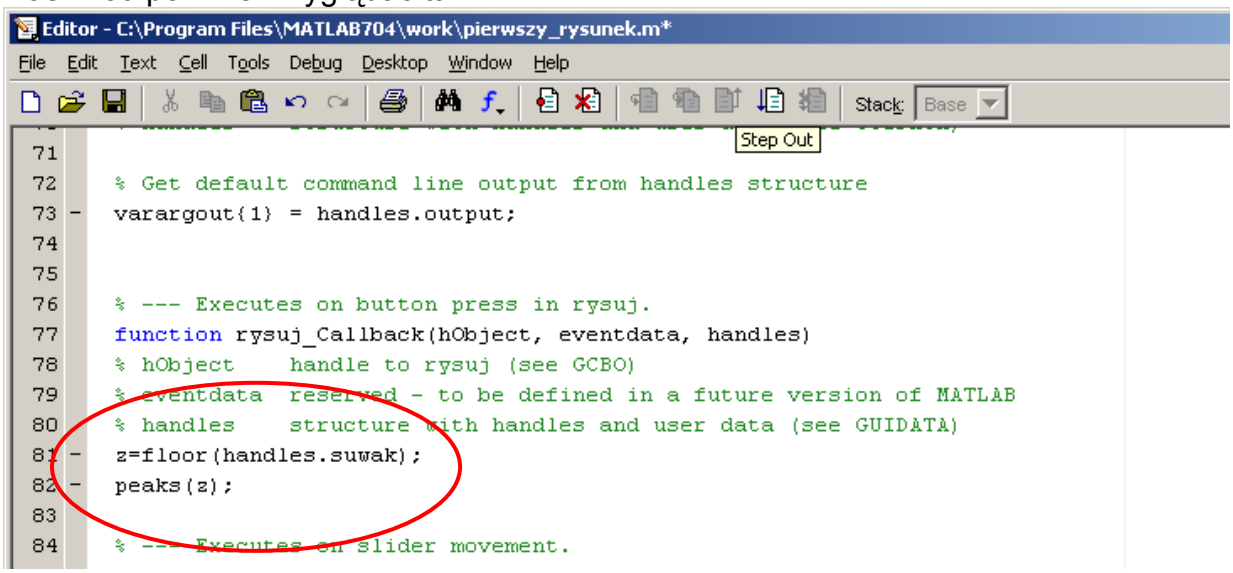
```
104
105
106     % --- Executes on slider movement.
107     function suwak_Callback(hObject, eventdata, handles)
108     % hObject    handle to suwak (see GCBO)
109     % eventdata  reserved - to be defined in a future version of MATLAB
110     % handles    structure with handles and user data (see GUIDATA)
111
112     % Hints: get(hObject,'Value') returns position of slider
113     %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
114 -   handles.suwak=get(hObject,'Value');
115
116     % Update handles structure
117 -   guidata(hObject, handles);
118
119     % --- Executes during object creation, after setting all properties.
120     function suwak_CreateFcn(hObject, eventdata, handles)
121     % hObject    handle to suwak (see GCBO)
122     % eventdata  reserved - to be defined in a future version of MATLAB
123     % handles    empty - handles not created until after all CreateFcns called
124
```

Teraz odczytajmy tą wartość przy funkcji rysuj_Callback.

Odczytanie odbywa się po prostu przez przypisanie zmiennej wartości handles.suwak.
Ograniczmy wartość handles.suwak do wartości całkowitych. Weźmy podłogę z pierwotnej wartości.

```
z=floor(handles.suwak);
```

Zmieńmy teraz funkcję tak, aby rysowała peaks(z); zamiast peaks(12);.
Nasz kod powinien wyglądać tak:



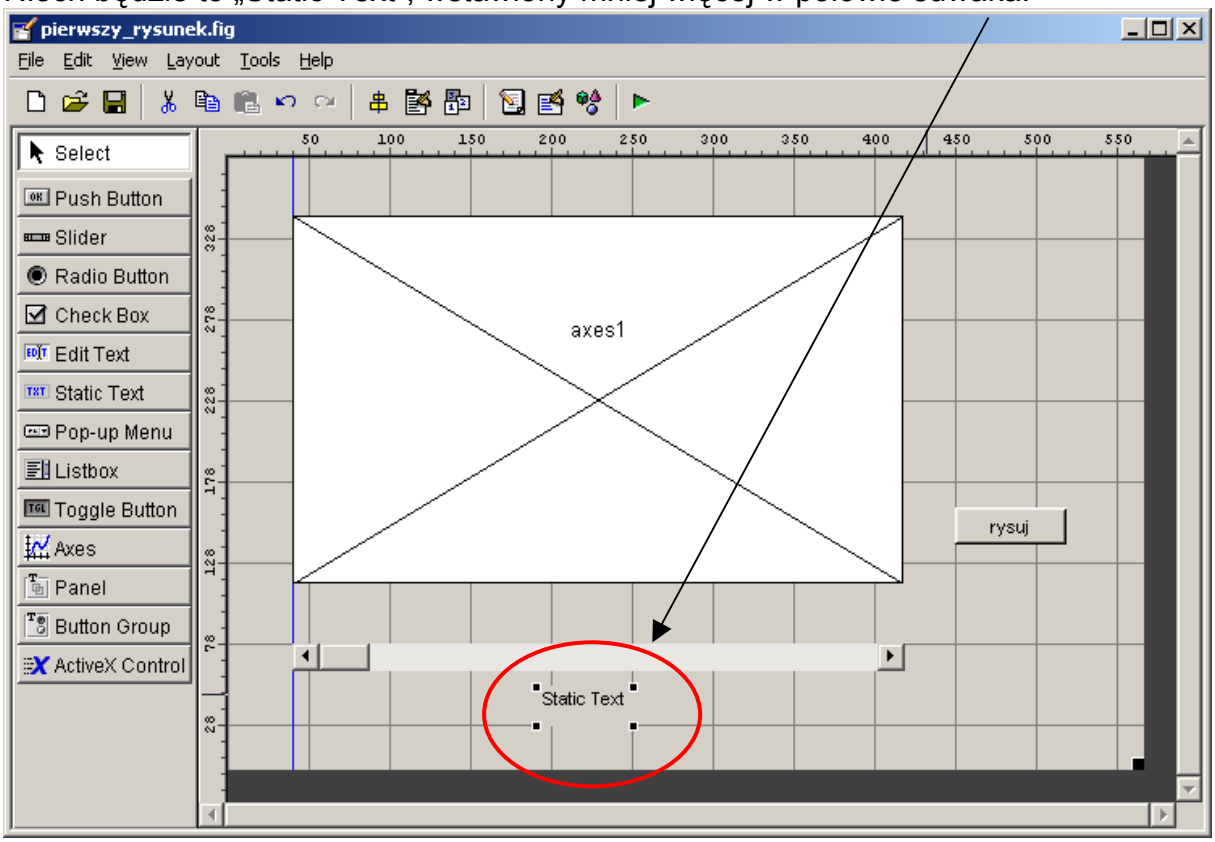
Uruchommy teraz nasz kod.

Przestawiając suwak, a następnie naciskając przycisk „rysuj” będziemy otrzymywać różne wykresy.

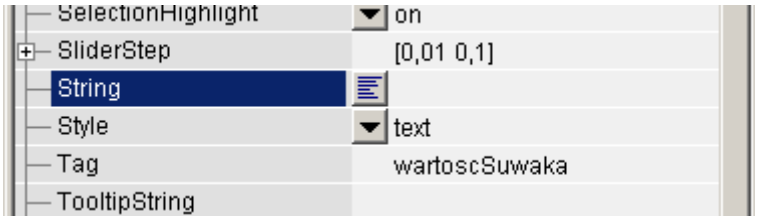
Zmienianie właściwości innych obiektów

W tym momencie przesuwając suwak możemy tylko mniej więcej domyślać się jego wartości. Nie wiemy dokładnie na jaką pozycję żeśmy go przestawili. Dodajmy teraz pole tekstowe, które będzie zawierało wartość suwaka.

Niech będzie to „Static Text”, wstawiony mniej więcej w połowie suwaka.



Zmieńmy teraz właściwość Tag na wartoscSuwaka (Tag nie może zawierać polskich znaków), i usuńmy wpis we właściwości String



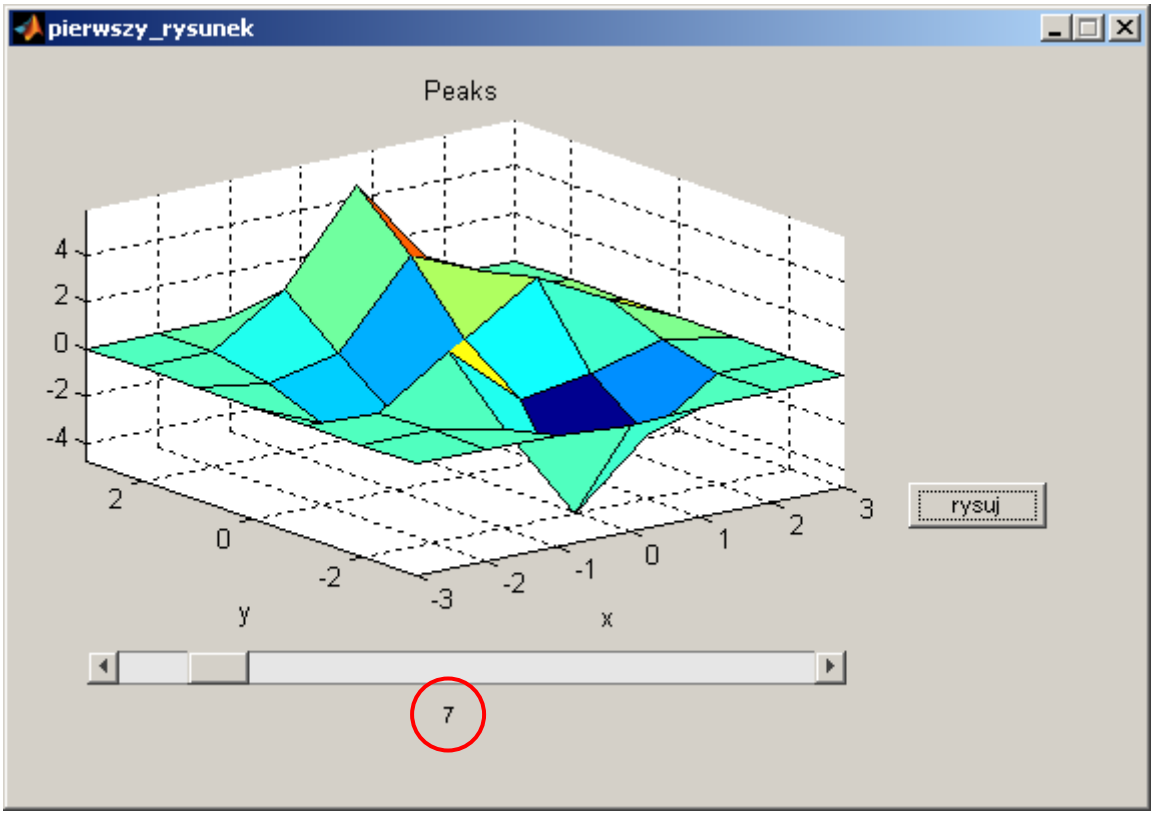
Przejdźmy do m-pliku do funkcji suwak_Callback i dopiszmy tam komendę zmieniająca wartość pola „wartoscSuwaka” jeszcze przed zapisaniem zmian do struktury handles.

```
set(handles.wartoscSuwaka,'String',floor(get(hObject,'Value')));
```

```
106    % --- Executes on slider movement.
107    function suwak_Callback(hObject, eventdata, handles)
108    % hObject    handle to suwak (see GCBO)
109    % eventdata  reserved - to be defined in a future version of MATLAB
110    % handles    structure with handles and user data (see GUIDATA)
111
112    % Hints: get(hObject,'Value') returns position of slider
113    %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
114 -  handles.suwak=get(hObject,'Value');
115 -  set(handles.wartoscSuwaka,'String',floor(get(hObject,'Value')));
116    % Update handles structure
117 -  guidata(hObject, handles);
118
119    % --- Executes during object creation, after setting all properties.
120    function suwak_CreateFcn(hObject, eventdata, handles)
```

Przy pomocy funkcji set możemy zmieniać wszystkie właściwości dostępne w Property Inspector. Wystarczy adres obiektu (handles.*nazwa*), adres właściwości (w pojedynczym cudzysłowie) i wartość końcowa. Należy tylko uważać, czy zgadzają się formaty (przydatne komendy str2int, oraz int2str), oraz czy dana zmiana nie jest sprzeczna z innymi własnościami (np. czy zmiana wartości suwaka mieści się w jego przedziale (min,max), można próbować rozwiązaniem typu „try..... catch.....").

W uruchomionym programie, po przesunięciu suwaka (i puszczeniu go), pod suwakiem powinna pojawić się liczba całkowita i to od tej liczby powinien być tworzony wykres peaks.



nazwa_OpeningFcn

Rozwiążmy teraz pewien błąd naszego programu. Po uruchomieniu, jeśli nie przesuniemy suwaka, program nie wie, jaką wartość ma z (lub co na jedno wychodzi handles.suwak). Jeśli bez zmiany suwaka naciśniemy przycisk „rysuj”, to albo program zgłupieje, albo przypisze jakąś przypadkową liczbę. Zmieńmy to.

W momencie uruchamiania naszego GUI na samym początku program uruchamia funkcję *nazwaProgramu*_OpeningFcn. Wstawienie tam odpowiednich komend, budowanie odpowiedniej struktury już na tym poziomie pozwala często uniknąć błędów w programie.

Dodajmy w funkcji pierwszy_rysunek_OpeningFcn następujący wiersz

handles.suwak=2;

```
46
47    % --- Executes just before pierwszy_rysunek is made visible.
48    function pierwszy_rysunek_OpeningFcn(hObject, eventdata, handles, varargin)
49    % This function has no output args, see OutputFcn.
50    % hObject    handle to figure
51    % eventdata  reserved - to be defined in a future version of MATLAB
52    % handles    structure with handles and user data (see GUIDATA)
53    % varargin   command line arguments to pierwszy_rysunek (see VARARGIN)
54
55    % Choose default command line output for pierwszy_rysunek
56 -  handles.output = hObject;
57 -  handles.suwak=2;
58    % Update handles structure
59 -  guidata(hObject, handles);
60
61    % UIWAIT makes pierwszy_rysunek wait for user response (see UIRESUME)
62    % uiwait(handles.figure1);
```

Dzięki temu powinniśmy uniknąć błędu przy uruchamianiu się funkcji. Możemy jeszcze zmienić właściwość String w polu tekstowym „wartoscSuwaka” na 2. Dzięki temu w momencie włączenia programu od początku będziemy widzieć wartość suwaka.

Ćwiczenia

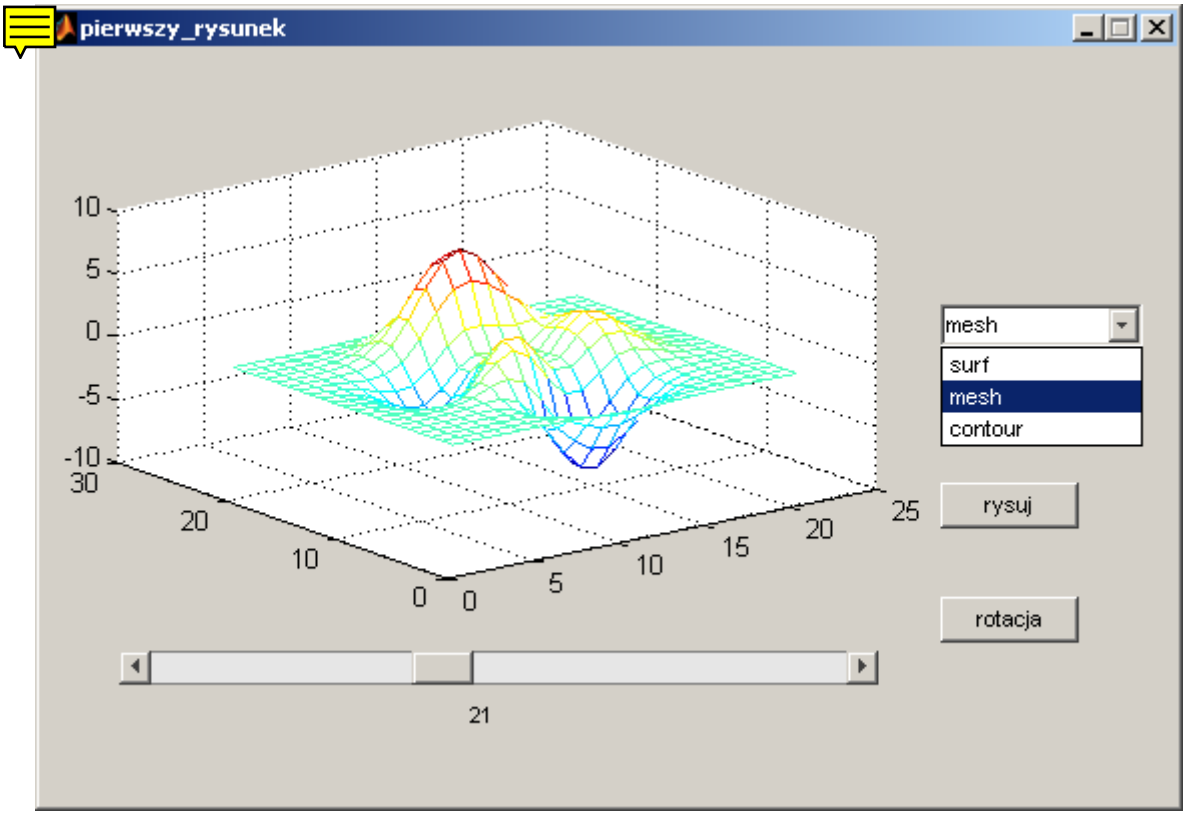
Ćwiczenie 1.

Wstaw nowy przycisk (Pushbutton), który będzie pozwalał obracać wykres. Nazwij go rotacja i zmień jego właściwość Tag na rotacja. W m-filu ustaw właściwość rotacji.

Podpowiedź

Ćwiczenie 2.

Wstaw „Pop-up Menu” o wartościach surf, mesh, contour. W zależności od wyboru naciśnięcie na przycisk rysuj ma powodować rysowanie peaks(z) jako surf, jako mesh, lub jako contour.

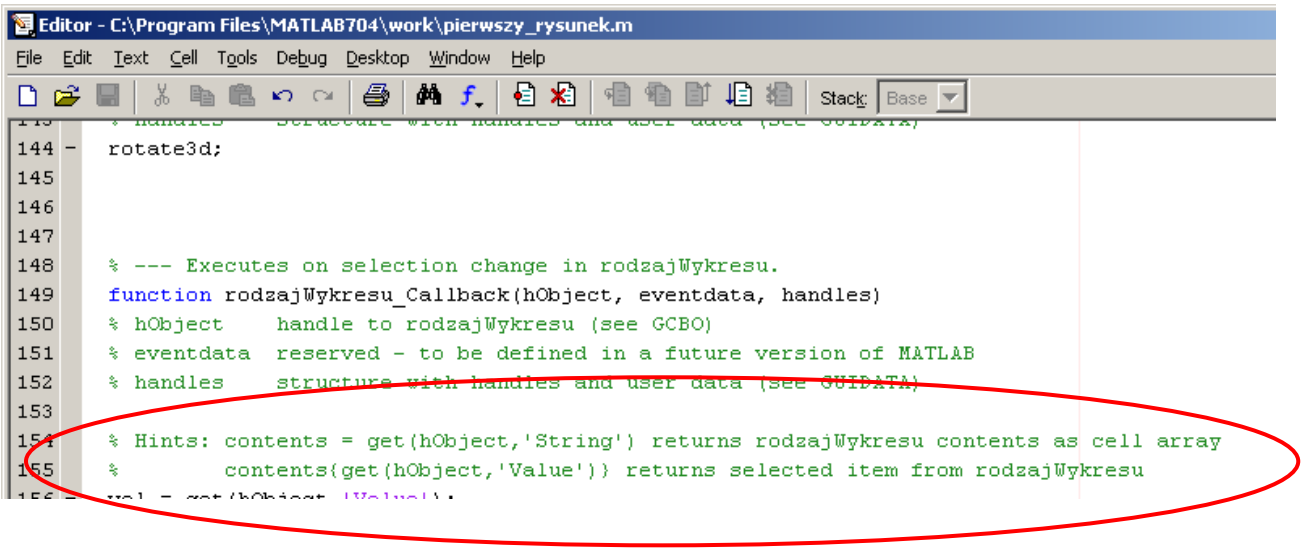


Podpowiedzi przykładowego rozwiązania

1. Wstaw „Pop-up Menu”, którego wartości będą surf, mesh i contour zmień Tag (np. na rodzajWykresu)

2. Niech w OpeningFcn będzie tworzony nowy obiekt struktury handles. Np. handles.rodzaj o początkowej wartości np. surf (najlepiej tej samej co pierwszy obiekt w Pop-up Menu).

3. Dodaj w funkcji Pop-up Menu (np. rodzajWykresu_Callback) odczytanie wyboru. Przyjrzyj się opisowi Hints



Jeśli nadal nie wiesz spójrz tutaj

4. Zapisz odczytany wynik do struktury hints, (np. handles.rodzaj)

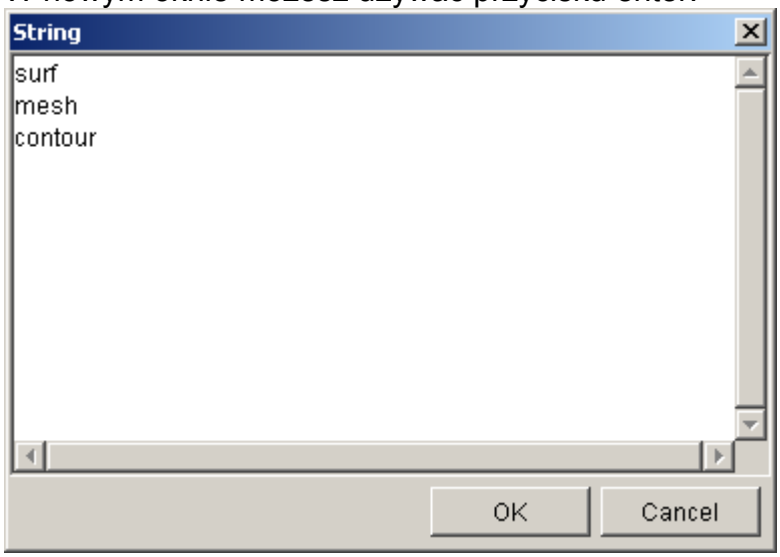
5. W funkcji przycisku „rysuj” wstaw odpytanie (if.... else.....) o wartość obiektu struktury (np. handles.rodzaj). Pamiętaj – do porównywania stringów służy komenda strcmp. W zależności od wyniku niech odbywa się mesh(peaks(z)), surf(peaks(z)), contour(peaks(z));

Dokładniejsze rozwiązanie na następnej stronie

Ad. 1.

Aby Pop-up Menu dobrze wyglądało kliknij dwa razy w ten symbol przy właściwości string

W nowym oknie możesz używać przycisku enter.



Ad. 2.

Np. kod

```
handles.rodzaj='surf';
```

surf musi być podane jako string. Samo surf (bez cudzysłowu) się nie uda!

Ad. 3. i 4.

Np. kod

```
val = get(hObject,'Value');
str = get(hObject, 'String');
handles.rodzaj=str{val};
```

Dzięki temu handles.rodzaj będzie zawierać wybraną wartość.

```
148
149
150    % --- Executes on selection change in rodzajWykresu.
151    function rodzajWykresu_Callback(hObject, eventdata, handles)
152    % hObject    handle to rodzajWykresu (see GCBO)
153    % eventdata  reserved - to be defined in a future version of MATLAB
154    % handles    structure with handles and user data (see GUIDATA)
155
156    % Hints: contents = get(hObject,'String') returns rodzajWykresu contents as cell array
157    %        contents{get(hObject,'Value')} returns selected item from rodzajWykresu
158 -  val = get(hObject,'Value');
159 -  str = get(hObject, 'String');
160 -  handles.rodzaj=str{val};
161
162    % Update handles structure
163 -  guidata(hObject, handles);
164
165    % --- Executes during object creation, after setting all properties.
166    function rodzajWykresu_CreateFcn(hObject, eventdata, handles)
```

Ad. 5.

Np. komenda

```
r=handles.rodzaj
if strcmp(r,'mesh')
   mesh(peaks(z));
elseif strcmp(r,'surf')
   surf(peaks(z));
else
   contour(peaks(z));
end;
```

```
76
77    % --- Executes on button press in rysuj.
78    function rysuj_Callback(hObject, eventdata, handles)
79    % hObject    handle to rysuj (see GCBO)
80    % eventdata  reserved - to be defined in a future version of MATLAB
81    % handles    structure with handles and user data (see GUIDATA)
82 -  z=floor(handles.suwak);
83 -  r=handles.rodzaj
84 -  if strcmp(r,'mesh')
85 -      mesh(peaks(z));
86 -  elseif strcmp(r,'surf')
87 -      surf(peaks(z));
88 -  else
89 -      contour(peaks(z));
90 -  end;
91
92    % --- Executes on slider movement.
```