

# Laboratorium MATLA/MTL

## *Ćwiczenie 5*

Opracowali:

- dr inż. Beata Leśniak-Plewińska
- dr inż. Jakub Żmigrodzki

Zakład Inżynierii Biomedycznej,  
Instytut Metrologii i Inżynierii Biomedycznej,  
Wydział Mechatroniki Politechniki Warszawskiej.

Warszawa, 2016

## 1. Cel ćwiczenia

W ramach ćwiczenia studenci zapoznają się z projektowaniem graficznego interfejsu użytkownika w MATLAB'ie. W instrukcji do ćwiczenia zamieszczono opis modułu GUIDE (Graphical User Interface Design Environment) – narzędzia służącego do interaktywnego tworzenia GUI (ang. Graphical User Interface, GUI).

## 2. Wymagane wiadomości

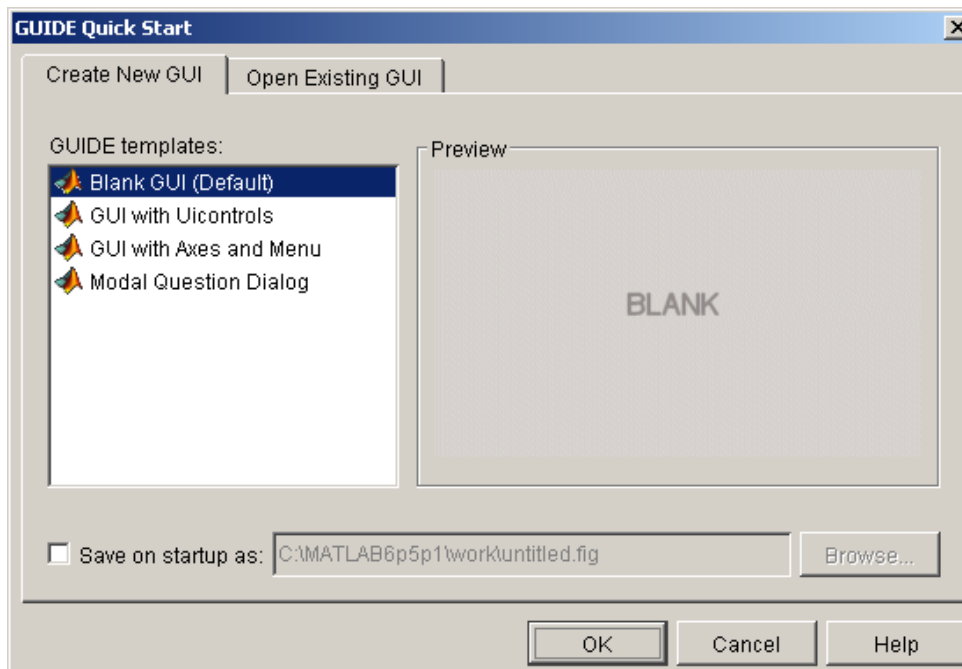
Informacje zawarte w niniejszej instrukcji.

## 3. Informacje podstawowe

Graficzny interfejs użytkownika to ogólne określenie sposobu prezentacji informacji przez komputer polegającego na rysowaniu elementów (widżetów, zwanych często kontrolkami, takich jak: okna, przyciski, pola wyboru, pola radiowe, pola edycyjne, paski menu, ikony, listy, zakładki, okna dialogowe, suwaki, paski narzędzi, etykiety) z dokładnością do piksela, w odróżnieniu od interfejsu tekstowego, gdzie najmniejszą jednostką rysowaną jest znak.

## 4. GUIDE

W MATLAB'ie do interaktywnego projektowania GUI służy narzędzie **GUIDE**. **GUIDE** wspomaga rozmieszczanie elementów GUI oraz generuje szkielet programu (m-plik) obsługującego te elementy (obiekty).



Rysunek 1: Okno **GUIDE Quick Start**

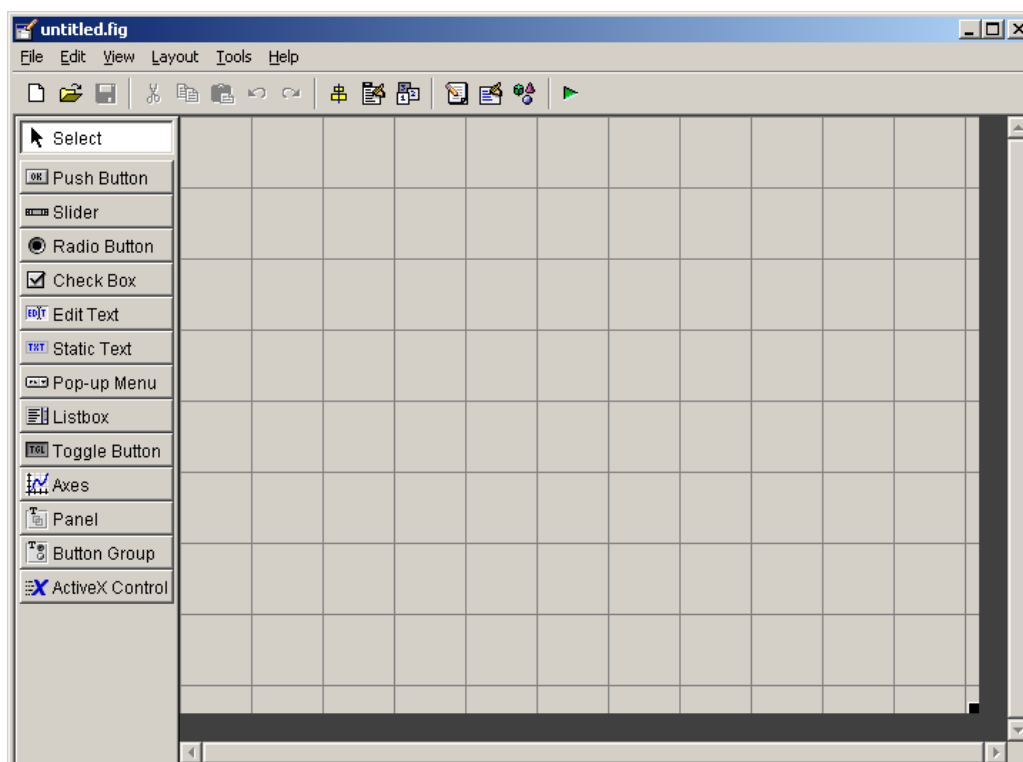
Aplikację **GUIDE** możemy wywoływać następująco:

- wpisując w oknie poleceń komendy `>>guide` lub `>>guide('nazwa_pliku.fig');`
- wybierając z menu **Start MATLAB** → **GUIDE (GUI Builder)**;

- wybierając z menu **MATLAB File New** → **GUI**;
- klikając na pasku narzędzi MATLABa przycisku .




Aplikacje tworzone w **GUIDE** są zapisywane jako pliki graficzne z rozszerzeniem *.fig* i w formie m-pliku. Polecenie `guide` otwiera okno **GUIDE Quick Start** (Rysunek 1). Użytkownik uzyskuje możliwość wyboru pustego edytora GUI, wyboru spośród kilku typowych wzorców rozmieszczenia elementów interfejsu oraz otworzenia istniejącego GUI w celu jego modyfikacji.













W celu stworzenia nowego GUI należy wybrać opcję **Blank GUI (Default)** i kliknąć OK otwierając okno edytora **Layout – untitled.fig** (Rysunek 2).



Rysunek 2: Okno edytora **Layout – untitled.fig**

W centralnej części okna edytora **Layout Editor** (część zakratkowana) znajduje się obszar wyświetlany po uruchomieniu aplikacji, dla której tworzony jest interfejs graficzny. W tym obszarze wstawiane są elementy/obiekty GUI. Z lewej strony okna **GUIDE** znajduje się menu pozwalające wybrać obiekty używane w GUI. Są to kolejno:

- Push Button**  - przycisk polecenia; generuje określoną akcję wtedy, kiedy jest naciśnięty (wykonywane jest wówczas wywołanie wsteczne – *callback*).
- Slider**  - suwak; akceptuje liczbowe parametry wejściowe mieszczące się w określonym zakresie; położenie suwaka wskazuje procent całego zakresu przemieszczania.
- Radio Button**  - przycisk wyboru jednej z opcji; wybranie jednego z nich anuluje wybór pozostałych.

<b>Checkbox</b>		- pole wyboru; generuje akcję, kiedy jest wybrane; jeśli jest w grupie, to wybranie jednego z nich nie anuluje wyboru pozostałych.
<b>Edit Text</b>		- pole tekstowe umożliwiające wprowadzanie i edycję tekstu; wywołanie wsteczne jest wykonywane po naciśnięciu klawisza ENTER.
<b>Static Text</b>		- etykieta opisująca inne obiekty graficzne w GUI lub pole obliczeniowe; użytkownik nie może interaktywnie wpływać na ten obiekt; nie posiada on wywołania wstecznego.
<b>Popup Menu</b>		- pole listy rozwijanej umożliwiające wybór jednego z jej elementów.
<b>Listbox</b>		- pole listy umożliwiające wybór jednego lub więcej elementów z listy.
<b>Toggle Button</b>		- przycisk przełącznika bistabilnego – wciśnięty pozostaje w tym stanie aż zostanie wyciśnięty; wywołanie wsteczne jest wykonywane po zwolnieniu naciśniętego przycisku myszki wskazującej ten przycisk.
<b>Table</b>		- tabela
<b>Axes</b>		- osie współrzędnych
<b>Panel</b>		- panel/ramka; może zawierać grupę dowolnych elementów; zgrupowanie podobnych elementów ułatwia korzystanie z interfejsu; panel ma tytuł, obramowanie oraz może zawierać inne panele, grupę opcji, osie współrzędnych i kontrolki; położenie każdego elementu wewnątrz panelu jest interpretowane jako względne do panelu.
<b>Button Group</b>		- grupa przycisków opcji ( <i>radio buttons</i> ) lub przyciski przełącznika ( <i>Toggle Buttons</i> ).
<b>Toolbar</b>		- pasek narzędzi.
<b>ActiveX Control</b>		- kontrolki ActiveX.

Właściwe działanie GUI gwarantują wywołania wsteczne (**Callbacks**). Są to funkcje wykonywane wtedy, gdy wystąpi określona akcja na ekranie, np. kliknięcie przycisku. Wybrane rodzaje i własności wywołań wstecznych podaje Tabela 1.

Pobranie elementu możliwe jest za pomocą myszy. Każdy obiekt GUI można dopasowywać do aktualnych potrzeb zmieniając jego położenie, rozmiar i inne atrybuty.

Korzystając w **GUIDE** z myszy należy używać obu jej przycisków przy czym:

- lewy przycisk służy do wyboru obiektu graficznego, zmiany jego wymiarów oraz do jego przesuwania; bardziej precyzyjne pozycjonowanie obiektów, po ich wskazaniu myszką, możliwe jest dzięki klawiszom kierunkowym: ←↓↑→;

- prawy przycisk myszy otwiera lokalne menu kontekstowe umożliwiające między innymi:
  - duplikowanie elementu (**Duplicate**);
  - zmianę warstwy: przesunąć na wierzch/pod spód (**Bring to Front/Send to Back**);
  - określenie i modyfikację właściwości obiektu (**Property Inspector**);
  - przeglądanie zdefiniowanych obiektów (**Object Browser**);
  - definiowanie wywołań zwrotnych (**View Callbacks**).


Tabela 1

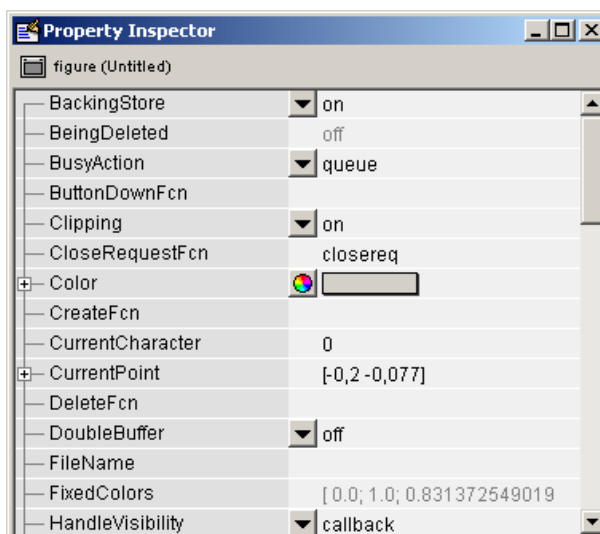
Własność wywołania wstecznego	Opis
ButtonDownFcn	Jest wykonywane wtedy, gdy użytkownik naciśnie przycisk myszki, a jej kursor wskazuje dany komponent
CreateFcn	Inicjalizuje komponent podczas jego tworzenia, ale przed pojawieniem się na ekranie
Callback	Wykonuje podstawowe zadania w odpowiedzi na akcję użytkownika.
DeleteFcn	Jest wykonywane przed usuwaniem obiektu, tj. gdy uruchomione zostanie polecenie delete albo close w stosunku do okna aplikacji zawierającego dany element (nie ma zastosowania do obiektu root).
KeyPressFcn	Jest wykonywane wtedy, gdy użytkownik naciśnie klawisz na klawiaturze a wywołanie wsteczne komponentu jest aktywne.
ResizeFcn	Jest wykonywane wtedy, gdy użytkownik zmienia rozmiar obiektu <b>Panel</b> lub <b>Button Group</b> .
SelectionChangeFcn	Jest wykonywane wtedy, gdy użytkownik wybierze inny przycisk opcji ( <b>Radio Button</b> ) lub przełącznika ( <b>Toggle Button</b> )

Programowanie interfejsu graficznego rozpoczyna się od określenia właściwości jego obiektów. W pierwszej kolejności należy definiować wartości własności **Tag** użytych obiektów GUI. Wartość własności **Tag** jest bowiem wykorzystywana do automatycznego tworzenia funkcji związanych z wywołaniami wstecznymi dla danego obiektu (np. **wartość\_Tag\_Callback**) oraz do aktualizacji struktury handles (dodanie elementu/pola struktury dla każdego nowo utworzonego obiektu wg schematu `handles.wartość_Tag`) co pozwala na łatwe pobieranie/modyfikowanie wartości własności użytych obiektów za pomocą funkcji `get` i `set`.

Określanie i modyfikację wartości własności obiektów inspektor właściwości (**Property Inspector**) (Rysunek 3).

W celu uruchomienia inspektora właściwości dla danego obiektu można:

- kliknąć obiekt dwukrotnie,
- kliknąć przycisk  na pasku narzędzi **GUIDE Layout Editor**
- z menu **View** wybrać opcję **Property Inspector**,
- z lokalnego menu kontekstowego wywołanego prawym przyciskiem myszy wybrać opcję **Property Inspector**.



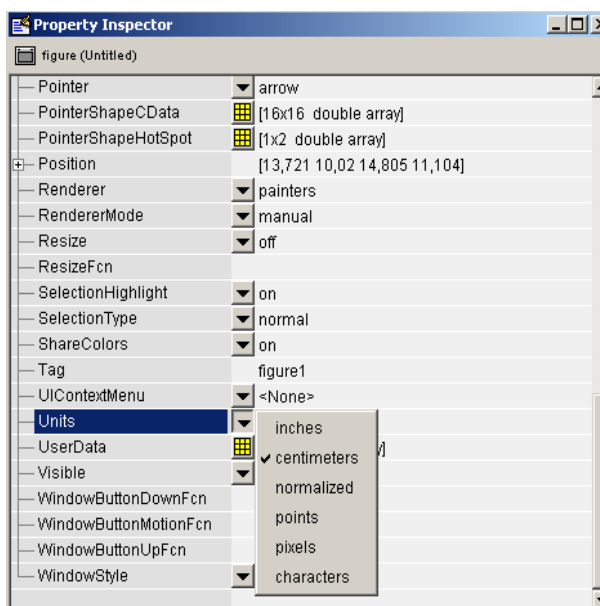
Rysunek 3: Okno inspektora właściwości obiektu (**Property Inspector**)

Uruchomienia wykonanego projektu można dokonać albo wybierając opcję **Run** z menu **Tools**, albo naciskając przycisk  na pasku narzędzi **GUIDE Layout Editor**.

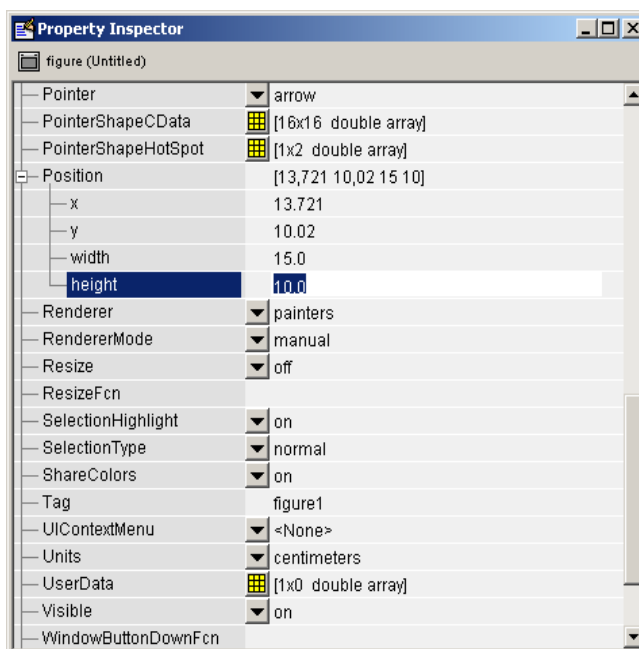
## 5. Wykonanie ćwiczenia

### Zadanie 1

1. Uruchom aplikację **GUIDE** wybierając „puste” GUI (**Blank GUI (Default)**)
2. Ustal wielkość otwieranego okna na  $10 \times 15$  cm. W tym celu najpierw określ nową wartość dla właściwości **Units** na centymetry - **centimeters** (Rysunek 4).



Rysunek 4: Ustalenie wartości właściwości **Units**

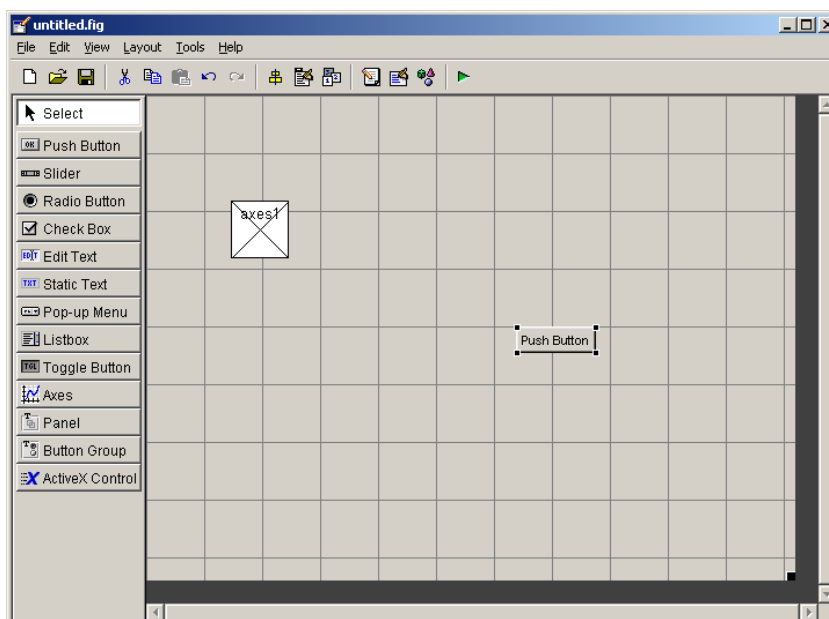


Rysunek 5: Ustalenie wartości właściwości **Position**

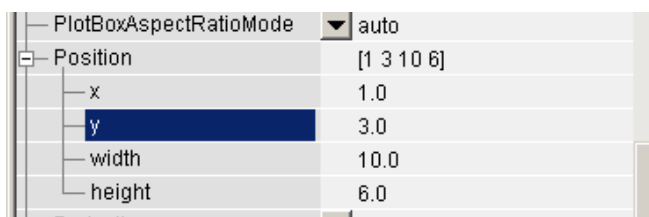
Następnie naciskając znak „+” przy zakładce właściwości **Position** ustal wartości pól: **width** na **15** oraz **height** na **10**. (Zatwierdzamy klawiszem **ENTER**) (Rysunek 5). Zamknij okno inspektora właściwości.

3. Wstaw osie układu współrzędnych (**Axes**) i przycisk polecenia (**Push button**) (Rysunek 6)
4. Otwórz **Property Inspector** obiektu **Axes**.
5. Zmień wartości własności: **Tag** na '**Osie**', **Units** na centymetry oraz własności **Position**: wielkość wykresu na **10x6** (jak w p. 2) i pozycje jego lewego dolnego rogu na:  $x=1$ ,  $y=3$  (w stosunku do lewego dolnego rogu obszaru tworzonego interfejsu) (Rysunek 7)
6. Otwórz **Property Inspector** obiektu **Push Button**.
7. Zmień wartości właściwość: **Tag** na '**Push\_Rysuj**', a **String** na '**Rysuj**'. Dzięki temu po włączeniu GUI nasz przycisk będzie miał etykietę 'Rysuj'.
8. Jeśli to potrzebne, przesuń obiekt **Push Button** tak aby nie kolidował on z obiektem **Axes**. (Rysunek 8)
9. Uruchom aplikację.

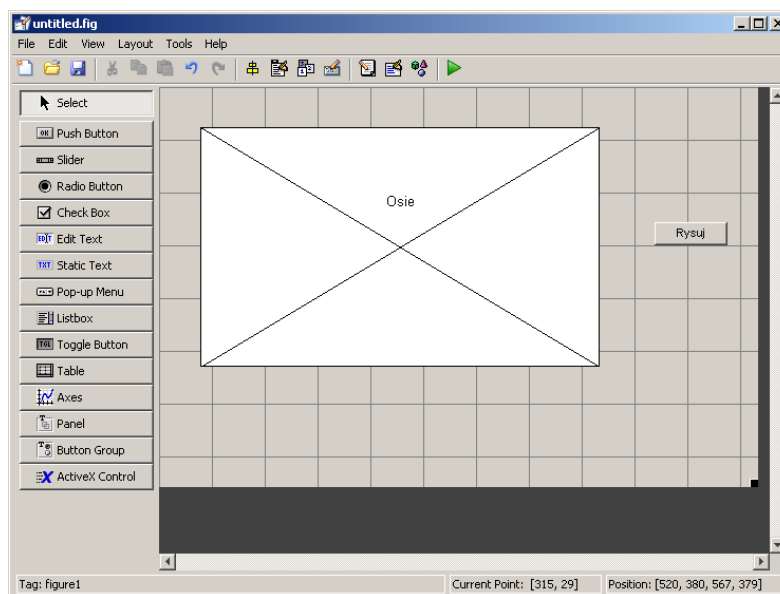
MATLAB zada pytanie, czy chcemy zapisać zmiany. (Rysunek 9) Bez zapisania zmian nie da się uruchomić GUI.



Rysunek 6: Wstawienie osi układu współrzędnych (*Axes*) i przycisku polecenia (*Push Button*)

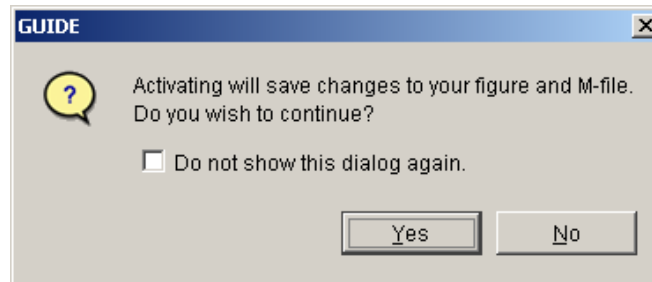


Rysunek 7: Określenie pozycji lewego górnego rogu obiektu



Rysunek 8: Przykładowe rozmieszczenie obiektów w *Layout Editor*






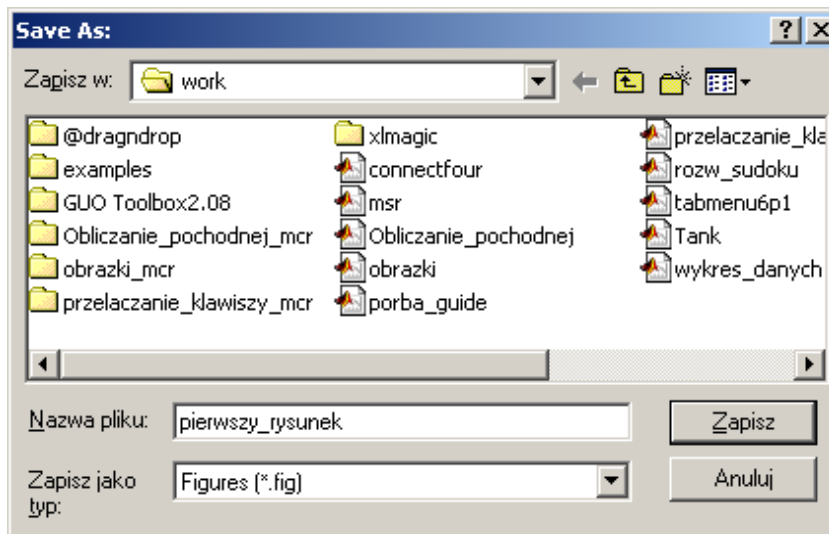
Rysunek 9: Pytanie MATLAB'a o zapisanie zmiany

Po potwierdzeniu (kliknięciu przycisku *Yes*) nastąpi przejście do domyślnego katalogu roboczego (Rysunek 10).

10. Zapisz plik pod nazwą „pierwsze\_gui.fig”. Po zapisaniu automatycznie zostanie wygenerowany m-file, o tej samej nazwie co nazwa plik *.fig*. Wspólnie tworzą one informacje dla MATLAB'a jak ma wyglądać GUI (Rysunek 11). GUI już „działa” – można przyciskać przycisk 'Rysuj'. Co się stało?

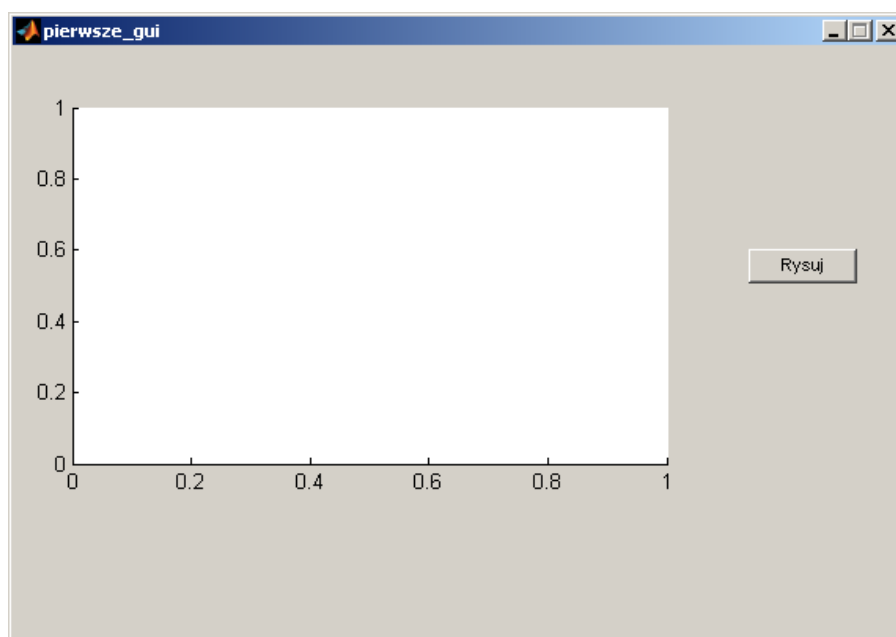
Przyciśnięcie przycisku 'Rysuj' nie powoduje żadnej reakcji, ponieważ nie została określona akcja dla wywołania wstecznego (rysowanie obiektu `peaks(12)`).

11. Zamknij okno uruchomionego GUI i przejdź do m-pliku.
12. Dzięki określeniu wartości właściwości *Tag* specyficznej dla każdego z użytych obiektów można łatwo odnaleźć funkcję, w której zdefiniujemy akcję wykonywaną jako wywołanie wsteczne (*Callback*). Nazwa funkcji tworzona jest automatycznie z wykorzystaniem wartości właściwości *Tag* wg schematu: *wartość\_Tag\_Callback*.
13. Kliknięcie przycisku *Show function*  na pasku narzędzi *Edytora/Debuggera* powoduje rozwinięcie listy nazw funkcji zawartych w danym m-pliku.



Rysunek 10: Okno zapisywanie pliku GUI

14. Wybierz z listy funkcję o nazwie *Push\_Rysuj\_Callback* (Rysunek 12).
15. Ustaw kursor pod linia definicji funkcji i jej komentarzami (Rysunek 13).
16. Wpisz polecenie: `peaks(12)`. Zwróć uwagę, że wpisanie polecenia bez średnika spowoduje wyświetlenie zwracanych wartości w głównym oknie MATLAB.
17. Zapisz m-plik.
18. Ponownie uruchom projektowane GUI. Przyciśnij przycisk 'Rysuj'. Co się stało?
19. Zamknij okno uruchomionego GUI i przejdź do okna *Layout Editor*.



Rysunek 11: Wygląd GUI

20. Dodaj suwak, który umożliwi ustalenie wartości parametru wejściowego  $n$  dla polecenia `peaks(n)` (Rysunek 14).
21. Najpierw wstaw obiekt *Slider*.
22. Ustal wielkość suwaka na: *height* = 0.45 i *width* = 10 cm (pamiętaj o zmianie jednostek - *Units* - na centymetry).
23. Ustaw teraz w jednej linii wykres i obiekt *Slider*. W tym celu z menu *Tools* wybierz *Grid and Rulers* (Rysunek 14).

```

1 function varargout = pierswsze_gui(varargin)
2 % PIERWSZE_GUI M-file for pierswsze_gui.
3 %
4 % PIERWSZE_GUI, by itself, creates a new PIERWSZE_GUI or raises the existing
5 % singleton*.
6 %
7 % H = PIERWSZE_GUI returns the handle to a new PIERWSZE_GUI or the handle to
8 % the existing singleton*.
9 %
10 % PIERWSZE_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
11 % function named CALLBACK in PIERWSZE_GUI.M with the given input arguments.
12 %
13 % PIERWSZE_GUI('Property','Value',...) creates a new PIERWSZE_GUI or raises the
14 % existing singleton*. Starting from the left, property value pairs are
15 % applied to the GUI before pierswsze_gui_OpeningFcn gets called. An
16 % unrecognized property name or invalid value makes property application
17 % stop. All inputs are passed to pierswsze_gui_OpeningFcn via varargin.
18 %
19 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
20 % instance to run (singleton)".
21 %
22 % See also: GUIDE, GUIDATA, GUIHANDLES
23 %
24 % Edit the above text to modify the response to help pierswsze_gui
25 %
26 % Last Modified by GUIDE v2.5 04-May-2010 19:38:03
27 %
28 % Begin initialization code - DO NOT EDIT
29 gui_Singleton = 1;
30 gui_State = struct('gui_Name',       mfilename, ...
31                   'gui_Singleton',  gui_Singleton, ...
32                   'gui_State',      gui_State, ...
33                   'gui_Callback',    []);
34 %
35 % Only allow one instance to run.
36 if exist('gui_Singleton','var') && gui_Singleton == 1
37     return;
38 end
39 gui_State.gui_Callback = str2func(varargin{1});
40
41 if nargin < 2
42     hObject = [];
43 else
44     hObject = varargin{1};
45 end
46 if nargin < 3
47     eventData = [];
48 else
49     eventData = varargin{2};
50 end
51 handles = varargin{3};
52
53 % Choose default command line output for pierswsze_gui
54 handles.output = hObject;
55
56 % Update handles structure
57 guidata(hObject, handles);
58
59 % UIWAIT makes pierswsze_gui wait for user response (see UIRESUME)
60 % uiwait(handles.figure1);
61
62 % --- Outputs from this function are returned to the command line.
63 %
64 % function varargout = pierswsze_gui_OutputFcn(hObject, eventdata, handles)
65 %
66 % varargout cell array for returning output args (see VARARGOUT);
67 % hObject handle to figure
68 % eventdata reserved - to be defined in a future version of MATLAB
69 % handles structure with handles and user data (see GUIDATA)
70 %
71 % Get default command line output from handles structure
72 varargout{1} = handles.output;
73
74 % --- Executes on button press in Push_Rysuj.
75 %
76 function Push_Rysuj_Callback(hObject, eventdata, handles)
77 % hObject handle to Push_Rysuj (see GCBO)
78 % eventdata reserved - to be defined in a future version of MATLAB
79 % handles structure with handles and user data (see GUIDATA)
80
81

```

Rysunek 12: Wybór funkcji wywołania wstecznego

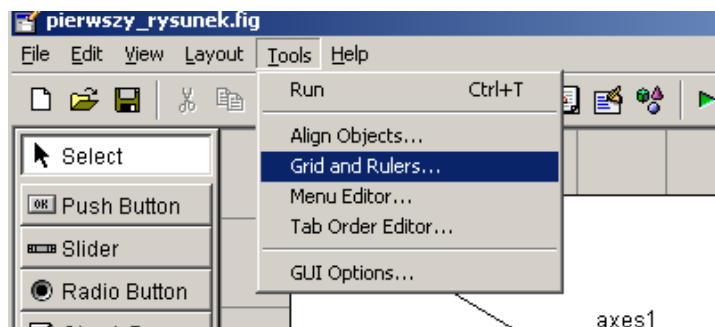
```

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 pierswsze_gui (see VARARGIN)
54
55 % Choose default command line output for pierswsze_gui
56 handles.output = hObject;
57
58 % Update handles structure
59 guidata(hObject, handles);
60
61 % UIWAIT makes pierswsze_gui wait for user response (see UIRESUME)
62 % uiwait(handles.figure1);
63
64 % --- Outputs from this function are returned to the command line.
65 %
66 % function varargout = pierswsze_gui_OutputFcn(hObject, eventdata, handles)
67 %
68 % varargout cell array for returning output args (see VARARGOUT);
69 % hObject handle to figure
70 % eventdata reserved - to be defined in a future version of MATLAB
71 % handles structure with handles and user data (see GUIDATA)
72 %
73 % Get default command line output from handles structure
74 varargout{1} = handles.output;
75
76 % --- Executes on button press in Push_Rysuj.
77 %
78 function Push_Rysuj_Callback(hObject, eventdata, handles)
79 % hObject handle to Push_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)

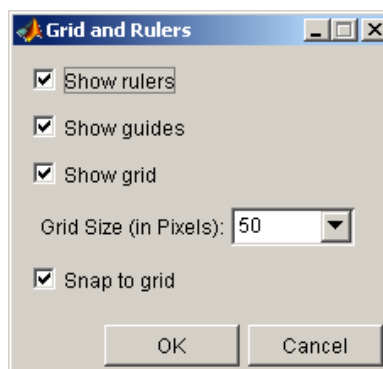
```

Rysunek 13: Funkcja *Push\_Rysuj\_Callback*

24. W otwartym oknie zaznacz opcję *Show rulers*.(Rysunek 15)



Rysunek 14: Wybór opcji *Grid and rulers*

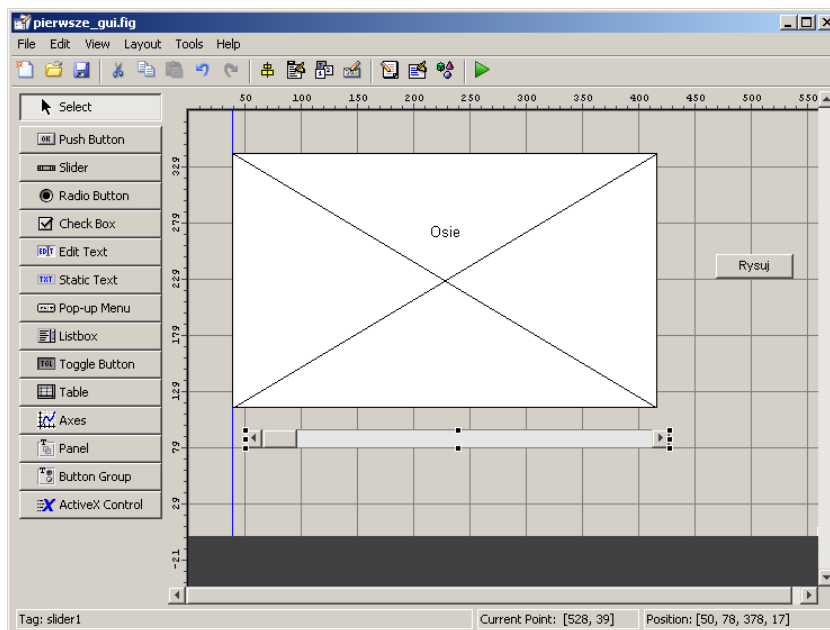


Rysunek 15: Zaznaczenie opcji linijek

25. Z lewej krawędzi okna *Layout Editor* wyciągnij linię. Dociągnij ją do lewego boku osi układu współrzędnych wykresu, tworząc prowadnicę, do której można wyrównywać obiekty (Rysunek 16).

Dociągnij obiekt *Slider* do utworzonej (niebieskiej) linii.

26. Zmień w *Property Inspector* właściwość *Tag* obiektu *Slider* na '*Suwak*'.
27. Zmień właściwości *Min* i *Max* na 2 i 50. Dzięki temu wartości odpowiadające położeniu suwaka będą się zawierały w podanym zakresie (pomiędzy *Min* i *Max*). Domyślnie są to wartości od 0 do 1.
28. Zmień wartość właściwości *Value*. Jest to wartość, którą przyjmuje suwak po uruchomieniu GUI. Domyślnie jest to 0. Ponieważ zero jest poza zakresem wartości naszego suwaka, musimy zmienić wartość właściwości *Value*. Ustaw najmniejszą możliwą wartość, tzn. 2 (Rysunek 17).
29. Zamknij okno *Property Inspector*.
30. Zapisz GUI (w oknie *Layout Editor*)



Rysunek 16: Linia wyrównywania



Rysunek 17: Zmiana wartości *Value*

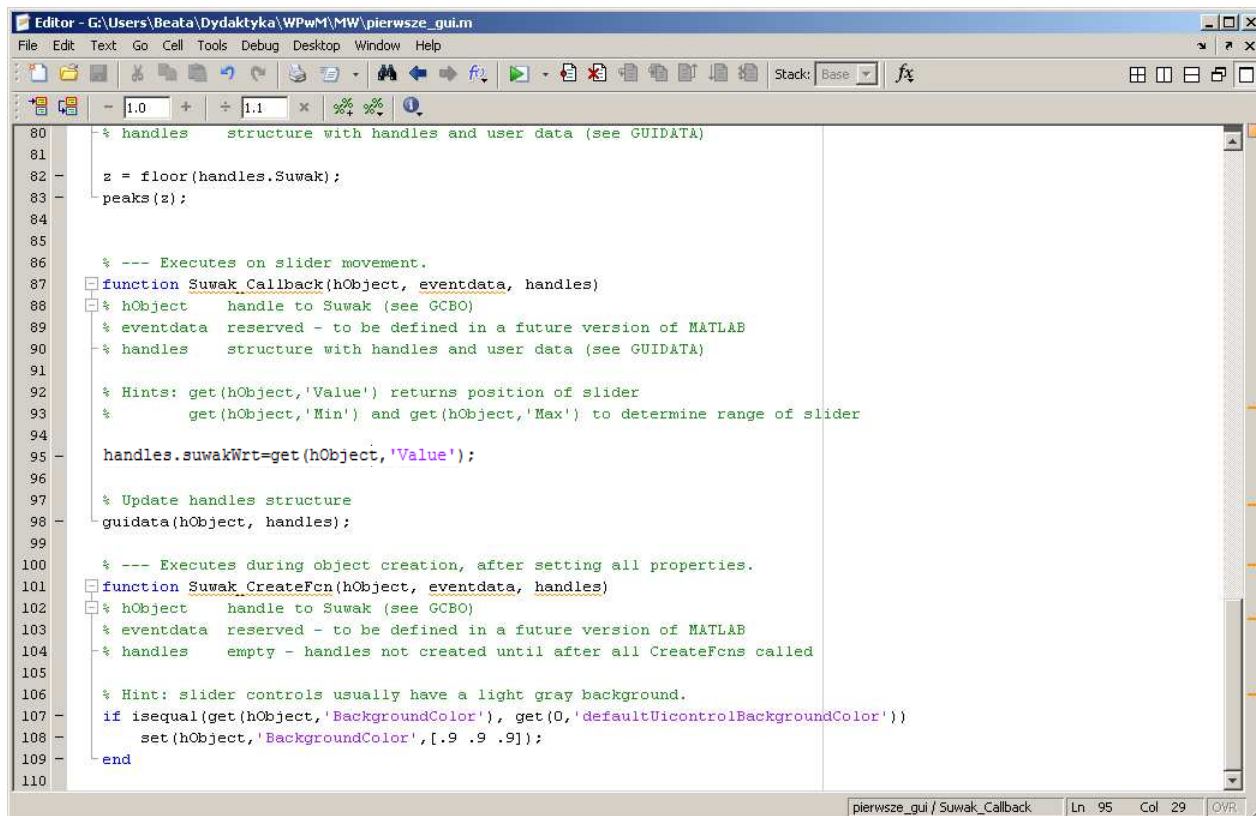
31. Przejdź w m-pliku, do funkcji *Suwak\_Callback*.

Funkcja `get(hObject, 'Value')` zwraca aktualną wartość suwaka (w formacie double). Wartość tę należy zapamiętać w strukturze `handles`, tak aby można ją było wykorzystać w momencie przyciśnięcia przycisku *Rysuj* jako parametr wejściowy dla polecenia `peaks(n)`.

W ciele funkcji *Suwak\_callback* wpisz `handles.suwakWrt=get(hObject, 'Value')`. Dzięki temu w strukturze `handles` w polu o nazwie `suwakWrt` będzie zapisana wartość odpowiadająca aktualnej pozycji suwaka. Następnie musimy zapisać (odświeżyć) strukturę

handles. Odbywa się to za pomocą komendy `guidata(hObject, handles)` (Rysunek 18)

**Uwaga:** WSZELKIE ZMIANY WARTOŚCI ZMIENNYCH POWINNY BYĆ ZAPISYWANE JAKO HANDLES I ODSWIEŻANE KOMENDĄ `guidata(hObject, handles); !!!!!!!`



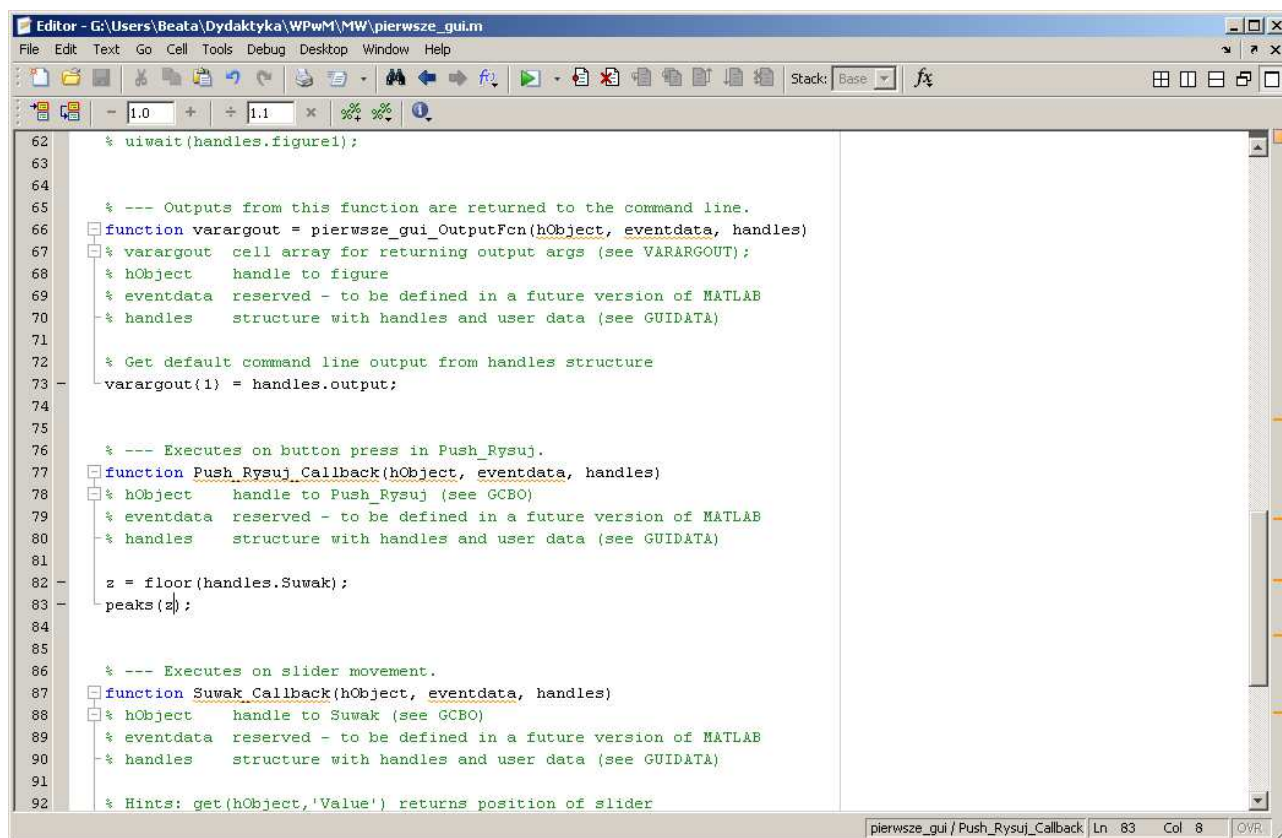
```
80 % handles structure with handles and user data (see GUIDATA)
81
82 z = floor(handles.Suwak);
83 peaks(z);
84
85
86 % --- Executes on slider movement.
87 function Suwak_Callback(hObject, eventdata, handles)
88 % hObject handle to Suwak (see GCBO)
89 % eventdata reserved - to be defined in a future version of MATLAB
90 % handles structure with handles and user data (see GUIDATA)
91
92 % Hints: get(hObject,'Value') returns position of slider
93 % get(hObject,'Min') and get(hObject,'Max') to determine range of slider
94
95 handles.suwakWrt=get(hObject,'Value');
96
97 % Update handles structure
98 guidata(hObject, handles);
99
100 % --- Executes during object creation, after setting all properties.
101 function Suwak_CreateFcn(hObject, eventdata, handles)
102 % hObject handle to Suwak (see GCBO)
103 % eventdata reserved - to be defined in a future version of MATLAB
104 % handles empty - handles not created until after all CreateFcns called
105
106 % Hint: slider controls usually have a light gray background.
107 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
108 set(hObject,'BackgroundColor',[.9 .9 .9]);
109 end
110
```

Rysunek 18: Kod funkcji *Suwak\_Callback*

32. Odczytaj wartość właściwości *Value* obiektu *Suwak* w funkcji *Push\_Rysuj\_Callback*. Odczytanie odbywa się przez przypisanie zmiennej *z* wartości pola *handles.suwak*. Ogranicz wartość zmiennej *z* do wartości całkowitych korzystając z polecenia `floor`: `z=floor(handles.suwakWrt)`. Zmodyfikuj teraz funkcję *Push\_Rysuj\_Callback* tak, aby rysowała `peaks(z)` zamiast `peaks(12)` (Rysunek 19).
33. Zapisz m-plik.
34. Uruchommy GUI. Przesuwając suwak, a następnie naciskając przycisk 'Rysuj' uzyskaj różne wykresy funkcji `peaks`.
35. Zamknij GUI.
36. W oknie *Layout Editor* dodaj pole tekstowe, które będzie zawierało wartość suwaka. Niech będzie to *Static Text*, wstawiony mniej więcej w połowie długości suwaka pod nim (Rysunek 20).

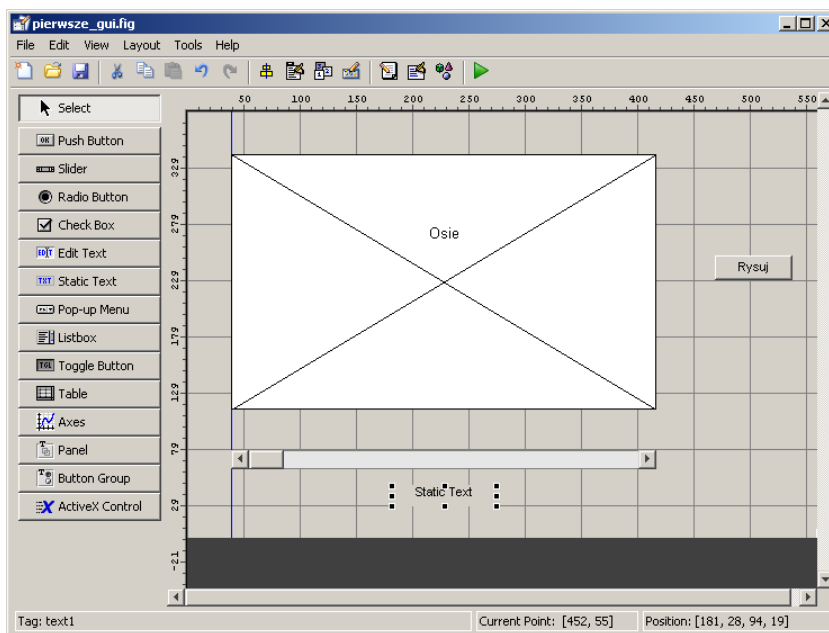


37. Zmień właściwość **Tag** obiektu **Static Text** na **wartosc\_Suwaka** i usuń wpis we właściwości **String** (Rysunek 21).

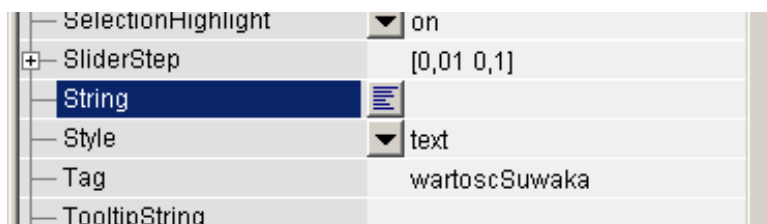


```
62 % uiwait(handles.figure1);
63
64
65 % --- Outputs from this function are returned to the command line.
66 function varargout = pierwsze_gui_OutputFcn(hObject, eventdata, handles)
67 % varargout cell array for returning output args (see VARARGOUT);
68 % hObject handle to figure
69 % eventdata reserved - to be defined in a future version of MATLAB
70 % handles structure with handles and user data (see GUIDATA)
71
72 % Get default command line output from handles structure
73 varargout{1} = handles.output;
74
75
76 % --- Executes on button press in Push_Rysuj.
77 function Push_Rysuj_Callback(hObject, eventdata, handles)
78 % hObject handle to Push_Rysuj (see GCBO)
79 % eventdata reserved - to be defined in a future version of MATLAB
80 % handles structure with handles and user data (see GUIDATA)
81
82 z = floor(handles.Suwak);
83 peaks(z);
84
85
86 % --- Executes on slider movement.
87 function Suwak_Callback(hObject, eventdata, handles)
88 % hObject handle to Suwak (see GCBO)
89 % eventdata reserved - to be defined in a future version of MATLAB
90 % handles structure with handles and user data (see GUIDATA)
91
92 % Hints: get(hObject,'Value') returns position of slider
```

Rysunek 19: Kod funkcji **Push\_Rysuj\_Callback**



Rysunek 20: Dodanie obiektu **Static Text**



Rysunek 21: Usunięcie wartości właściwości **String**

38. Zamknij okno **Property Inspector**.
39. Zapisz GUI (w oknie **Layout Editor**)
40. Przejdź do m-pliku do funkcji **Suwak\_Callback**. Przed zapisaniem zmian do struktury handles dopisz komendę zmieniająca wartość pola **wartosc\_Suwaka** na łańcuch znakowy stanowiący wartość właściwości **String** obiektu **Static Text** i:  
`set(handles.wartosc_Suwaka, 'String', floor(get(hObject, 'Value')))`  
 (Rysunek 22).

Przy pomocy polecenia `set` możemy zmieniać wszystkie właściwości dostępne w **Property Inspector**. Wystarczy identyfikator obiektu zapisanego w strukturze handles (`handles.Tag`), nazwa właściwości (w pojedynczym cudzysłowie) i wartość końcowa dla tej właściwości.

```

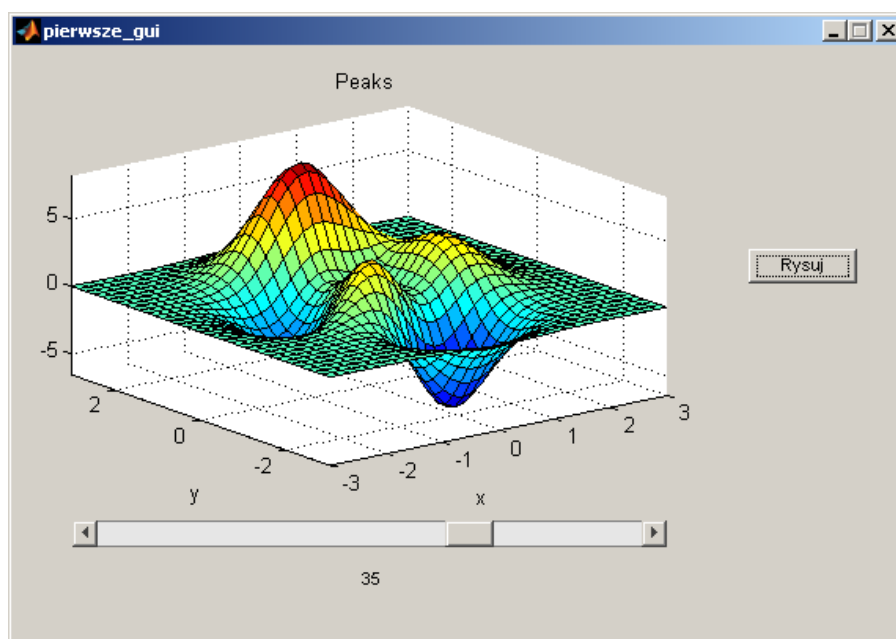
80  % handles structure with handles and user data (see GUIDATA)
81
82  z = floor(handles.Suwak);
83  peaks(z);
84
85
86  % --- Executes on slider movement.
87  function Suwak_Callback(hObject, eventdata, handles)
88  % hObject handle to Suwak (see GCBO)
89  % eventdata reserved - to be defined in a future version of MATLAB
90  % handles structure with handles and user data (see GUIDATA)
91
92  % Hints: get(hObject,'Value') returns position of slider
93  % get(hObject,'Min') and get(hObject,'Max') to determine range of slider
94
95  handles.Suwak = get(hObject, 'Value');
96  handles.suwakWrt=get(hObject, 'Value');
97  set(handles.wartosc_Suwaka, 'String', floor(handles.suwakWrt));
98  % Update handles structure
99  guidata(hObject, handles)
100
101  % --- Executes during object creation, after setting all properties.
102  function Suwak_CreateFcn(hObject, eventdata, handles)
103  % hObject handle to Suwak (see GCBO)
104  % eventdata reserved - to be defined in a future version of MATLAB
105  % handles empty - handles not created until after all CreateFcns called
106
107  % Hint: slider controls usually have a light gray background.
108  if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
109  set(hObject,'BackgroundColor',[.9 .9 .9]);
110  end

```

Rysunek 22: Kod zmodyfikowanej funkcji **Suwak\_Callback**



41. Zapisz m-plik.
42. Uruchom zmodyfikowane GUI. Po przesunięciu suwaka (i puszczeniu go), pod suwakiem powinna pojawić się liczba całkowita, której wartość odpowiada wartości parametru wejściowego funkcji `peaks` (Rysunek 23).
43. Bezpośrednio po uruchomieniu GUI, zanim suwak zostanie przesunięty, wartość zmiennej `z` (lub `handles.suwakWrt`) jest nieokreślona. Przyciśnięcie przycisku 'Rysuj' powoduje wówczas narysowanie wykresu `peaks` (`z`) dla przypadkowej wartości zmiennej `z`. Zmień to.

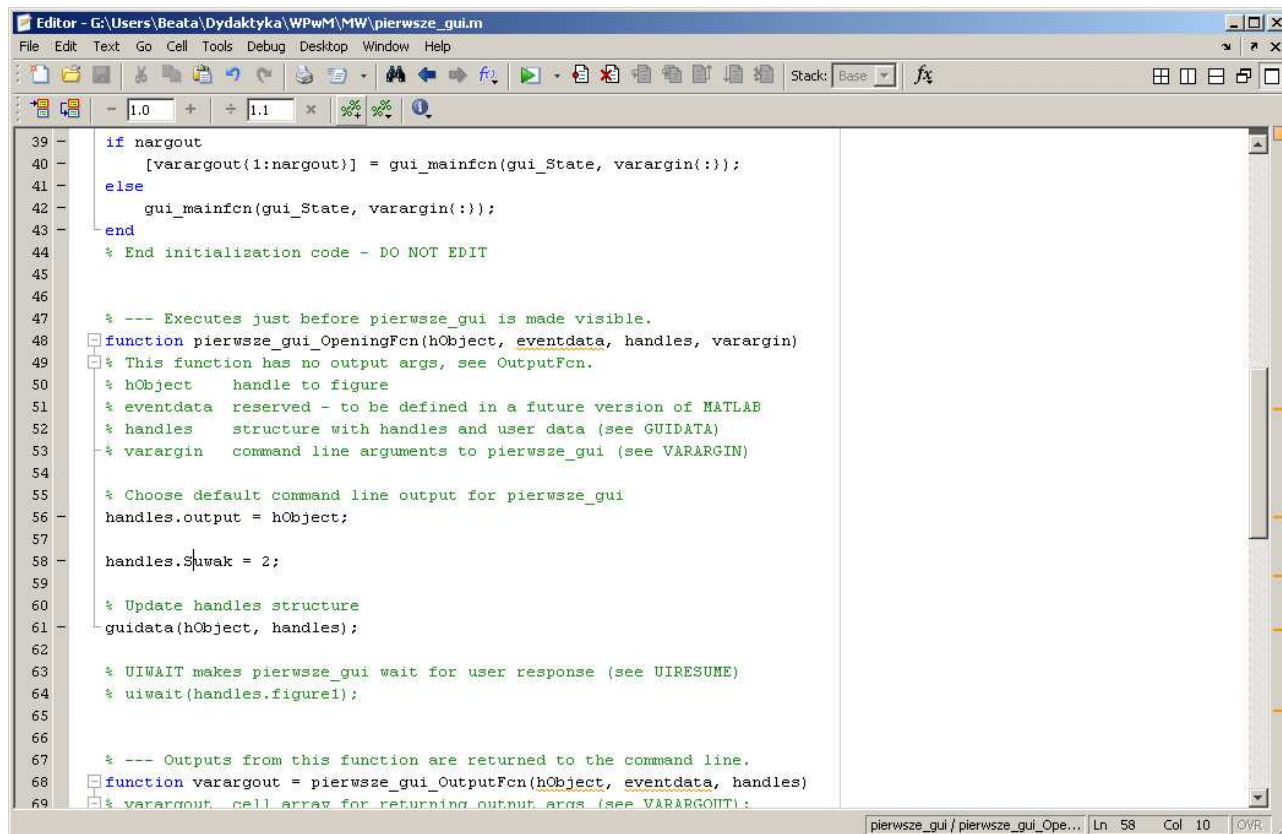


Rysunek 23: Widok uruchomionej aplikacji dla wartości suwaka równej 35

44. W momencie uruchamiania GUI uruchamiana jest funkcja ***nazwaProgramu\_OpeningFcn***.  
  
Dodaj w funkcji ***pierwsze\_gui\_OpeningFcn*** (przed `guidata(hObject, handles);`) następujący wiersz `handles.suwakWrt=2;` (Rysunek 24).
45. Zapisz m-plik.
46. W oknie okno ***Property Inspector*** zmień wartość właściwości ***String*** obiektu ***wartosc\_Suwaka*** na **2**. Dzięki temu w momencie włączenia programu od początku widoczna będzie wartość suwaka.
47. Zamknij okno ***Property Inspector*** i zapisz GUI.
48. Ponownie uruchom GUI i przetestuj jego działanie.

## Zadanie 2

Wstaw nowy przycisk (dobierz właściwy element), który będzie aktywował i dezaktywował możliwość obracania wykresu. Nazwij go 'Rotacja' i zmień jego właściwość *Tag* (np. na *Button\_Rotacja*). Kolejne użycia przycisku 'Rotacja' powinny przełączać (włączać/wyłączać) możliwość obracania wykresu (wbudowana funkcja MATLAB'a `rotate3d`). W m-pliku wpisz odpowiedni kod.



```
39     if narginout
40         [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41     else
42         gui_mainfcn(gui_State, varargin{:});
43     end
44     % End initialization code - DO NOT EDIT
45
46
47     % --- Executes just before pierwsze_gui is made visible.
48     function pierwsze_gui_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 pierwsze_gui (see VARARGIN)
54
55     % Choose default command line output for pierwsze_gui
56     handles.output = hObject;
57
58     handles.Suwak = 2;
59
60     % Update handles structure
61     guidata(hObject, handles);
62
63     % UIWAIT makes pierwsze_gui wait for user response (see UIRESUME)
64     % uiwait(handles.figure1);
65
66
67     % --- Outputs from this function are returned to the command line.
68     function varargout = pierwsze_gui_OutputFcn(hObject, eventdata, handles)
69     % varargout cell array for returning output args (see VARARGOUT):
```

Rysunek 24: Kod zmodyfikowanej funkcji `pierwsze_gui_OpeningFcn`

## Sprawozdanie

### Ćwiczenie nr 5

L.p.	Imię i nazwisko	Grupa	Data

Punkt ćw./ L. punktów	Wynik	Uwagi prowadzącego
1/4		
2/1		