Informatyka

Wykład 5

Witold Dyrka
witold.dyrka@pwr.wroc.pl

16/04/2012

Program wykładów

0. Informatyka. Wprowadzenie do Matlaba (13.02.12)
1. Matlab dla programistów C/C++ (20.02.12)
2. Optymalizacja obliczeń. Grafika (05.03.12)
3. Złożone typy danych. Programowanie zorientowane obiektowo (OOP) (19.03.12)
4. OOP część 2 (02.04.12)
5. **Graficzny interfejs użytkownika** **(16.04.12)**
6. Obliczenia numeryczne (30.04.12)
7. Kolokwium (14.05.12)

Dzisiejszy wykład w oparciu o...

- B. Mrozek, Z. Mrozek. MATLAB i Simulink. Poradnik użytkownika. Wydanie III. Helion 2010. Rozdział 6.
- B. Stroustrup. Programowanie. Helion 2010. Rozdziały 12-16
- MATLAB Product Documentation
  - Handle Graphics Objects, http://www.mathworks.com/help/techdoc/creating_plots/f7-20419.html
  - Creating Graphical User Interfaces, http://www.mathworks.com/help/techdoc/creating_guis/bqz79mu.html
    - Examples of GUIDE GUIs
      - GUI with Multiple Axis: http://www.mathworks.com/help/techdoc/creating_guis/f6-15783.html
      - GUI to Interactively Explore Data in a Table http://www.mathworks.com/help/techdoc/creating_guis/brpat2g.html

---

- Dla dociekliwych:
  - http://undocumentedmatlab.com

Na poprzednich wykładach...

- Programowanie zorientowane obiektowo
  - enkapsulacja
  - dziedziczenie
  - polimorfizm
  - klasy uchwytowe
    - czyli przekazywanie obiektu przez referencję

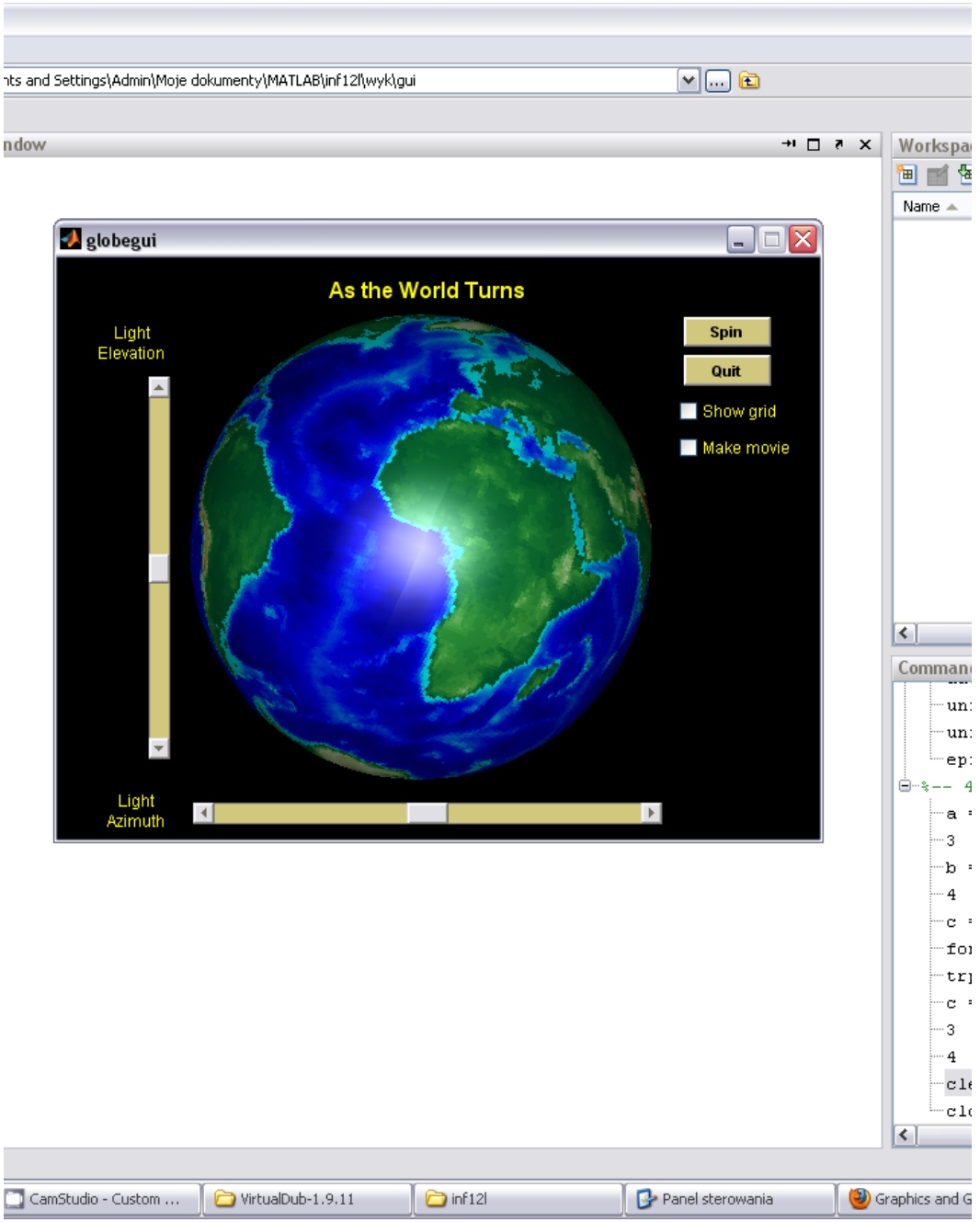
Program na dziś

- Grafika uchwytów
- Graficzny interfejs użytkownika
  - programowanie sterowane zdarzeniami
  - funkcje zwrotne (*callback*)
  - GUIDE – narzędzie tworzenia interfejsu

Elementy grafiki

- Ekran
  - Okno
    - kontrolki
    - wykresy
    - adnotacje

- Jak MATLAB reprezentuje elementy grafiki?
  - jako obiekty

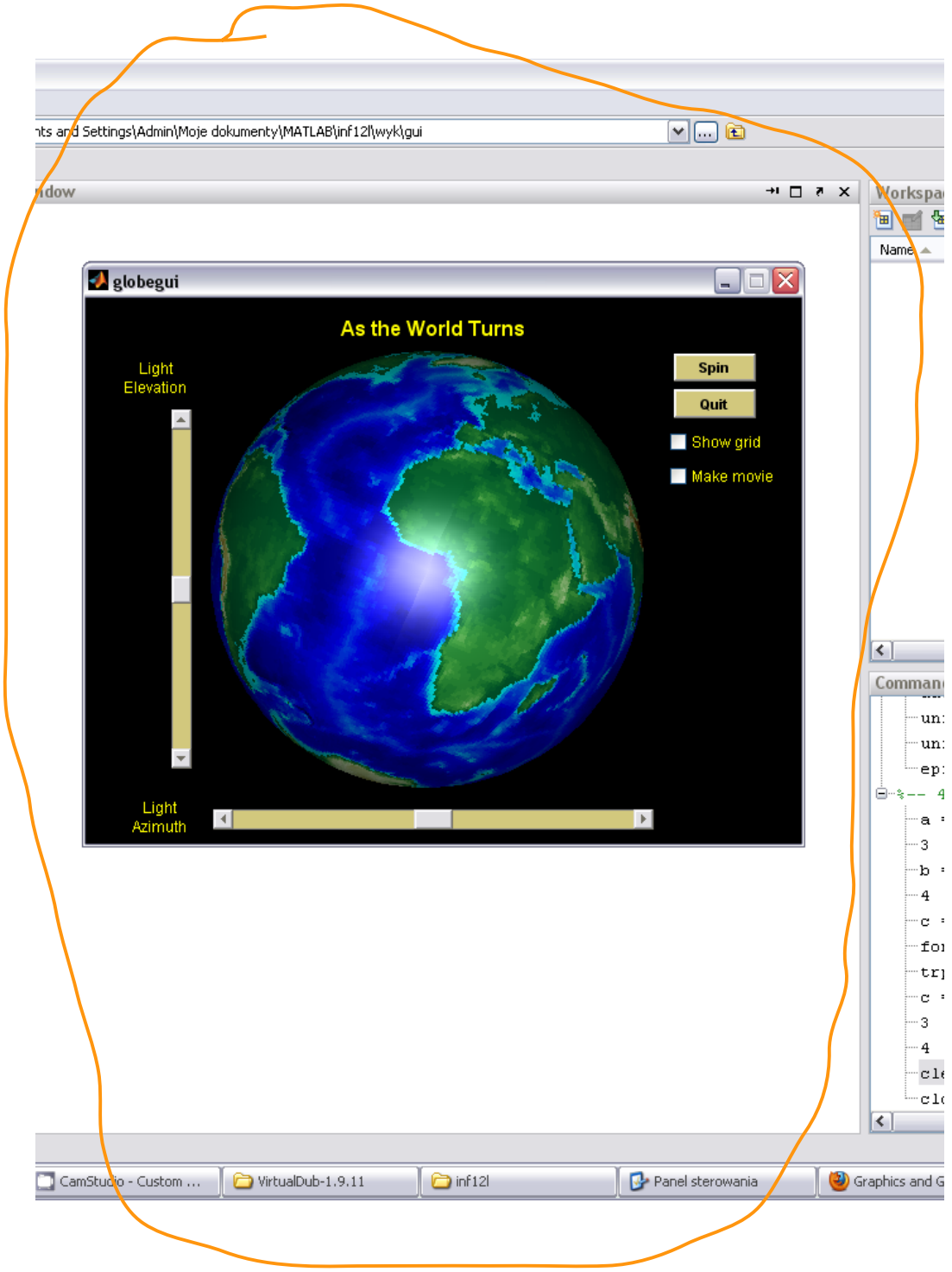


http://www.mathworks.com/help/techdoc/creating_guis/brob1jb-1.html

Elementy grafiki

- Ekran
  - Okno
    - kontrolki
    - wykresy
    - adnotacje

- Jak MATLAB reprezentuje elementy grafiki?
  - jako obiekty

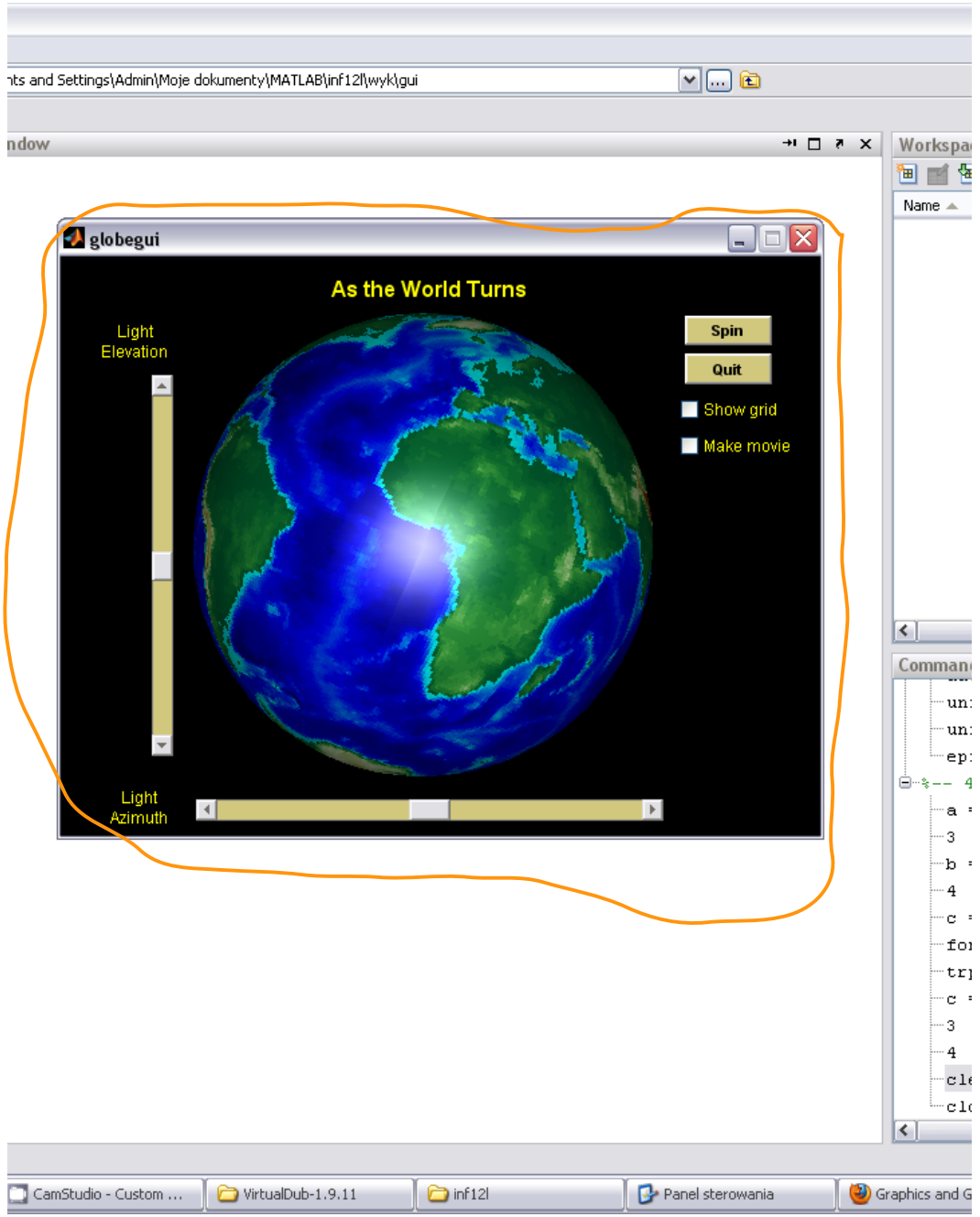


http://www.mathworks.com/help/techdoc/creating_guis/brob1jb-1.html

Elementy grafiki

- Ekran
  - Okno
    - kontrolki
    - wykresy
    - adnotacje

- Jak MATLAB reprezentuje elementy grafiki?
  - jako obiekty

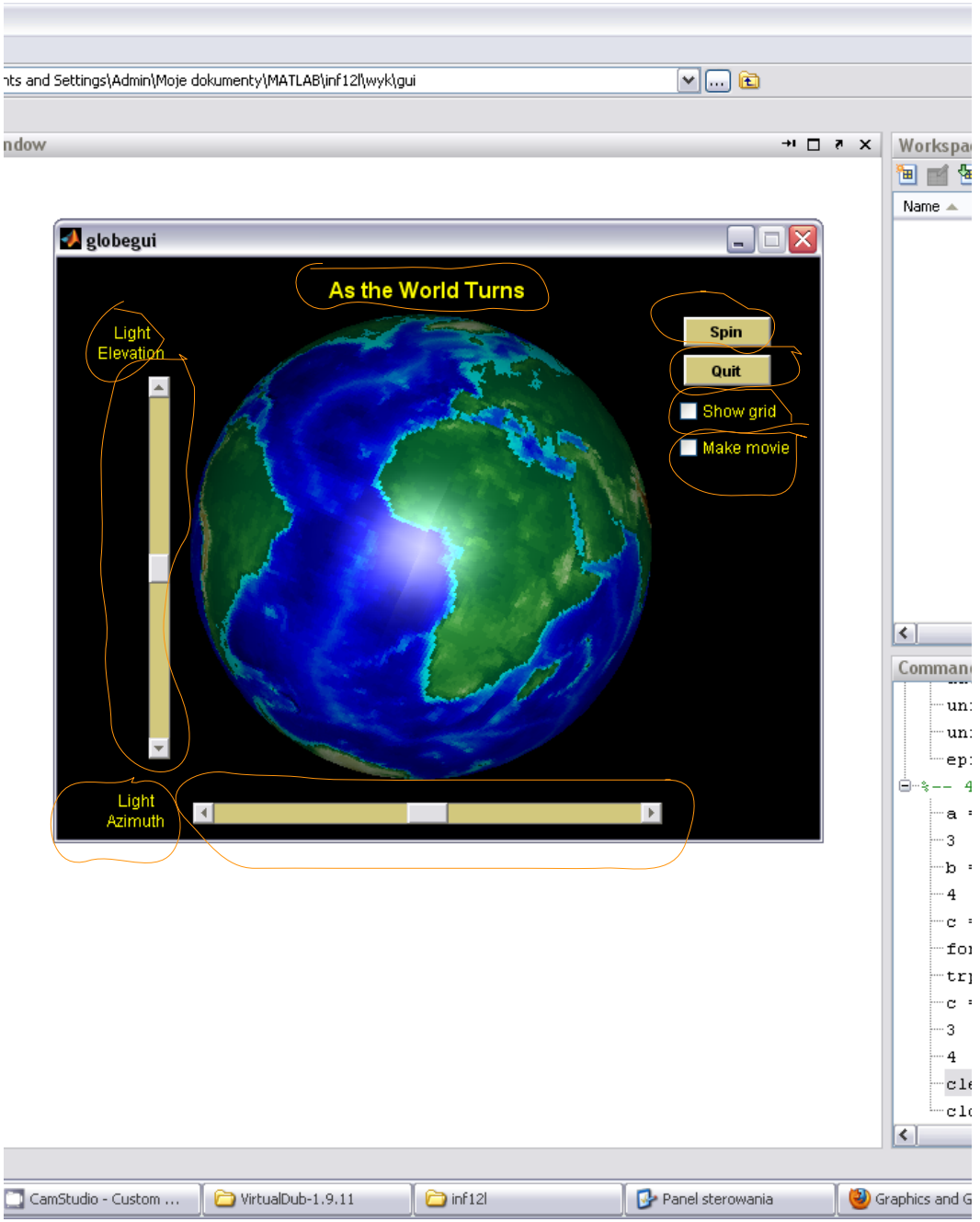


http://www.mathworks.com/help/techdoc/creating_guis/brob1jb-1.html

Elementy grafiki

- Ekran
  - Okno
    - kontrolki
    - wykresy
    - adnotacje

- Jak MATLAB reprezentuje elementy grafiki?
  - jako obiekty

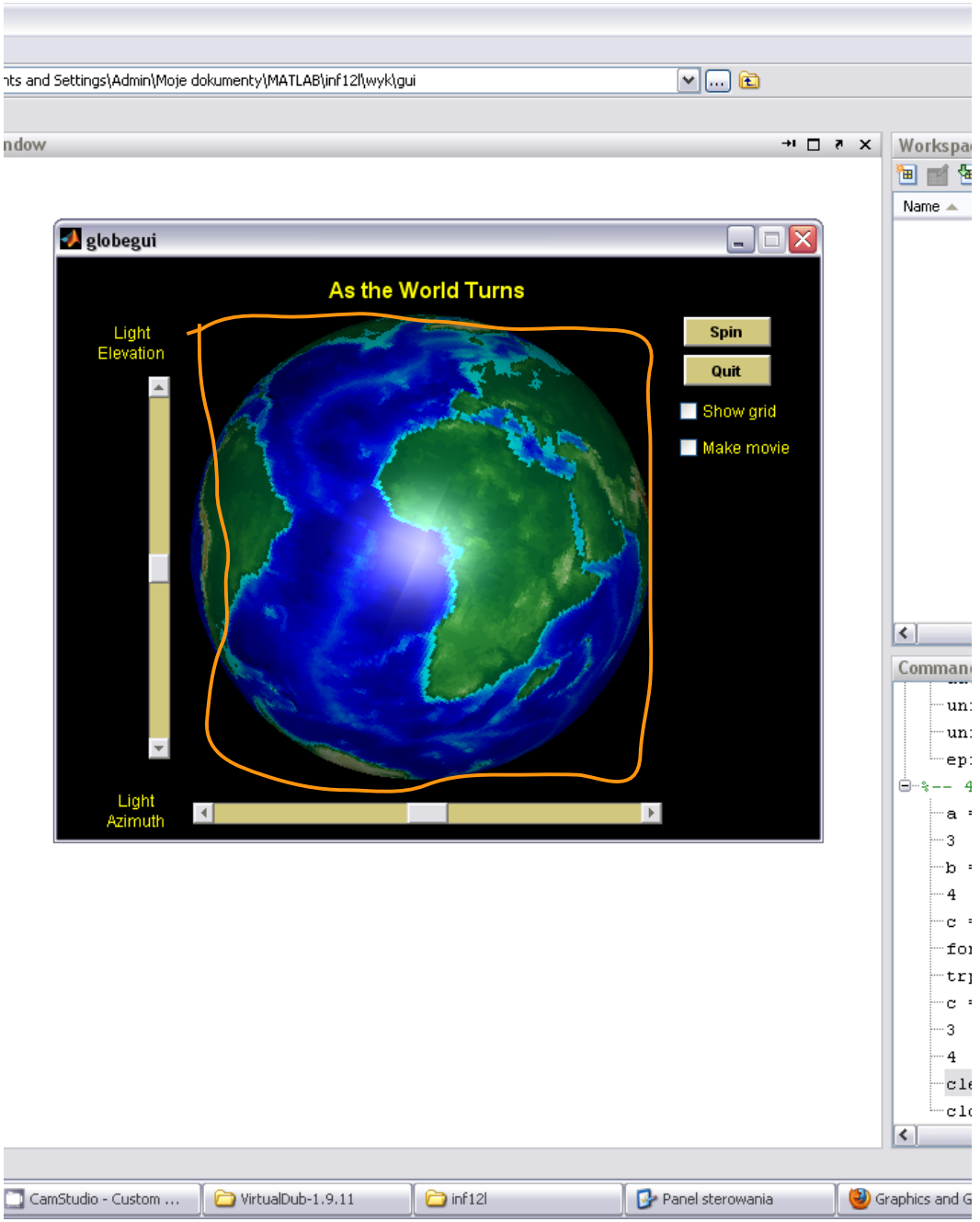


http://www.mathworks.com/help/techdoc/creating_guis/brob1jb-1.html

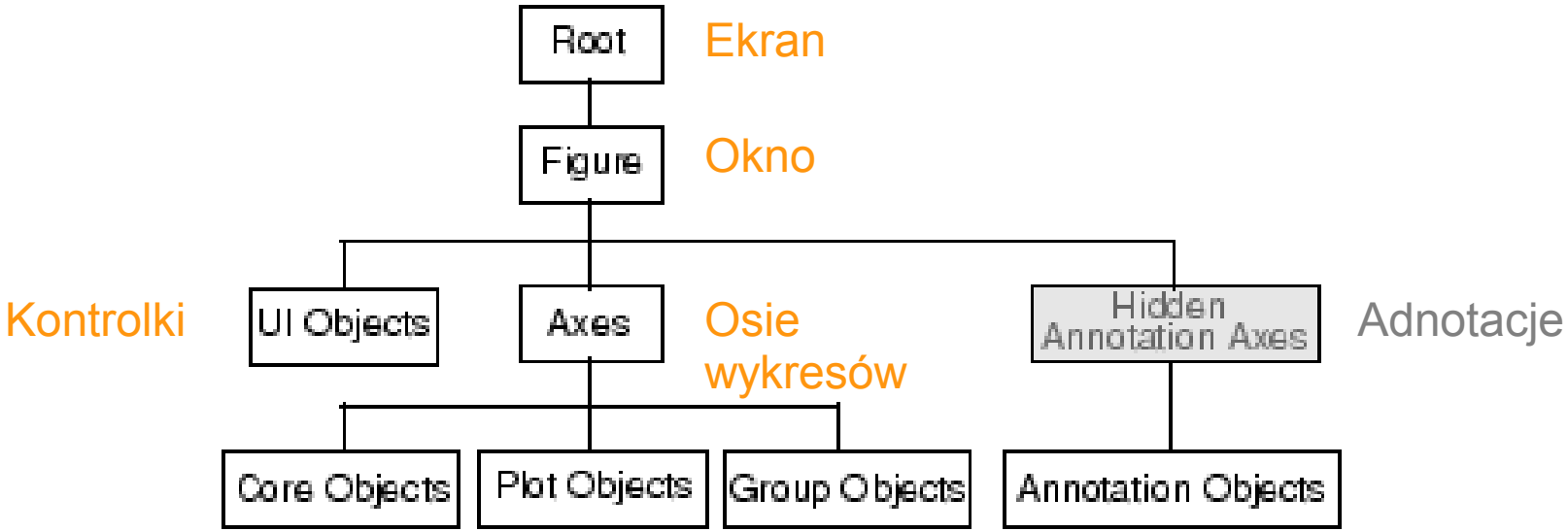
Elementy grafiki

- Ekran
  - Okno
    - kontrolki
    - wykresy
    - adnotacje

- Jak MATLAB reprezentuje elementy grafiki?
  - jako obiekty



Hierarchia obiektów graficznych



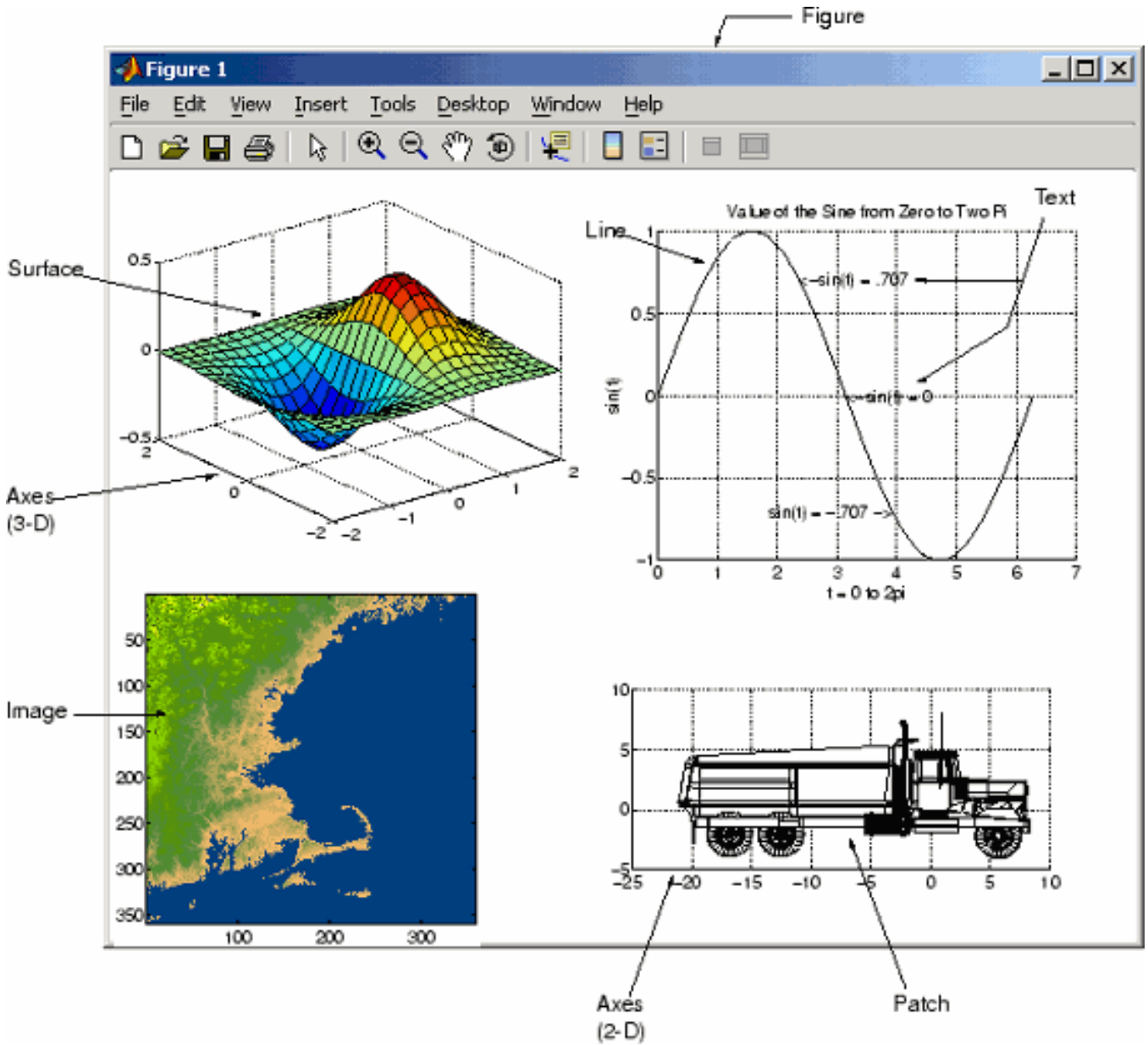
http://www.mathworks.com/help/techdoc/creating_plots/f7-41259.html

Obiekty, które można umieszczać na wykresie

- **Obiekty podstawowe** (*Core Objects*)
  - obrazek, linia, kształt prosty, wielokąt, powierzchnia 3D, tekst, puste osie współrzędnych
- **Wykresy** (*Plot Objects*)
  - tworzone funkcjami wysokiego poziomu
    - `plot, plot3, bar, scatter3, surf, mesh` itp.
- **Obiekty grupowe** (*Group Objects*: `hggroup,hgtransform`)
  - jednoczesne przekształcanie grupy obiektów

http://www.mathworks.com/help/techdoc/creating_plots/f7-41198.html

Obiekty podstawowe (tzw. prymitywy)



http://www.mathworks.com/help/techdoc/creating_plots/f7-57481.html

Wykresy

- **area**series - tworzone funkcją **area**
- **bar**series - tworzone funkcją **bar**
- **contour**group - tworzone funkcją **contour**
- **errorbar**series - tworzone funkcją **errorbar**
- lineseries - tworzone funkcjami **plot, plot3**
- **quiver**group - tworzone funkcjami **quiver, quiver3**
- **scatter**group - tworzone funkcjami **scatter, scatter3**
- **stair**series - tworzone funkcją **stairs**
- **stem**series - tworzone funkcjami **stem, stem3**
- surfaceplot - tworzone funkcjami **surf, mesh**

http://www.mathworks.com/help/techdoc/creating_plots/f7-45202.html

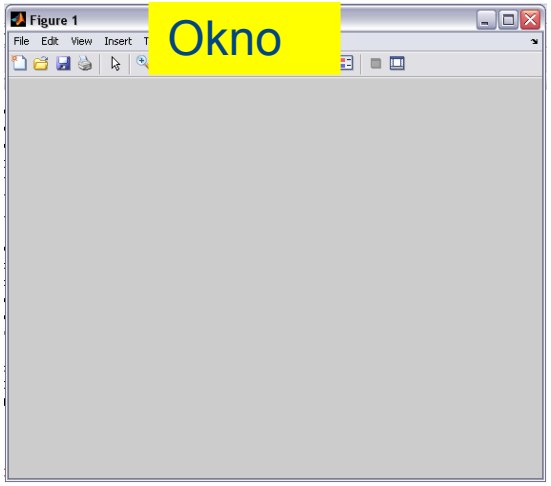
Obiekty graficzne

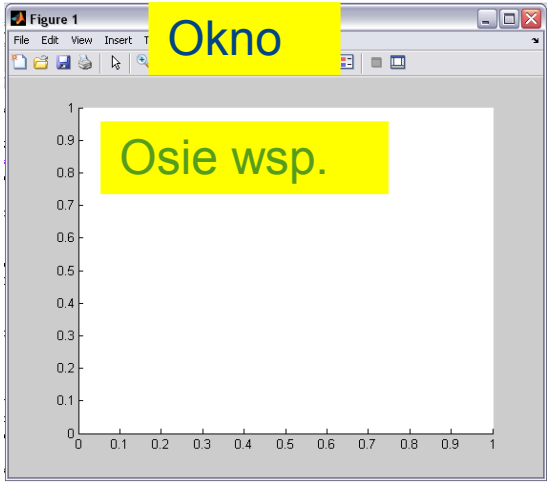
- Są klasy uchwytowej

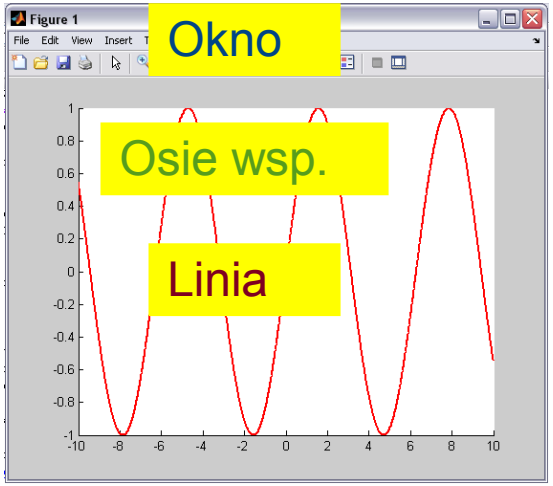
```
>> hf = figure

hf =  1          % uchwyt do rysunku (okna)

>> ha = axes

ha =  196.0021 % uchwyt do osi wsp.

>> hl = line(x,y,'Color','r','LineWidth',2)

hl =  197.0023 % uchwyt do linii
```

- ale nie klasy `handle` :/

```
>> isa(hl,'handle')     % czy hl klasy handle?
ans = 0                 % NIE

>> ishandle(hl)         % czy hl klasy
ans = 1                 % graficznej? TAK
```



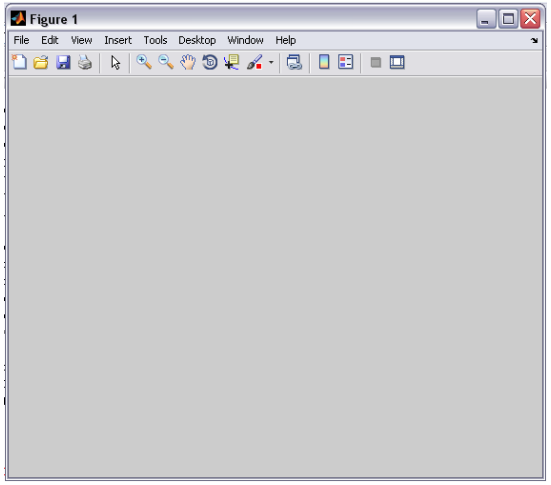


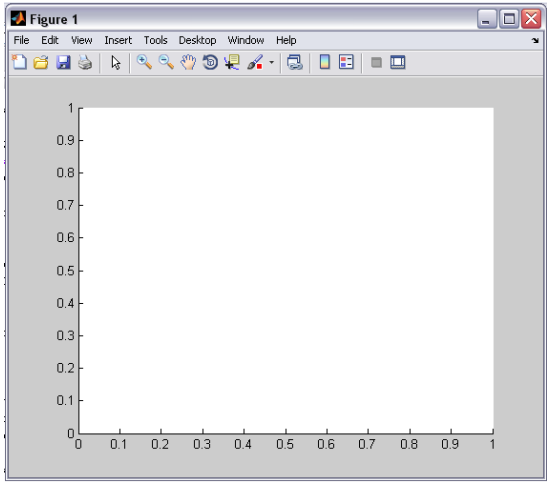


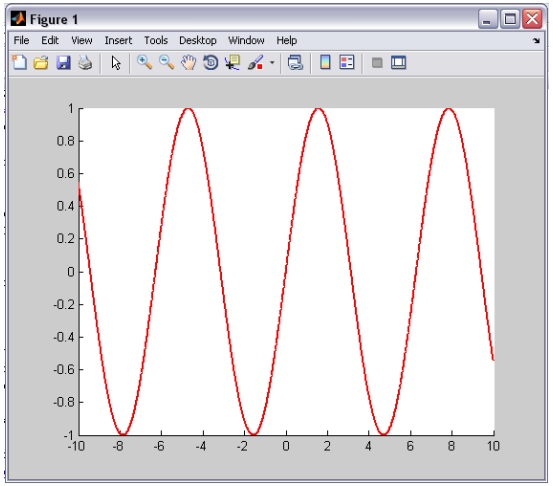
Obiekty graficzne (2)

- Posiadają własności

```
>> hl = line(x,y,'Color','r','LineWidth',2)
hl =  197.0023 % uchwyt do linii

>> get(hl)

    DisplayName =
    Annotation = [ (1 by 1) hg.Annotation array]
    Color = [1 0 0]
    LineStyle = -
    LineWidth = [2]
    Marker = none
    MarkerSize = [6]
    MarkerEdgeColor = auto
    MarkerFaceColor = none
    XData = [ (1 by 2001) double array]
    YData = [ (1 by 2001) double array]
    ZData = []
    ...

    Parent = [196.002]
```







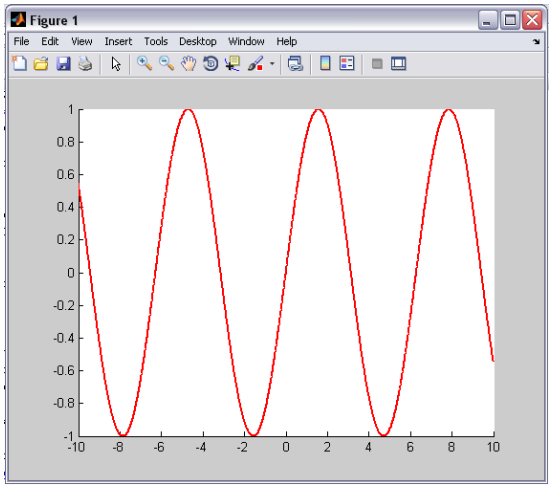
Własności obiektów graficznych

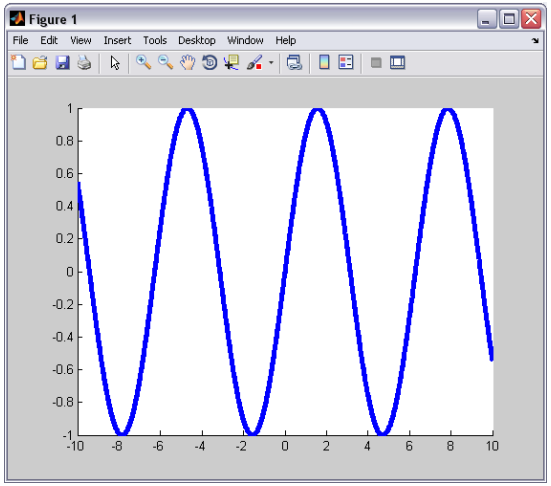
- Dostępne programowo przez
  - konstruktor

```
>> hl = line(x,y,'Color','r','LineWidth',2)
hl =  197.0023
```

  - metody `get` / `set`

```
>> get(hl,'LineWidth')
ans = 2

>> set(hl,'LineWidth',4)

>> set(hl,'Color','b')
```





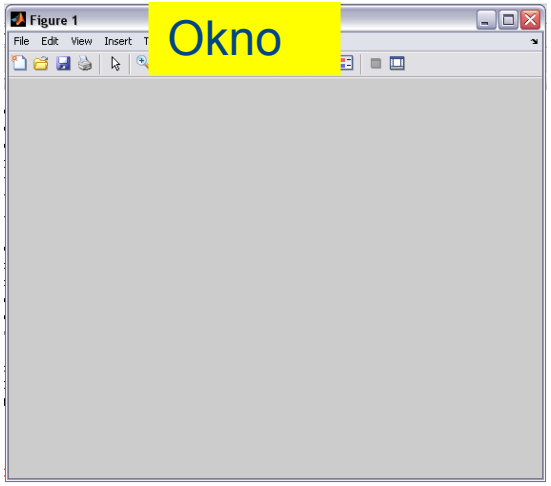
- Dostępne interaktywnie przez
  - inspektora własności (patrz wykład 2)

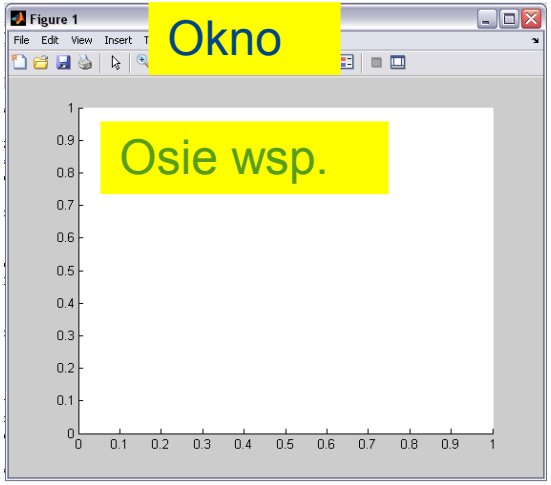
http://www.mathworks.com/help/techdoc/creating_plots/f7-18922.html

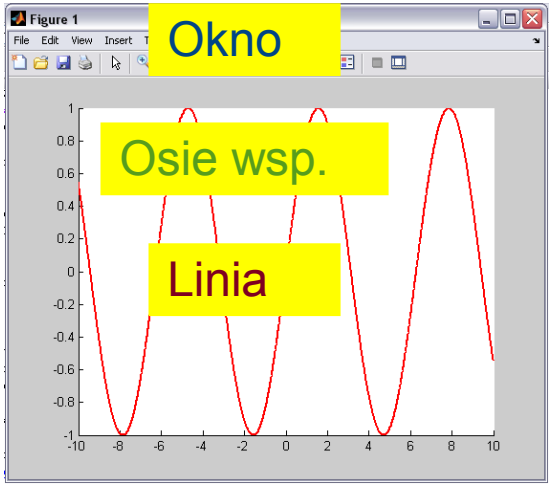
Rodzice i dzieci

- Szczególnymi własnościami są uchwyty
  - do rodzica (`Parent`)
  - do dzieci (`Children`)

```
>> hf = figure
hf =  1

>> ha = axes
ha =  196.0021

>> hl = line(x,y,'Color','r','LineWidth',2)
hl =  197.0023

>> get(ha,'Children')
ans = 197.0023

>> get(ha,'Parent')
ans = 1
```

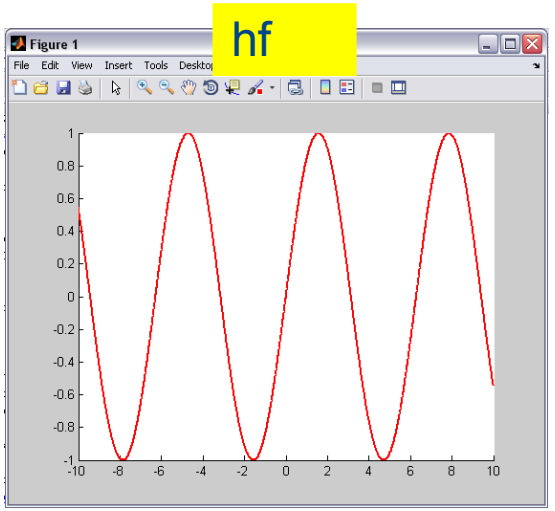




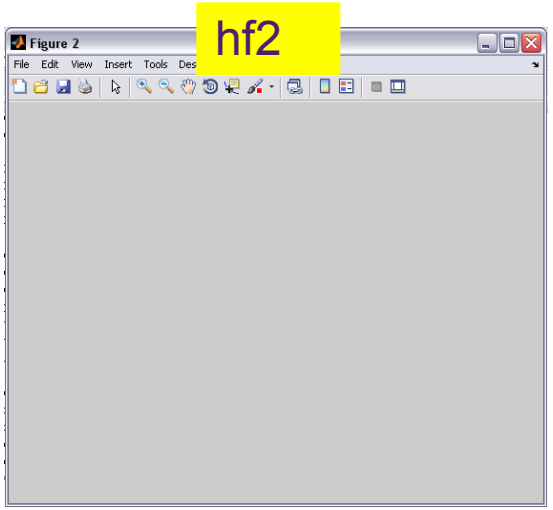


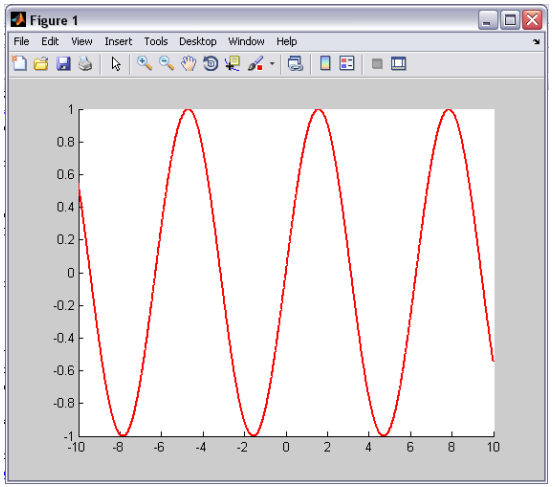
Rodzice i dzieci (2)

- Dzieci przynależą do rodziców

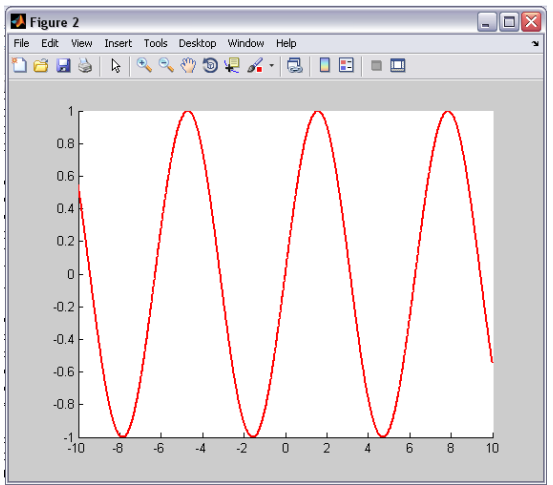


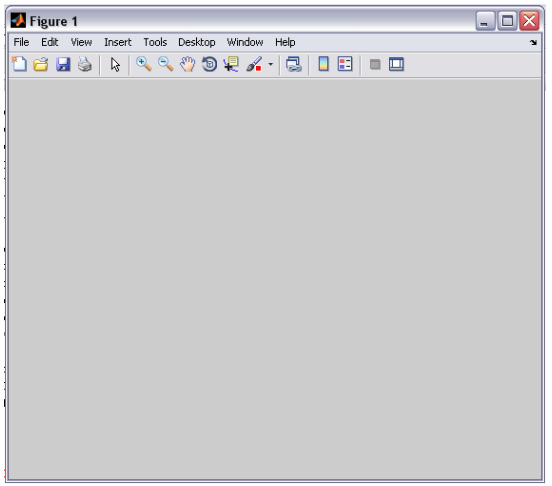

```
>> hf2 = figure
hf2 =   2
```





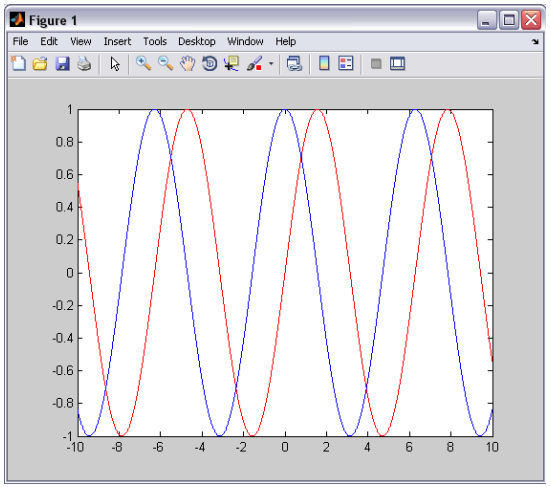

```
>> set(ha,'Parent', hf2)
```



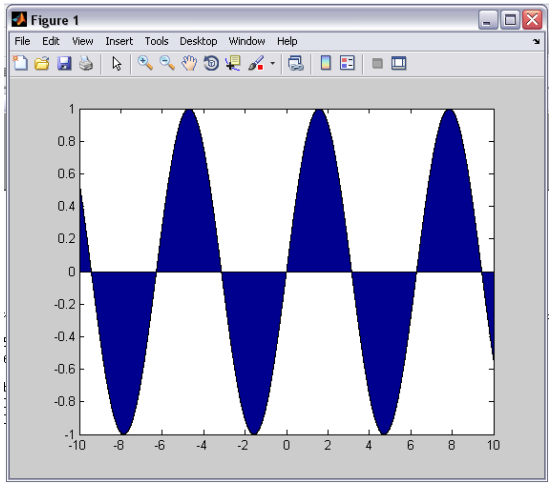


Wykresy – obiekty złożone

```
>> hp = plot(x,sin(x),'r',x,cos(x),'b')
hp = 175.0016        % uchwyty do dwóch obiektów
     176.0011

>> get(hp,'Type')
ans = 'line'         % są to linie klasy lineseries
      'line'         % typu line

>> ha = get(hp(1),'Parent');
        % rodzicem każdej linii są osie wsp.
>> hf = get(ha,'Parent');
        % rodzicem osi wsp. jest okno
```




```
>> hr = area(x,sin(x))
hp = 175.0060        % uchwyt do jednego obiektu

>> get(hr,'Type')
ans = hggroup        % ... klasy areaseries
                     % typu obiekt grupowy hggroup

>> hx = get(hr,'Children')
hx = 176.0045        % który składa się z 1 obiektu

>> get(hx,'Type')
ans = patch          % typu patch
```



Użyteczne funkcje

- `0` – uchwyt do całego ekranu
- `gcf` – zwraca uchwyt do bieżącego okna
- `gca` – zwraca uchwyt do bieżącej osi współrzędnych
- `gco` – zwraca uchwyt do bieżącego obiektu
- `findobj` – znajduje obiekty o podanej własności, np.

```
>>findobj(gcf,'Color','r')  % zwraca uchwyty do obiektów
                            % w kolorze czerwonym ('r')
                            % w bieżącym oknie (gcf)
```

- `copyobj` – kopiuje obiekt wraz z potomstwem, np.

```
>>hp2=copyobj(gcf,0)      % kopiuje bieżący rysunek (gcf)
                         % i oddaje do adopcji ekranowi (0)
```

- `delete` – usuwa obiekt wraz z potomstwem

http://www.mathworks.com/help/techdoc/creating_plots/f7-18922.html

Grafika uchwytów (ang. ***H**andle **G**raphics*) - podsumowanie

- Grafika w MATLABie
  - jest <u>obiektowo zorientowana</u>
    - dostęp do własności obiektów odbywa się przez funkcje `set` i `get`
  - obiekty graficzne są przekazywane przez rodzaj referencji
    - <u>uchwyt</u>
      - stąd nazwa: grafika uchwytów (*Handle Graphics*)

Graficzny interfejs użytkownika
(ang. ***G****raphical* ***U****ser* ***I****nterface*)

- Graficzna prezentacja aplikacji
  - w jednym lub wielu oknach
  - zawiera kontrolki
    - umożliwiają interaktywne wykonywanie zadań
    - np. menu, paski narzędzi, przyciski, listy, suwaki

- GUI współpracuje z innymi programami (funkcjami)
  - wykonują obliczenia, obsługują pliki, tworzą wykresy itp.
    - stanowią tzw. logikę programu (w odróżnieniu od interfejsu)

http://www.mathworks.com/help/techdoc/creating_guis/f2-998436.html

Konsola vs. GUI

- Problem:
  - obliczanie długości przeciwprostokątnej trójkąta prostokątnego

- Konsola

```
function c = przpr
    a = input('Przyprostokatna a:');
    b = input('Przyprostokatna b:');
    c = sqrt(a^2+b^2);
```

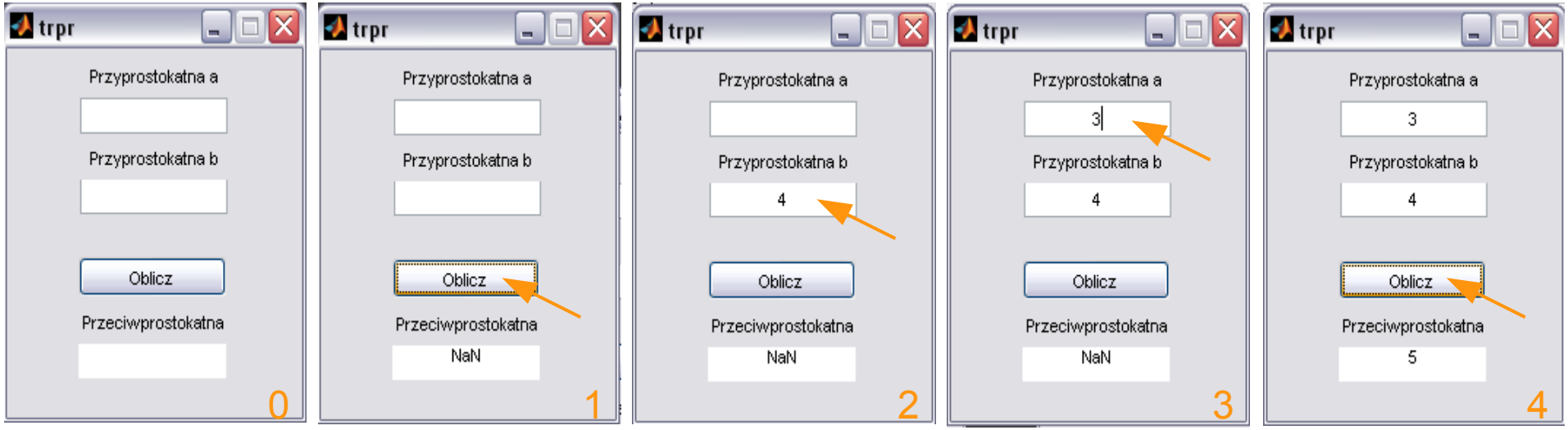
```
>> c = przpr
Przyprostokatna a: 3
Przyprostokatna b: 4
c = 5
```

- Kto kontroluje bieg zdarzeń? (tu: kolejność wprowadzania danych)
  - program (funkcja `przpr`)

Konsola vs. GUI (2)

- Problem:
  - obliczanie długości przeciwprostokątnej trójkąta prostokątnego

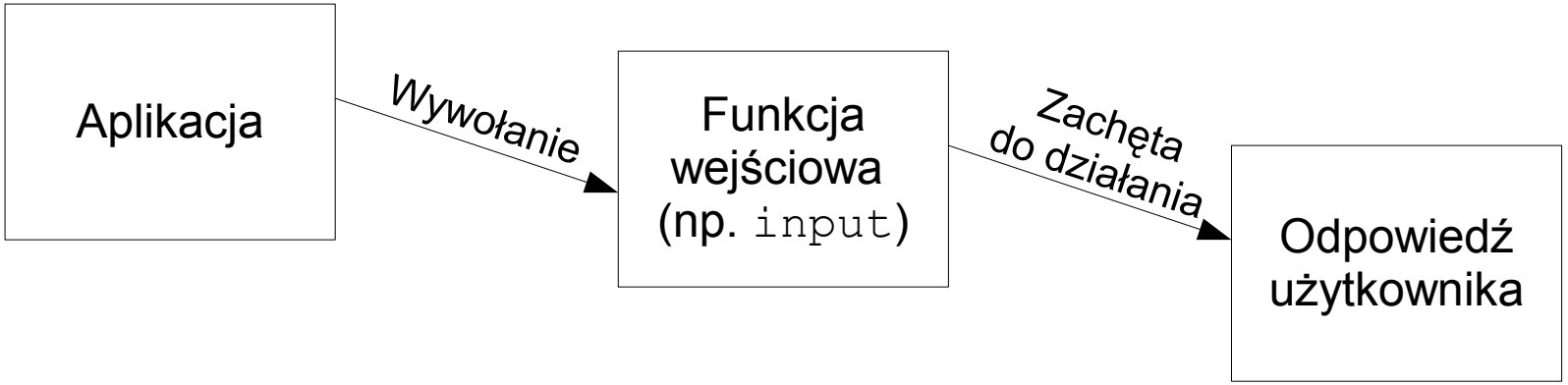
- GUI



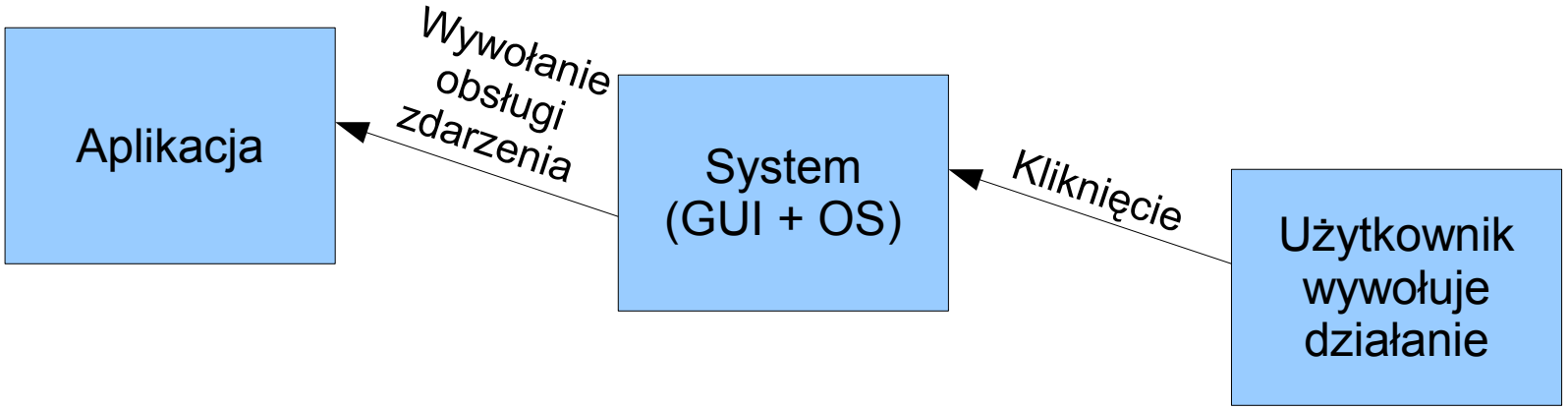
- Kto kontroluje bieg zdarzeń? (tu: kolejność wprowadzania danych i obliczania)
  - **użytkownik**

Inwersja kontroli

- Aplikacja konsolowa



- Aplikacja GUI



na podst. B.Stroustrup, Programowanie. Rozdział 16. s.512. Helion 2010

Programowanie sterowane zdarzeniami

- Użytkownik kontroluje przebieg zdarzeń
  - aplikacja musi reagować na akcje użytkownika
    - GUI rejestruje zdarzenia i wywołuje ich obsługę
- Jest to programowanie sterowane zdarzeniami
  - ang. *event-driven programming*

Dziesięć zasad projektowania interfejsu użytkownika

- Projektuj dla użytkowników i ich zadań
- Bądź spójny
- Stosuj proste i naturalne dialogi
- Ogranicz niepotrzebny wysiłek umysłowy użytkownika (związany z obsługą GUI, a nie z zadaniem)
- Dostarcz informacji zwrotnej (czy działa? dlaczego nie?)
- Zapewnij czytelną nawigację (w tym wyjście z błędnej ścieżki)
- Przekaż kontrolę użytkownikowi
- Prezentuj informacje czytelnie
- Bądź pomocny
- Ogranicz możliwość popełnienia błędu

http://www.usabilitynet.org/management/b_design.htm

Projektowanie aplikacji z graficznym interfejsem użytkownika

- Spojrzeć na zadanie z punktu widzenia użytkownika
  - Projekt interfejsu
    - szkic, projekt
  - Przypadki użycia
    - sposoby korzystania z aplikacji
  - Diagram klas
    - ideałem jest wyrażenie pojęć dziedziny w kodzie

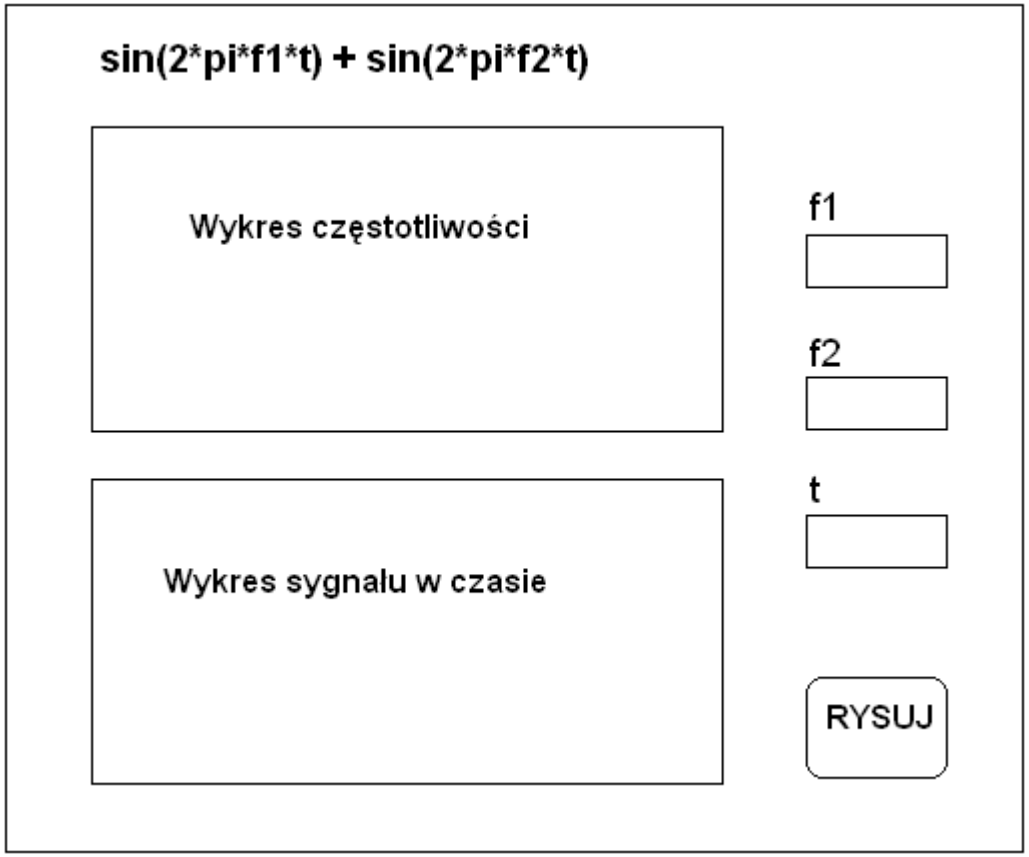
http://www.mathworks.com/help/techdoc/creating_guis/f2-998551.html

Wizualne środowisko programistyczne

- Zaprojektuj interfejs użytkownika
  - na papierze, w Paincie, jakkolwiek
- Utwórz interfejs
  - programistycznie (proste lub bardzo złożone)
  - używając narzędzi wizualnego środowiska programistycznym (średnio złożone – coś dla nas:-)

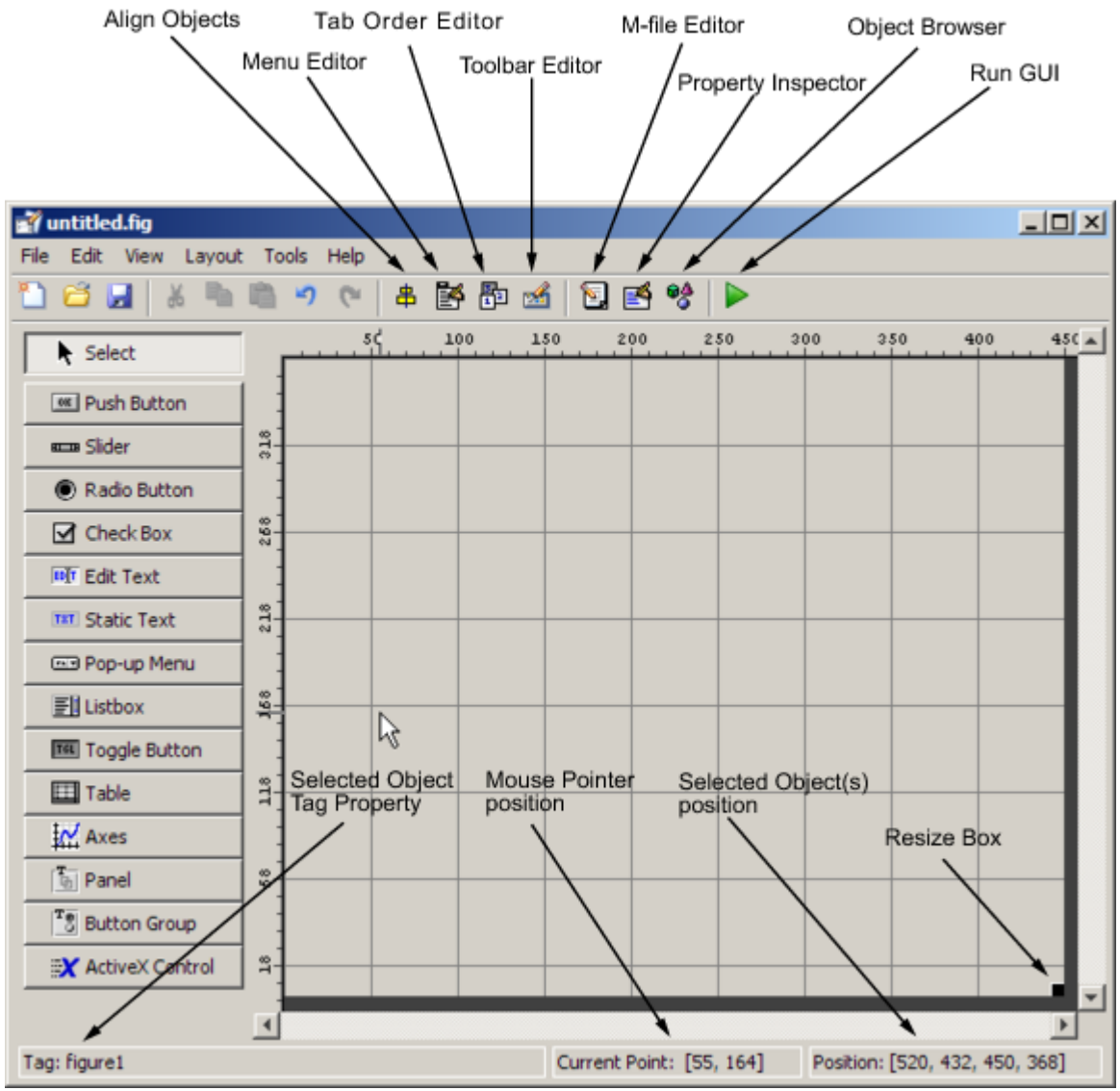
http://www.mathworks.com/help/techdoc/creating_guis/brtk_rg.html

Przykład: generowanie i analiza sygnału sinusoidalnego



GUIDE – GUI Design Environment

środowisko projektowania GUI



http://www.mathworks.com/help/techdoc/creating_guis/f7-998368.html

Projekt GUI

MATLAB R2011b

File Edit View Debug Desktop Window Help

Current Folder: C:\Documents and Settings\Admin\Moje dokumenty\MATLAB\inf12l\wyk\ansyg

Shortcuts How to Add What's New

Current Folder

« inf12l ▸ wyk ▸ ansyg

Name

Open Ctrl+D
Open as Text
Open Outside MATLAB
Show in Explorer
Create Zip File
Rename F2
Delete Delete
Compare Selected Files/Folders
Compare Against
Source Control
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
✔ Indicate Files Not on Path
Import Data...

ansyg.JPG (Obraz JPEG)

Command Window

>>

Workspace

Stack: Select data to plot

Name Value

Command History

```
unique(datestr(datenum([g1(:,1:2) 15
unique(datenum([g1(:,1:2) 15*ones(si
epidem
%-- 4/12/2012 2:48 PM --%
a = input('Przyprostokatna a: ');
3
b = input('Przyprostokatna b: ');
4
c = sqrt(a^2+b^2)
format compact
trpr
c = przpr
3
4
clear
clc
```

Start

Start INF_wyklad_5.odp - ... MATLAB R2011b ansyg.JPG - Podgląd ... Flashing EN 5:04 PM

Wskazówki

- Nadaj znaczące nazwy obiektom
  - użyj własności `Tag`
  - unikniesz błędów i bełkotliwego kodu!
- Zrób to od razu
  - zanim zapiszesz GUI
  - unikniesz bałaganu w pliku kodem!
- Stosuj etykiety (`text`)
  oraz wprowadź sensowne przykładowe dane
  - ułatwisz testowanie
  - uzyskasz bardziej przejrzysty interfejs

Interakcja z użytkownikiem

- Przypadki użycia (PU) aplikacji `ansyg`
  - otworzenie aplikacji
    - generujemy przykładowy wykres czy czekamy na przycisk RYSUJ?
  - zamknięcie aplikacji
    - może wyświetlimy zachętę do wpłaty na nasze konto?
  - wciśniecie przycisku RYSUJ
    - rysowanie sygnału i widma częstotliwościowego
  - edycja zmiennych f1, f2, t
    - jak sprawdzimy poprawność?

Przypadek użycia: wciśnięcie klawisza RYSUJ (1)

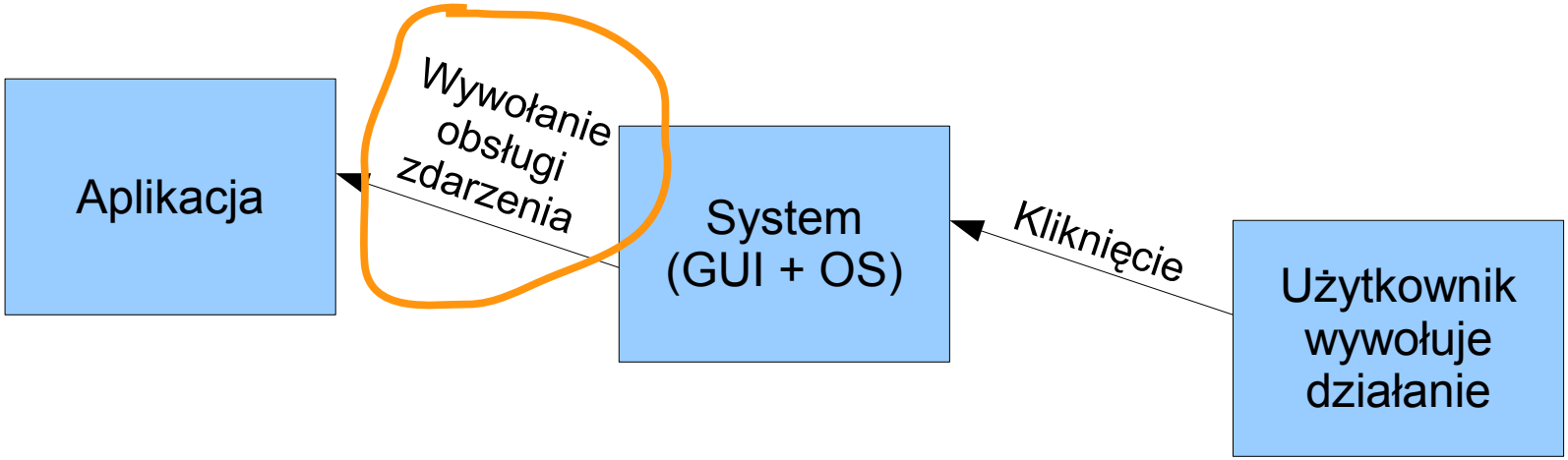
- **Cel**
  - wygenerowanie sygnału
  - wykres w dziedzinie czasu oraz częstotliwości
- **Warunki początkowe**
  - Użytkownik kliknął przycisk RYSUJ
  - Poprawność danych
    - f1, f2 – są skalarami rzeczywistymi
    - t – jest wektorem
- **Warunki końcowe**
  - PU narysował wykresy sygnału w dziedzinie czasu i częstotliwości

Przypadek użycia: wciśnięcie klawisza RYSUJ (2)

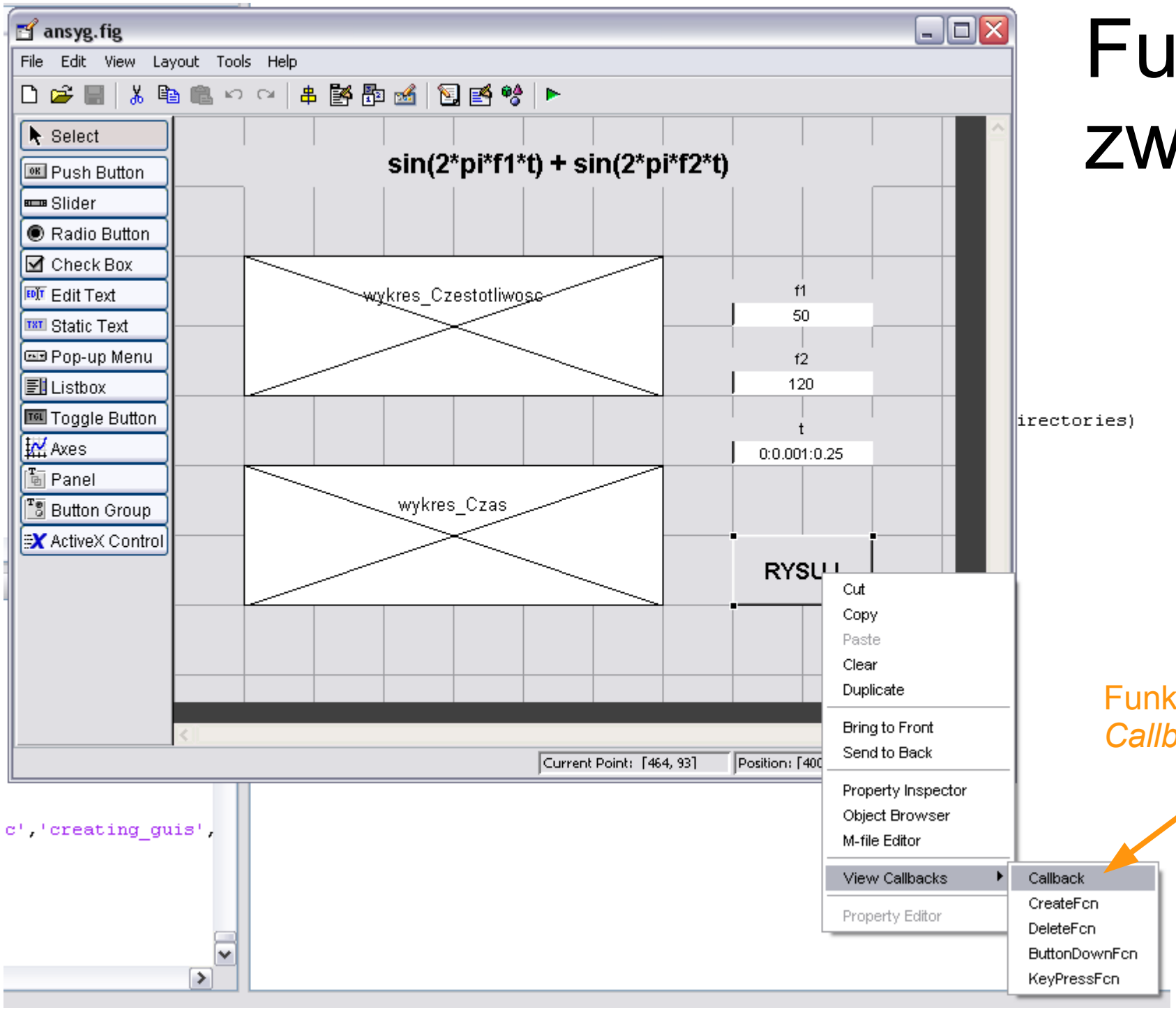
- **Przebieg działania:**
  - Użytkownik klika przycisk RYSUJ
  - PU pobiera f1, f2 i t z kontrolek GUI
  - PU oblicza wartości funkcji
    x = sin(2*pi*f1*t) + sin(2*pi*f2*t)
  - PU wykonuje szybką transformację Fouriera
    y = fft(x)
  - PU rysuje wykres |y(f)| w odpowiednich osiach wsp.
  - PU rysuje wykres x(t) w odpowiednich osiach wsp.

Jak to zrobić? Funkcje zwrotne

- Użytkownik kliknął przycisk RYSUJ
  - system wychwycił to zdarzenie
    - programista wie jak je obsłużyć (patrz nasz PU):
      - przebieg działania umieszcza się w **funkcji zwrotnej** (*Callback*)
        - przypisanej do danej **kontrolki** i **zdarzenia** (*event*)



Funkcje zwrotne (1)



Funkcja zwrotna
Callback function

Funkcje zwrotne (2)

- W pliku `ansyg.m`
  - został dopisany nagłówek funkcji zwrotnej przycisku RYSUJ
    - o nazwie `przycisk_Rysuj_Callback`

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

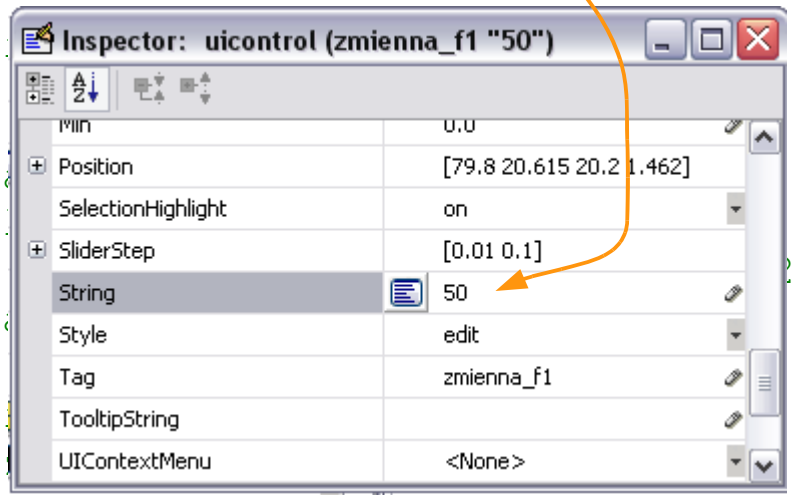
- Parametry funkcji zwrotnej
  - `hObject` – uchwyt obiektu (tu: przycisk RYSUJ)
  - `eventdata` – dodatkowe informacje o zdarzeniu (tu: nieużywane)
  - `handles` – struktura zawierająca wszystkie uchwyty GUI oraz ewentualne dane użytkownika

przycisk_Rysuj_Callback

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

% Pobieramy f1, f2 i t z kontrolek GUI
f1 = str2double(get(handles.zmienna_f1,'String'));
f2 = str2double(get(handles.zmienna_f2,'String'));
t = eval(get(handles.zmienna_t,'String'));

% Obliczamy wartości funkcji x  = sin(2*pi*f1*t) + sin(2*pi*f2*t)
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);

% Wykonujemy szybką transformację Fouriera y = fft(x)
fs = 1/median(diff(t));          % częstotliwośc próbkowania
len = 2.^nextpow2(length(t));    % dla FFT długość sygnału musi wynosić 2^k
                                 % "brakujące" dane funkcja fft uzupełni 0
y = fft(x,len);                  % funkcja fft
m = abs(y);                      % moduł liczby zespolonej
f = fs*(0:len/2)/len;            % skala częstotliwości 0 .. fs/2

% Rysujemy wykres |y(f)| w odpowiednich osiach wsp.
plot(handles.wykres_Czestotliwosc,f,m(1:len/2+1))

% Rysujemy wykres x(f) w odpowiednich osiach wsp.
plot(handles.wykres_Czas,t,x)
```

przycisk_Rysuj_Callback

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

% Pobieramy f1, f2 i t z kontrolek GUI
f1 = str2double(get(handles.zmienna_f1,'String'));
f2 = str2double(get(handles.zmienna_f2,'String'));
t = eval(get(handles.zmienna_t,'String'));

% Obliczamy wartości funkcji x  = sin(2*p
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);

% Wykonujemy szybką transformację Fourier
fs = 1/median(diff(t));          % częstot
len = 2.^nextpow2(length(t));    % dla FFT                    2^k
                                 % "brakuj                    0
y = fft(x,len);                  % funkcja
m = abs(y);                      % moduł l
f = fs*(0:len/2)/len;            % skala c

% Rysujemy wykres |y(f)| w odpowiednich osiach wsp.
plot(handles.wykres_Czestotliwosc,f,m(1:len/2+1))

% Rysujemy wykres x(f) w odpowiednich osiach wsp.
plot(handles.wykres_Czas,t,x)
```

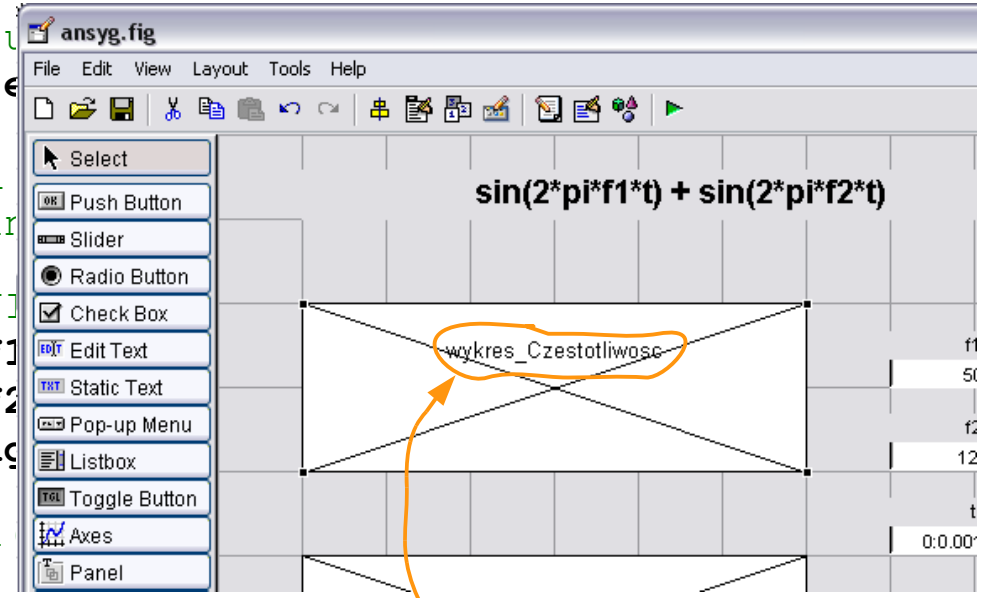


przycisk_Rysuj_Callback

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

% Pobieramy f1, f2 i t z kontrolek GUI
f1 = str2double(get(handles.zmienna_f1,'String'));
f2 = str2double(get(handles.zmienna_f2,'String'));
t = eval(get(handles.zmienna_t,'String'));

% Obliczamy wartości funkcji x  = sin(2*pi*f1*t) + sin(2*pi*f2*t)
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);

% Wykonujemy szybką transformację Fouriera y = fft(x)
fs = 1/median(diff(t));          % częstotliwośc próbkowania
len = 2.^nextpow2(length(t));    % dla FFT długość sygnału musi wynosić 2^k
                                 % "brakujące" dane funkcja fft uzupełni 0
y = fft(x,len);                  % funkcja fft
m = abs(y);                      % moduł liczby zespolonej
f = fs*(0:len/2)/len;            % skala częstotliwości 0 .. fs/2

% Rysujemy wykres |y(f)| w odpowiednich osiach wsp.
plot(handles.wykres_Czestotliwosc,f,m(1:len/2+1))

% Rysujemy wykres x(f) w odpowiednich osiach wsp.
plot(handles.wykres_Czas,t,x)
```

przycisk_Rysuj_Callback

```
% --- Użytkownik kliknął przycisk_Rysu
function przycisk_Rysuj_Callback(hObje
% hObject    handle to przycisk_Rysuj
% eventdata  reserved - to be defined
% handles    structure with handles ar

% Pobieramy f1, f2 i t z kontrolek GUI
f1 = str2double(get(handles.zmienna_f1
f2 = str2double(get(handles.zmienna_f2
t = eval(get(handles.zmienna_t,'String

% Obliczamy wartości funkcji x  = sin
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);

% Wykonujemy szybką transformację Fouriera y = fft(x)
fs = 1/median(diff(t));          % częstotliwośc próbkowania
len = 2.^nextpow2(length(t));    % dla FFT długość sygnału musi wynosić 2^k
                                 % "brakujące" dane funkcja fft uzupełni 0
y = fft(x,len);                  % funkcja fft
m = abs(y);                      % moduł liczby zespolonej
f = fs*(0:len/2)/len;            % skala częstotliwości 0 .. fs/2

% Rysujemy wykres |y(f)| w odpowiednich osiach wsp.
plot(handles.wykres_Czestotliwosc,f,m(1:len/2+1))

% Rysujemy wykres x(f) w odpowiednich osiach wsp.
plot(handles.wykres_Czas,t,x)
```



Testujemy...

Editor - C:\Documents and Settings\Admin\Moje dokumenty\MATLAB\inf12l\wyk\ansyg\ansyg.m*

File Edit Text Go Cell Tools Debug Desktop Window Help

```
132     % --- Executes during object creation, after setting all properties.
133     function zmienna_t_CreateFcn(hObject, eventdata, handles)
134     % hObject    handle to zmienna_t (see GCBO)
135     % eventdata  reserved - to be defined in a future version of MATLAB
136     % handles    empty - handles not created until after all CreateFcns called
137
138     % Hint: edit controls usually have a white background on Windows.
139     %       See ISPC and COMPUTER.
140     if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
141         set(hObject,'BackgroundColor','white');
142     end
143
144
145     % --- Użytkownik kliknął przycisk_Rysuj
146     function przycisk_Rysuj_Callback(hObject, eventdata, handles)
147     % hObject    handle to przycisk_Rysuj (see GCBO)
148     % eventdata  reserved - to be defined in a future version of MATLAB
149     % handles    structure with handles and user data (see GUIDATA)
150
151     % Pobieramy f1, f2 i t z kontrolek GUI
152     f1 = str2double(get(handles.zmienna_f1,'String'));
153     f2 = str2double(get(handles.zmienna_f2,'String'));
154     t = eval(get(handles.zmienna_t,'String'));
155
156     % Obliczamy wartości funkcji x  = sin(2*pi*f1*t) + sin(2*pi*f2*t)
157     x = sin(2*pi*f1*t) + sin(2*pi*f2*t);
158
159     % Wykonujemy szybką transformację Fouriera y = fft(x)
160     fs = 1/median(diff(t));          % częstotliwośc próbkowania
161     len = 2.^nextpow2(length(t));    % dla FFT dlugosc sygnału musi wynosić 2^k
162                                      % "brakujące" dane funkcja fft uzupełni 0
163     y = fft(x,len);                  % funkcja fft
164     m = abs(y);                      % moduł liczby zespolonej
165     f = fs*(0:len/2)/len;            % skala częstotliwości 0 .. fs/2
166
167     % Rysujemy wykres |y(f)| w odpowiednich osiach wsp.
168     plot(handles.wykres_Czestotliwosc,f,m(1:len/2+1))
169
170     % Rysujemy wykres x(f) w odpowiednich osiach wsp.
171     plot(handles.wykres_Czas,t,x)
172
173
```

ansyg / przycisk_Rysuj_Callback Ln 171 Col 30 OVR

Start | AVI ReComp 1.4.5 | Windows Media P... | 3 MATLAB | Szybka transform... | INF_wyklad_5.od... | info5.ppt - Open... | wyklad_uml_bios... | bez tytułu - Paint | PL | 3:14 PM

W czym problem?

- Opisując przypadek użycia przycisku RYSUJ
  - zauważyliśmy, że warunkami początkowym są
    - f1, f2 – są skalarami rzeczywistymi
    - t – jest wektorem
  - Powinniśmy <u>gdzieś</u> o to zadbać
    - przed przystąpieniem do generowania sygnału
      - w obsłudze przycisku RYSUJ
    - po zmianie wartości zmiennych
      - w obsługach pól edycyjnych

W czym problem?

- Opisując przypadek użycia przycisku RYSUJ
  - zauważyliśmy, że warunkami początkowym są
    - f1, f2 – są skalarami rzeczywistymi
    - t – jest wektorem
  - Powinniśmy gdzieś o to zadbać
    - przed przystąpieniem do generowania sygnału
      - w obsłudze przycisku RYSUJ
    - po zmianie wartości zmiennych
      - w obsługach pól edycyjnych

PU: Weryfikacja zmiennej f1 (1)

- **Cel:**
  - zapewnienie poprawności f1
- **Warunki początkowe**
  - Użytkownik zakończył edycję pola zmienna_f1
- **Warunki końcowe**
  - wartość pola zmienna_f1 jest nieprawidłowa – przycisk RYSUJ jest zablokowany

    albo
  - wartość pola zmienna_f1 jest prawidłowa – przycisk RYSUJ jest odblokowany

PU: Weryfikacja zmiennej f1 (2)

- **Przebieg działania:**
  - Użytkownik kończy edycję pola zmienna_f1 przechodząc do innego elementu GUI
  - PU pobiera wartość zmienna_f1
  - PU sprawdza czy zmienna_f1 jest skalarem rzeczywistym
    - jeśli <u>nie</u>
      - blokuje przycisk RYSUJ
      - wyświetla komunikat błędu w polu statusu
      - ustawia kursor na polu zmienna_f1
    - jeśli <u>tak</u>,
      - odblokowuje przycisk RYSUJ
      - usuwa komunikat błędu w polu statusu

zmienna_f1_Callback

```
% --- Użytkownik ukończył edycję zmienna_f1 przechodząc do innego el. GUI
function zmienna_f1_Callback(hObject, eventdata, handles)

f1 = str2double(get(hObject,'String'));     % zwraca NaN jesli nie liczba

%Sprawdzamy czy wartość zmienna_f1 jest skalarem rzeczywistym
if isnan(f1) || ~isreal(f1) % jeśli nie
    % Blokujemy przycisk RYSUJ
    set(handles.przycisk_Rysuj,'Enable','off')
    % Wyświetlamy komunikat błędu w polu statusu
    set(handles.tekst_Status,'String','Nieprawidlowa wartosc f1')
    % Ustawiamy kursor na polu zmienna_f1
    uicontrol(hObject)
else                            % jeśli tak
    % Odblokowujemy przycisk RYSUJ
    set(handles.przycisk_Rysuj,'Enable','on')
    % Usuwamy komunikat błędu w polu statusu
    set(handles.tekst_Status,'String','')
end
```

Uwagi

- dodaliśmy nowy komponent typu *text* o nazwie `tekst_Status`
- edycję `zmienna_f2` obłsugujemy analogicznie

Teraz zmienna_t_Callback

```
% --- Użytkownik ukończył edycję zmienna_t przechodząc do innego el. GUI
function zmienna_t_Callback(hObject, eventdata, handles)

% Blokujemy przycisk_Rysuj... dopóki nie sprawdziliśmy, że t prawidłowe
set(handles.przycisk_Rysuj,'Enable','off')

t = eval(get(handles.zmienna_t,'String'));

if ~isnumeric(t) || length(t) < 2   % t nie jest wektorem liczbowym
    set(handles.tekst_Status,'String','t musi byc wektorem liczbowym')
elseif min(diff(t)) < 0             % t nie rośnie monotonicznie
    set(handles.tekst_Status,'String','t musi rosnac monotonicznie')
else                                % wszystko w porządku
    % Usuwamy komunikat błędu w polu statusu
    set(handles.tekst_Status,'String','')
    % Odblokowujemy przycisk RYSUJ
    set(handles.przycisk_Rysuj,'Enable','on')
    return
end

% Jeśli jednak nie było ok, to ustawiamy kursor na polu zmienna_t
uicontrol(hObject)
```

Testujemy i poprawiamy

Editor - C:\Documents and Settings\Admin\Moje dokumenty\MATLAB\inf12l\wyk\ansyg\ansyg.m

```
147     % eventdata  reserved - to be defined in a future version of MATLAB
148     % handles    empty - handles not created until after all CreateFcns called
149
150     % Hint: edit controls usually have a white background on Windows.
151     %       See ISPC and COMPUTER.
152     if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
153         set(hObject,'BackgroundColor','white');
154     end
155
156
157     % --- Użytkownik ukończył edycję zmienna_t przechodząc do innego el. GUI
158     function zmienna_t_Callback(hObject, eventdata, handles)
159     % hObject    handle to zmienna_t (see GCBO)
160     % eventdata  reserved - to be defined in a future version of MATLAB
161     % handles    structure with handles and user data (see GUIDATA)
162
163     % Hints: get(hObject,'String') returns contents of zmienna_t as text
164     %        str2double(get(hObject,'String')) returns contents of zmienna_t
165     %        as a double
166
167     % Blokujemy przycisk_Rysuj... dopóki nie sprawdziliśmy, że t prawidłowe
168     set(handles.przycisk_Rysuj,'Enable','off')
169
170     t = eval(get(handles.zmienna_t,'String'));
171     if ~isnumeric(t) || length(t) < 2   % t nie jest wektorem liczbowym
172         set(handles.tekst_Status,'String','t musi byc wektorem liczbowym')
173     elseif min(diff(t)) < 0             % t nie rośnie monotonicznie
174         set(handles.tekst_Status,'String','t musi rosnac monotonicznie')
175     else                                % wszystko w porządku
176         % Usuwamy komunikat błędu w polu statusu
177         set(handles.tekst_Status,'String','')
178         % Odblokowujemy przycisk RYSUJ
179         set(handles.przycisk_Rysuj,'Enable','on')
180         return
181     end
182     % Jeśli jednak nie było ok, to ustawiamy kursor na polu zmienna_t
183     uicontrol(hObject)
184
185
186     % --- Executes during object creation, after setting all properties.
187     function zmienna_t_CreateFcn(hObject, eventdata, handles)
```

zmienna_t_Callback (v2)

```
% --- Użytkownik ukończył edycję zmienna_t przechodząc do innego el. GUI
function zmienna_t_Callback(hObject, eventdata, handles)

% Blokujemy przycisk_Rysuj... dopóki nie sprawdziliśmy, że t prawidłowe
set(handles.przycisk_Rysuj,'Enable','off')

try
    t = eval(get(handles.zmienna_t,'String'));

    if ~isnumeric(t) || length(t) < 2   % t nie jest wektorem liczbowym
        set(handles.tekst_Status,'String','t musi byc wektorem liczbowym')
    elseif min(diff(t)) < 0             % t nie rośnie monotonicznie
        set(handles.tekst_Status,'String','t musi rosnac monotonicznie')
    else                                % wszystko w porządku
        % Usuwamy komunikat błędu w polu statusu
        set(handles.tekst_Status,'String','')
        % Odblokowujemy przycisk RYSUJ
        set(handles.przycisk_Rysuj,'Enable','on')
        return
    end

    % Jeśli jednak nie było ok, to ustawiamy kursor na polu zmienna_t
    uicontrol(hObject)

catch EM
    set(handles.tekst_Status,'String','Nieprawidlowa wartosc t')
    uicontrol(hObject)
end
```

ansyg – podsumowanie

- Nauczyliśmy się
  - jak projektować i jak tworzyć **GUI**
  - jak opisywać **przypadki użycia**
    - jako sposób dokumentacji projektu
    - jako wstęp do implementacji
  - jak tworzyć **funkcje zwrotne**
    - jak korzystać z innych elementów GUI (uchwyty)

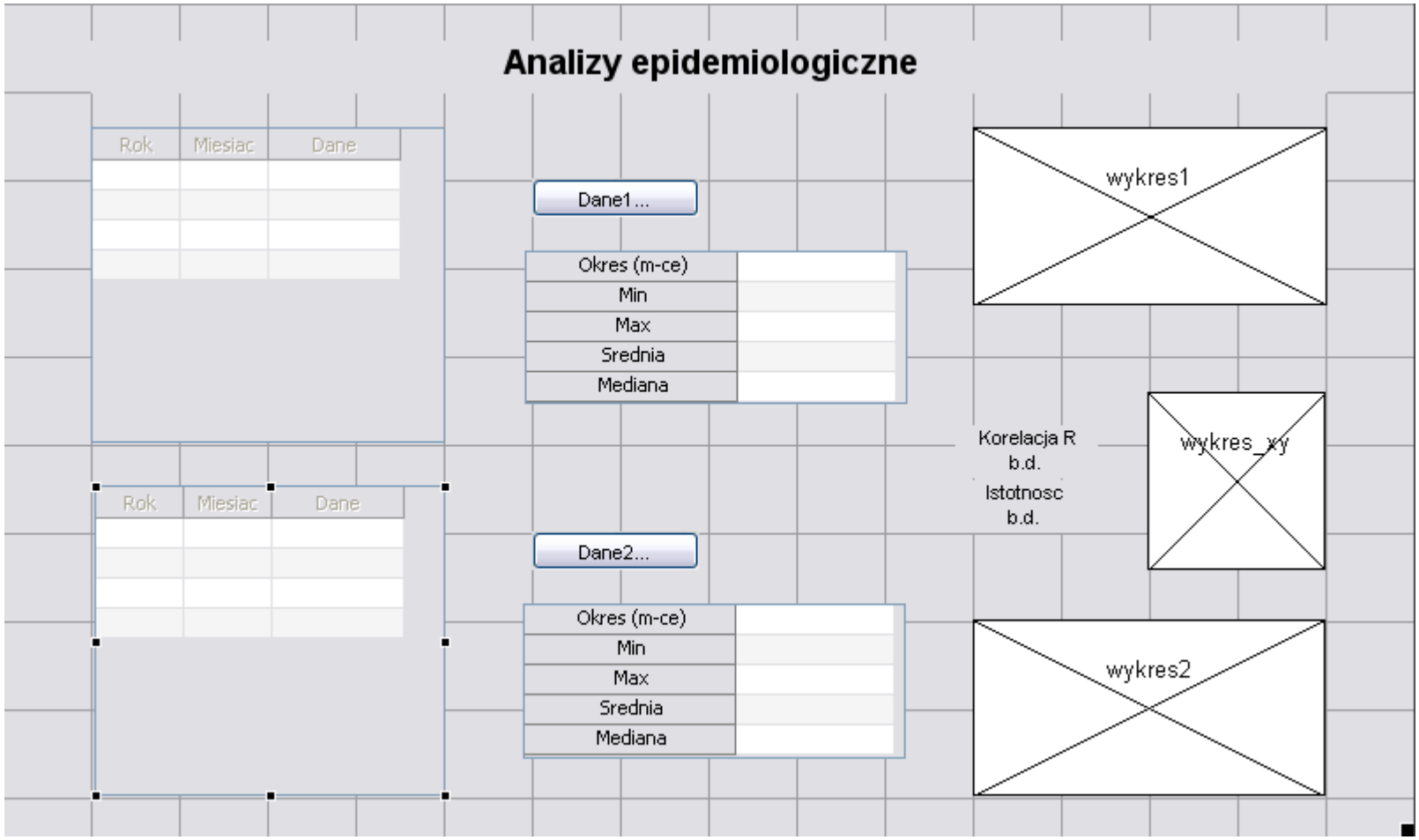
Nauczymy się jeszcze kilku technik

- Jak?
  - otwierać pliki w GUI
  - tworzyć i dostosowywać tabele `uitable`
  - korzystać ze struktury `evendata`
  - przechowywac dane aplikacji
  - liczyć korelacje w MATLABIE

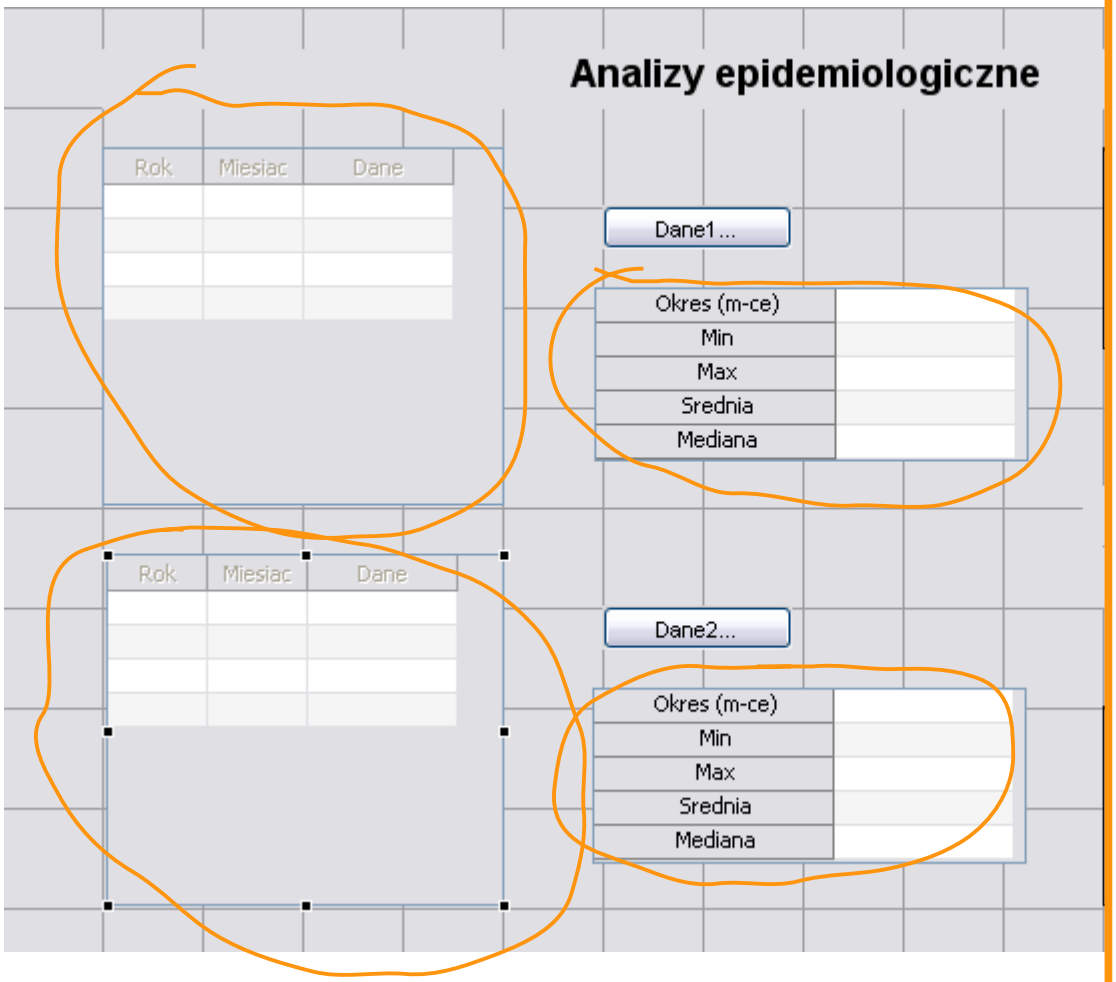
Przykład 2 – epidem

- **Cel**
  - Wizualizacja i porównywanie danych epidemiologicznych
- **Założenia**
  - Dane dostępne w układzie miesięcznym
    - EPIMELD http://www.pzh.gov.pl/oldpage/epimeld/index_p.html
    - GUS http://www.stat.gov.pl/cps/rde/xbcr/gus/PUBL_rs_rocznik_demograficzny_2011.pdf
  - Jednoczesna wizualizacja i analiza dwóch zbiorów, np.
    - zachorowalność na dwie choroby
    - zachorowalność a zgodny
  - Interesują nas
    - statystka danych
    - korelacja liniowa (Pearsona)

epidem – projekt



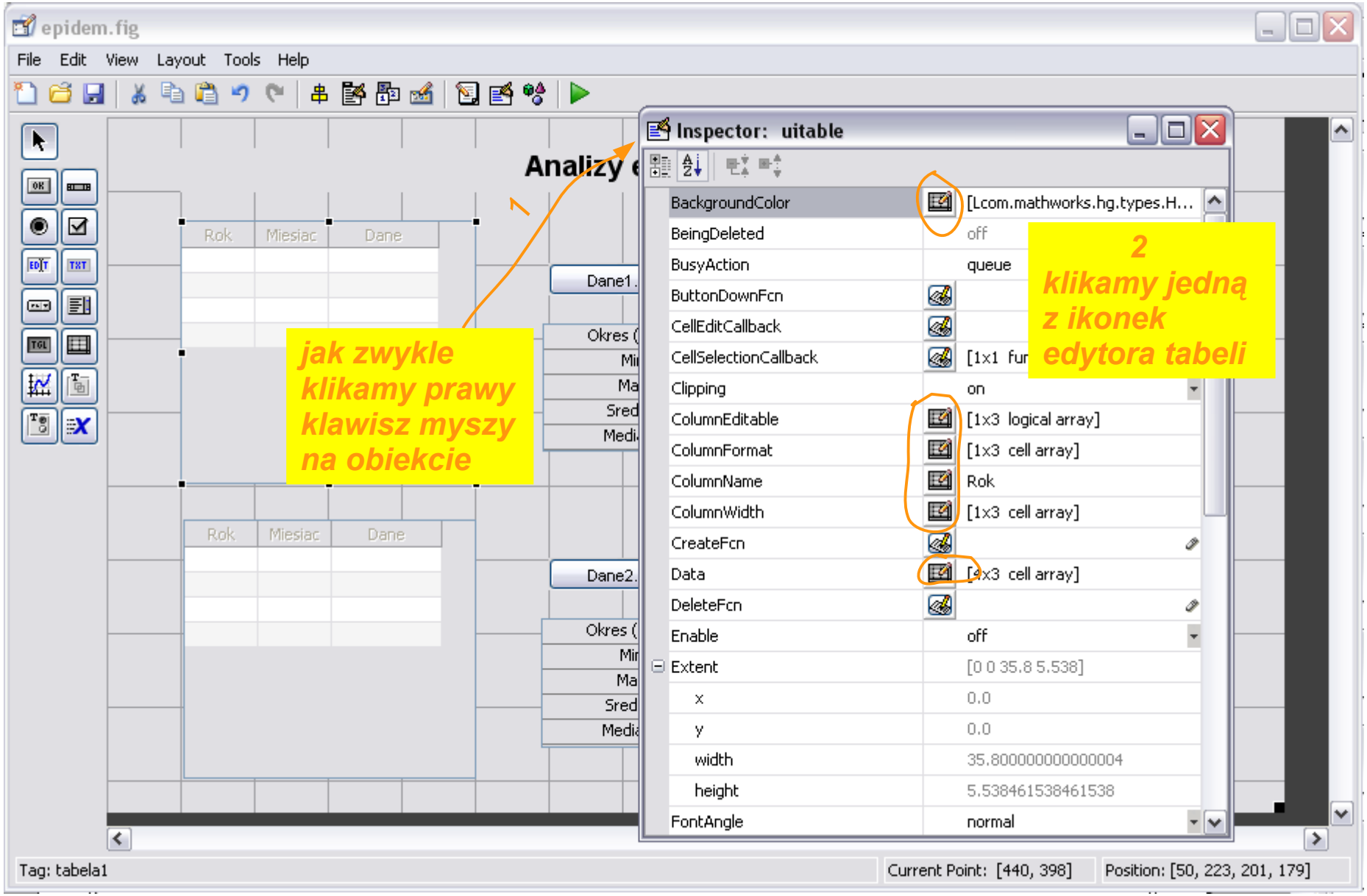
epidem – projekt



`uitable`
- nowy element GUI

- obiekt graficzny przechowujący dane
- umożliwia m.in.
  - zaznaczanie danych
  - edycję danych

Edycja własności tablicy (1)



Edycja własności tablicy (2)

epidem.fig

File Edit View Layout Tools Help

Analizy

Rok Miesiac Dane

Dane1

Okres (

Inspector: uitable

BackgroundColor	[Lcom.mathworks.hg.types.H...
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
CellEditCallback	
CellSelectionCallback	[1x1 function_handle arra...
	on
	[1x3 logical array]
	[1x3 cell array]
	Rok
	[1x3 cell array]
	[4x3 cell array]
	off
	[0 0 35.8 5.538]
	0.0
	0.0
	35.800000000000004
	5.538461538461538
	normal

Current Point: [781, 235] Position: [50, 223, 201, 179]

Table Property Editor

Columns
Rows
Data
Colors

Columns

Unless the "Show names entered below..." option is selected, the number of columns appearing in the table is determined by the number of columns in Data.

Column Headers

- Do not show column headers
- Show numbered column headers
- Show names entered below as the column headers:

3 edytujemy własności tabeli

Column Definitions

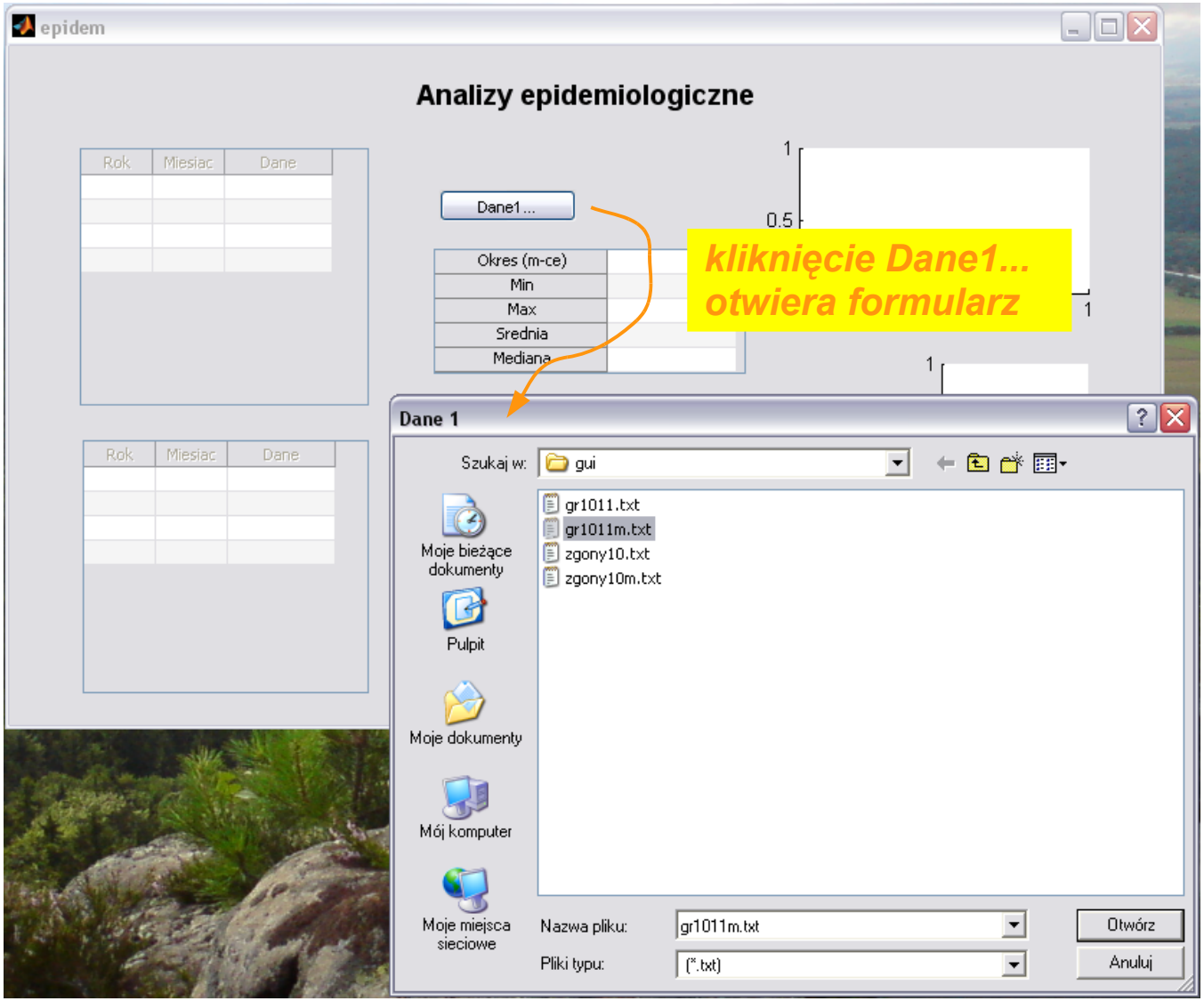
#	Name	Auto Width	Width (px)	Editable	Format
1	Rok	☐	50	☐	Let MATLAB Choose
2	Miesiac	☐	50	☐	Let MATLAB Choose
3	Dane	☑		☐	Let MATLAB Choose

Insert
Copy
Delete
Up
Down

OK Cancel Apply Help

```
figure(1)
clear(gcf)
clf
help clf
epidem
```

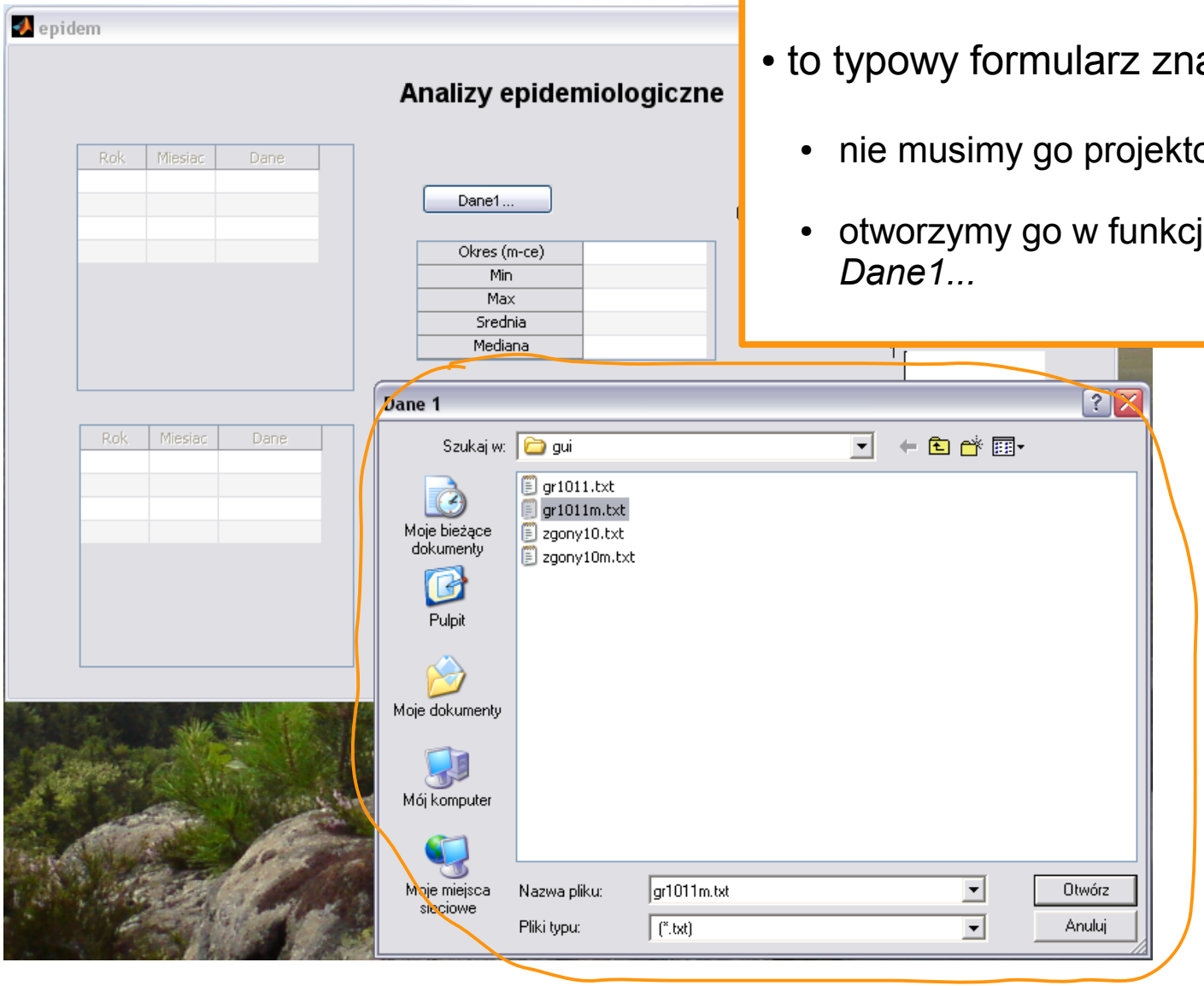
Otwieranie plików w GUI (1)



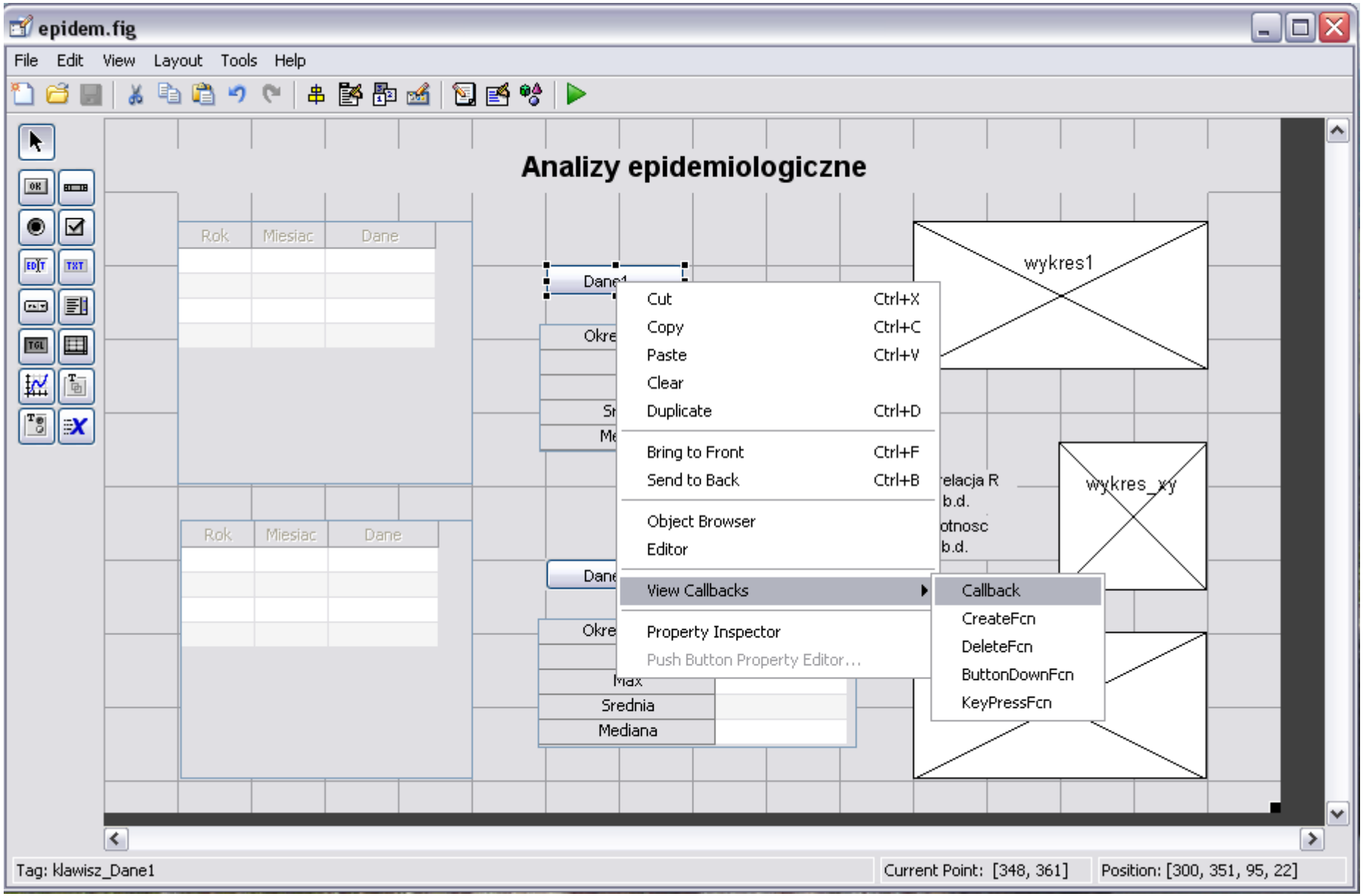
Otwieranie plików w GUI (2)

To okno to obiekt `uigetfile`

- to typowy formularz znany z Windows
  - nie musimy go projektować
  - otworzymy go w funkcji zwrotnej przycisku *Dane1...*



To już znamy: tworzenie funkcji zwrotnej



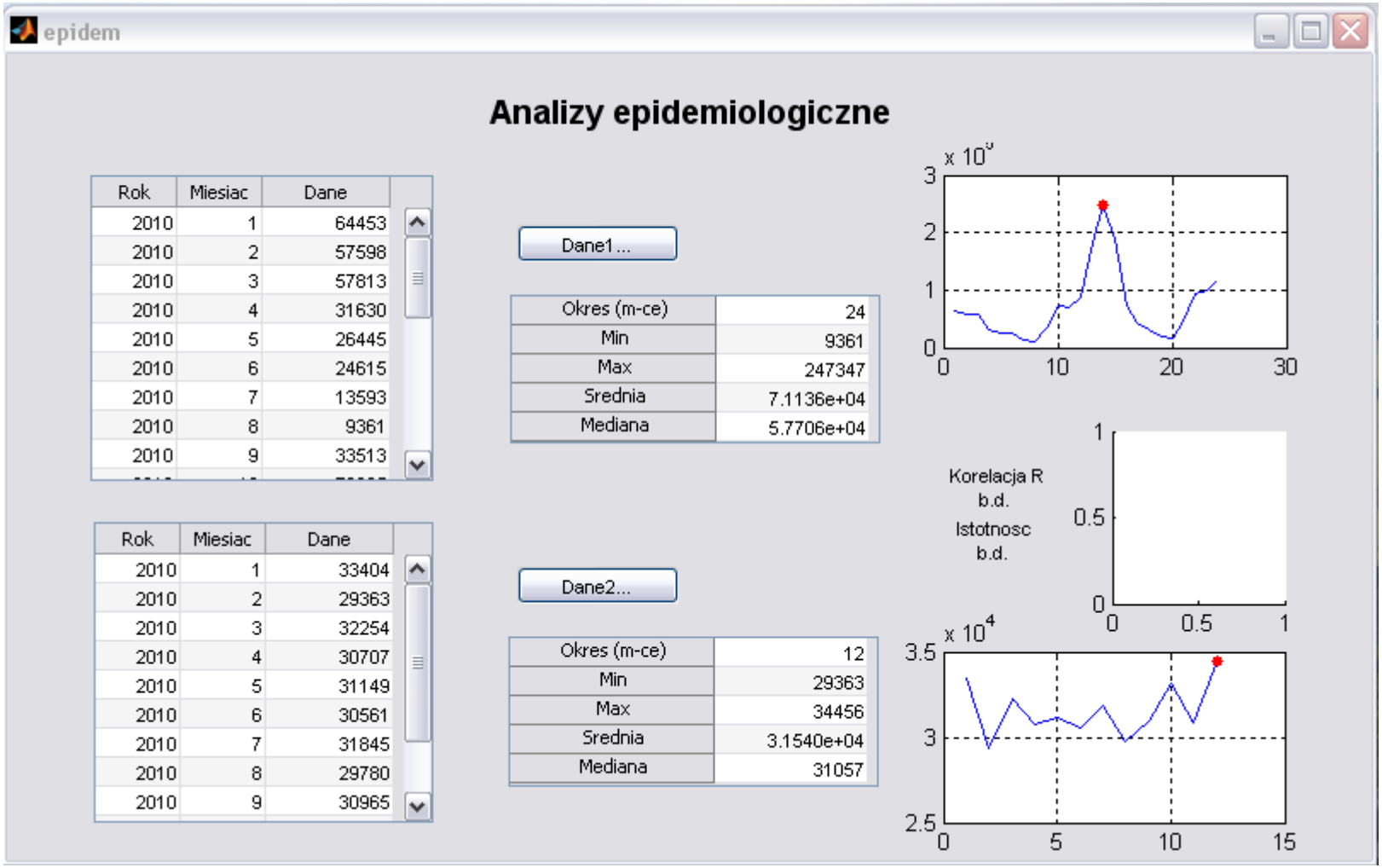
Otwieranie pliku jest bardzo proste

```
% --- Executes on button press in klawisz_Dane1.
function klawisz_Dane1_Callback(hObject, eventdata, handles)
% hObject    handle to klawisz_Dane1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Wywołujemy formularz otwierania pliku
% - parametrami są maska nazwy pliku oraz nazwa okna
% - formularz zwraca nazwę pliku i ścieżkę dostępu
%   lub 0 jeśli wybór pliku został anulowany
[plik, sciezka] = uigetfile('*.txt','Dane 1');

if plik~=0                                          % jeśli wybrano plik
    dane1 = load([sciezka '\' plik]);               % załaduj do zmiennej dane1
    set(handles.tabela1,'Data',dane1);              % wprowadź dane1 do
                                                    % tabeli handles.tabela1

    set(handles.tabela1,'Enable','on');             % umożliw zaznaczanie tabeli

                                                    % Dwie proste funkcje:
    statystyki(handles.tabela_stat1, dane1);        % - licz statystyki danych
    rysuj_wykres(handles.wykres1, dane1);           % - rysuj wykres
end
```

Umieszczanie danych w `uitable` też

```
% --- Executes on button press in klawisz_Dane1.
function klawisz_Dane1_Callback(hObject, eventdata, handles)
% hObject    handle to klawisz_Dane1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Wywołujemy formularz otwierania pliku
% - parametrami są maska nazwy pliku oraz nazwa okna
% - formularz zwraca nazwę pliku i ścieżkę dostępu
%   lub 0 jeśli wybór pliku został anulowany
[plik, sciezka] = uigetfile('*.txt','Dane 1');

if plik~=0                                      % jeśli wybrano plik
    dane1 = load([sciezka '\' plik]);           % załaduj do zmiennej dane1
    set(handles.tabela1,'Data',dane1);          % wprowadź dane1 do
                                                % tabeli handles.tabela1

    set(handles.tabela1,'Enable','on');         % umożliw zaznaczanie tabeli

                                                % Dwie proste funkcje:
    statystyki(handles.tabela_stat1, dane1);    % - licz statystyki danych
    rysuj_wykres(handles.wykres1, dane1);       % - rysuj wykres
end
```

Testujemy... z grubsza działa :-)

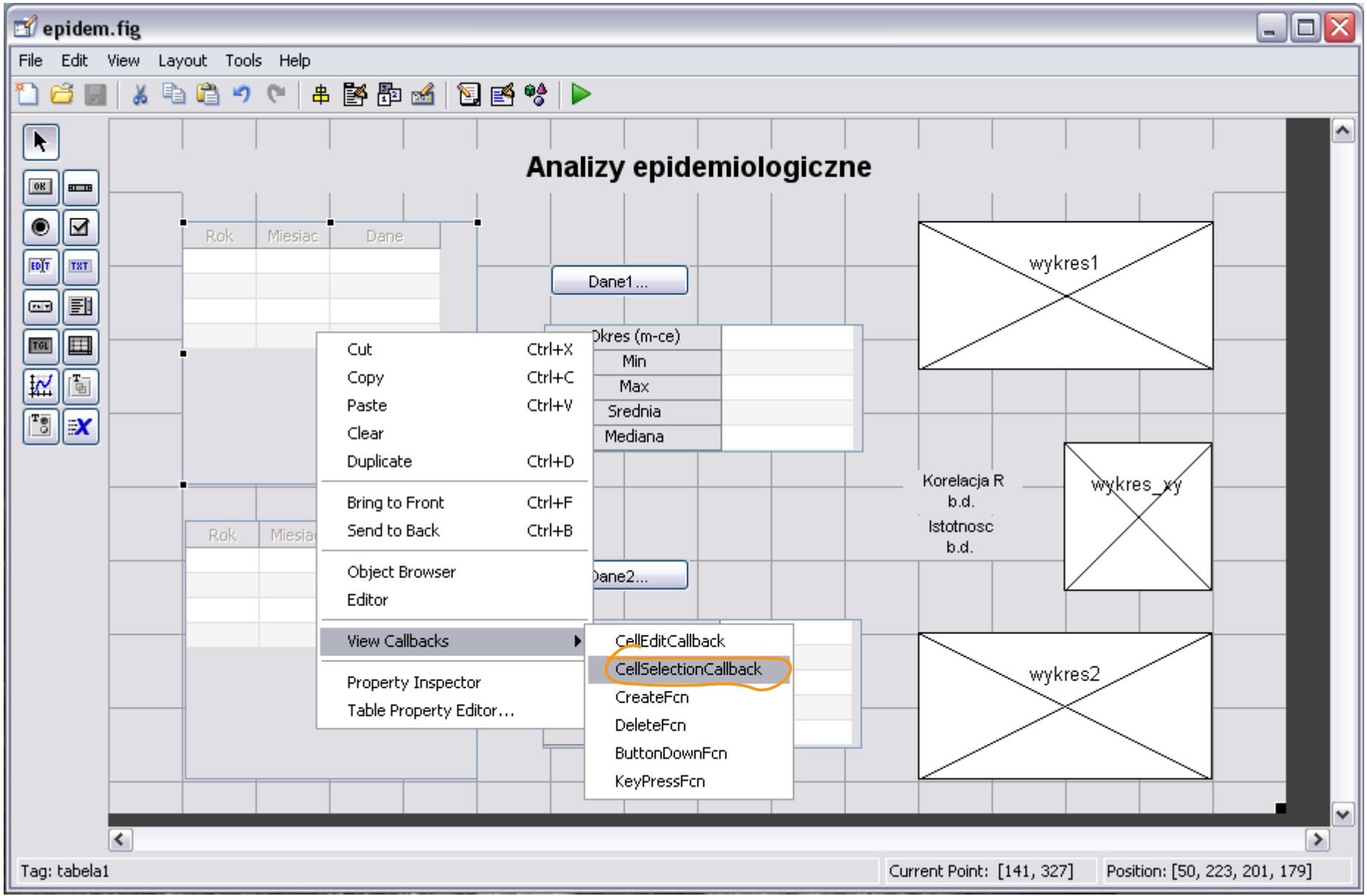


Pamiętamy

- Chcemy sprawdzić zależności pomiędzy danymi, np.
  - czy liczba zgonów koreluje z zachorowalnością na grypę?
  - potrzebujemy zaznaczyć dane za ten sam okres w obydwu tabelach

Rok	Miesiac	Dane
2011	2	247347
2011	3	190068
2011	4	73308
2011	5	42247
2011	6	31193
2011	7	18212
2011	8	15031
2011	9	45185
2011	10	93107

`uitable` obsługuje zdarzenie
zaznaczenie komórek



PU: Zaznaczenie danych (1)

- **Cel:**
  - zawężenie zakresu analiz statystycznych i wizualizacji
  - policzenie korelacji pomiędzy dwoma zbiorami danych
- **Warunki początkowe:**
  - Użytkownik zaznaczył zakres komórek tabeli
- **Warunki końcowe:**
  - PU zaktualizował analizy statystyczne i wykresy
  - PU obliczył korelacje liniową i narysował jej wykres
    - jeśli długości okresu z obydwu tabel zgodne

PU: Zaznaczenie danych (2)

- **Przebieg działania**
  - pobiera dane z tabeli 1
  - pobiera indeksy wierszy zaznaczonych komórek w tabeli 1
  - wywołuje wykonanie analiz statystycznych
    oraz wykresu danych dla zaznaczonego okresu w tabeli 1
  - pobiera indeksy wierszy zaznaczonych komórek w tabeli 2
  - porównuje liczbę wierszy zaznaczonych w tabelach 1 i 2
  - jeśli liczba wierszy równa
    - pobiera dane z tabeli 2
    - wywołuje obliczanie korelacji liniowej dla zaznaczonych wierszy z tabel 1 i 2
    - wyświetla wartość korelacji liniowej oraz rysuje wykres rozrzutu
  - jeśli liczba zaznaczonych wierszy w obydwu tabelach nie jest równa
    - czyści wartość korelacji oraz wykres rozrzutu

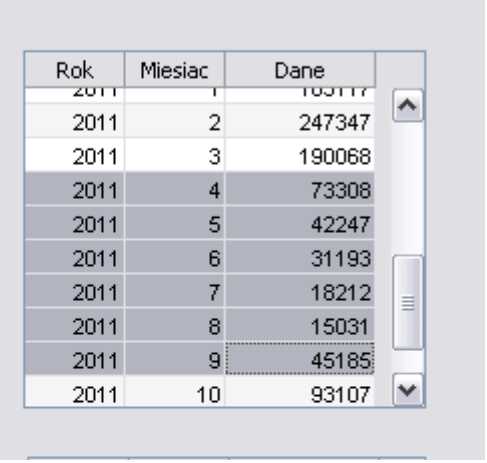
PU: Zaznaczenie danych (2)

- **Przebieg działania**
  - pobiera dane z tabeli 1
  - pobiera indeksy wierszy zaznaczonych komórek w tabeli 1
  - wywołuje wykonanie analiz statystycznych
    oraz wykresu danych dla zaznaczonego okresu w tabeli 1
  - pobiera indeksy wierszy zaznaczonych komórek w tabeli 2
  - porównuje liczbę wierszy zaznaczonych w tabelach 1 i 2
  - jeśli liczba wierszy równa
    - pobiera dane z tabeli 2
    - wywołuje obliczanie korelacji liniowej dla zaznaczonych wierszy z tabel 1 i 2
    - wyświetla wartość korelacji liniowej oraz rysuje wykres rozrzutu
  - jeśli liczba zaznaczonych wierszy w obydwu tabelach nie jest równa
    - czyści wartość korelacji oraz wykres rozrzutu

Zaznaczanie

- Sama informacja o zaznaczeniu nie wystarczy
  - musimy wiedzieć co zostało zaznaczone
  - dane te znajdują się w strukturze **eventdata**

```
% --- Executes when selected cell(s) is changed in tabela1.
function tabela1_CellSelectionCallback(hObject, eventdata, handles)
% hObject    handle to tabela1 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) currently selecteds
% handles    structure with handles and user data (see GUIDATA)

% Nas interesują tylko wiersze
selekcja = eventdata.Indices(:,1);
% Usuwamy ewentulane powtórzenia
selekcja = unique(selekcja);

% ...
```



Rok	Miesiac	Dane
2011	2	247347
2011	3	190068
2011	4	73308
2011	5	42247
2011	6	31193
2011	7	18212
2011	8	15031
2011	9	45185
2011	10	93107

Zapis wybranych indeksów (1)

- W `tabela1_CellSelectionCallback(hObject,eventdata,handles)`
  - w strukturze `eventdata` znajdują się tylko indeksy wybrane z tabeli 1
  - jak dotrzeć do indeksów wybranych z tabeli 2?
    - nie ma cudownych rozwiązań – trzeba je tymczasowo zapisać

Zapis wybranych indeksów (2)

- W `tabela1_CellSelectionCallback(hObject,eventdata,handles)`
  - w strukturze `eventdata` znajdują się tylko indeksy wybrane z tabeli 1
  - jak dotrzeć do indeksów wybranych z tabeli 2?
    - nie ma cudownych rozwiązań – trzeba je tymczasowo zapisać

```
function tabela1_CellSelectionCallback(hObject, eventdata, handles)

% Nas interesują tylko wiersze
selekcja = eventdata.Indices(:,1);
% Usuwamy ewentulane powtórzenia
selekcja = unique(selekcja);

% Zrobimy pewien trik - dodamy pole wybrane1 do struktury handles
% i zapiszemy w nim selekcję indeksów
handles.wybrane1 = selekcja;

% Ważne! Musimy powiedzieć GUI aby zaktualizował swoją kopię handles
guidata(hObject, handles);

% ...
```

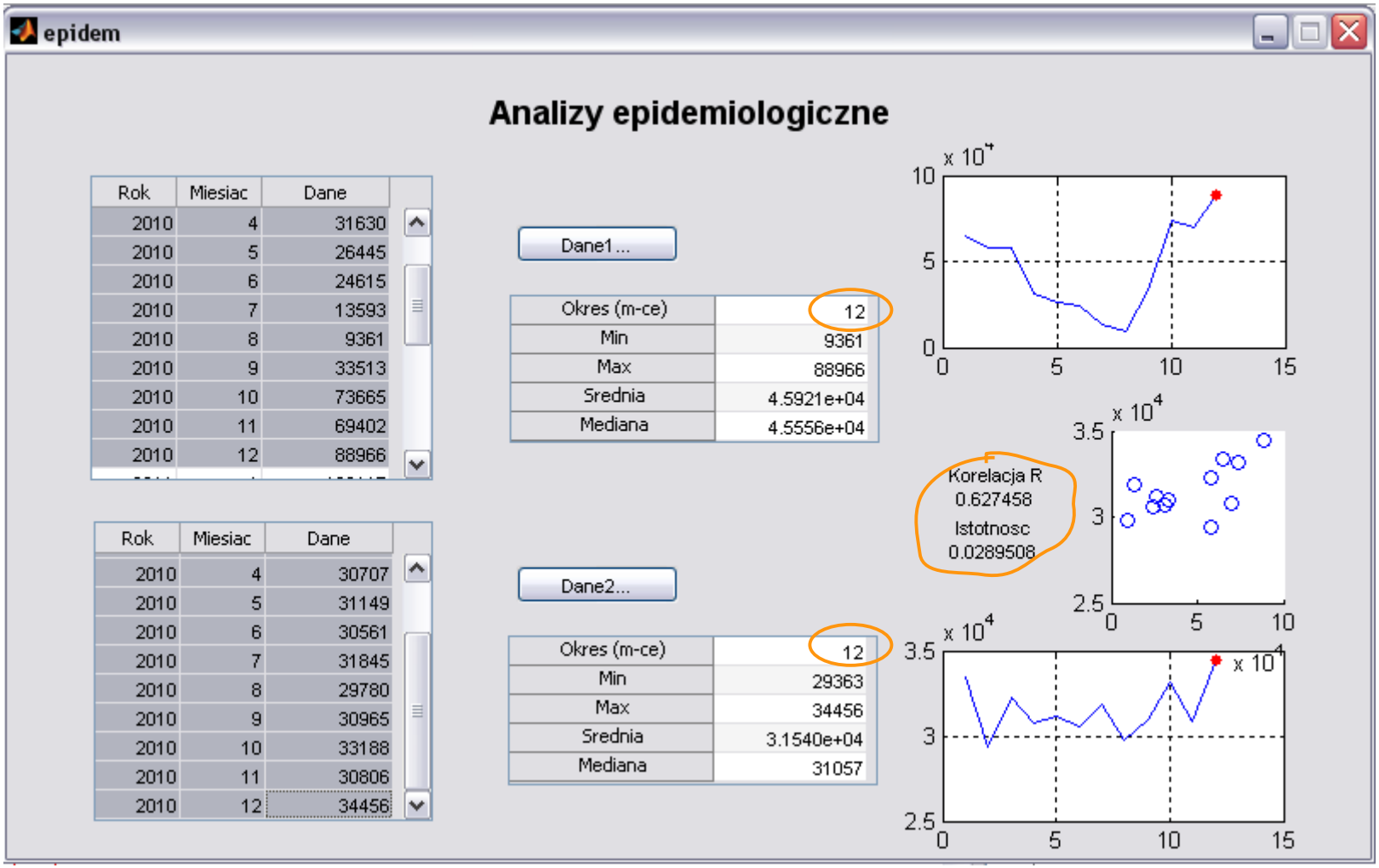
tabela1_CellSelectionCallback

```
function tabela1_CellSelectionCallback(hObject, eventdata, handles)

% Nas interesują tylko wiersze
selekcja = eventdata.Indices(:,1);
% Usuwamy ewentulane powtórzenia
selekcja = unique(selekcja);

% Zrobimy pewien trik - dodamy pole wybrane1 do struktury handles
% i zapiszemy w nim selekcję indeksów
handles.wybrane1 = selekcja;

% Ważne! Musimy powiedzieć GUI aby zaktualizował swoją kopię handles
guidata(hObject, handles);

% Pobieramy dane z tabeli 1
tabela = get(hObject,'Data');

% Aktualizujemy analizy statystyczne oraz wykres dla zaznaczonych danych
statystyki(handles.tabela_stat1, tabela(selekcja,:));
rysuj_wykres(handles.wykres1, tabela(selekcja,:));

% Liczymy i wyświetlamy korelację liniową
korelacja(handles)     % struktura handles zawiera wybrane indeksy
```

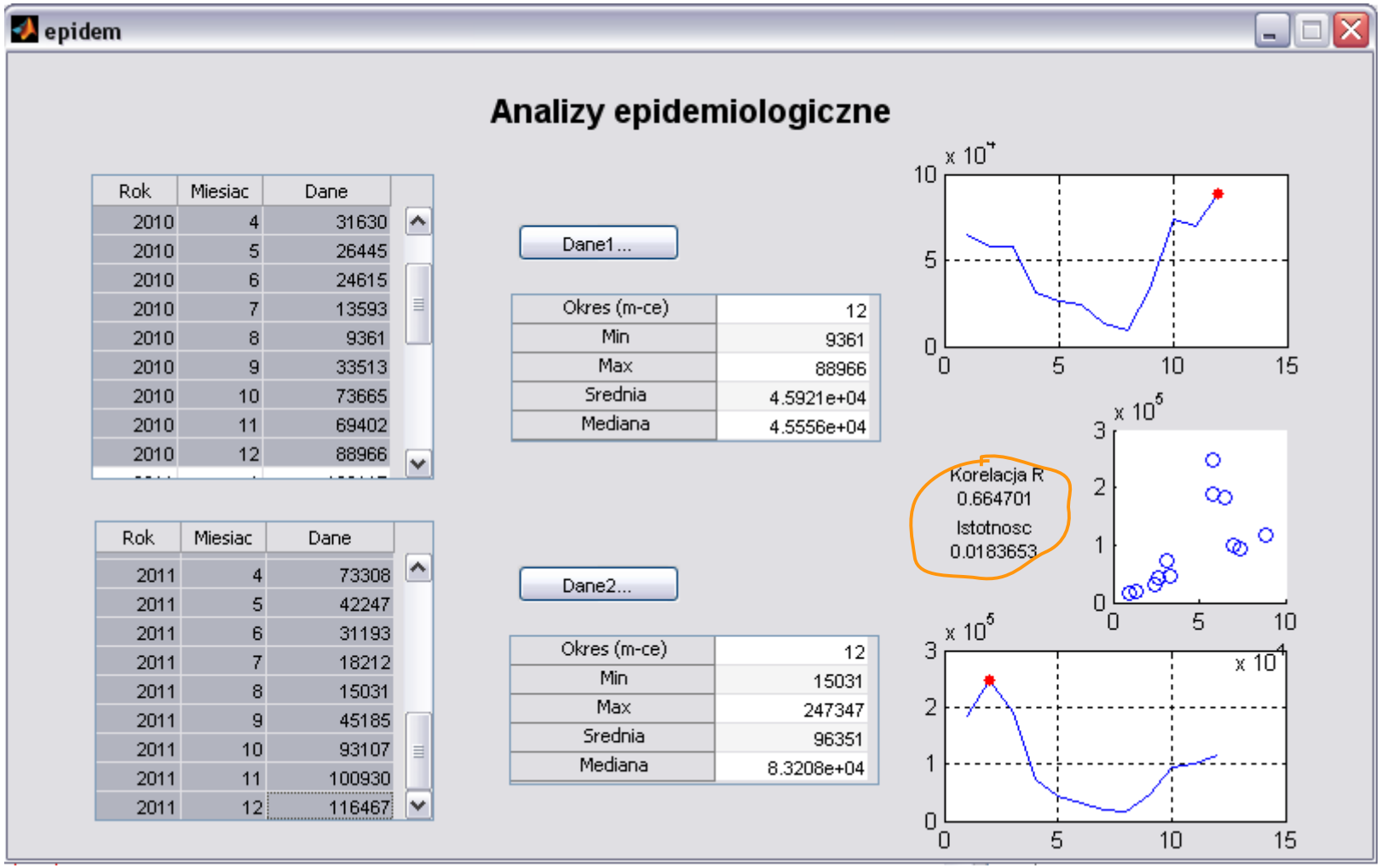
korelacja

```
function korelacja(handles)
%KORELACJA - liczy i wyświetla korelację liniową dla aplikacji EPIDEM

    dane1 = get(handles.tabela1,'Data');      % pobiera dane z tabeli 1
    dane2 = get(handles.tabela2,'Data');      % pobiera dane z tabeli 2
    wybrane1 = handles.wybrane1;  % pobiera zaznaczone indeksy w tab.1
    wybrane2 = handles.wybrane2;  % pobiera zaznaczone indeksy w tab.2

    % Czy oba zaznaczenie obejmują tyle samo wierszy?
    % Czy obejmują chociaż dwa wiersze?
    if length(wybrane1)~=length(wybrane2) || length(wybrane1)<2     % nie?

        cla(handles.wykres_xy);                           % czysc wykres
        set(handles.zmienna_Korelacja,'String','b.d.'); % czysc wart. kor.
        set(handles.zmienna_Istotnosc,'String','b.d.'); % czysc ist. kor.

    else                                                          % tak?

        % Wykres rozrzutu:
        scatter(handles.wykres_xy,dane1(wybrane1,3),dane2(wybrane2,3));
        % Korelacja liniowa:
        [kor, ist] = corr(dane1(wybrane1,3),dane2(wybrane2,3));
        set(handles.zmienna_Korelacja,'String',kor);   % ustawia wart.kor.
        set(handles.zmienna_Istotnosc,'String',ist);   % ustawia ist.kor.

    end
```

Zależność zgonów od zachorowalności na grypę



Korelacja pomiędzy zachorowalnością na grypę w latach 2010 i 2011



Sposoby zapisywania danych użytkownika

- GUI Data
  - dane powiązane z oknem (czyli z GUI)
  - jeśli GUI stworzony przez GUIDE
    - dane należy umieścić w strukturze `handles`
    - wymagane zapisanie funkcją `guidata` po modyfikacji
- UserData
  - każdy obiekt graficzny posiada własność `UserData`, której można przypisać dowolną zmienną
    - dostęp przez funkcje `set` i `get`
- Application Data
  - dane można przypisać do okna lub komponentu używając funkcji `setappdata` i odczytać `getappdata`

http://www.mathworks.com/help/techdoc/creating_guis/f5-998352.html#f5-998820

epidem - podsumowanie

- Nauczyliśmy się
  - otwierać pliki `uigetfile`
  - tworzyć i dostosowywać tabele `uitable`
  - zaznaczać komórki tabeli
    - korzystać ze struktury `evendata`
  - przechowywac dane aplikacji
    - jako pole struktury `handles`
  - liczyć korelacje

Dziś najważniejsze było...

- Grafika w Matlabie jest
  - obiektowa
  - uchwytowa
  - hierarchiczna

- Graficzny interfejs użytkownika
  - programowanie sterowane zdarzeniami
  - projektowanie: postaw się w miejscu użytkownika
  - implementacja: funkcje zwrotne

A za 2 tygodnie...

- Obliczenia numeryczne
  - ostatni wykład przed kolokwium 14.05
    - PN godz. 13.15-15.00

Kontrolki

- klasa `uicontrol`
  - `pushbutton` - zwykły przycisk
  - `edit` - pole edycji tekstu
  - `text` - tekst statyczny (etykieta)
  - `slider` - suwak
  - `checkbox` - pole zaznaczenia
  - `listbox` - lista wyboru
  - ...
- klasa `uitable` - tabela danych
- klasa `uimenu` - element menu okna

...