

Laboratorium MATLA

Ćwiczenie 5.

*Elementy programowania obiektowego.
Graficzny Interfejs Użytkownika (GUI)*

Opracowali:

- dr inż. Beata Leśniak-Plewińska

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

Warszawa, 2019

I. Cel ćwiczenia

W ramach ćwiczenia studenci zapoznają się z różnymi metodami tworzenia w MATLAB'ie graficznego interfejsu użytkownika (GUI z ang. *Graphical User Interface*).

II. Przygotowanie do ćwiczenia

W celu przygotowania się do ćwiczenia zalecane jest zapoznanie się z podstawami dotyczącymi metod tworzenia GUI w MATLAB'ie, w szczególności z następującymi materiałami dostępnymi na stronie firmy Mathworks:

1. [Tworzenie w MATLAB'ie aplikacji korzystających z graficznego interfejsu użytkownika](#)
2. [Tworzenie prostej aplikacji z użyciem narzędzia GUIDE](#)

III. Informacje podstawowe

Większość programów wymaga interfejsu umożliwiającego pobieranie od użytkownika danych wejściowych i zwracanie użytkownikowi wyników. Dotychczasowe programy tworzone na zajęciach laboratoryjnych korzystały z interfejsu tekstowego. Bardziej intuicyjną komunikację umożliwia graficzny interfejs użytkownika (GUI z ang. *Graphical User Interface*). GUI 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.

W MATLAB'ie GUI można utworzyć w sposób programistyczny oraz za pomocą dwóch interaktywnych narzędzi: **GUIDE** (z ang. *Graphical User Interface Development Designer*) oraz **App Designer**.

Tworzenie GUI w sposób programistyczny wymaga wykorzystania właściwych funkcji MATLAB'a niezbędnych do utworzenia okna graficznego i umieszczenia w nim wymaganych komponentów interfejsu, tzw. obiektów **UIcontrol**, oraz zaimplementowania funkcji niezapewniających obsługi tych obiektów. Więcej o obiektach **UIcontrol** można znaleźć [tutaj](#).

GUIDE i **App Designer** wspomagają programistę w tym procesie dzięki interaktywnemu środowisku, które generuje szkielet programu (m-plik) obsługującego elementy GUI (obiekty graficzne). **App Designer** został po raz pierwszy wprowadzony przez firmę Mathworks wraz z programem MATLAB w wersji 2016a i jest on obecnie zalecanym narzędziem służącym do tworzenia GUI. Jego przewagą nad **GUIDE** jest poszerzony zestaw komponentów i bardziej intuicyjny sposób tworzenia interfejsu i całej aplikacji.

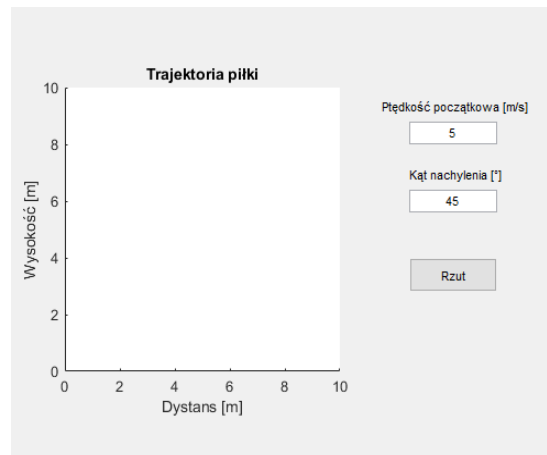
A. Tworzenie GUI w sposób programistyczny

Poniżej przedstawiono przykładową funkcję tworzący prosty GUI.

```
1 function guiTrajektoria_A
2
3 f = figure('Name','guiTrajektoria_A',...
4           'Position',[650,350,500,400]);
```

```
5  osie = axes('Units','pixels','Position',[50,80,250,250], ...
6      'xlim',[0 10], 'ylim',[0 10]);
7  osie.XLabel.String = 'Dystans [m]';
8  osie.YLabel.String = 'Wysokość [m]';
9  osie.Title.String = 'Trajektoria piłki';
10
11  textPredk = uicontrol('Style','text',...
12      'String','Prędkość początkowa [m/s]',...
13      'Position',[330,300,150,20]);
14  velocityBox = uicontrol('Style','edit',...
15      'String','5',...
16      'Tag','velocityBox',...
17      'Position',[363,270,80,20]);
18  textKat = uicontrol('Style','text',...
19      'String','Kąt wyrzutu [stopnie]',...
20      'Position',[330,200,150,20]);
21  angleBox = uicontrol('Style','edit',...
22      'String','45',...
23      'Tag','angleBox',...
24      'Position',[363,170,80,20]);
25  throwPush = uicontrol('Style','pushbutton',...
26      'String','Rzut',...
27      'Tag','throwPush',...
28      'Position',[363,90,80,20],...
29      'Callback',@pushbuttonCallback);
30
31  end
32
33  function pushbuttonCallback(source,~)
34
35      g = 9.81
36      velObj = findobj('Tag','velocityBox');
37      v0 = str2double(velObj.String);
38      angleObj = findobj('Tag','angleBox');
39      theta = str2double(angleObj.String)*pi/180;
40      t1 = 2*v0*sin(theta)/g;
41      t=0:0.01:t1;
42      x = v0*cos(theta)*t;
43      y = v0*sin(theta)*t - g*t.^2/2;
44
45      hold on
46      comet(x,y)
47
48  end
```

Uruchomienie funkcji powoduje utworzenie GUI przedstawionego na Rys. 1, w którym użytkownik może wprowadzić dwie wartości liczbowe (wartość prędkości początkowej piłki i kąt, w stosunku do poziomu, pod jakim zostaje wyrzucona piłka, a następnie dla wprowadzonych wartości zostają uruchomione obliczenia i animacja graficzna.



Rysunek 1: GUI będący wynikiem uruchomienia przykładowego skryptu

Linie 3-29 m-pliku służą do konfiguracji interfejsu w tym do zdefiniowania obiektów **figure** i **axes** (osie) oraz pięciu elementów będących obiektami **UIcontrol**, w tym odpowiednio: dwóch obiektów **static text** (tekst statyczny, etykieta), dwóch **editable text** (tekst edytowalny wprowadzany z klawiatury przez użytkownika) i obiektu **pushbutton** (przycisk).

Linia 3 służy do utworzenia głównego okna interfejsu (obektu **figure**). Jednocześnie, poprzez podanie wartości właściwości **Name** zdefiniowano tytuł okna głównego GUI, a przez podanie wartości właściwości **Position** obiektu, zdefiniowano jego pozycję (odległości wzdłuż osi x i y lewego dolnego rogu okna graficznego od lewego dolnego rogu ekranu) oraz jego wymiary (wysokość i szerokość) (w pikselach – jednostka domyślna). Więcej informacji o właściwościach obiektu **figure** można uzyskać [tutaj](#).

W liniach 5-9 utworzono obiekt **axes**. W linii 5, jednocześnie z utworzeniem tego obiektu, zdefiniowano jednostkę, w której określane są pozycja i wymiary tego obiektu jako piksele (wartość właściwości 'Units' obiektu **axes**) oraz określono pozycję obiektu **axes** (w odniesieniu do lewego dolnego rogu obiektu zawierającego obiekt **axes**, np. obiektu **figure**) oraz jego wymiary. Ponadto, tworząc obiekt **axes** zdefiniowano również zakresy wartości dla obu osi. Dalej (linie 10-12) zdefiniowano etykiety oraz tytuł osi. Więcej informacji o właściwościach obiektu **axes** można znaleźć [tutaj](#).

W kolejnych liniach (11-29) zdefiniowano pozostałe elementy interfejsu pozwalające na komunikację użytkownika z aplikacją. Są to obiekty **UIcontrol**: dwa o stylu **text** (tekst statyczny), dwa o stylu **edit** (tekst edytowalny) oraz jeden o stylu **pushbutton** (przycisk). Więcej o obiektach **UIcontrol** można znaleźć [tutaj](#).

Dla każdego obiektu **UIcontrol** można zdefiniować funkcje **callback** (wywołania zwrotnego). Funkcje te nie są wywoływane bezpośrednio, lecz przez system zarządzania obiektami **UIcontrol** w odpowiedzi na **event** (zdarzenie), np. naciśnięcie przycisku (obektu **pushbutton**) lub zmianę wartości obiektu **editable**. System zarządzania obiektami **UIcontrol** przekazuje do funkcji wywołania zwrotnego dwie zmienne, który przekazuje do

nich dwie zmienne. Pierwszą z nich (`pushButton` w linii 33) jest uchwyt do struktury danych definiującej właściwości obiektu **UIcontrol** (np. pole `pushButton.BackgroundColor` przechowuje definicję koloru tła przycisku) który wyzwala dane wywołanie zwrotne – w wypadku powyższego przykładu jest to przycisk. Druga zmienna (~ w linii 28) jest zarezerwowana dla przyszłych wersji MATLAB'a (tylko jest w MATLAB'ie wykorzystywana w miejscu nieużywanego argumentu wejściowego). Więcej informacji o funkcjach wywołania zwrotnego można znaleźć [tutaj](#).

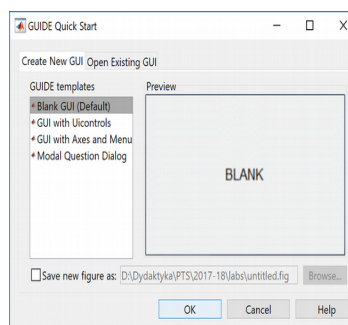
W naszym przykładzie zdefiniowana została funkcja wywołania zwrotnego jedynie dla przycisku `Rzut` (linie 33-48). Naciśnięcie przycisku `Rzut` powoduje jej wywołanie, tzn. zawsze gdy naciśnięty zostanie przycisk `Rzut` uruchomiona zostaje funkcja `pushbuttonCallback`. Funkcja ta wyszukuje uchwyty do dwóch obiektów o znanych wartościach właściwości `Tag` (linie 36 i 68) i pobiera wartość właściwości `String` dwóch obiektów `editable` (linie 30-31) jednocześnie przekształcając je z postaci łańcuchów znakowych reprezentujących wartości liczbowe do postaci liczbowej. Następnie wyznacza trajektorię piłki (linie 40-43) i uruchamia w obiekcie `axes` graficzną animację lotu piłki (linia 46).

B. Tworzeni GUI za pomocą narzędzia **GUIDE**

Aplikację z GUI o fikcjonalności identycznej jak ta z p. A możemy utworzyć również korzystając z narzędzia **GUIDE**. Narzędzie **GUIDE** możemy uruchomić wywołując w oknie poleceń komendę `guide`. Polecenie to otwiera okno **GUIDE Quick Start** (Rys. 2). Użytkownik uzyskuje możliwość wyboru pustego edytora GUI, wyboru spośród kilku typowych wzorców rozmieszczenia elementów interfejsu oraz otworzenia istniejącego GUI w celu jego modyfikacji.

Aplikacje tworzone w **GUIDE** są zapisywane jako pliki graficzne z rozszerzeniem `.fig` i w formie m-pliku.

Polecenie `guide` otwiera okno **GUIDE Quick Start** (Rys. 2). W celu stworzenia nowego GUI należy wybrać opcję **Blank GUI (Default)** i kliknąć OK otwierając okno edytora **Layout – untitled.fig** (Rys. 3).

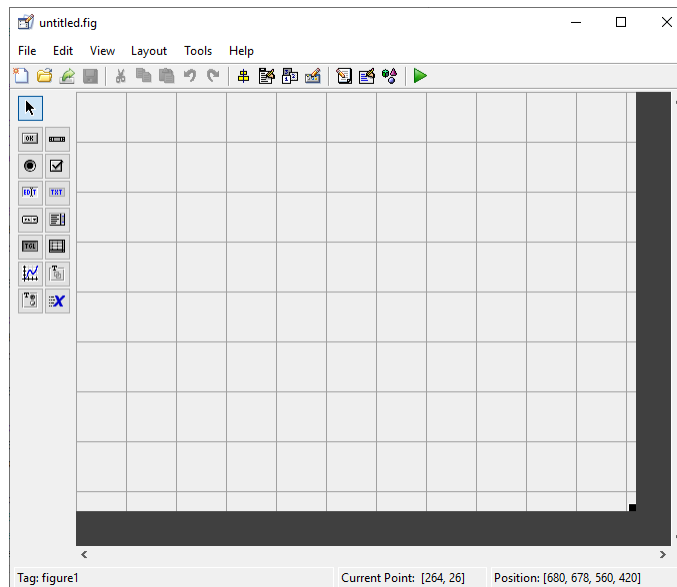


Rysunek 2: Okno **GUIDE Quick Start**

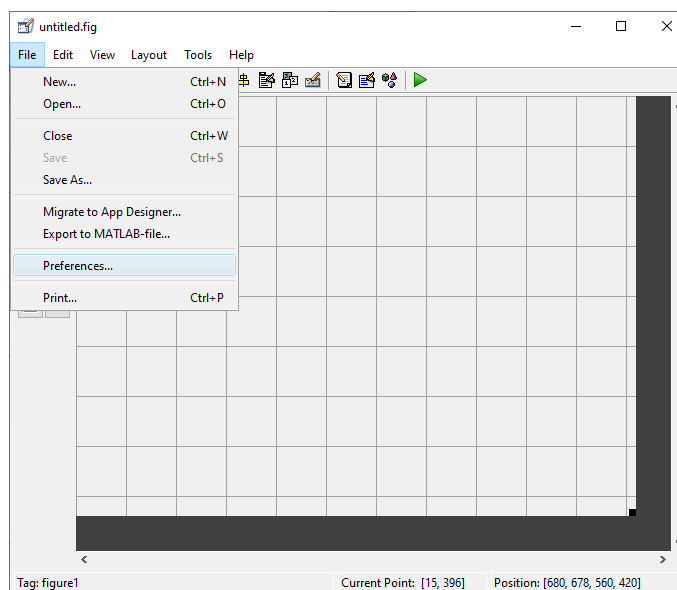
W centralnej części okna edytora **Layout** (część zakratkowana liniami siatki) znajduje się obszar wyświetlany po uruchomieniu aplikacji, dla której tworzony jest interfejs graficzny. W tym obszarze wstawiane są elementy/obiekty GUI. Z lewej strony okna **GUIDE** znajduje

się pasek narzędzi z ikonami odpowiadającymi poszczególnym komponentom, z których może zostać zbudowane. Aby poznać nazwy poszczególnych komponentów należy albo najechać kursorem myszy na ikonę danego komponentu, albo zmienić ustawienia **GUIDE**. W tym celu należy z menu okna **GUIDE** wybrać pozycję **File** → **Preferences** (Rys. 4). Następnie należy zaznaczyć pole wyboru dla pozycji *Show names in component palette* (Rys. 5).

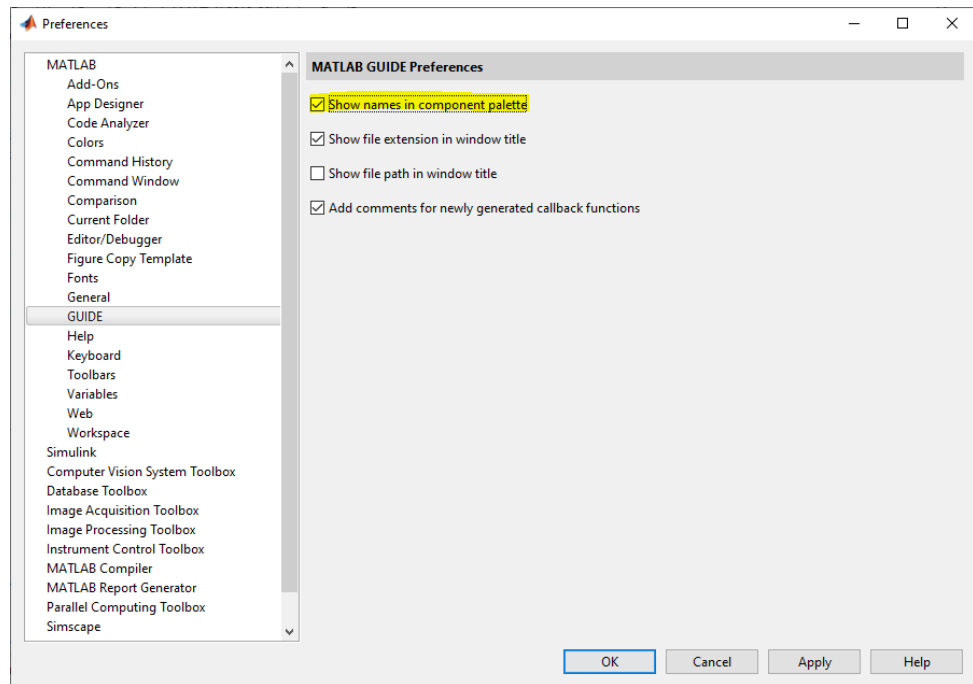
Po zaakceptowaniu zmiany okno edytora GUI będzie miało postać przedstawioną na Rys. 6.



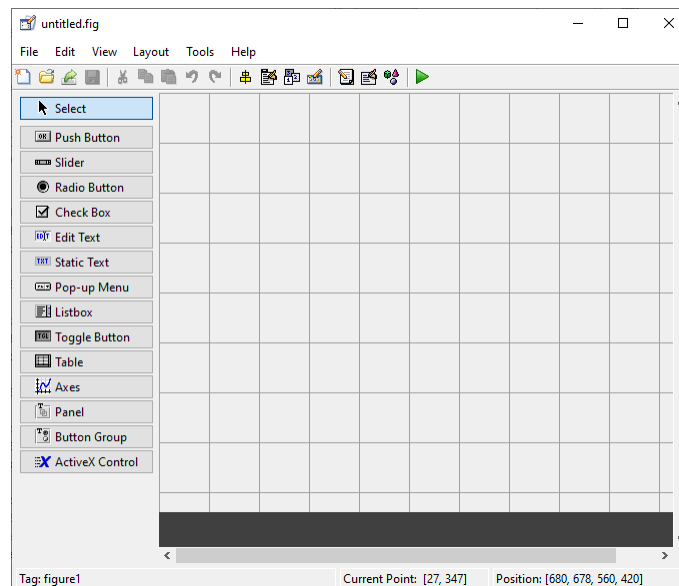
*Rysunek 3: Okno edytora **Layout***



*Rysunek 4: Okno edytora **Layout** . Pozycja **Preferencje** menu **File***



Rysunek 5: Zaznaczenie pola wyboru dla pozycji *Show names in component palette*



Rysunek 6: Okno edytora *Layout*. Pozycja *Preferencje* menu *File*

Poszczególne narzędzia służą do:

- Select** - wyboru/wskazania komponentu
- Push Button** - utworzenia przycisku, który po naciśnięciu wywołuje akcję zdefiniowaną w funkcji wywołania zwrotnego (*Callback*)
- Slider** - utworzenia suwaka

- Radio Button** - utworzenia przycisku, który posiada dwa stany: włączony i wyłączony; jeśli przycisk ten jest w grupie to w momencie aktywacji dezaktywuje on pozostałe przyciski opcji z grupy.
- Checkbox** - utworzenia przycisku wyboru; jeśli przycisk ten jest w grupie, to wybranie jednego z nich nie anuluje wyboru pozostałych
- Edit Text** - utworzenia pola tekstowego umożliwiającego wprowadzanie i edycję tekstu; wywołanie zwrotne jest wykonywane po naciśnięciu klawisza ENTER
- Static Text** - utworzenia etykiety opisującej inne obiekty graficzne w GUI lub pole obliczeniowe; użytkownik nie może interaktywnie wpływać na ten obiekt; nie posiada on wywołania zwrotnego
- Popup Menu** - utworzenia menu kontekstowego umożliwiającego wybór jednego z elementów na liście
- Listbox** - utworzenia listy rozwijanej umożliwiającej wybór jednego lub więcej jej elementów
- Toggle Button** - utworzenia przycisku przełącznika bistabilnego – wciśnięty pozostaje w tym stanie aż zostanie wyciśnięty; wywołanie zwrotne jest wykonywane po zwolnieniu naciśniętego przycisku myszki wskazującej ten przycisk
- Table** - utworzenia tabeli
- Axes** - utworzenia osi współrzędnych
- Panel** - utworzenia panelu; panel 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
- Button Group** - utworzenia grupy przycisków **Radio Buttons** lub przycisków **Toggle Buttons**.
- Toolbar** - utworzenia paska narzędzi
- ActiveX Control** - utworzenia kontrolki ActiveX

Właściwe działanie GUI zapewniają znane nam już wywołania zwrotne (**Callbacks**), o których wspomniano w pp. A. Wybrane rodzaje i własności wywołań zwrotnych 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 (Rys. 7):
 - „wycięcie” elementu (**Cut**)
 - skopiowanie elementu (**Copy**)

- wklejenie „wyciętego” elementu (dostępna po użyciu opcji **Cut**) (**Paste**)
- „wyczyszczenie” elementu z okna (**Clear**)
- 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

Właściwość wywołania zwrotnego	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 zwrotne komponentu jest aktywne.
ResizeFcn	Jest wykonywane wtedy, gdy użytkownik zmienia rozmiar obiektu Panel lub Button Group .
SelectionChangeFcn	Jest wykonywane wtedy, gdy użytkownik wybierze inny przycisk opcji (Radio Button) lub przełącznika (Toggle Button)

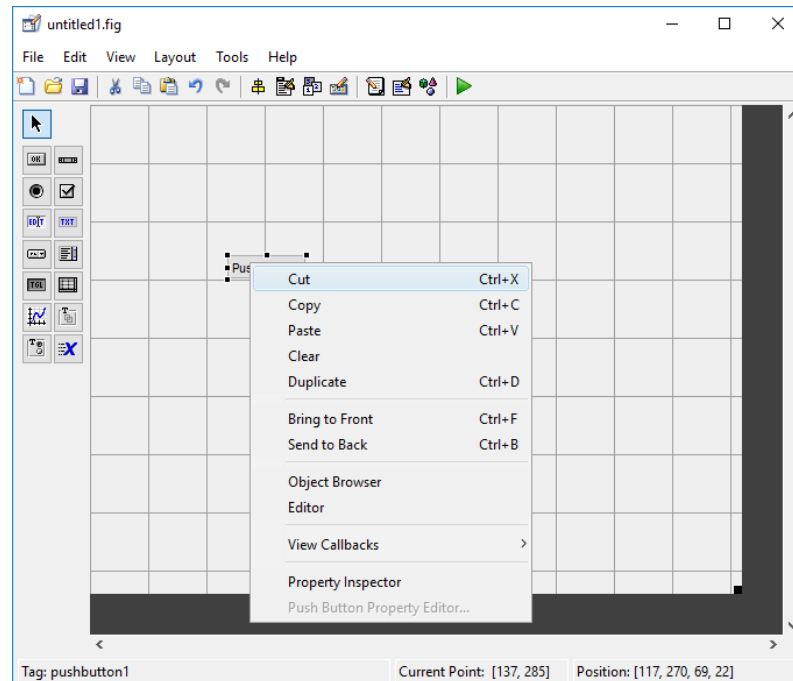
Programowanie interfejsu graficznego rozpoczyna się od określenia właściwości jego elementów (obiektów graficznych). Właściwości tych elementów są przechowywane w odpowiednich polach struktury o nazwie handles. Struktura ta zawiera uchwyty do wszystkich elementów GUI. Może ona zawierać także dane użytkownika służącą do przekazywania danych między wywołaniami zwrotnymi dla różnych elementów GUI.

W pierwszej kolejności należy zdefiniować wartości właściwości Tag definiowanego komponentu GUI. Wartość właściwości Tag jest bowiem wykorzystywana do automatycznego tworzenia funkcji związanych z wywołaniami zwrotnymi 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 właściwości użytych obiektów za pomocą funkcji get i set.

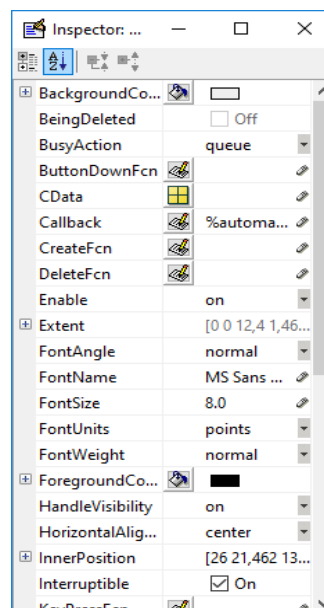
Określanie i modyfikację wartości właściwości obiektów umożliwia *Inspektor Właściwości (Property Inspector)* (Rys. 8). Okno inspektora właściwości dla danego obiektu można otworzyć wykorzystując jedną z poniższych możliwości:

- kliknąć obiekt dwukrotnie


- kliknąć ikonę **Property Inspector** na pasku narzędzi **GUIDE Layout Editor** (Rys. 9)
- z menu **View** wybrać opcję **Property Inspector**
- z lokalnego menu kontekstowego wywołanego prawym przyciskiem myszy wybrać opcję **Property Inspector**.



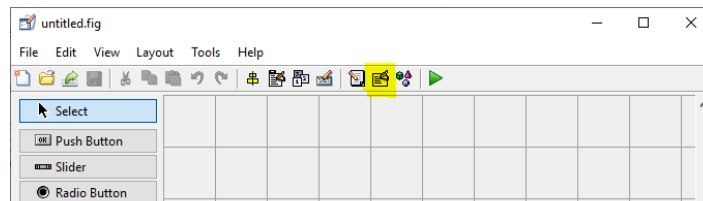
Rysunek 7: Okno edytora **Layout**. Menu kontekstowe otwarte za pomocą prawego przycisku myszy.



Rysunek 8: 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**.

GUI jest zapisywane w dwóch plikach, pliku *.m i pliku *.fig. W m-pliku zawarty jest kod aplikacji, natomiast w fig-pliku zawarta jest struktura GUI.



Rysunek 9: Okno edytora **Layout**. Ikona **Property Inspector**.

C. Tworzenie GUI za pomocą *App Designer*

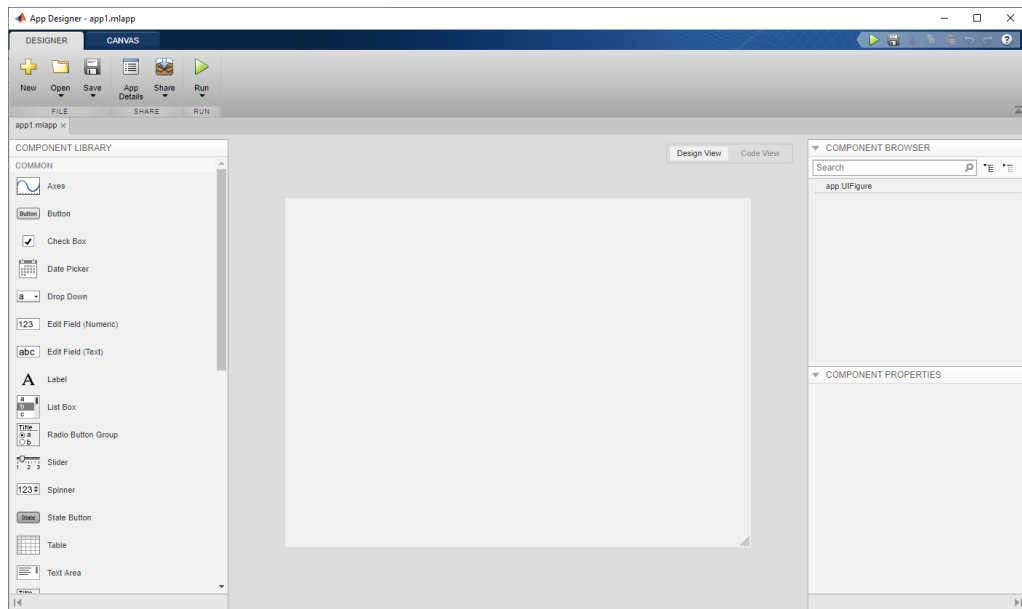
Wprowadzone do środowiska MATLAB w wersji 2016a narzędzie **App Designer** również umożliwia utworzenie GUI, jednak w porównaniu do **GUIDE** oferuje ono więcej komponentów interfejsu oraz bardziej intuicyjny proces tworzenia interfejsu zwanego aplikacją (lub w skrócie app'ką). Ponadto **App Designer** w pełni korzysta z obiektowości środowiska MATLAB (np. miejsce funkcji zajmują tu metody).

App Designer może zostać otwarty albo poprzez uruchomienie w oknie **Command Window** polecenia `appdesigner` albo poprzez wybranie pozycji **New->App** z menu **Home**. W wyniku otwarte zostaje nowe okno (Rys. 10), w którym odbywa się cały proces tworzenia GUI.

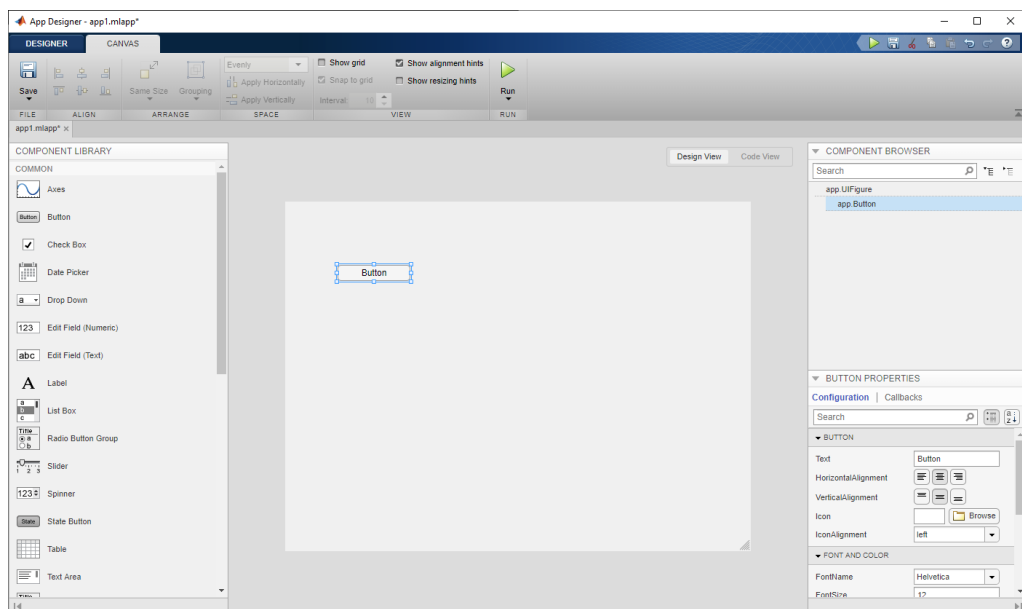
Okno **App Designer** jest podzielone na 3 części. Z lewej strony znajduje się panel **Component Library**, zawierający bibliotekę dostępnych komponentów (suwaków, wskaźników, przycisków itp.). Biblioteka ta zawiera komponenty zbliżone do tych dostępnych w narzędziu **GUIDE** oraz szereg nowych komponentów, wśród których na szczególną uwagę zasługuje grupa komponentów **INSTRUMENTATION**. Wybrane komponenty możemy przeciągnąć na obszar projektowanego interfejsu znajdujący się w środkowej części okna - panelu **Design View**. Panel ten umożliwia aranżowanie przestrzenne komponentów GUI w obrębie obszaru obiektu **figure** (jaśniejszy obszar panelu **Design View**).

Po prawej stronie, w panelu **Component Browser**, wyświetlona jest lista nazw użytych komponentów. Komponent `app.UIFigure` jest tworzony automatycznie w momencie uruchomienia narzędzia **App Designer**. Po zaznaczeniu danego komponentu umieszczonego w panelu **Design View**, w oknie **Properties** znajdującym się w prawej dolnej części ekranu, wyświetlane są właściwości odpowiedniego mu obiektu. Okno **Properties** umożliwia modyfikację tych właściwości. (Rys. 11). Należy zwrócić uwagę na fakt, że komponenty o nazwach identycznych jak te występujące w narzędziu **GUIDE** mogą w **App Designer** posiadać właściwości inne niż w **GUIDE**.

Więcej informacji na temat komponentów dostępnych w **App Designer** można znaleźć [tutaj](#).



Rysunek 10: Okno narzędzia *App Designer*.

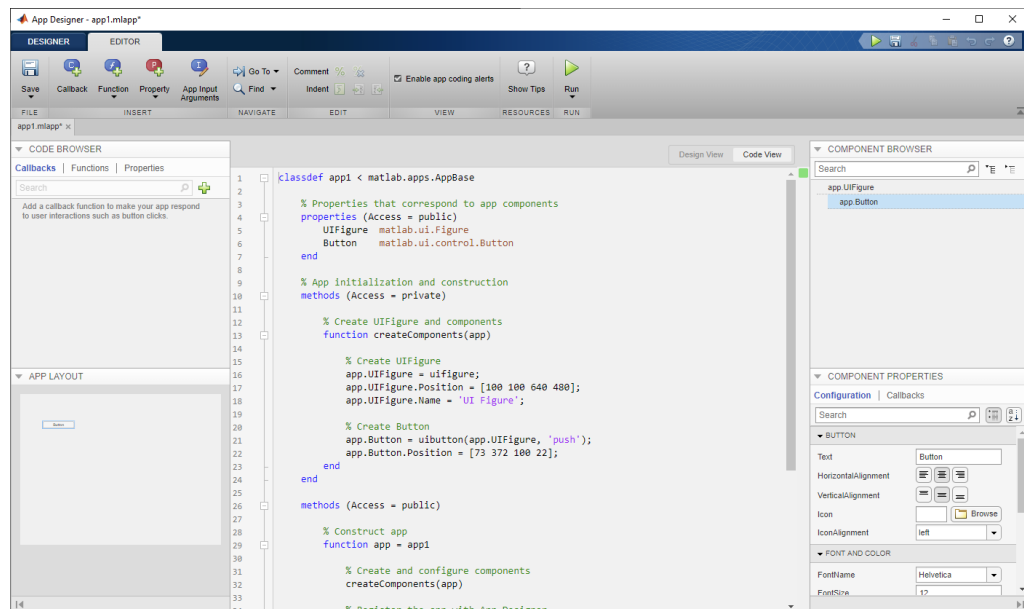


Rysunek 11: Okno narzędzia *App Designer* z elementem umieszczonym w panelu *Desing View* .

Narzędzie *App Designer* automatycznie generuje kod służący do obsługi elementów graficznych pobranych z *Component Library*. Nie jest to jednak kompletna aplikacja, ponieważ *App Designer* nie może zdefiniować wzajemnych relacji pomiędzy tymi elementami. Aktualny kod aplikacji jest widoczny po przełączeniu centralnego panelu z trybu *Design View* na *Code View*. Dodatkowo zakładka *CANVAS* zmienia się na *EDITOR*, po lewej stronie u góry pojawia się panel *CODE BROWSER* (umożliwiający

modyfikację kodu, np. dodawania funkcji prywatnych), a po lewej na dole pojawia się panel **APP LAYOUT** (służący do ilustracji wyglądu GUI) (Rys. 12).

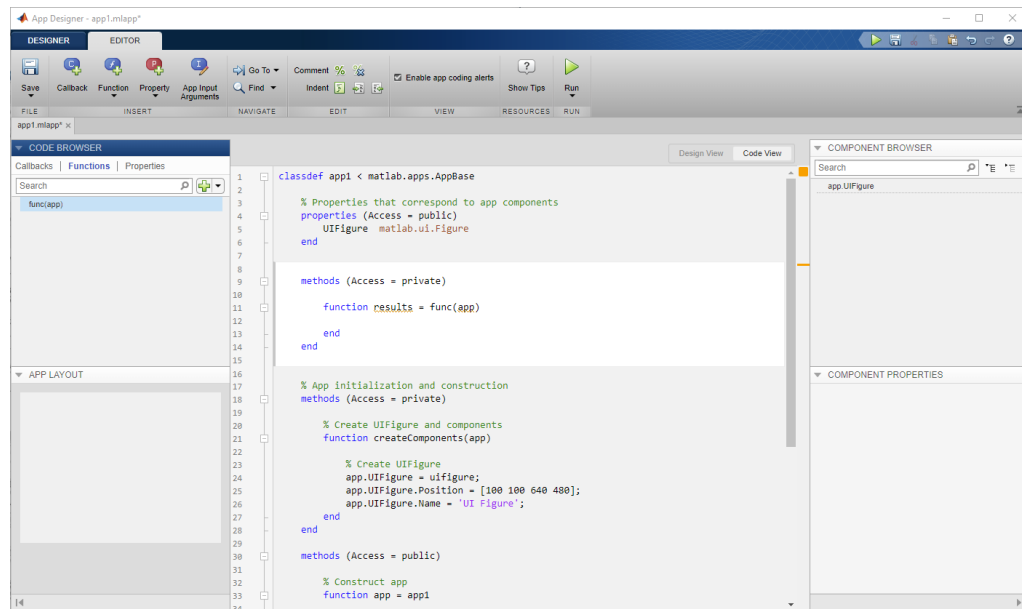
W przypadku **App Designer** użytkownik może dopisać i edytować wywołania zwrotne oraz dodatkowe funkcje i zmienne jednak tylko w przewidzianych do tego miejscach programu wygenerowanego dla użytych elementów GUI. Pozostała część automatycznie generowanego kodu nie może być edytowana i jest wyświetlana na szarym tle. Ma to na celu zwiększenie niezawodności kodu, gdyż jego istotne fragmenty nie mogą być przypadkowo zmienione przez użytkownika.



Rysunek 12: Okno narzędzia **App Designer** – widok panelu **Code View**.

Podobnie jak w przypadku narzędzia **GUIDE**, **App Designer** proponuje miejsce wstawienia wywołania zwrotnego i jego kod szkieletowy dla każdego elementu GUI. Rolą użytkownika jest zaakceptowanie lub modyfikacja tych wywołań oraz wprowadzenie nazw i parametrów funkcji, które będą wykonane w odpowiedzi na zdarzenia przewidziane w projekcie aplikacji.

Własne funkcje można utworzyć w trybie **Code View**, klikając zakładkę **Functions** w oknie **CODE BROWSER**. Następnie, z menu rozwijanego **+** należy wybrać pozycję **Private Function**. W efekcie w panelu **Functions** okna **CODE BROWSER** pojawia się nowa funkcja `func(app)`. Funkcję tę można odszukać samodzielnie w oknie **Code View**, można też ją zlokalizować, klikając nazwę funkcji `func(app)` w panelu **Functions** okna **CODE BROWSER**. Jest ona edytowalna. Może też zostać usunięta lub zastąpiona innymi funkcjami (Rys. 13).



Rysunek 13: Okno narzędzia *App Designer* – widok panelu *Code View*.

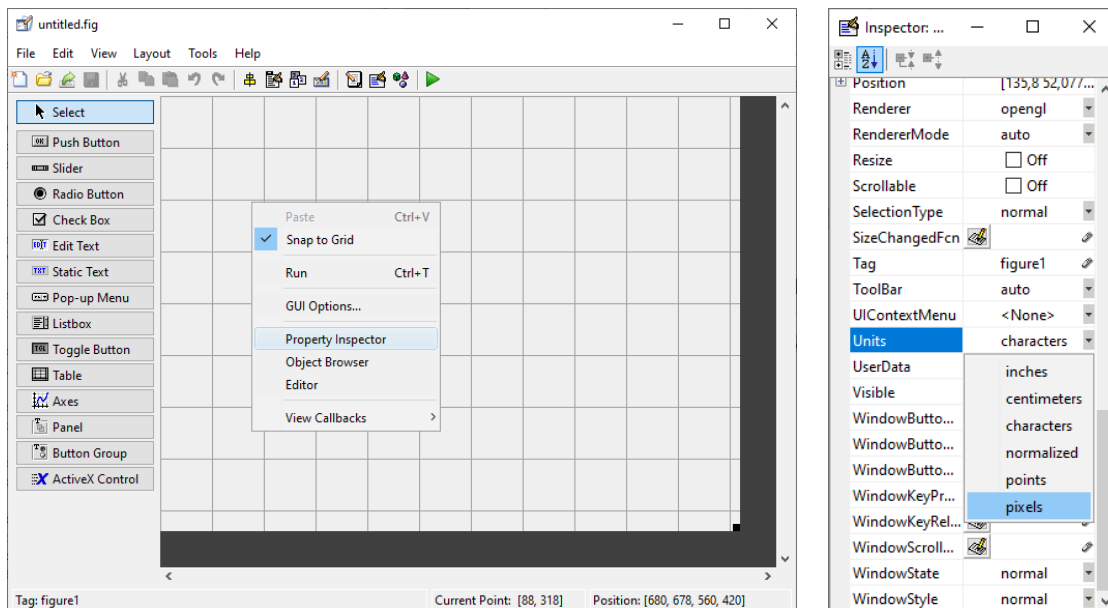
IV. Realizacja ćwiczenia

A) Tworzenie GUI w sposób programistyczny

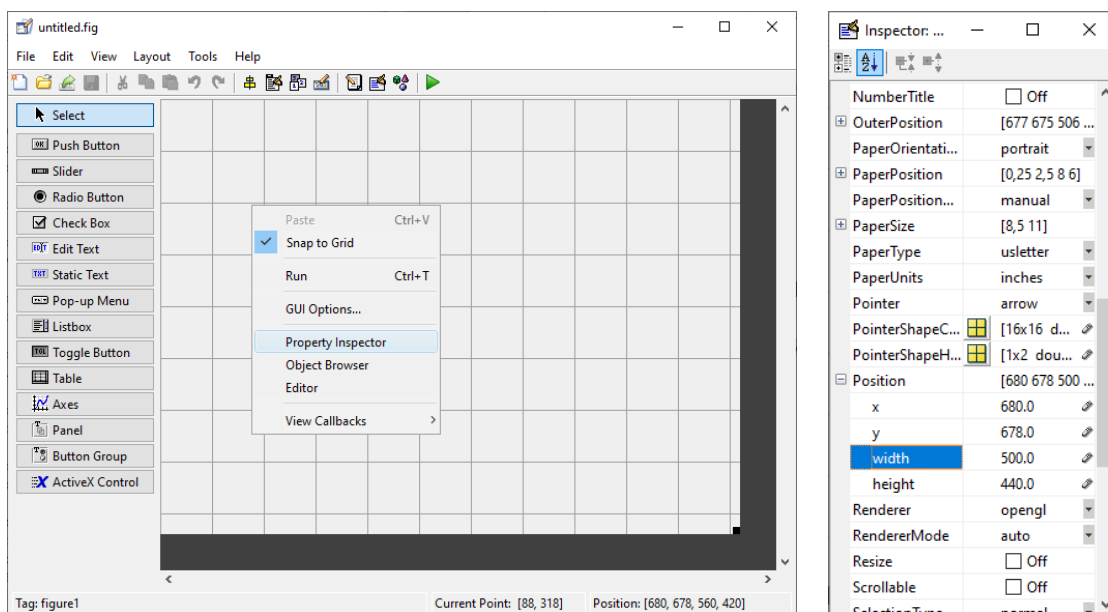
1. Przepisz przykładowy kod z p. III.A. Następnie, skrypt zapisz w pliku o nazwie *guiTrajektoria_A.m* i uruchom.
2. Naciśnij przycisk

B) Tworzenie GUI z użyciem narzędzia GUIDE

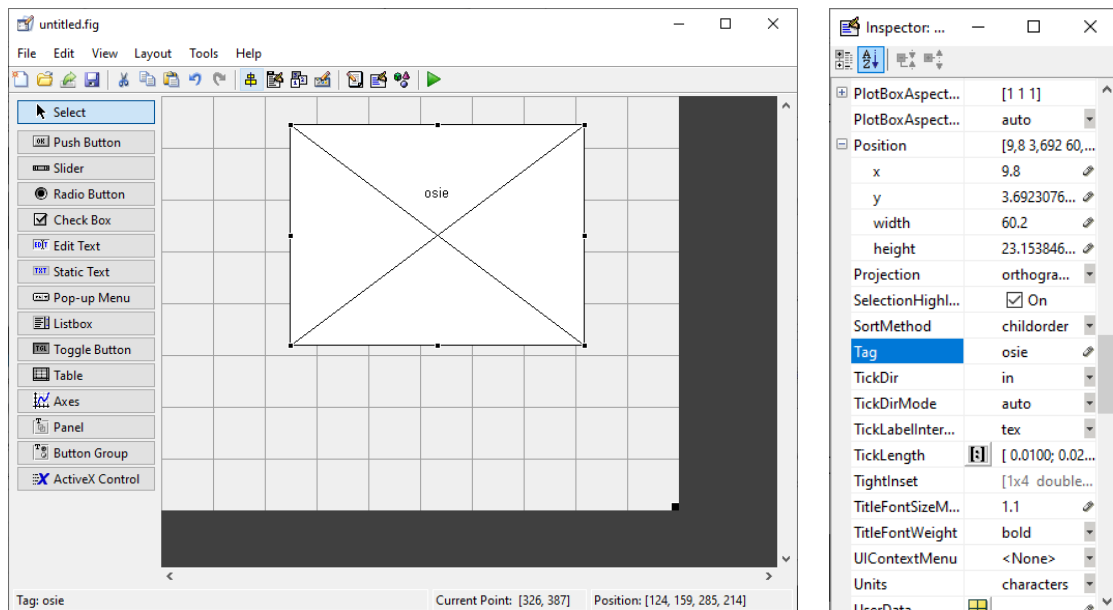
1. Uruchom aplikację **GUIDE** wybierając „puste” GUI (**Blank GUI (Default)**).
2. Za pomocą *Inspektora Właściwości* ustal nazwę oraz rozmiar i pozycję otwieranego okna na zgodną z tymi zdefiniowanymi w przykładowym skrypcie w pp. A. W tym celu najpierw sprawdź i w razie potrzeby zmodyfikuj wartość dla właściwości *Units* obszaru okna graficznego (z kratką linii siatki) na piksele (*pixels*) (Rys. 14). Następnie w zakładce właściwości *Position* zmodyfikuj wartości pól: *x*, *y*, *width* i *height*. (Zatwierdzenie wartości odbywa się klawiszem *ENTER* lub następuje ono po wybraniu innej właściwości) (Rys. 15).
3. Wstaw osie układu współrzędnych (*Axes*).
4. Otwórz okno *Inspektora Właściwości* dla obiektu *Axes*. Zmień wartości właściwości: *Tag* na *Osie*, *Units* na centymetry oraz wartości właściwości na zgodne ze zdefiniowanymi w przykładowym skrypcie w pp. A. (Rys. 16 i 17).
5. Wstaw obiekt *Static text* i zmodyfikuj wartość jego właściwości *String* i *Position* tak aby były zgodne z właściwościami odpowiedniego obiektu w przykładowym skrypcie z pp. A. (Rys. 18).



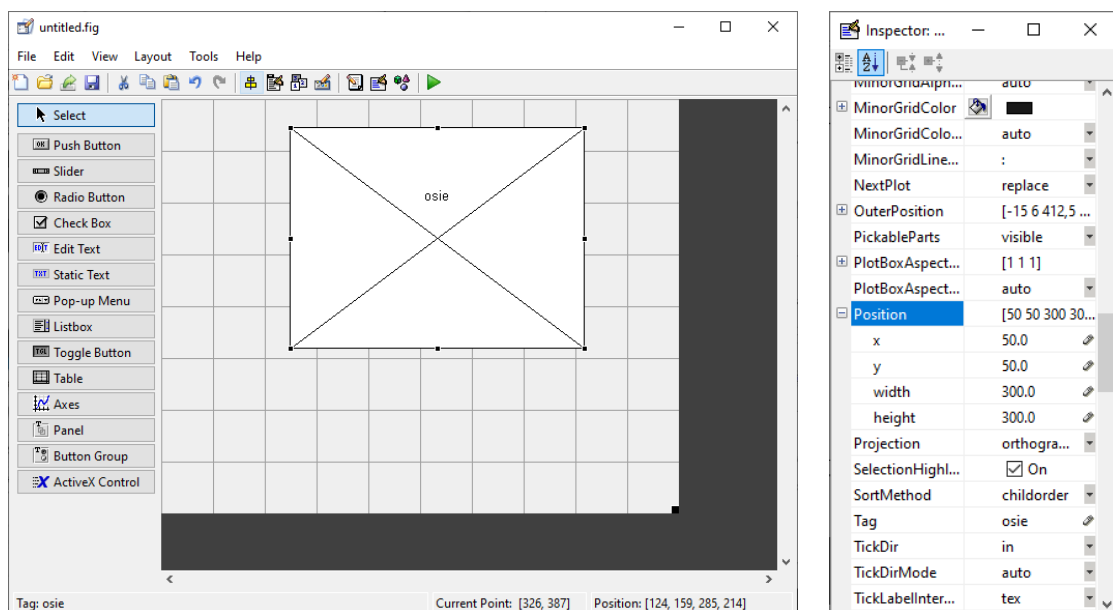
Rysunek 14: Widok i modyfikacja wartości właściwości Units jego głównego okna



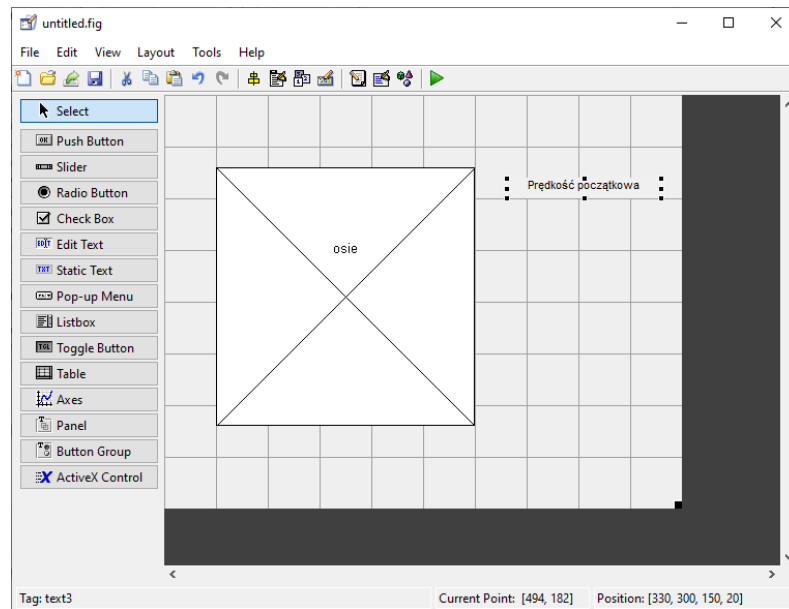
Rysunek 15: Widok i modyfikacja wartości właściwości Position jego głównego okna



Rysunek 16: Widok GUI i modyfikacja wartości właściwości Tag obiektu Axes

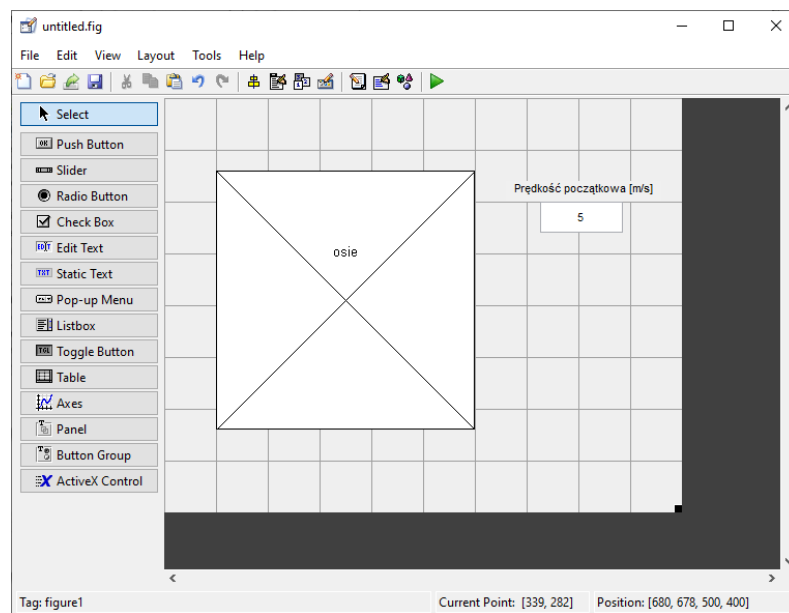


Rysunek 17: Widok GUI i modyfikacja wartości właściwości Position obiektu Axes

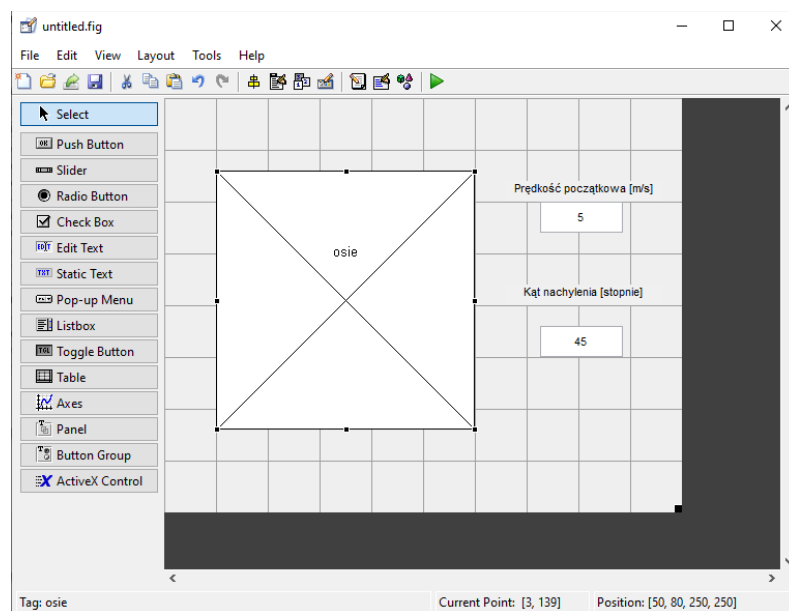


Rysunek 18: Widok GUI z dodanym obiektem Static text (po modyfikacji właściwości)

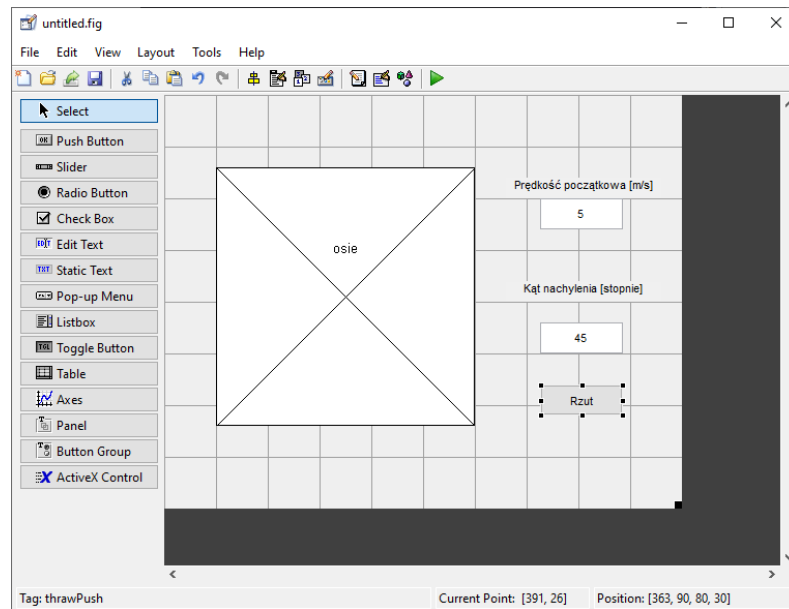
6. Wstaw obiekt **Edit text**, zmień wartości właściwości: Tag na velocityBox oraz zmodyfikuj wartość jego właściwości String i Position tak aby były zgodne z właściwościami odpowiedniego obiektu w przykładowym skrypcie z pp. A. (Rys. 19).
7. Wstaw pozostałe dwa obiekty: **Static text** i **Edit text** i zmodyfikuj odpowiednio ich właściwości (m.in. zmień wartości właściwość Tag obiektu Edit **Text** na **angleBox**) tak, aby były zgodne z właściwościami odpowiadających im elementów w przykładowym skrypcie z pp. A. (Rys. 20).
8. Wstaw obiekt **Push Button**. I zmodyfikuj jego właściwości tak aby były zgodne z właściwościami odpowiadającego mu obiektu w przykładowym skrypcie z pp. A. (m.in. zmień wartości właściwość: Tag na throwPush) (Rys. 21).
9. Uruchom zaprojektowane GUI. Potwierdź chęć kontynuacji (Rys. 22) i zapisz projekt w plikach o nazwie *guiTrajektoria_B* (m-plik zostanie utworzony automatycznie).
10. Naciśnij przycisk **Rzut**. Czy coś się wydarzyło? Dlaczego? Odpowiedzi wpisz we właściwych rubrykach *Sprawozdania*.
11. Zamknij uruchomione GUI.
12. Przejdź do Edytora/Debuggera kodu do m-pliku dla utworzonego interfejsu.
13. Zlokalizuj funkcję `guiTrajektoria_B_OpeningFcn` (funkcja ta jest wykonywana przed wyświetleniem GUI).
14. Na końcu funkcji dodaj instrukcje definiujące zakres wartości dla osi x i y oraz etykiety tych osi identycznie jak to miało miejsce w przykładowym skrypcie z pp. A.(Rys. 23).



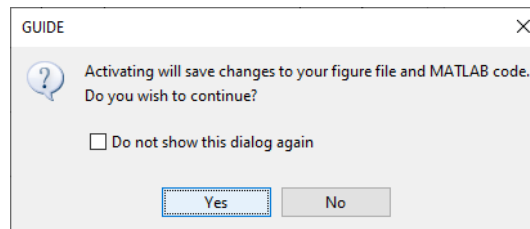
Rysunek 19: Widok GUI z dodanym obiektem Edit text (po modyfikacji właściwości)



Rysunek 20: Widok GUI z 2 obiektami Static Text i 2 obiektami Edit Text (po modyfikacji właściwości)



Rysunek 21: Końcowy widok GUI



Rysunek 22: Końcowy widok GUI

10. Zlokalizuj funkcję `velocityBox_CreateFcn` (funkcja ta jest wykonywana w momencie utworzenia obiektu ***velocityBox***).
11. Na końcu funkcji dodaj instrukcje definiujące zmienną globalną `velocityBox` i przypisującą jej wartość zmiennej `hObject` (pierwszy argument wejściowy funkcji `velocityBox_CreateFcn`) (Rys. 24).
12. Zlokalizuj funkcję `angleBox_Callback` (funkcja ta jest wykonywana w momencie utworzenia obiektu ***angleBox***).
13. Na końcu funkcji dodaj instrukcje definiujące zmienną globalną `angleBox` i przypisującą jej wartość zmiennej `hObject` (pierwszy argument wejściowy funkcji `angleBox_CreateFcn`) (Rys. 25).
14. Zlokalizuj funkcję `throwPush_Callback` (funkcja ta wyznacza i wykreśla trajektorię lotu piłki i jest wykonywana za każdym razem gdy naciśnięty zostanie przycisk ***throwPush***).
15. Na końcu funkcji dodaj instrukcję definiującą dwie zmienne globalne: `velocityBox` i `angleBox` oraz instrukcje służące do wyznaczenia

i wykreślenia trajektorii lotu piłki, analogicznie do instrukcji zawartych w funkcji lokalnej w przykładowym skrypcie z p. A. (Rys. 26).

16. Uruchom *m*-plik.

17. Naciśnij przycisk **Rzut**. Czy coś się wydarzyło? Dlaczego? Odpowiedzi wpisz we właściwych rubrykach *Sprawozdania*.

18. Zamknij aplikację.

```
function guiTrajektor_B_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to guiTrajektor_B (see VARARGIN)

% Choose default command line output for guiTrajektor_B
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guiTrajektor_B wait for user response (see UIRESUME)
% uiwait(handles.figure1);
axis([0,10,0,10])
xlabel('Dystans [m]')
ylabel('Wysokość [m]')
```

Rysunek 23: Funkcja `guiTrajektor_B_OpeningFcn` (po dodaniu wymaganych instrukcji).

```
function velocityBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to velocityBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global velocityBox
velocityBox = hObject;
```

Rysunek 24: Funkcja `velocityBox_CreateFcn` (po dodaniu wymaganych instrukcji).

```
function angleBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angleBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global angleBox
angleBox = hObject;
```

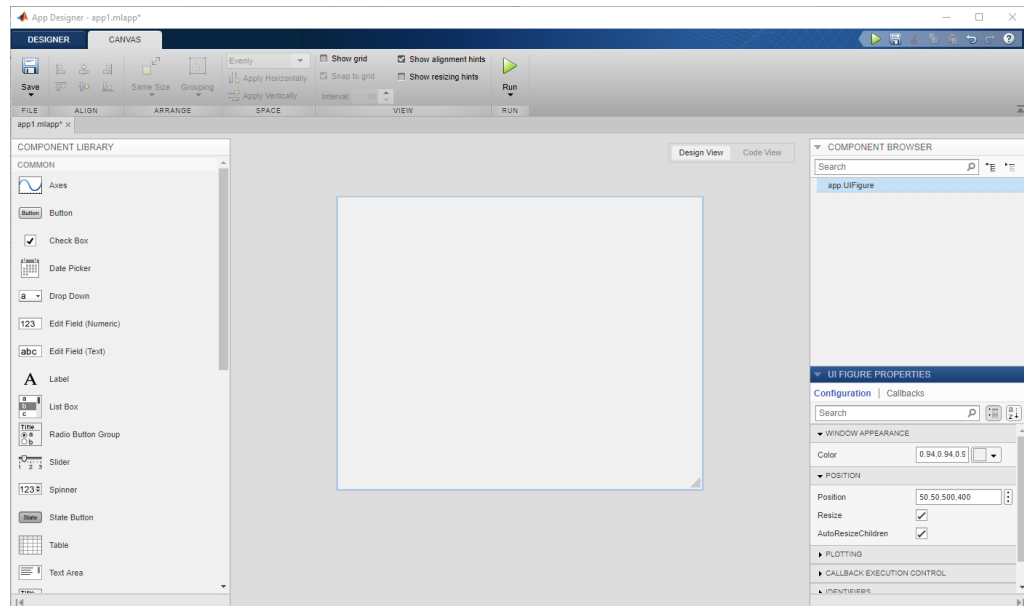
Rysunek 25: Funkcja `angleBox_CreateFcn` (po dodaniu wymaganych instrukcji).

```
function throwPush_Callback(hObject, eventdata, handles)
% hObject      handle to throwPush (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global velocityBox angleBox
g = 9.81;
v0 = str2double(velocityBox.String);
theta = str2double(angleBox.String)*pi/180;
t1 = 2*v0*sin(theta)/g;
t=0:0.01:t1;
x = v0*cos(theta)*t;
y = v0*sin(theta)*t - g*t.^2/2;
hold on
comet(x,y)
```

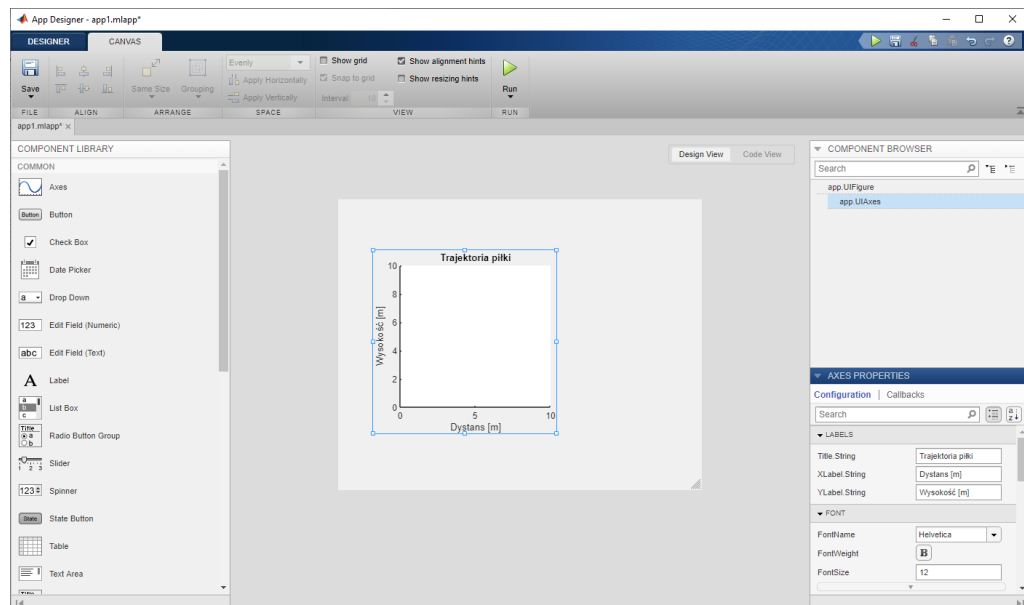
Rysunek 26: Funkcja throwPush_Callback (po dodaniu wymaganych instrukcji).

C) Tworzenie GUI za pomocą App Designer

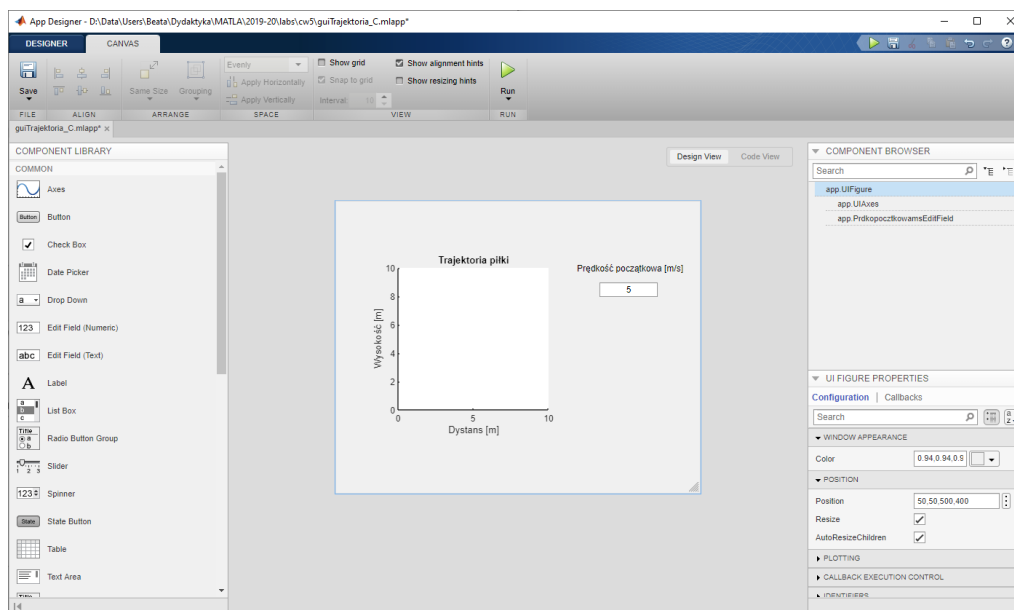
1. Uruchom aplikację *App Designer*.
2. W oknie **UI FIGURE PROPERTIES** ustal rozmiar i pozycję otwieranego okna na zgodne z tymi zdefiniowanymi w przykładowym skrypcie w pp. A i B. W tym celu zmień wartości pól: x, y, width i height właściwości Position. (Zatwierdzenie wartości odbywa się klawiszem *ENTER* lub *TAB* albo następuje po wybraniu myszą innego pola/właściwości). W tym samym oknie ustal tytuł wykresu oraz etykiety i zakresy wartości osi dla osi x i y (właściwości **LABELS** i **RULERS**). (Rys. 27).
3. Wstaw osie układu współrzędnych (obiekt *Axes*). Sprawdź listę komponentów w oknie **COMPONENT BROWSER**.
4. W oknie **UI FIGURE PROPERTIES** zmień wartości właściwości Position na zgodne ze zdefiniowanymi w przykładowym skrypcie w pp. A. (Rys. 28).
5. Wstaw obiekt **Edit Field** i zmodyfikuj wartość jego właściwości Label i Value tak aby były miały one następujące wartości, odpowiednio: Prędkość początkowa [m/s] i 5 oraz zmodyfikuj wartości właściwości Position elementów składowych tego komponentu: pole tekstowe zawierające etykietę oraz okno edycyjne o wartości początkowej równej 5, miały następujące wartości, odpowiednio: [330,300,150,20] i [363,270,80,20] (wartości zgodne z wartościami właściwości Position dwóch obiektów odpowiadających tym elementom składowym obiektu **Edit Field**) (Rys. 29).
6. Analogicznie jak w pp. C.5. wstaw drugi obiekt **Edit Field** i zmodyfikuj wartość jego właściwości tak aby obiekt ten odpowiadał dwóm obiektom z pp. A i C reprezentującym etykietę i pole edycyjne dla wartości kąta, pod jakim została wyrzucona piłka (Rys. 30).
7. Wstaw obiekt **Button** i zmodyfikuj jego właściwości Text i Position tak, aby były zgodne z właściwościami odpowiadającego mu obiektu (**Push Button**) z pp. A i B. (Rys. 31).
8. Zapisz projekt w pliku o nazwie *guiTrajektoria_C* i uruchom aplikację. Naciśnij przycisk **Rzut**. Czy coś się wydarzyło? Dlaczego? Odpowiedzi wpisz we właściwych rubrykach *Sprawozdania*.



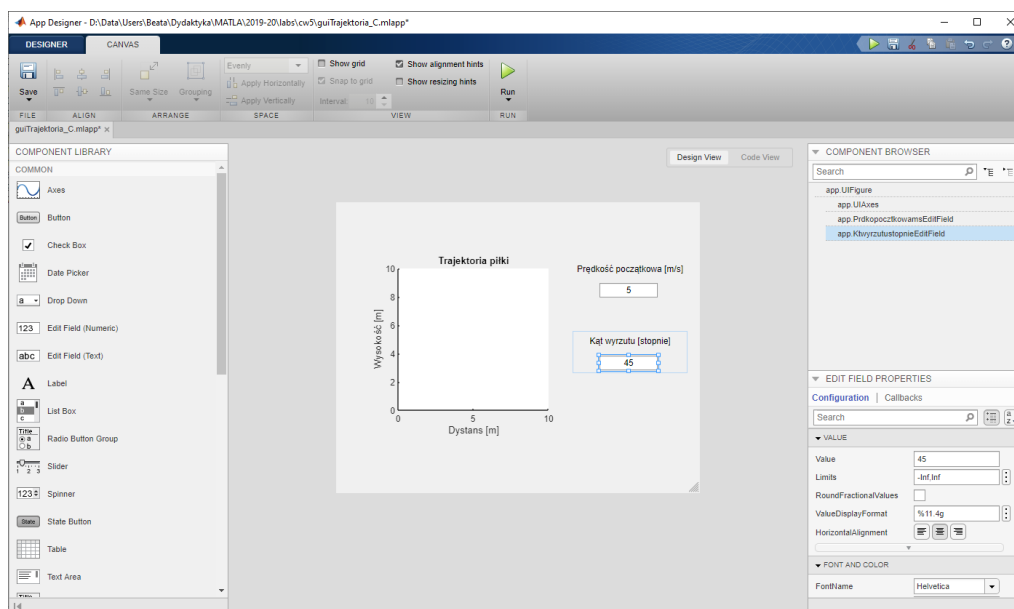
Rysunek 27: Okno aplikacji **App Designer** z modyfikacją właściwości *Position* obiektu *Figure*..



Rysunek 28: Okno aplikacji **App Designer** po dodaniu komponentu *Axes*..

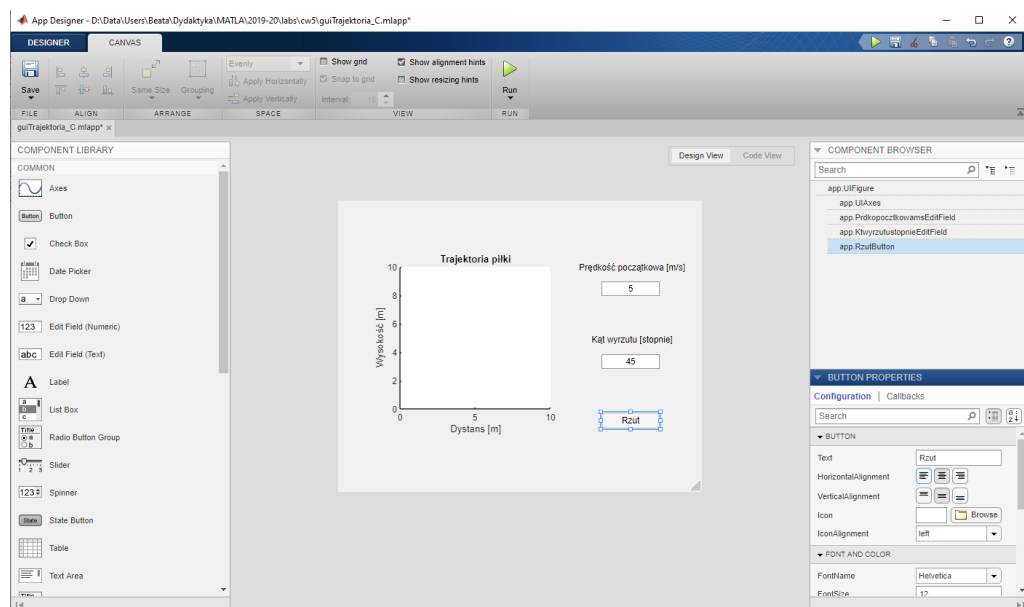


Rysunek 29: Okno aplikacji *App Designer* po dodaniu komponentu *Edit Field*..

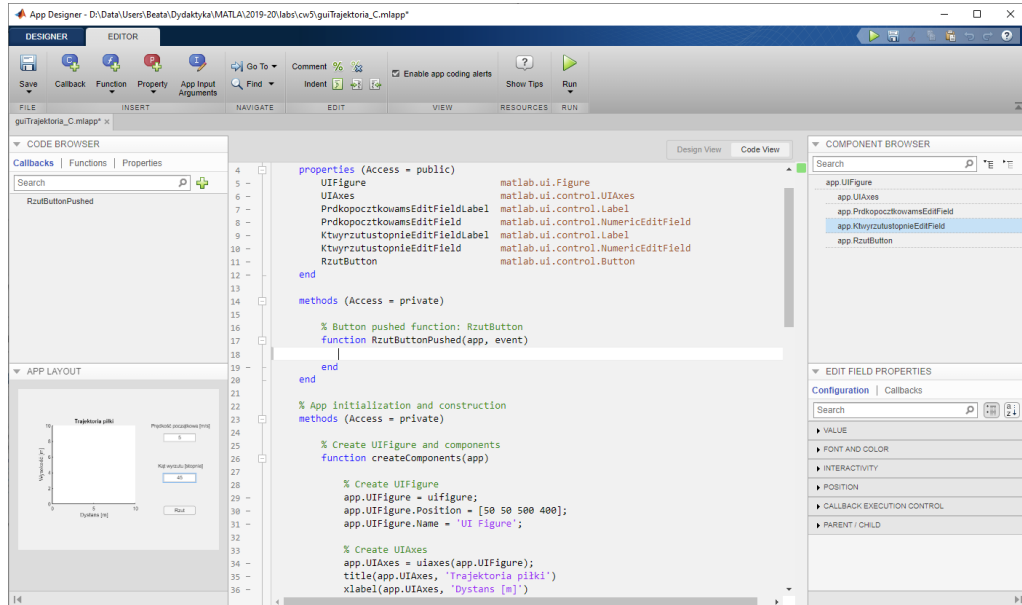


Rysunek 30: Okno aplikacji *App Designer* po dodaniu drugiego komponentu *Edit Field*..

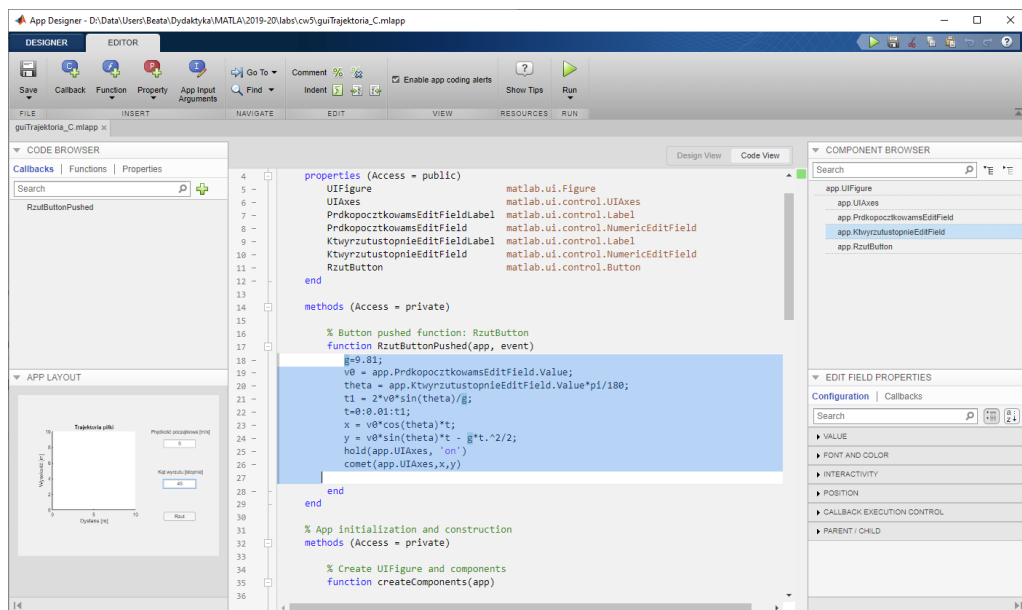
9. Zaznacz przycisk `Rzut` i w oknie **BUTTON PROPERTIES** otwórz zakładkę **Callbacks**.
10. Z rozwijanej listy **ButtonPushFcn** wybierz pozycję **<add ButtonPushFcn callback>**. W efekcie otwarta zostaje zakładka **EDITOR**, w panelu centralnym wyświetlony zostanie kod a kursor będzie automatycznie ustawiony w pierwszej pustej linii ciała edytowanej funkcji, która jest tworzona automatycznie (Rys. 32). Funkcja ta będzie wykonywana zawsze gdy przyciśnięty będzie przycisk `Rzut`.
11. W miejscu w którym znajduje się kursor (bezpośrednio pod linią definicji funkcji `RzutButtonPushed`) wprowadź następujące instrukcje służące do wyznaczenia i wykreślenia trajektorii lotu piłki, analogicznie do instrukcji zawartych w funkcji lokalnej w przykładowym skrypcie z pp. A czy w funkcji wywołania wstecznego dla przycisku `Rzut` w pp. B. Zwróć uwagę na postać argumentów wejściowych w wywołaniu funkcji funkcję `hold` i `comet!!!` (Rys. 33).
12. Uruchom aplikację.
13. Naciśnij przycisk `Rzut`. Czy coś się wydarzyło? Dlaczego? Odpowiedzi wpisz we właściwych rubrykach *Sprawozdania*.
14. Zamknij aplikację.



Rysunek 31: Okno aplikacji App Designer po dodaniu komponentu **Button**.



Rysunek 32: Kod z prototypem funkcji wywołania zwrotnego dla przycisku Rzut .



Rysunek 33: Kod ciała funkcji wywołania zwrotnego dla przycisku Rzut .

D) Testy i modyfikacje interfejsów

1. Przetestuj działanie aplikacji utworzonych w pp. A, B i C dla wartości prędkości początkowej i kąta wyrzutu podanych w Tabeli 2.

Tabela 2. Dane testowe

Prędkość początkowa [m/s]	Kąt wyrzutu [°]
5	45
10	45
10	60
10	80
12	80
14	80
14	100

2. Czy wygląd i zachowanie aplikacji utworzonych w pp. A, B i C jest jednakowe.
3. Zmodyfikuj kod funkcji *guiTrajektoria_A*, tak aby w oknie GUI nie było widoczne menu oraz numer okna graficznego (właściwości `MenuBar` oraz `NumberTitle`).
4. Zmodyfikowane fragmenty kodu wpisz w odpowiedniej rubryce *Sprawozdania*.

Sprawozdanie

Ćwiczenie 5. Elementy programowania obiektowego. Graficzny Interfejs Użytkownika (GUI).

L.p.	Imię i nazwisko	Grupa	Data
Punkt cw./ L. punktów	Realizacja/wynik		Uwagi prowadzącego
A / 1			
B / 1,5			
C / 1,5			
D / 1			