Popup Menu Math Drill Demo
Created by Dr. C. S. Tritt
Last revised: January 24, 2007

Background and Equations

None required.

Operations Description

This GUI program demonstrates the use of popup menu. The user specifies the type of problems desired using a popup menu and clicks the "New Problem" button. The program then generates a random arithmetic problem of this type. Each time the user clicks on the new problem button a new problem is displayed. The user enters their answer to the problem in the edit box. No problem is presented until the New Problem button is pressed. Note that answers are evaluated when the edit box loses focus. If this loss is due to clicking the "New Problem" button, the display will be cleared as soon as the button is released.

A natural extension to this program would be use a timer to limit the time the user has to enter their answer. This could easily be done by having the timer started by the "New Problem" button and the timer call the answer edit box callback when it expires. The timer would have to be stopped in the answer edit box callback if it had not yet expired (the user would have entered their answer prior to the timer expires).

Another extension would be to keep a track of the number of questions asked and the number answered correctly.

User Interface Description

A popup menu (opPopup) that allows the user to specify the problem type ("Addition," "Subtraction," "Multiplication" or "Division).''

A static text box (probText) that displays the generated problem.

An edit box (ansEdit) in which the answer is entered. The ansEdit callback evaluates the entered answer and displays the corresponding result. Note that this callback is also called when the timer expires.

A static text box (resultText) indicating if the last answer was "Correct" or "Incorrect."

Other static text labels as needed (probLabel, opLabel).

A new problem button (newButton) that clears any existing output and generates and displays a new problem.

handles Proprerties

Ans – Answer to problem.

Algorithms

OpeningFcn

Initialize (seed) the uniform random number (rand) generator using system clock.

newButton Callback

```
Clear any preexisting output (ansEdit & resultText controls).
Generate term1 (0 to 9) and term2 (1 to 9).
Get operation (opValue) from opPopup.
Switch opValue
        1: opStr = ‘+’
        2: opStr = ‘-’
        3: opStr = ‘*’
        4: opStr = ‘/’
Build problem text (probStr) and display it in probText control.
Evaluate problem text and store answer in handles structure.
Save modified handles structure.
```

ansEdit Callback

```
Get user answer as a double (userAns).
Echo it.
if (user answer equals stored answer (handles.Ans)
        display “Correct!” message in resultText control.
 else
        display “Incorrect.: message in resultText control.
```

Source Code

```
function varargout = mathDrill(varargin)
% MATHDRILL M-file for mathDrill.fig
%
% See Popup Menu Math Drill Demo docummentation
% (popupMenuMathDemo.doc).
%
% Created by Dr. C. S. Tritt
% Last revised: 1/24/07 (version 1.0)
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help mathDrill

% Last Modified by GUIDE v2.5 24-Jan-2007 07:51:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mathDrill_OpeningFcn, ...
                   'gui_OutputFcn',  @mathDrill_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before mathDrill is made visible.
function mathDrill_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 mathDrill (see VARARGIN)

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

% Initialize (seed) random number generator using system clock.
rand('state', sum(100*clock));

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mathDrill wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = mathDrill_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in opPopup.
function opPopup_Callback(hObject, eventdata, handles)
% hObject    handle to opPopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns opPopup contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from opPopup

% --- Executes during object creation, after setting all properties.
function opPopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to opPopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

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

% Hints: get(hObject,'String') returns contents of ansEdit as text
%        str2double(get(hObject,'String')) returns contents of ansEdit as a
double

% Get answer and echo it.
userAns = str2double(get(hObject,'String'));
echoStr = sprintf('%f', userAns);
set(hObject, 'String', echoStr);

% Determine if correct.

if (userAns == handles.Ans)
   set(handles.resultText, 'String', 'Correct!');
else
   set (handles.resultText, 'String', 'Incorrect.');
```

```
end

% --- Executes during object creation, after setting all properties.
function ansEdit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ansEdit (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
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

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

% Clear existing output.
set(handles.ansEdit, 'String', '');
set(handles.resultText, 'String', '');

% Generate numeric terms.
term1 = floor(10*rand()); % Rounds down to produce values 0 to 9.
term2 = 1 + floor(9*rand()); % Rounds down to produce values 1 to 9.

% Get operation from popup menu.
opValue = get(handles.opPopup, 'Value');
switch opValue
   case 1
      opStr = '+';
   case 2
      opStr = '-';
   case 3
      opStr = '*';
   case 4
      opStr = '/';
end

% Build problem string, display it & evaluate answer.
probStr = sprintf('%.0f %s %.0f' , term1, opStr, term2);
set(handles.probText, 'String', probStr);
handles.Ans = eval(probStr);

% Update handles structure to save Ans
guidata(hObject, handles);
```