Ein interaktiver Editor zum Verschieben von 3D-Objekten in Grafikfenstern und Subplots von Matlab

Tim Lueth, Professor an der TU München

Spechtweg 11, 85386 Eching-Dietersheim

Die hier beschriebene Methode wurde auf dem MAC mit OS 10.15.7 (Catilina) und Matlab Version R2020b Update 6 und der SG-Lib 5.1 getestet

Motivation

Wenn wir in einem Fenster mehrere Oberflächen gezeichnet haben, beispielsweise mit *SGplot* oder *VLFLplot,* dann werden diese Oberflächen in dem Fenster als Grafikobjekte vom Typ "*Patch*" abgelegt. Möchte man diese nun direkt am Bildschirm anordnen, dann benötigt man dafür einen grafischen Editor. Das folgende Beispiel zeichnet drei Quader zum Verschieben.

```
SGbox([30 20 10;30 40 20;10 10 10],'','randxy',true);
```

Aufbau einer Editorfunktion

Die folgende Vorgehensweise verdeutlich wie die Callback-Funktionen von Matlab dazu verwendet werden können einen interaktiven Editor aufzubauen. Diese Schritte "installieren" die entsprechenden Callback-Funktionen, so dass die Editorfunktion automatisch aufgerufen wird. Eine solche Editor-Funktion könnte beispielsweise *SGfigureInteractiveMove* genannt werden.

Zuerst werden alle Patch-Objekte des Fensters "deselected', damit man nicht verwirrt wird.

```
%% Deselect all patches in the gcf current figure
shg; set(findobj(gcf,'Type','Patch'),'Selected','off')
```

Dann wird eine Callback-Funktion global für das gesamte Fenster eingeführt, die bei einem Mouseclick aufgerufen wird. Callback-Funktionen für *Grafik-Fenster* lassen sich nur installieren und ausführen, wenn der Darstellungsmodus *rotate3d* auf off gestellt ist. Später kann man von Hand dann die Rotation einschalten und die Maus zum Rotieren verwenden oder die Rotation ausschalten und die Maus für den Editor verwenden.

```
%% CREATE A GLOBAL CALLBACK FOR A MOUSE CLICK
rotate3d('off');                                    % required before setting WindowsButtonDownFcn
set(gcf,'WindowButtonDownFcn',@gcfWindowButtonDownFcn);   % Installing the gcf WindowsButtonDownFcn
```

Die installierte WindowButtonDownFcn-Callback-Funktion wertet den Mausklick aus. Sie ermittelt die Position des Klicks, ob es ein Doppelklick war, welches Objekt angeklickt wurde.

```
function gcfWindowButtonDownFcn(src,evt);
  pause(0.1)                    % required not to be too fast
  pp=get(gca,'CurrentPoint')    % returns current points crossing the clipping box
  p=pp(1,1:2)                   % returns the 2D version of the first entry point
  t=src.SelectionType           % returns the mous click style 'double', 'normal
  if isempty(gco); return; end;  % rotate3d is "on" ==> no further action
  gco                           % axis or object only if rotate3d is 'off'
```

Wenn *rotate3d* eingeschaltet ist, wird immer ein leeres gco (Graphics Object) zurückgeliefert. Eine weitere Verarbeitung macht dann keinen Sinn. Sofern jedoch ein Patch angeklickt wurde, werden jetzt alle bisherigen selektierten Patches *deselektiert* und dann das ausgewählte Patch auf *selektiert* gesetzt. Dann werden die Daten des Patchs ausgelesen und in der Callback-Funktion persistent gespeichert. Es wird jetzt temporär eine Callback-Funktion für die Mausbewegung installiert, die durch eine zweite Callback-Funktion beim Loslassen der Maus wieder deinstalliert wird. In diesen zwei Funktionen wird dann das interaktive Verschieben durch einfache Vektoraddition realisiert.

```
    if ~isempty(gco)
        [~,~,~,~,fi,ax]=select3d(gco)  % returns facet index and patch of gco
        %% A facet of a patch has been selected
        if ~isempty(fi) && isequal(get(ax,'Type'),'patch')   % A patch has been selected
           set(findobj(gca,'Type','Patch'),'Selected','off') % Deselect all patches in gca
           set(ax,'Selected','on')                           % Select the new gco
           set(gcf,'WindowButtonMotionFcn',@gcfWindowButtonMotionFcn);
           set(gcf,'WindowButtonUpFcn',@gcfWindowButtonUpFcn);
           VL=ax.Vertices; FL=ax.Faces;
        end
    end

      %% LOCAL gcfWindowButtonMotionFcn FOR PATCHES - CALCS REALATIVE MOUSE MOVEMENT
      function gcfWindowButtonMotionFcn(~,~)
          np=get(gca,'CurrentPoint');          % returns current points crossing the clipping box
          dp=np(1,1:3)-pp(1,1:3);              % returns the differenz vector
          dp=dp.*(abs(dp)==max(abs(dp)));      % select only the maximum dimension
          ax.Vertices=VL+dp;                   % Change the position of the patch
          axis manual;                         % Avoid automatic axis scaling
      end
      %% LOCAL gcfWindowButtonUpFcn FOR PATCHES - STOPS MOTION PROCEDURES
      function gcfWindowButtonUpFcn(~,~)
          pause(0.1)                           % required not to be too fast
          set(gcf,'WindowButtonMotionFcn',[]); % remove the callback procedure for movement
%         ax.Vertices=VL;                      % reset the position of the gco
          axis padded; axis tight; drawnow;    % Switch on automatic scaling to find objects moved out
         set(ax,'Selected','off')
      end
```

Es ist leicht vorstellbar wie man durch Auswertung der linken ('normal'), mittleren ('alt'), rechten ('extend') Maustaste oder einem Doppelklick ('open') unterschiedliche Verhalten wie das Verschieben, Drehen oder auch das Verändern der Blickrichtung ermöglichen könnte

```
function SGfigureInteractiveMove

%% Show the figure in the foreground
shg
% NN=SGsurfaces(SGofgca); cla; SGplotsurfaces(NN,'',0);


%% Deselect all patches in the gcf current figure
set(findobj(gcf,'Type','Patch'),'Selected','off')

%% CREATE A GLOBAL CALLBACK FOR A MOUSE CLICK
rotate3d('off');                                        % required before setting WindowsButtonDownFcn
set(gcf,'WindowButtonDownFcn',@gcfWindowButtonDownFcn);   % Installing the gcf WindowsButtonDownFcn
end


%% Mouse button down function  % DOES WORK ONLY IF ROTATE3D IS OFF
function gcfWindowButtonDownFcn(src,evt);
  pause(0.1)                      % required not to be too fast
  pp=get(gca,'CurrentPoint');     % returns current points crossing the clipping box
  p=pp(1,1:2);                    % returns the 2D version of the first entry point
  t=src.SelectionType;            % returns the mous click style 'double', 'normal
  if isempty(gco); return; end;   % rotate3d is "on" ==> no further action
  gco;                             % axis or object only if rotate3d is 'off'

%% NOW PROCESS THE PATCH MOVE BY TWO ADDITIONAL CALLBACK-FUNCTIONs


  %% If rotate3d is off, gco is either an axis or a selected gco
  if ~isempty(gco)
      [~,~,~,~,fi,ax]=select3d(gco)  % returns facet index and patch of gco
      %% A facet of a patch has been selected
      if ~isempty(fi) && isequal(get(ax,'Type'),'patch')   % A patch has been selected
         set(findobj(gca,'Type','Patch'),'Selected','off') % Deselect all patches in gca
         set(ax,'Selected','on')                           % Select the new gco
         set(gcf,'WindowButtonMotionFcn',@gcfWindowButtonMotionFcn);
         set(gcf,'WindowButtonUpFcn',@gcfWindowButtonUpFcn);
         VL=ax.Vertices; FL=ax.Faces;
      end
  end

    %% LOCAL gcfWindowButtonMotionFcn FOR PATCHES - CALCS REALATIVE MOUSE MOVEMENT
    function gcfWindowButtonMotionFcn(~,~)
        np=get(gca,'CurrentPoint');          % returns current points crossing the clipping box
        dp=np(1,1:3)-pp(1,1:3);              % returns the differenz vector
        dp=dp.*(abs(dp)==max(abs(dp)));      % select only the maximum dimension
        ax.Vertices=VL+dp;                   % Change the position of the patch
        axis manual;                         % Avoid automatic axis scaling
    end
    %% LOCAL gcfWindowButtonUpFcn FOR PATCHES - STOPS MOTION PROCEDURES
    function gcfWindowButtonUpFcn(~,~)
        pause(0.1)                           % required not to be too fast
        set(gcf,'WindowButtonMotionFcn',[]); % remove the callback procedure for movement
%         ax.Vertices=VL;                    % reset the position of the gco
        axis padded; axis tight; drawnow;    % Switch on automatic scaling to find objects moved out
       set(ax,'Selected','off')
    end
end
```

Abb. 1: Listing eines kleinen grafischen Editors zum Verschieben von Objekten