Documentation for the Initialize View Window function for BerenGUI:

I attached a function "initialize_view_window". Put this function intothe main beren bin folder and call it in

Installation:

handles = initialize_view_window(handles) is to be placed in berengui.m in

function berengui_OpeningFcn(hObject, eventdata, handles, varargin).

Put this line of code towards the end of *OpeningFcn()* just before the line

guidata(hObject, handles); This will fix the problem on any screen of reasonable size and resolution.

Usage:

> MATLAB defines the Position property of an object to be [X, Y, width, height] where [X, Y] are measured from the bottom left-hand corner of the object's parent (the figure or panel which it exists inside) to the bottom left-hand corner of the object. This means an increase in X shifts the object to the right and an increase in Y shifts the object up. In this code, many times only a 2-value vector describes an object's position. In this case, the code refers only to the [X, Y] portion of the Position property.
>
> It is also valuable to note that MATLAB always reads the Position property in terms of the Units property. Units are default set to 'characters' which have ambiguous functionality. In this code, all objects have their Units property forced to 'pixels', so specify Position appropriately.

To change the size or position of any of the large panel area positions:

> Navigate to the code subsection labeled "Setting sizes and positions for panels and figure". It should be very near to the top of the function. You should in general not change the figure size and placement definitions. The pixel position of the lower left hand corner of each big panel relative to its parent object (either the figure or another panel if the panel in question is nested) is defined by the first two (of four) vector values assigned to the appropriately named ___PanPos variable. Changing these values changes the position of the panel. In general, do not change the sizes of the panels.

Example 1: Swapping the Mod A and Mod B panel areas

1) Navigate to the "Setting sizes and positions for panels and figures" subsection and find these lines of code:

```
% Mod B right panel
rightLowPanPos = [5, 5, rightPanPos(3)-10, (rightPanPos(4)-19)/2];

% Mod A right panel
rightHighPanPos = [5, rightLowPanPos(4)+14, rightPanPos(3)-10,
(rightPanPos(4)-19)/2];
```

2) Modify these so that every instance of rightLowPanPos is replaced with rightHighPanPos and vice versa:

```
% Mod A right panel
rightHighPanPos = [5, 5, rightPanPos(3)-10, (rightPanPos(4)-19)/2];

% Mod B right panel
rightLowPanPos = [5, rightHighPanPos(4)+14, rightPanPos(3)-10,
(rightPanPos(4)-19)/2];
```

3) Note that it might be confusing to switch these as rightHighPanPos now refers to the lower right-hand panel. It may be better to perform a more thorough code modification for such a fundamental change.

To change the size or any other non-position property of any control object:

> This can still be done in the .fig window. The only thing that the initialize view window function overrides is the position of every element, and the sizes of the main figure, parent panels, and axes displays.

Example 2: Changing the size of the Event Gram Play button

1) Type 'guide' into the MATLAB command prompt and open berengui.fig

2) Find the main Play button in the Event Gram section, click on it, and drag it to the appropriate size. This new size will be preserved in initialize_view_window.m.

3) Note that if you accidentally change the position of the main Play button while resizing it, this change in position in the .fig file will have no effect on the position assigned to the button in initialize_view_window.m.

To change the positioning of any control object:

This is what the code really does. There are 4 typical cases to change an object's position:

1. **Change the properties of the object's group:** Every object is placed in a group of similarly positioned objects. If you want to change the position or spacing of a control group, navigate to the code subsection labeled "Setting positions for groups of buttons, controls, and axes". Then, find the control group you are looking for--these are labeled in the comments and by the variable name which defines their lower left hand corner anchor position. Finally, make the change to either the position variable to change the position of the group of controls or to the Hspace or Vspace variables to change the horizontal or vertical spacing of the group respectively.

2. **Nudge an object within a group:** If you want to slightly shift the position of an object inside a group or place it in a different slot in the group, navigate to the code subsection labeled " Placing every individual control object". Every object's [X, Y] position vector is set here. The objects are organized and labeled by their groups to be more easily found. Every position defining line of code is formatted:

   ```
   OBJECTLABEL = GROUPPOS + [M*Hspace, N*Vspace];
   ```

   To change the object's slot in the group, adjust the M and N values accordingly. To nudge the object, insert an additional "nudge vector" as follows:

   ```
   OBJECTLABEL = GROUPPOS + [M*Hspace, N*Vspace] + [Xnudge, Ynudge];
   ```

3. **Completely reposition an object:** If you want to do a hard override of the position of a certain object, navigate to the code subsection labeled " Assigning the positions to the existing control objects". Find the Tag property associated with the object you want to modify by clicking on it in the .fig file and opening the Property Inspector. Scroll to the handles.Tag line in the code subsection (again, these are organized by group). The line should look like this:

   ```
   handles.TAG = set_new_position(handles.TAG, P, POSITIONLABEL);
   ```

   Where POSITIONLABEL is the same label as OBJECTLABEL in step 2, and P is the handle of the parent object. To override the position described by POSITIONLABEL, replace POSITIONLABEL with an [X, Y] vector of the pixel position you want the object to have.

4. **Reposition an axes object:** Axes objects are a bit different than normal objects, because their sizes are also controlled by the initialize view window function. To do this, in most cases, you should only change the bottom left hand corner position of the axes object or group. This can be done by navigating to the bottom of the "Setting sizes and positions for panels and figure" subsection. Controlling the anchor position of the 4 Event Gram axes is done with the first two values of the line:

   ```
   leftPanGraphicsPos = [50, 80, leftInsidePanPos(3)-100, leftInsidePanPos(4)-160];
   ```

   Controlling the position of the Mod A axes is done with the first two values of the line:

   ```
   rightPanGraphicsPos = [40, 60, rightHighPanPos(3)-80, rightHighPanPos(4)-140];
   ```

   Controlling the position of the Mod B axes is done with the values of the line:

   ```
   rightLowPanGraphicsPos(1:2) = [40, 90];
   ```

Example 3: Changing the vertical space between the Event Gram top menu and the top of the panel

1) Navigate to the "Setting positions for groups of buttons, controls, and axes" subsection and find the following lines of code:

```
% Exp, Talker, Cons, Vowels: Menu to specify the speech file
leftPanTopMenuPos = [100, leftInsidePanPos(4)-60];
    leftPanTopMenuHspace = 100; %horizontal spacing between members
    leftPanTopMenuVspace = 25; %vertical spacing between rows of members
```

2) The first assignment assigns [X, Y] to the lower left-hand corner of the group. To lower this anchor point and increase the space by 10 pixels, we would change this first assignment line to the following:

```
% Exp, Talker, Cons, Vowels: Menu to specify the speech file
leftPanTopMenuPos = [100, leftInsidePanPos(4)-70];
```

3) Note that this shift might encroach on the space of other groups, so additional work may need to be done to even things out and make them look nice.

Example 4: Moving the Open button to the top row of the top menu and giving it a bit more space

1) Navigate to the " Placing every individual control object" subsection and find the following lines of code:

```
%left panel top menu, 'EVENT GRAM' sound selection:
lptm_lbl1 = leftPanTopMenuPos + [leftPanTopMenuHspace*0,
leftPanTopMenuVspace*1];
lptm_lbl2 = leftPanTopMenuPos + [leftPanTopMenuHspace*1,
leftPanTopMenuVspace*1];
lptm_lbl3 = leftPanTopMenuPos + [leftPanTopMenuHspace*2,
leftPanTopMenuVspace*1];
lptm_lbl4 = leftPanTopMenuPos + [leftPanTopMenuHspace*3,
leftPanTopMenuVspace*1];
lptm_pop1 = leftPanTopMenuPos + [leftPanTopMenuHspace*0, 0];
lptm_pop2 = leftPanTopMenuPos + [leftPanTopMenuHspace*1, 0];
lptm_pop3 = leftPanTopMenuPos + [leftPanTopMenuHspace*2, 0];
lptm_pop4 = leftPanTopMenuPos + [leftPanTopMenuHspace*3, 0];
lptm_btn1 = leftPanTopMenuPos + [leftPanTopMenuHspace*4, 0];
```

2) Because the order of labels always reads from top-left to bottom-right like reading a book in English, and because we expect the Open button to be labeled "btn", we can safely assume that lptm_btn1 is the label for the Open button. To shift it to the top row, we need to add a Vspace argument as so:

```
lptm_btn1 = leftPanTopMenuPos + [leftPanTopMenuHspace*4,
leftPanTopMenuVspace*1];
```

3) To give the Open button a little more horizontal spacing than the other controls to set it apart, we can nudge it to the right by 20 pixels like this:

```
lptm_btn1 = leftPanTopMenuPos + [leftPanTopMenuHspace*4,
leftPanTopMenuVspace*1] + [20, 0];
```

4) Note that now it would be good practice to move the above line of code right above the lptm_pop1 label so that the top-left to bottom-right standard is maintained.

Example 5: Completely overriding the position of the main Event Gram Play button

1) Navigate to the "Assigning the positions to the existing control objects" subsection and find the following lines of code:

```
%left panel graphics area, 'EVENT GRAM':
handles.plotAIgram = set_new_full_position(handles.plotAIgram, P, [lpga_axes1,
leftPanGraphicsAxesSize]);
handles.plotEventGram = set_new_full_position(handles.plotEventGram, P,
[lpga_axes2, leftPanGraphicsAxesSize]);
handles.butPlay = set_new_position(handles.butPlay, P, lpga_btn1);
handles.plotCC = set_new_full_position(handles.plotCC, P, [lpga_axes3,
leftPanGraphicsAxesSize]);
handles.plotPI = set_new_full_position(handles.plotPI, P, [lpga_axes4,
leftPanGraphicsAxesSize]);
```

2) These labels are ordered from top-left to bottom-right like reading an English book, so we expect handles.butPlay to be the Play button. The tag also seems to fit, and we can check this assumption by opening the .fig file, right clicking on the Play button, checking the Property Inspector, and looking at the object's Tag property. We again find that butPlay is correct.

3) To completely reposition this object, we can change the third argument to set_new_position to be a 2-value vector of form [X, Y] defining the lower-left hand corner pixel values of the Play button relative to the specified parent panel P. P can be replaced by a handle to any figure or panel if you want to override even the parent. To arbitrarily set the Play button's position to Event Gram [100, 100], change the relevant line of code to the following:

```
handles.butPlay = set_new_position(handles.butPlay, P, [100, 100]);
```

4) Note that a set_new_full_position call's 3rd argument would need a 4-value vector of form [X, Y, wid, ht]. Also, it may be good coding practice to change the code position of the Play button's position allocation line so that it does not appear to be in a control group for which it is no longer a member.

Example 6: Carving out more space between the Event Gram axes for the Play button

1) Navigate to the "Setting positions for groups of buttons, controls, and axes" subsection and find the following lines of code:

```
% 'EVENT GRAM' graphics area with 4 axes and a play button
%   This code sizes the graphics area so that every axes has the same size
%   with the appropriate aspect ratio and a constant pixel width between
%   axes.
leftPanGraphicsPos = [50, 80, leftInsidePanPos(3)-100, leftInsidePanPos(4)-
160];
if(leftPanGraphicsPos(3) < leftPanGraphicsPos(4)*1.3)
    leftPanGraphicsPos(2) = leftPanGraphicsPos(2) + (leftPanGraphicsPos(4) -
(leftPanGraphicsPos(3)/1.3))/2;
    leftPanGraphicsPos(4) = leftPanGraphicsPos(3)/1.3;
elseif(leftPanGraphicsPos(3) > leftPanGraphicsPos(4)*1.3)
```

```
    leftPanGraphicsPos(1) = leftPanGraphicsPos(1) + (leftPanGraphicsPos(3) -
(leftPanGraphicsPos(4)*1.3))/2;
    leftPanGraphicsPos(3) = leftPanGraphicsPos(4)*1.3;
end
leftPanGraphicsHspace = 50;
leftPanGraphicsVspace = 50; leftPanGraphicsPlaySpace = 15;
leftPanGraphicsAxesSize = [(leftPanGraphicsPos(3)-leftPanGraphicsHspace)/2,
...
                            (leftPanGraphicsPos(4)-leftPanGraphicsVspace)/2];
```

2) This looks like a lot, but you should always be able to ignore the hefty if statement. To carve out space in the middle, let's take some space from the outside edges and some more from the margins between axes. To take from the outside edges, we should shift the anchor point of the graphics area down by 10 pixels and extend its total height up by 10 pixels. Modify the first line to do this:

```
leftPanGraphicsPos = [50, 70, leftInsidePanPos(3)-100, leftInsidePanPos(4)-
150];
```

3) This gives us 20 additional pixels to work with in the middle. If we need another 10 pixels for the play button, we can increase the vertical space between the two rows of axes by increasing leftPanGraphicsVspace:

```
leftPanGraphicsVspace = 60; leftPanGraphicsPlaySpace = 15;
```

4) This should do it. Note that leftPanGraphicsPlaySpace is unused in the current revision. It was left in on accident. Also note that if the vertical and horizontal spacing are not equal, the aspect ratio of the axes may change slightly across screens and be different from the aspect ratio of Mod A and Mod B axes. Special care was taken to ensure equal aspect ratios for all axes on all platforms when the horizontal and vertical spacing are equivalent. This might cause you to want to modify leftPanGraphicsHspace whenever you modify leftPanGraphicsVspace.