Implement a New Figure Panel

The BRAPH 2 Developers

August 14, 2024

This is the developer tutorial for implementing a new figure panel. In this tutorial, you will learn how to create the generator file *.gen.m for a new figure panel, which can then be compiled by braph2genesis. All figure panels are (direct or indirect) extensions of the element PanelFig. You will use the figure panels BrainSurfacePF and BrainAtlasPF as an examples.

Contents

Implementation of figure panel (BrainSurfacePF)

To understand the general concepts of a figure panel, you will start by implementing in detail the figure panel `BrainSurfacePF`, which is a direct extension of the element `PanelFig`.

Code 1: BrainSurfacePF element header. The `header` section of the generator code in `_BrainSurfacePF.gen.m` provides the general information about the `BrainSurfacePF` element.

```
1  %% ¡header!
2  BrainSurfacePF < PanelFig (pf, panel figure brain surface) is a plot of a
       brain surfce. ①
3
4  %%% ¡description!
5  BrainSurfacePF manages the plot of the brain surface choosen by the user.
6  A collection of brain surfaces in NV format can be found in the folder
7  ./braph2/brainsurfs/.
8  This class provides the common methods needed to manage the plot of
9  the surface. In particular, the user can change lighting, material,
10 camlight, shadning, colormap, facecolor, brain color, face color,
11 edge color, and background color.
12
13 %%% ¡seealso!
14 BrainSurface
15
16 %%% ¡build!
17 1
```

① The element `BrainSurfacePF` is defined as a subclass of `PabelFig`. The moniker will be pf.

Code 2: BrainSurfacePF element constants. The `constants` section of the generator code in `_BrainSurfacePF.gen.m` introductes some element constants. These will simplify the management of the visualization of the brain surface.

```
1  %% ¡constants!
2
3  % fixed 3d view
4  VIEW_3D = 1 % 3D view numeric code
5  VIEW_3D_CMD = '3D' % 3D view name
6  VIEW_3D_AZEL = [-37.5 30] % 3D view azimutal and polar angles
7
8  % sagittal left view
9  VIEW_SL = 2 % sagittal left view numeric code
10 VIEW_SL_CMD = 'Sagittal left' % sagittal left view name
11 VIEW_SL_AZEL = [-90 0] % sagittal left view azimutal and polar angles
12
13 % sagittal right view
14 VIEW_SR = 3 % sagittal right view numeric code
15 VIEW_SR_CMD = 'Sagittal right' % sagittal right view name
16 VIEW_SR_AZEL = [90 0] % sagittal right view azimutal and polar angles
17
18 % axial dorsal view
19 VIEW_AD = 4 % axial dorsal view numeric code
20 VIEW_AD_CMD = 'Axial dorsal' % axial dorsal view name
21 VIEW_AD_AZEL = [0 90] % axial dorsal view azimutal and polar angles
22
23 % axial ventral view
24 VIEW_AV = 5 % axial ventral view numeric code
```

```
25  VIEW_AV_CMD = 'Axial ventral' % axial ventral view name
26  VIEW_AV_AZEL = [0 -90] % axial ventral view azimutal and polar angles
27
28  % coronal anterior view
29  VIEW_CA = 6 % coronal anterior view numeric code
30  VIEW_CA_CMD = 'Coronal anterior' % coronal anterior view name
31  VIEW_CA_AZEL = [180 0] % coronal anterior view azimutal and polar angles
32
33  % coronal posterior view
34  VIEW_CP = 7 % coronal posterior view numeric code
35  VIEW_CP_CMD = 'Coronal posterior' % coronal posterior view name
36  VIEW_CP_AZEL = [0 0] % coronal posterior view azimutal and polar angles
37
38  VIEW_CMD = { ... % vector of view names
39      BrainSurfacePF.VIEW_3D_CMD ...
40      BrainSurfacePF.VIEW_SL_CMD ...
41      BrainSurfacePF.VIEW_SR_CMD ...
42      BrainSurfacePF.VIEW_AD_CMD ...
43      BrainSurfacePF.VIEW_AV_CMD ...
44      BrainSurfacePF.VIEW_CA_CMD ...
45      BrainSurfacePF.VIEW_CP_CMD ...
46      }
47
48  VIEW_AZEL = { ... % vector of view azimutal and polar angle
49      BrainSurfacePF.VIEW_3D_AZEL ...
50      BrainSurfacePF.VIEW_SL_AZEL ...
51      BrainSurfacePF.VIEW_SR_AZEL ...
52      BrainSurfacePF.VIEW_AD_AZEL ...
53      BrainSurfacePF.VIEW_AV_AZEL ...
54      BrainSurfacePF.VIEW_CA_AZEL ...
55      BrainSurfacePF.VIEW_CP_AZEL ...
56      }
```

Code 3: **BrainSurfacePF element new props.** The props section of
the generator code in _BrainSurfacePF.gen.m defines the necessary
user interface objects and their callbacks.

```
1   %% iprops!
2
3   %%% iprop!
4   H_AXES (evanescent, handle) is the handle for the axes. (1)
5   %%%% icalculate!
6   h_axes = uiaxes( ...
7       'Parent', pf.memorize('H'), ... (2)
8       'Tag', 'H_AXES', ...
9       'Units', 'normalized', ...
10      'OuterPosition', [0 0 1 1] ...
11      );
12  h_axes.Toolbar.Visible = 'off';
13  h_axes.Interactions = [];
14  value = h_axes;
15
16  %%% iprop!
17  VIEW (figure, rvector) sets the desired view as the line-of-sight azimuth
        and elevation angles. (3)
18  %%%% icheck_prop!
19  check = length(value) == 2;
20  %%%% idefault!
21  BrainSurfacePF.VIEW_SL_AZEL
```

(1) defines the evanescent handle of the axes where the brain surface will be plotted. It also defines its general properties.

(2) ensures that the parent panel is memorized.

(3) determines the view of the brain surface.

```
22  %%%% ¡postset! ④
23  if pf.get('DRAWN')
24      view(pf.get('H_AXES'), pf.get('VIEW'))
25
26      % reset the ambient lighting
27      pf.memorize('ST_AMBIENT').set('PANEL', pf, 'PROP', pf.H_AXES).get('SETUP
        ')
28  end
29  %%%% ¡gui!
30  pr = PanelPropRVectorView('EL', pf, 'PROP', BrainSurfacePF.VIEW, varargin
        {:});
31
32  %%% ¡prop!
33  ST_AXIS (figure, item) determines the axis settings. ⑤
34  %%%% ¡settings!
35  'SettingsAxis'
36  %%%% ¡default!
37  SettingsAxis('GRID', false, 'AXIS', false) ⑥
38  %%%% ¡gui! ⑦
39  pr = SettingsAxisPP('EL', pf, 'PROP', BrainSurfacePF.ST_AXIS, varargin{:});
40
41  %%% ¡prop!
42  SURFFILE (figure, option) is the name of the file of the brain surface to be
        plotted. ⑧
43  %%%% ¡settings!
44  {dir([fileparts(which('braph2')) filesep() 'brainsurfs' filesep() '*.nv']).
        name}
45  %%%% ¡default!
46  'human_ICBM152.nv'
47  %%%% ¡postset! ⑨
48  bs = ImporterBrainSurfaceNV('FILE', pf.get('SURFFILE')).get('SURF');
49  pf.set('SURF', bs)
50
51  if pf.get('DRAWN')
52      delete(pf.get('H_BRAIN'))
53      pf.set('H_BRAIN', Element.getNoValue())
54
55      pf.memorize('H_BRAIN')
56
57      pf.set('BRAIN', pf.get('BRAIN'))
58
59      pf.memorize('ST_SURFACE').set('PANEL', pf, 'PROP', pf.H_BRAIN).get('
        SETUP')
60
61      pf.memorize('ST_AMBIENT').set('PANEL', pf, 'PROP', pf.H_AXES).get('SETUP
        ')
62  end
63
64  %%% ¡prop!
65  SURF (metadata, item) is the brain surface to be plotted. ⑩
66  %%% ¡settings!
67  'BrainSurface'
68  %%% ¡default!
69  ImporterBrainSurfaceNV('FILE', BrainSurfacePF.getPropDefault('SURFFILE')).
        get('SURF')
70
71  %%% ¡prop!
72  H_BRAIN (evanescent, handle) is the handle for brain surface. ⑪
```

④ is executed only when the VIEW property is set. It takes care of adjusting the view and resetting the lightning.

⑤ determines the axis setting through the container property SettingsAxis, which derives from Settings.

⑥ defines the default values by instantiating a default instance of SettingsAxis.

⑦ employs the property panel SettingsAxisPP, which is specialized for SettingsAxis and derives from SettingsPP.

⑧ contains the file from which the brain surface is plotted.

⑨ is executed only when the SURFILE property is set. It updates the property SURF loading the data from the file. It the figure panel is already drawn, it refreshes the brain handle and redraws it.

⑩ contains the BrainSurface element.

⑪ is the evanescent handle for the brain surface. This is calculated by ⑫.

```matlab
73  %%%% ¡calculate!  (12)
74  triangles = pf.get('SURF').get('TRIANGLES');
75  coordinates = pf.get('SURF').get('COORDINATES');
76  h_brain = trisurf( ...
77      triangles, ...
78      coordinates(:, 1), ...
79      coordinates(:, 2), ...
80      coordinates(:, 3), ...
81      'Parent', pf.memorize('H_AXES'), ...
82      'Tag', 'H_BRAIN' ...
83      );
84  xlabel(pf.get('H_AXES'), 'Sagittal')
85  ylabel(pf.get('H_AXES'), 'Axial')
86  zlabel(pf.get('H_AXES'), 'Coronal')
87  value = h_brain;
88
89  %%% ¡prop!
90  BRAIN (figure, logical) determines whether the brain surface is shown.  (13)
91  %%%% ¡default!
92  true
93  %%%% ¡postset!  (14)
94  if pf.get('DRAWN')
95      if pf.get('BRAIN')
96          set(pf.get('H_BRAIN'), 'Visible', 'on')
97      else % ~pf.get('BRAIN')
98          set(pf.get('H_BRAIN'), 'Visible', 'off')
99      end
100 end
101
102 %%% ¡prop!
103 ST_SURFACE (figure, item) determines the surface settings.  (15)
104 %%%% ¡settings!
105 'SettingsSurface'
106 %%%% ¡gui!
107 pr = SettingsSurfacePP('EL', pf, 'PROP', BrainSurfacePF.ST_SURFACE, varargin
        {:});  (16)
108
109 %%% ¡prop!
110 ST_AMBIENT (figure, item) determines the ambient settings.  (17)
111 %%%% ¡settings!
112 'SettingsAmbient'
113 %%%% ¡default!
114 SettingsAmbient('LIGHTING', 'gouraud', 'MATERIAL', 'dull', 'CAMLIGHT', '
        headlight (x2)', 'SHADING', 'none', 'COLORMAP', 'none')  (18)
115 %%%% ¡gui!
116 pr = SettingsAmbientPP('EL', pf, 'PROP', BrainSurfacePF.ST_AMBIENT, varargin
        {:});  (19)
```

(13) determines whether the brain surface is shown.

(14) is executed whenever BRAIN is set in order to show or hide the brain surface.

(15) determines the brain surface settings throught the container property SettingsSurface, which derives from Settings.

(16) employs the property panel SettingsSurfacePP, which is specialized for SettingsSurface and derives from SettingsPP.

(17) determines the ambient lighting settings throught the container property SettingsAmbient, which is derived from Settings.

(18) defines the default values by instantiating a default instance of SettingsAmbient.

(19) employs the property panel SettingsAmbientPP, which is specialized for SettingsAmbient and derives from SettingsPP.

Code 4: **BrainSurfacePF element props update.** The props_update section of the generator code in _BrainSurfacePF.gen.m updates the properties of the PanelFig element. This defines the core properties of the figure panel.

```matlab
1  %% ¡props_update!
2
3  ...
```

```
4
5  %%% ¡prop!
6  DRAW (query, logical) draws the figure brain surface. (1)
7  %%%% ¡calculate!
8  value = calculateValue@PanelFig(pf, PanelFig.DRAW, varargin{:}); (2)
9  if value
10     pf.memorize('H_AXES') (3)
11
12     pf.memorize('ST_AXIS').set('PANEL', pf, 'PROP', BrainSurfacePF.H_AXES).
          get('SETUP') (4)
13
14     pf.memorize('H_BRAIN') (5)
15
16     pf.memorize('ST_SURFACE').set('PANEL', pf, 'PROP', BrainSurfacePF.
          H_BRAIN).get('SETUP') (6)
17
18     pf.memorize('ST_AMBIENT').set('PANEL', pf, 'PROP', BrainSurfacePF.H_AXES
          ).get('SETUP') (7)
19 end
20
21 %%% ¡prop!
22 DELETE (query, logical) resets the handles when the panel figure brain
       surface is deleted. (8)
23 %%%% ¡calculate!
24 value = calculateValue@PanelFig(pf, PanelFig.DELETE, varargin{:}); % also
       warning
25 if value
26     pf.set('H_AXES', Element.getNoValue())
27     pf.set('H_BRAIN', Element.getNoValue())
28 end
```

(1) initializes the figure panel by drawing its graphical elements.

(2) calls the constructor of the parent. It returns value = true if the panel is drawn correctly. It gives a warning if the panel is not drawn correctly.

(3) ensures that the axes are memorized.

(4) creates, memorizes, and sets up the property ST_AXES containing the axes settings.

(5) memorizes the property H_BRAIN.

(6) creates, memorizes, and sets up the property ST_SURFACE containing the brain surface settings.

(7) creates, memorizes, and sets up the property ST_AMBIENT containing the lightning settings.

(8) deletes all evanescent handles when the figure containing the panel is deleted.

Code 5: **BrainSurfacePF element tests.** The tests section of the generator code in _BrainSurfacePF.gen.m determines how the unit tests are performed.

```
1  %% ¡tests!
2
3  %%% ¡excluded_props! (1)
4  [BrainSurfacePF.PARENT BrainSurfacePF.H BrainSurfacePF.ST_POSITION
       BrainSurfacePF.ST_AXIS BrainSurfacePF.ST_SURFACE BrainSurfacePF.
       ST_AMBIENT]
5
6  %%% ¡warning_off!
7  true
8
9  %%% ¡test!
10 %%%% ¡name!
11 Remove Figures
12 %%%% ¡code!
13 warning('off', [BRAPH2.STR ':BrainSurfacePF'])
14 assert(length(findall(0, 'type', 'figure')) == 1) (2)
15 delete(findall(0, 'type', 'figure')) (3)
16 warning('on', [BRAPH2.STR ':BrainSurfacePF'])
```

(1) some properties need to be excluded from the tests, mainly because they are initialized by other proprties and therefore could give some spurious errors when unit tested.

(2) throws an error if there remains a different number of figures than expected.

(3) removes the figures remaining from the testing.

Addition of toolbar buttons

You will now see how to add the pushbuttons in the toolbar of the figure, opportunely altering the code so far implemented.

Code 6: **BrainSurfacePF element props update.** The props_update section of the generator code in _BrainSurfacePF.gen.m with the additions needed to have the toolbar pushbuttons. ← Code 4

```matlab
1  %% ¡props_update!
2
3  ...
4
5  %%% ¡prop!
6  H_TOOLS (evanescent, handlelist) is the list of panel-specific tools from
          the first. (1)
7  %%%% ¡calculate!
8  toolbar = pf.memorize('H_TOOLBAR'); (2)
9  if check_graphics(toolbar, 'uitoolbar') (3)
10     value = calculateValue@PanelFig(pf, PanelFig.H_TOOLS);
11
12     tool_separator_1 = uipushtool(toolbar, 'Separator', 'on', 'Visible', '
       off');
13
14     % Brain
15     tool_brain = uitoggletool(toolbar, ...
16         'Tag', 'TOOL.Brain', ...
17         'Separator', 'on', ...
18         'State', pf.get('BRAIN'), ...
19         'Tooltip', 'Show Brain', ...
20         'CData', imread('icon_brain.png'), ...
21         'OnCallback', {@cb_brain, true}, ...
22         'OffCallback', {@cb_brain, false});
23
24     % Axis
25     tool_axis = uitoggletool(toolbar, ...
26         'Tag', 'TOOL.Axis', ...
27         'State', pf.get('ST_AXIS').get('AXIS'), ...
28         'Tooltip', 'Show axis', ...
29         'CData', imread('icon_axis.png'), ...
30         'OnCallback', {@cb_axis, true}, ...
31         'OffCallback', {@cb_axis, false});
32
33     % Grid
34     tool_grid = uitoggletool(toolbar, ...
35         'Tag', 'TOOL.Grid', ...
36         'State', pf.get('ST_AXIS').get('GRID'), ...
37         'Tooltip', 'Show grid', ...
38         'CData', imread('icon_grid.png'), ...
39         'OnCallback', {@cb_grid, true}, ...
40         'OffCallback', {@cb_grid, false});
41
42     tool_separator_2 = uipushtool(toolbar, 'Separator', 'on', 'Visible', '
       off');
43
44     % View 3D
45     tool_view3D = uitoggletool(toolbar, ...
46         'Tag', 'TOOL.View3D', ...
47         'Separator', 'on', ...
48         'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_3D_AZEL), ...
```

(1) provides a list of evanescent handles to toolbar pushbuttons.

(2) retrieves the toolbar, ensuring that the handle is memorized, and (3) checks that it is actually drawn.

```matlab
49          'Tooltip', BrainSurfacePF.VIEW_3D_CMD, ...
50          'CData', imread('icon_view_3d.png'), ...
51          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_3D_AZEL});
52
53      % View SL
54      tool_viewSL = uitoggletool(toolbar, ...
55          'Tag', 'TOOL.ViewSL', ...
56          'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_SL_AZEL), ...
57          'Tooltip', BrainSurfacePF.VIEW_SL_CMD, ...
58          'CData', imread('icon_view_sl.png'), ...
59          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_SL_AZEL});
60
61      % View SR
62      tool_viewSR = uitoggletool(toolbar, ...
63          'Tag', 'TOOL.ViewSR', ...
64          'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_SR_AZEL), ...
65          'Tooltip', BrainSurfacePF.VIEW_SR_CMD, ...
66          'CData', imread('icon_view_sr.png'), ...
67          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_SR_AZEL});
68
69      % View AD
70      tool_viewAD = uitoggletool(toolbar, ...
71          'Tag', 'TOOL.ViewAD', ...
72          'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_AD_AZEL), ...
73          'Tooltip', BrainSurfacePF.VIEW_AD_CMD, ...
74          'CData', imread('icon_view_ad.png'), ...
75          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_AD_AZEL});
76
77      % View AV
78      tool_viewAV = uitoggletool(toolbar, ...
79          'Tag', 'TOOL.ViewAV', ...
80          'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_AV_AZEL), ...
81          'Tooltip', BrainSurfacePF.VIEW_AV_CMD, ...
82          'CData', imread('icon_view_av.png'), ...
83          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_AV_AZEL});
84
85      % View CA
86      tool_viewCA = uitoggletool(toolbar, ...
87          'Tag', 'TOOL.ViewCA', ...
88          'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_CA_AZEL), ...
89          'Tooltip', BrainSurfacePF.VIEW_CA_CMD, ...
90          'CData', imread('icon_view_ca.png'), ...
91          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_CA_AZEL});
92
93      % View CP
94      tool_viewCP = uitoggletool(toolbar, ...
95          'Tag', 'TOOL.ViewCP', ...
96          'State', isequal(pf.get('VIEW'), BrainSurfacePF.VIEW_CP_AZEL), ...
97          'Tooltip', BrainSurfacePF.VIEW_CP_CMD, ...
98          'CData', imread('icon_view_cp.png'), ...
99          'ClickedCallback', {@cb_view, BrainSurfacePF.VIEW_CP_AZEL});
100
101     value = {value{:}, ...   ④
102         tool_separator_1, ...
103         tool_brain, tool_axis, tool_grid, ...
104         tool_separator_2, ...
105         tool_view3D, tool_viewSL, tool_viewSL, tool_viewSR, tool_viewAD,
       tool_viewAV, tool_viewCA, tool_viewCP ...
106         };
107 else
108     value = {};
```

④ reorders the pushbuttons.

```
109  end
110  %%%% ¡calculate_callbacks!  (5)
111  function cb_brain(~, ~, brain) % (src, event)
112      pf.set('BRAIN', brain)
113  end
114  function cb_axis(~, ~, axis) % (src, event)
115      pf.get('ST_AXIS').set('AXIS', axis);
116
117      % triggers the update of ST_AXIS
118      pf.set('ST_AXIS', pf.get('ST_AXIS'))
119  end
120  function cb_grid(~, ~, grid) % (src, event)
121      pf.get('ST_AXIS').set('GRID', grid);
122
123      % triggers the update of ST_AXIS
124      pf.set('ST_AXIS', pf.get('ST_AXIS'))
125  end
126  function cb_view(~, ~, azel) % (src, event)
127      pf.set('VIEW', azel)
128  end
129
130  %%% ¡prop!
131  DRAW (query, logical) draws the figure brain surface.
132  %%%% ¡calculate!
133  value = calculateValue@PanelFig(pf, PanelFig.DRAW, varargin{:});
134  if value
135      pf.memorize('H_AXES')
136
137      pf.set('VIEW', pf.get('VIEW'))  (1)
138
139      pf.memorize('ST_AXIS').set('PANEL', pf, 'PROP', BrainSurfacePF.H_AXES).
          get('SETUP')
140      pf.memorize('LISTENER_ST_AXIS');  (2)
141
142      pf.memorize('H_BRAIN')
143
144      pf.set('BRAIN', pf.get('BRAIN'))  (3)
145
146      pf.memorize('ST_SURFACE').set('PANEL', pf, 'PROP', BrainSurfacePF.
          H_BRAIN).get('SETUP')
147
148      pf.memorize('ST_AMBIENT').set('PANEL', pf, 'PROP', BrainSurfacePF.H_AXES
          ).get('SETUP')
149  end
150
151  %%% ¡prop!
152  DELETE (query, logical) resets the handles when the panel figure brain
        surface is deleted.
153  %%%% ¡calculate!
154  value = calculateValue@PanelFig(pf, PanelFig.DELETE, varargin{:});
155  if value
156      pf.set('H_AXES', Element.getNoValue())
157      pf.set('H_BRAIN', Element.getNoValue())
158
159      pf.set('LISTENER_ST_AXIS', Element.getNoValue())  (4)
160  end
```

(5) provides the callback functions for the pushbuttons.

(1) ensures that the postset code is executed by resetting VIEW to its current value. This is needed to update the toolbar pushbuttons when the figure panel is first drawn.

(2) memorizes also the listener to the changes in ST_AXIS. This is needed to ensure that the toolbar pushbuttons are synchronized with the content of ST_AXIS.

(3) ensures that the postset code is executed by resetting BRAIN to its current value. This is needed to update the toolbar pushbuttons when the figure panel is first drawn.

(4) deletes also the evanescent handle for the LISTENER_ST_AXIS when the figure panel is deleted.

Code 7: **BrainSurfacePF element new props with toolbar pushbuttons.** The props section of the generator code in

_BrainSurfacePF.gen.m with the additions needed to have the tool-
bar pushbuttons for the brain surface. ← Code 3

```matlab
1  %% iprops!
2
3  ...
4
5  %%% iprop!
6  VIEW (figure, rvector) sets the desired view as the line-of-sight azimuth
       and elevation angles.
7  %%%% icheck_prop!
8  check = length(value) == 2;
9  %%%% idefault!
10 BrainSurfacePF.VIEW_SL_AZEL
11 %%%% ipostset!
12 if pf.get('DRAWN')
13     view(pf.get('H_AXES'), pf.get('VIEW'))
14
15     % reset the ambient lighting
16     pf.memorize('ST_AMBIENT').set('PANEL', pf, 'PROP', pf.H_AXES).get('SETUP
        ')
17
18     % update state of toggle tools  ①
19     toolbar = pf.get('H_TOOLBAR');
20     if check_graphics(toolbar, 'uitoolbar')
21         set(findobj(toolbar, 'Tag', 'TOOL.View3D'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_3D_AZEL))
22         set(findobj(toolbar, 'Tag', 'TOOL.ViewSL'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_SL_AZEL))
23         set(findobj(toolbar, 'Tag', 'TOOL.ViewSR'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_SR_AZEL))
24         set(findobj(toolbar, 'Tag', 'TOOL.ViewAD'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_AD_AZEL))
25         set(findobj(toolbar, 'Tag', 'TOOL.ViewAV'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_AV_AZEL))
26         set(findobj(toolbar, 'Tag', 'TOOL.ViewCA'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_CA_AZEL))
27         set(findobj(toolbar, 'Tag', 'TOOL.ViewCP'), 'State', isequal(pf.get(
       'VIEW'), BrainSurfacePF.VIEW_CP_AZEL))
28     end
29 end
30 %%%% igui!
31 pr = PanelPropRVectorView('EL', pf, 'PROP', BrainSurfacePF.VIEW, varargin
       {:});
32
33 %%% iprop!
34 ST_AXIS (figure, item) determines the axis settings.
35 %%%% isettings!
36 'SettingsAxis'
37 %%%% idefault!
38 SettingsAxis('GRID', false, 'AXIS', false)
39 %%%% ipostset!  ②
40 if pf.get('DRAWN')
41     toolbar = pf.get('H_TOOLBAR');
42     if check_graphics(toolbar, 'uitoolbar')
43         set(findobj(toolbar, 'Tag', 'TOOL.Grid'), 'State', pf.get('ST_AXIS')
       .get('GRID'))
44         set(findobj(toolbar, 'Tag', 'TOOL.Axis'), 'State', pf.get('ST_AXIS')
       .get('AXIS'))
45     end
46 end
```

① ensures that toolbar pushbuttons
are updated with the current view.

② ensures that the toolbar
pushbuttons are updated whenever
the ST_AXIS property is updated.

```
47  %%%% igui!
48  pr = SettingsAxisPP('EL', pf, 'PROP', BrainSurfacePF.ST_AXIS, varargin{:});
49
50  %%% iprop!  ③
51  LISTENER_ST_AXIS (evanescent, handle) contains the listener to the axis
          settings to update the pushbuttons.
52  %%%% icalculate!
53  value = listener(pf.get('ST_AXIS'), 'PropSet', @cb_listener_st_axis);
54  %%%% icalculate_callbacks!
55  function cb_listener_st_axis(~, ~)
56      if pf.get('DRAWN')
57          toolbar = pf.get('H_TOOLBAR');
58          if check_graphics(toolbar, 'uitoolbar')
59              set(findobj(toolbar, 'Tag', 'TOOL.Grid'), 'State', pf.get('
      ST_AXIS').get('GRID'))
60              set(findobj(toolbar, 'Tag', 'TOOL.Axis'), 'State', pf.get('
      ST_AXIS').get('AXIS'))
61          end
62      end
63  end
64
65  ...
66
67  %%% iprop!
68  BRAIN (figure, logical) determines whether the brain surface is shown.
69  %%%% idefault!
70  true
71  %%%% ipostset!
72  if pf.get('DRAWN')
73      if pf.get('BRAIN')
74          set(pf.get('H_BRAIN'), 'Visible', 'on')
75      else % ~pf.get('BRAIN')
76          set(pf.get('H_BRAIN'), 'Visible', 'off')
77      end
78
79      toolbar = pf.get('H_TOOLBAR');  ④
80      if check_graphics(toolbar, 'uitoolbar')
81          set(findobj(toolbar, 'Tag', 'TOOL.Brain'), 'State', pf.get('BRAIN'))
82      end
83  end
84
85  ...
```

③ ensures that the toolbar pushbuttons are updated whenever the ST_AXIS property is updated.

④ ensures that the toolbar pushbuttons are updated whenever the BRAIN property is updated.

Code 8: **BrainSurfacePF element tests with toolbar pushbuttons.**
The tests section of the generator code in _BrainSurfacePF.gen.m
with the additions needed to have the tool- bar pushbuttons for the
brain surface. ← Code 5

```
1  %% itests!
2
3  %%% iexcluded_props!
4  [BrainSurfacePF.PARENT BrainSurfacePF.H BrainSurfacePF.ST_POSITION
       BrainSurfacePF.ST_AXIS BrainSurfacePF.ST_SURFACE BrainSurfacePF.
       ST_AMBIENT BrainSurfacePF.LISTENER_ST_AXIS]  ①
5
6  ...
```

① excludes from testing also LISTENER_ST_AXIS.

Extension of figure panel (BrainAtlasPF)

You will now learn how to extend BrainSurfacePF to plot also brain regions. You will therefore implement BrainAtlasPF.

Code 9: **BrainAtlasPF element header.** The header section of the generator code in _BrainAtlasPF.gen.m provides the general information about the BrainAtlasPF element.

```
1  %% iheader!
2  BrainAtlasPF < BrainSurfacePF (pf, panel figure brain atlas) is a plot of a
       brain atlas. 1
3
4  %%% idescription!
5  BrainAtlasPF manages the plot of the brain regions symbols,
6  spheres, ids and labels. BrainAtlasPF utilizes the surface created
7  from PFBrainSurface to integrate the regions to a brain surface.
8
9  %%% iseealso!
10 BrainAtlas, BrainSurface
11
12 %%% ibuild!
13 1
```

① BrainAtlasPF is a child of BrainSurfacePF.

Code 10: **BrainAtlasPF spheres.** This code demonstrates how to add the spheres to the BrainAtlasPF.

```
1  %% iprops!
2
3  %%% iprop! 1
4  BA (metadata, item) is the brain atlas with the brain regions.
5  %%%% isettings!
6  'BrainAtlas'
7
8  %%% iprop! 2
9  H_SPHS (evanescent, handlelist) is the set of handles for the spheres.
10 %%%% icalculate! 3
11 L = pf.memorize('BA').get('BR_DICT').get('LENGTH');
12 h_sphs = cell(1, L);
13 for i = 1:1:L
14     h_sphs{i} = surf([], [], [], ...
15         'Parent', pf.memorize('H_AXES'), ...
16         'Tag', ['H_SPHS{' int2str(i) '}'], ...
17         'Visible', false ...
18         );
19 end
20 value = h_sphs;
21
22 %%% iprop! 4
23 SPHS (figure, logical) determines whether the spheres are shown.
24 %%%% idefault!
25 true
26 %%%% ipostset!
27 if ~pf.get('SPHS') % false 5
28     h_sphs = pf.get('H_SPHS');
29     for i = 1:1:length(h_sphs)
30         set(h_sphs{i}, 'Visible', false)
```

① contains the brain atlas to be visualized.

② contains the evanescent handles for the spheres to represent the brain regions.

③ draws the spheres and creates the handles.

④ determines whether the shperes are shown. When it is set to FALSE, ⑤ sets all spheres already drawn to invisible. When it is set to TRUE, ⑥ triggers the update of the sphere dictionary containing the elements corresponding to each sphere.

```
31      end
32  else % true  (6)
33      % triggers the update of SPH_DICT
34      pf.set('SPH_DICT', pf.get('SPH_DICT'))
35  end
36
37  %%% ¡prop!
38  SPH_DICT (figure, idict) contains the spheres of the brain regions.  (7)
39  %%%% ¡settings!
40  'SettingsSphere'
41  %%%% ¡postset!
42  if pf.get('SPHS') && ~isa(pf.getr('BA'), 'NoValue')  (8)
43
44      br_dict = pf.get('BA').get('BR_DICT');
45
46      if pf.get('SPH_DICT').get('LENGTH') == 0 && br_dict.get('LENGTH')  (9)
47          for i = 1:1:br_dict.get('LENGTH')
48              br = br_dict.get('IT', i);
49              sphs{i} = SettingsSphere( ...
50                  'PANEL', pf, ...
51                  'PROP', BrainAtlasPF.H_SPHS, ...
52                  'I', i, ...
53                  'VISIBLE', true, ...
54                  'ID', br.get('ID'), ...
55                  'X', br.get('X'), ...
56                  'Y', br.get('Y'), ...
57                  'Z', br.get('Z'), ...
58                  'FACECOLOR', BRAPH2.COL, ...
59                  'FACEALPHA', 1 ...
60                  );
61          end
62          pf.get('SPH_DICT').set('IT_LIST', sphs)
63      end
64
65      for i = 1:1:br_dict.get('LENGTH')  (10)
66          pf.get('SPH_DICT').get('IT', i).get('SETUP')
67      end
68
69      % reset the ambient lighting
70      pf.get('ST_AMBIENT').get('SETUP')
71  end
72  %%%% ¡gui!  (11)
73  pr = PanelPropIDictTable('EL', pf, 'PROP', BrainAtlasPF.SPH_DICT, ...
74      'COLS', [PanelPropIDictTable.SELECTOR SettingsSphere.VISIBLE
         SettingsSphere.X SettingsSphere.Y SettingsSphere.Z SettingsSphere.
         SPHERESIZE SettingsSphere.FACECOLOR SettingsSphere.FACEALPHA
         SettingsSphere.EDGECOLOR SettingsSphere.EDGEALPHA], ...
75      varargin{:});
76
77  %% ¡props_update!
78
79  ...
80
81  %%% ¡prop!
82  DRAW (query, logical) draws the figure brain atlas.
83  %%%% ¡calculate!
84  value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DRAW, varargin{:});
         % also warning
85  if value
```

(7) provides the dictionary with all sphere elements, which is only executed if (8) the brain atlas is set.

(9) creates the sphere elements if they do not already exist. Each sphere element is a `SettingsSphere` with all properties necessary to set the sphere.

(10) setups the sphere objects by calling the property SETUP on each of them.

(11) uses `PanelPropIDictTable` to provide a table where the sphere settings can be managed.

```
86      pf.memorize('H_SPHS') (12)
87      pf.set('SPHS', pf.get('SPHS')) (13)
88
89      % reset the ambient lighting
90      pf.get('ST_AMBIENT').get('SETUP')
91  end
92
93  %%% iprop!
94  DELETE (query, logical) resets the handles when the panel figure brain
        surface is deleted.
95  %%%% icalculate!
96  value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DELETE, varargin
        {:}); % also warning
97  if value
98      pf.set('H_SPHS', Element.getNoValue()) (14)
99  end
100
101 %% itests!
102
103 %%% iexcluded_props!
104 [BrainAtlasPF.PARENT BrainAtlasPF.H BrainAtlasPF.ST_POSITION BrainAtlasPF.
        ST_AXIS BrainAtlasPF.ST_SURFACE BrainAtlasPF.ST_AMBIENT]
105
106 ...
```

(12) memorizes the sphere handles.

(13) sets the sphere elements
`SettingsSphere` by triggering the
`postset` of SPHS.

(14) deletes the sphere handles when
the figure panel is deleted.

Code 11: **BrainAtlasPF symbols.** This code demonstrates how to add
the symbols to the `BrainAtlasPF`. ← Code 10

```
1   %% iprops!
2
3   ...
4
5   %%% iprop!
6   H_SYMS (evanescent, handlelist) is the set of handles for the symbols.
7   %%%% icalculate!
8   L = pf.memorize('BA').get('BR_DICT').get('LENGTH');
9   h_syms = cell(1, L);
10  for i = 1:1:L
11      h_syms{i} = plot3(0, 0, 0, ...
12          'Parent', pf.get('H_AXES'), ...
13          'Tag', ['H_SYMS{' int2str(i) '}'], ...
14          'Visible', false ...
15          );
16  end
17  value = h_syms;
18
19  %%% iprop!
20  SYMS (figure, logical) determines whether the symbols are shown.
21  %%%% idefault!
22  false
23  %%%% ipostset!
24  if ~pf.get('SYMS') % false
25      h_syms = pf.get('H_SYMS');
26      for i = 1:1:length(h_syms)
27          set(h_syms{i}, 'Visible', false)
28      end
29  else % true
30      % triggers the update of SYM_DICT
31      pf.set('SYM_DICT', pf.get('SYM_DICT'))
```

```matlab
32  end
33
34  %%% ¡prop!
35  SYM_DICT (figure, idict) contains the symbols of the brain regions.
36  %%%% ¡settings!
37  'SettingsSymbol'
38  %%%% ¡postset!
39  if pf.get('SYMS') && ~isa(pf.getr('BA'), 'NoValue')
40
41      br_dict = pf.get('BA').get('BR_DICT');
42
43    if pf.get('SYM_DICT').get('LENGTH') == 0 && br_dict.get('LENGTH')
44          for i = 1:1:br_dict.get('LENGTH')
45              br = br_dict.get('IT', i);
46              syms{i} = SettingsSymbol( ...
47                  'PANEL', pf, ...
48                  'PROP', BrainAtlasPF.H_SYMS, ...
49                  'I', i, ...
50                  'VISIBLE', true, ...
51                  'ID', br.get('ID'), ... % Callback('EL', br, 'TAG', 'ID'),
      ...
52                  'X', br.get('X'), ... % Callback('EL', br, 'TAG', 'X'), ...
53                  'Y', br.get('Y'), ... % Callback('EL', br, 'TAG', 'Y'), ...
54                  'Z', br.get('Z') ... % Callback('EL', br, 'TAG', 'Z') ...
55                  );
56          end
57          pf.get('SYM_DICT').set('IT_LIST', syms)
58      end
59
60      for i = 1:1:br_dict.get('LENGTH')
61          pf.get('SYM_DICT').get('IT', i).get('SETUP')
62      end
63  end
64  %%%% ¡gui!
65  pr = PanelPropIDictTable('EL', pf, 'PROP', BrainAtlasPF.SYM_DICT, ...
66      'COLS', [PanelPropIDictTable.SELECTOR SettingsSymbol.VISIBLE
       SettingsSymbol.X SettingsSymbol.Y SettingsSymbol.Z SettingsSymbol.
       SYMBOL SettingsSymbol.SYMBOLSIZE SettingsSymbol.EDGECOLOR
       SettingsSymbol.FACECOLOR], ...
67      varargin{:});
68
69  %% ¡props_update!
70
71  ...
72
73  %%% ¡prop!
74  DRAW (query, logical) draws the figure brain atlas.
75  %%%% ¡calculate!
76  value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DRAW, varargin{:});
          % also warning
77  if value
78      pf.memorize('H_SPHS')
79      pf.set('SPHS', pf.get('SPHS'))
80
81      pf.memorize('H_SYMS')
82      pf.set('SYMS', pf.get('SYMS'))
83
84      % reset the ambient lighting
85      pf.get('ST_AMBIENT').get('SETUP')
86  end
87
```

```
88  %%% iprop!
89  DELETE (query, logical) resets the handles when the panel figure brain
        surface is deleted.
90  %%%% icalculate!
91  value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DELETE, varargin
        {:}); % also warning
92  if value
93      pf.set('H_SPHS', Element.getNoValue())
94      pf.set('H_SYMS', Element.getNoValue())
95  end
96
97  ...
```

Code 12: **BrainAtlasPF ids.** This code demonstrates how to add the ids to the BrainAtlasPF. ← Code 11

```
1   %% iprops!
2
3   ...
4
5   %%% iprop!
6   H_IDS (evanescent, handlelist) is the set of handles for the ids.
7   %%%% icalculate!
8   L = pf.memorize('BA').get('BR_DICT').get('LENGTH');
9   h_ids = cell(1, L);
10  for i = 1:1:L
11      h_ids{i} = text(0, 0, 0, '', ...
12          'Parent', pf.get('H_AXES'), ...
13          'Tag', ['H_IDS{' int2str(i) '}'], ...
14          'Visible', false ...
15          );
16  end
17  value = h_ids;
18
19  %%% iprop!
20  IDS (figure, logical) determines whether the ids are shown.
21  %%%% idefault!
22  false
23  %%%% ipostset!
24  if ~pf.get('IDS') % false
25      h_ids = pf.get('H_IDS');
26      for i = 1:1:length(h_ids)
27          set(h_ids{i}, 'Visible', false)
28      end
29  else % true
30      % triggers the update of ID_DICT
31      pf.set('ID_DICT', pf.get('ID_DICT'))
32  end
33
34  %%% iprop!
35  ID_DICT (figure, idict) contains the ids of the brain regions.
36  %%%% isettings!
37  'SettingsText'
38  %%%% ipostset!
39  if pf.get('IDS') && ~isa(pf.getr('BA'), 'NoValue')
40
41      br_dict = pf.get('BA').get('BR_DICT');
42
43      if pf.get('ID_DICT').get('LENGTH') == 0 && br_dict.get('LENGTH')
44          for i = 1:1:br_dict.get('LENGTH')
45              br = br_dict.get('IT', i);
```

```matlab
46              ids{i} = SettingsText( ...
47                  'PANEL', pf, ...
48                  'PROP', BrainAtlasPF.H_IDS, ...
49                  'I', i, ...
50                  'VISIBLE', true, ...
51                  'ID', br.get('ID'), ... % Callback('EL', br, 'TAG', 'ID'),
       ...
52                  'X', br.get('X'), ... % Callback('EL', br, 'TAG', 'X'), ...
53                  'Y', br.get('Y'), ... % Callback('EL', br, 'TAG', 'Y'), ...
54                  'Z', br.get('Z'), ... % Callback('EL', br, 'TAG', 'Z'), ...
55                  'TXT', br.get('ID') ... % Callback('EL', br, 'TAG', 'ID')
       ...
56                  );
57          end
58          pf.get('ID_DICT').set('IT_LIST', ids)
59      end
60
61      for i = 1:1:br_dict.get('LENGTH')
62          pf.get('ID_DICT').get('IT', i).get('SETUP')
63      end
64 end
65 %%%% ¡gui!
66 pr = PanelPropIDictTable('EL', pf, 'PROP', BrainAtlasPF.ID_DICT, ...
67     'COLS', [PanelPropIDictTable.SELECTOR SettingsText.VISIBLE SettingsText.
      X SettingsText.Y SettingsText.Z SettingsText.ROTATION SettingsText.TXT
      SettingsText.FONTNAME SettingsText.FONTSIZE SettingsText.FONTCOLOR
      SettingsText.INTERPRETER], ...
68     varargin{:});
69
70 %% ¡props_update!
71
72 ...
73
74 %%% ¡prop!
75 DRAW (query, logical) draws the figure brain atlas.
76 %%%% ¡calculate!
77 value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DRAW, varargin{:});
       % also warning
78 if value
79     pf.memorize('H_SPHS')
80     pf.set('SPHS', pf.get('SPHS'))
81
82     pf.memorize('H_SYMS')
83     pf.set('SYMS', pf.get('SYMS'))
84
85     pf.memorize('H_IDS')
86     pf.set('SPHS', pf.get('SPHS'))
87
88     % reset the ambient lighting
89     pf.get('ST_AMBIENT').get('SETUP')
90 end
91
92 %%% ¡prop!
93 DELETE (query, logical) resets the handles when the panel figure brain
       surface is deleted.
94 %%%% ¡calculate!
95 value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DELETE, varargin
      {:}); % also warning
96 if value
97     pf.set('H_SPHS', Element.getNoValue())
98     pf.set('H_SYMS', Element.getNoValue())
```

```
99      pf.set('H_IDS', Element.getNoValue())
100 end
101
102 ...
```

Code 13: **BrainAtlasPF labels.** This code demonstrates how to add the labels to the `BrainAtlasPF`. ← Code 12

```
1  %% iprops!
2
3  ...
4
5  %%% iprop!
6  H_LABS (evanescent, handlelist) is the set of handles for the labels.
7  %%%% icalculate!
8  L = pf.memorize('BA').get('BR_DICT').get('LENGTH');
9  h_labs = cell(1, L);
10 for i = 1:1:L
11     h_labs{i} = text(0, 0, 0, '', ...
12         'Parent', pf.get('H_AXES'), ...
13         'Tag', ['H_LABS{' int2str(i) '}'], ...
14         'Visible', false ...
15         );
16 end
17 value = h_labs;
18
19 %%% iprop!
20 LABS (figure, logical) determines whether the labels are shown.
21 %%%% idefault!
22 false
23 %%%% ipostset!
24 if ~pf.get('LABS') % false
25     h_labs = pf.get('H_LABS');
26     for i = 1:1:length(h_labs)
27         set(h_labs{i}, 'Visible', false)
28     end
29 else % true
30   % triggers the update of LAB_DICT
31   pf.set('LAB_DICT', pf.get('LAB_DICT'))
32 end
33
34 %%% iprop!
35 LAB_DICT (figure, idict) contains the labels of the brain regions.
36 %%%% isettings!
37 'SettingsText'
38 %%%% ipostset!
39 if pf.get('LABS') && ~isa(pf.getr('BA'), 'NoValue')
40
41     br_dict = pf.get('BA').get('BR_DICT');
42
43     if pf.get('LAB_DICT').get('LENGTH') == 0 && br_dict.get('LENGTH')
44         for i = 1:1:br_dict.get('LENGTH')
45             br = br_dict.get('IT', i);
46             labs{i} = SettingsText( ...
47                 'PANEL', pf, ...
48                 'PROP', BrainAtlasPF.H_LABS, ...
49                 'I', i, ...
50                 'VISIBLE', true, ...
51                 'ID', br.get('ID'), ... % Callback('EL', br, 'TAG', 'ID'),
     ...
52                 'X', br.get('X'), ... % Callback('EL', br, 'TAG', 'X'), ...
```

```
53                    'Y', br.get('Y'), ... % Callback('EL', br, 'TAG', 'Y'), ...
54                    'Z', br.get('Z'), ... % Callback('EL', br, 'TAG', 'Z'), ...
55                    'TXT', br.get('LABEL') ... % Callback('EL', br, 'TAG', '
        LABEL') ...
56                    );
57            end
58            pf.get('LAB_DICT').set('IT_LIST', labs)
59        end
60
61        for i = 1:1:br_dict.get('LENGTH')
62            pf.get('LAB_DICT').get('IT', i).get('SETUP')
63        end
64 end
65 %%%% ¡gui!
66 pr = PanelPropIDictTable('EL', pf, 'PROP', BrainAtlasPF.LAB_DICT, ...
67        'COLS', [PanelPropIDictTable.SELECTOR SettingsText.VISIBLE SettingsText.
        X SettingsText.Y SettingsText.Z SettingsText.ROTATION SettingsText.TXT
        SettingsText.FONTNAME SettingsText.FONTSIZE SettingsText.FONTCOLOR
        SettingsText.INTERPRETER], ...
68        varargin{:});
69
70 %% ¡props_update!
71
72 ...
73
74 %%% ¡prop!
75 DRAW (query, logical) draws the figure brain atlas.
76 %%%% ¡calculate!
77 value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DRAW, varargin{:});
        % also warning
78 if value
79     pf.memorize('H_SPHS')
80     pf.set('SPHS', pf.get('SPHS'))
81
82     pf.memorize('H_SYMS')
83     pf.set('SYMS', pf.get('SYMS'))
84
85     pf.memorize('H_IDS')
86     pf.set('SPHS', pf.get('SPHS'))
87
88     pf.memorize('H_LABS')
89     pf.set('LABS', pf.get('LABS'))
90
91     % reset the ambient lighting
92     pf.get('ST_AMBIENT').get('SETUP')
93 end
94
95 %%% ¡prop!
96 DELETE (query, logical) resets the handles when the panel figure brain
        surface is deleted.
97 %%%% ¡calculate!
98 value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DELETE, varargin
        {:}); % also warning
99 if value
100     pf.set('H_SPHS', Element.getNoValue())
101     pf.set('H_SYMS', Element.getNoValue())
102     pf.set('H_IDS', Element.getNoValue())
103     pf.set('H_LABS', Element.getNoValue())
104 end
105
106 ...
```

Extension of toolbar buttons

You will now see how to add toolbar pushbuttons to the previous code.

Code 14: **BrainAtlasPF with toolbar.** This code demonstrates how to add the toolbar pushbuttons to the BrainAtlasPF. ← Code 13

```matlab
%% iprops!

...

%%% iprop!
SPHS (figure, logical) determines whether the spheres are shown.
%%%% idefault!
true
%%%% ipostset!
if ~pf.get('SPHS') % false
    h_sphs = pf.get('H_SPHS');
    for i = 1:1:length(h_sphs)
        set(h_sphs{i}, 'Visible', false)
    end
else % true
    % triggers the update of SPH_DICT
    pf.set('SPH_DICT', pf.get('SPH_DICT'))
end

% update state of toggle tool
toolbar = pf.get('H_TOOLBAR');
if check_graphics(toolbar, 'uitoolbar')
    set(findobj(toolbar, 'Tag', 'TOOL.Sphs'), 'State', pf.get('SPHS'))
end

...

%%% iprop!
SYMS (figure, logical) determines whether the symbols are shown.
%%%% idefault!
false
%%%% ipostset!
if ~pf.get('SYMS') % false
    h_syms = pf.get('H_SYMS');
    for i = 1:1:length(h_syms)
        set(h_syms{i}, 'Visible', false)
    end
else % true
    % triggers the update of SYM_DICT
    pf.set('SYM_DICT', pf.get('SYM_DICT'))
end

% update state of toggle tool
toolbar = pf.get('H_TOOLBAR');
if check_graphics(toolbar, 'uitoolbar')
    set(findobj(toolbar, 'Tag', 'TOOL.Syms'), 'State', pf.get('SYMS'))
end

...

%%% iprop!
IDS (figure, logical) determines whether the ids are shown.
%%%% idefault!
```

```matlab
false
%%%% ¡postset!
if ~pf.get('IDS') % false
    h_ids = pf.get('H_IDS');
    for i = 1:1:length(h_ids)
        set(h_ids{i}, 'Visible', false)
    end
else % true
    % triggers the update of ID_DICT
    pf.set('ID_DICT', pf.get('ID_DICT'))
end

% update state of toggle tool
toolbar = pf.get('H_TOOLBAR');
if check_graphics(toolbar, 'uitoolbar')
    set(findobj(toolbar, 'Tag', 'TOOL.Ids'), 'State', pf.get('IDS'))
end

...

%%% ¡prop!
LABS (figure, logical) determines whether the labels are shown.
%%%% ¡default!
false
%%%% ¡postset!
if ~pf.get('LABS') % false
    h_labs = pf.get('H_LABS');
    for i = 1:1:length(h_labs)
        set(h_labs{i}, 'Visible', false)
    end
else % true
  % triggers the update of LAB_DICT
  pf.set('LAB_DICT', pf.get('LAB_DICT'))
end

% update state of toggle tool
toolbar = pf.get('H_TOOLBAR');
if check_graphics(toolbar, 'uitoolbar')
    set(findobj(toolbar, 'Tag', 'TOOL.Labs'), 'State', pf.get('LABS'))
end

...

%% ¡props_update!

...

%%% ¡prop!
DRAW (query, logical) draws the figure brain atlas.
%%%% ¡calculate!
value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.DRAW, varargin{:});
       % also warning
if value
    pf.memorize('H_SPHS')
    pf.set('SPHS', pf.get('SPHS')) % sets also   SPH_DICT

    pf.memorize('H_SYMS')
    pf.set('SYMS', pf.get('SYMS')) % sets also   SYM_DICT

    pf.memorize('H_IDS')
    pf.set('SPHS', pf.get('SPHS')) % sets also   ID_DICT
```

```matlab
114
115     pf.memorize('H_LABS')
116     pf.set('LABS', pf.get('LABS')) % sets also   LAB_DICT
117
118     % reset the ambient lighting
119     pf.get('ST_AMBIENT').get('SETUP')
120 end
121 %%%% ¡calculate_callbacks!
122 function cb_sphs(~, ~, sphs) % (src, event)
123     pf.set('SPHS', sphs)
124 end
125 function cb_syms(~, ~, syms) % (src, event)
126     pf.set('SYMS', syms)
127 end
128 function cb_ids(~, ~, ids) % (src, event)
129     pf.set('IDS', ids)
130 end
131 function cb_labs(~, ~, labs) % (src, event)
132     pf.set('LABS', labs)
133 end
134
135 ...
136
137 %%% ¡prop!
138 H_TOOLS (evanescent, handlelist) is the list of panel-specific tools from
        the first.
139 %%%% ¡calculate!
140 toolbar = pf.memorize(PanelFig.H_TOOLBAR);
141 if check_graphics(toolbar, 'uitoolbar')
142   value = calculateValue@BrainSurfacePF(pf, BrainSurfacePF.H_TOOLS);
143
144     tool_separator_1 = uipushtool(toolbar, 'Separator', 'on', 'Visible', '
      off');
145
146     % Spheres
147     tool_sphs = uitoggletool(toolbar, ...
148         'Tag', 'TOOL.Sphs', ...
149         'Separator', 'on', ...
150         'State', pf.get('SPHS'), ...
151         'Tooltip', 'Show Spheres', ...
152         'CData', imread('icon_sphere.png'), ...
153         'OnCallback', {@cb_sphs, true}, ...
154         'OffCallback', {@cb_sphs, false});
155
156     % Symbols
157     tool_syms = uitoggletool(toolbar, ...
158         'Tag', 'TOOL.Syms', ...
159         'Separator', 'on', ...
160         'State', pf.get('SYMS'), ...
161         'Tooltip', 'Show Symbols', ...
162         'CData', imread('icon_symbol.png'), ...
163         'OnCallback', {@cb_syms, true}, ...
164         'OffCallback', {@cb_syms, false});
165
166     % IDs
167     tool_ids = uitoggletool(toolbar, ...
168         'Tag', 'TOOL.Ids', ...
169         'Separator', 'on', ...
170         'State', pf.get('IDS'), ...
171         'Tooltip', 'Show IDs', ...
172         'CData', imread('icon_id.png'), ...
```

```matlab
173            'OnCallback', {@cb_ids, true}, ...
174            'OffCallback', {@cb_ids, false});
175
176        % Labels
177        tool_labs = uitoggletool(toolbar, ...
178            'Tag', 'TOOL.Labs', ...
179            'Separator', 'on', ...
180            'State', pf.get('LABS'), ...
181            'Tooltip', 'Show Labels', ...
182            'CData', imread('icon_label.png'), ...
183            'OnCallback', {@cb_labs, true}, ...
184            'OffCallback', {@cb_labs, false});
185
186        value = {value{:}, ...
187            tool_separator_1, ...
188            tool_sphs, tool_syms, tool_ids, tool_labs ...
189            };
190    else
191      value = {};
192    end
193
194    ...
```