FESTIVAL_DATA_ENGINE OBJECT

1/ DESCRIPTION

The FESTIVAL_DATA_ENGINE object is called each time a new data has to be computed or read out of the HISTORY_STACK. The application has only one instance of FESTIVAL_DATA_ENGINE running. No GUI is associated to this object.

The main functionalities of the FESTIVAL_DATA_ENGINE are :

- Controlling the FESTIVAL_VISU_GUI, i.e. creation and destruction of the FESTIVAL_VISU_GUIs, modification of their content (images, polygons and grids) when the user changes some parameters (like projection, time, mode compact or not, etc),
- Sending to the FESTIVAL_SELECTION_GUI the information telling which data is visualized and which data (i.e. search result) is not visualized in the different FESTIVAL_VISU_GUI,
- Requesting the reading of FITS files, computing the calibrated data, applying the filters and the enhancements requested by the user, computing the projection of the data and masks,
- Creation and destruction of the FESTIVAL_DATA objects,
- Management of the FESTIVAL_HISTORY_STACK and of the data creation and destruction in this stack.
- Management of the current time(s) used in compact mode,
- Management of the “Load configuration” and “Save configuration” commands
- Management of the FESTIVAL_PROGRESS_BAR and printing the status of the application (in the log area of the FESTIVAL_SELECTION_GUI),
- Management of the profiles required by the user (creation of the data profile and the window to visualize it)
- Computation of the difference data (when the difference is required by the user in the FESTIVAL_VISU_GUI),
- Management of the sequence movies
- Management of any polygon (creation, projection, destruction) to be added to the visualized data.

2/ Index

Available methods

3/ SUPERCLASSES

4/ PROPERTIES

Preliminary : a few constants are defined for this object in FESTIVAL_DATA_TYPES :

Nb_Probes = 3 (STEREO_A, STEREO_B, SOHO)

STEREO_A_index = 0

STEREO_B_index = 1

SOHO_index = 2

MaxNb_of_instruments = 5 (STEREO A and B have 5 instruments, SOHO has 4 instruments)

EUVI_index = 0

COR1_index = 1

COR2_index = 2

HI1_index = 3

HI2_index = 4

EIT_index = 0

C1_index = 1

C2_index = 2

C3_index = 3

Properties :

- **VISU_GUI_of_probe[Nb_Probes]** : array of references to the FESTIVAL_VISU_GUI object of the 3 probes.

- **SELECTION_GUI :** reference to the FESTIVAL_SELECTION_GUI object.

- **HISTORY_STACK :** reference to the FESTIVAL_HISTORY_STACK object

- **compact_mode_is_ON_for_probe[Nb_Probes]** : array of 3 flags (TRUE/FLASE). Contains the information for each visualization window : is the "compact mode" set ON ?

- **linked_mode_is_ON** : flag (TRUE/FLASE). Contains the information : is the "linked mode" set ON ?

- **current_compact_date[Nb_Probes]:** array of 3 dates defining the current date in compact mode for each probe. If the "linked mode" is ON, the 3 dates are the same.

- **probe_is_active[Nb_Probes]:** array of 3 TRUE/FALSE flags. It indicates if the probe has searchresults associated to it.

- **data_for_instrument_is_available [Nb_Probes, MaxNb_of_instruments]** : array of TRUE/FALSE flags defining which instrument has been requested for the last query. For exemple, data_for_instrument_is_available[2,2]=TRUE means that instrument C2 of SOHO probe has been requested at the last query. This property will be updated at each new query and will be used to know which instruments can be selected in the FESTIVAL_VISU objects for a PREVIOUS/NEXT operation.

- **diff_requested_for_instrument [Nb_Probes, MaxNb_of_instruments]** : array of TRUE/FALSE flags defining for each instrument if the difference data is requested. This property will be updated when the user clicks in the diff cell of a VISU_GUI (the function DIFF_REQUESTED_BY_CLICK will be called in this case).

- **visualized_SearchResults [Nb_Probes, MaxNb_of_instruments]** : array of object references defining for each probe and for each instrument which SearchResult of the GLOBAL_RESULTS object is visualized at the current time.

- **visualized_Data [Nb_Probes, MaxNb_of_instruments]** : array of object references defining for each probe and for each instrument which FESTIVAL_DATA in the HISTORY_STACK is visualized at the current time.

- **Progressbar** : reference to the progressbar object

5/ METHODS

5.1/ FESTIVAL_DATA_ENGINE::INIT

Syntax :
Result = Obj->[FESTIVAL_DATA_ENGINE::]Init (SELECTION_GUI)

Description :
The FESTIVAL_DATA_ENGINE::INIT function method constructs a FESTIVAL_DATA_ENGINE object.
It also creates the FESTIVAL_HISTORY_STACK and the FESTIVAL_VISU_GUI objects (but the VISU_GUI widgets are not realized yet).

Return Value :
The FESTIVAL_DATA_ENGINE::INIT function method returns 1 if initialization was successful, or 0 otherwise.

Arguments :

SELECTION_GUI :

A reference to the FESTIVAL_SELECTION_GUI which is the object containing the main GUI of the FESTIVAL application.

Keywords :

5.2/ FESTIVAL_DATA_ENGINE::UPDATE_INSTRUMENT_LIST

Syntax :
Result = Obj->[FESTIVAL_DATA_ENGINE::]Update_Instrument_List (STEREOA_LIST=StereoA_list, STEREOB_LIST=StereoB_list, SOHO_LIST=Soho_list)

Description :
The FESTIVAL_DATA_ENGINE::Update_Instrument_List method is called when a new query has been done in order to update the property data_for_instrument_is_available[Nb_Probes, MaxNb_of_instruments].

If the value of the list is different from previous the one, the FESTIVAL_VISU_GUI are informed (and the widget is rebuild if necessary : the VISU::CHANGE_INSTRUMENT_LIST is called).
The property probe_is_active[Nprobes] is also updated.

Return Value :
The FESTIVAL_DATA_ENGINE::Update_Instrument_List function method returns 1 if successful, or 0 otherwise.

Arguments :

Keywords : (all keywords are mandatory)

STEREOA_LIST : an array of TRUE/FALSE flags, one for each STEREO A instrument.

STEREOB_LIST : an array of TRUE/FALSE flags, one for each STEREO B instrument.

SOHO_LIST : an array of TRUE/FALSE flags, one for each SOHO instrument.

5.3/ FESTIVAL_DATA_ENGINE::REDRAW_ALL_VISU

Syntax :
Result = Obj->[FESTIVAL_DATA_ENGINE::]Redraw_All_Visu ()

Description :
The FESTIVAL_DATA_ENGINE::Redraw_All_Visu is called when the redraw method of each FESTIVAL_VISU_GUI must be called (for example when the user has changed a color palette)

Return Value :
The FESTIVAL_DATA_ENGINE::Redraw_All_Visu function method returns 1 if successful, or 0 otherwise.

Arguments :

Keywords :

5.4/ FESTIVAL_DATA_ENGINE::REALIZE_ALL_VISU

Syntax :
Result = Obj->[FESTIVAL_DATA_ENGINE::]Realize_All_Visu ()

Description :
The FESTIVAL_DATA_ENGINE::Realize_All_Visu is called in order to build and realize each FESTIVAL_VISU_GUI widget (if a probe has not been queried in the SELECTION_GUI, it will not be realized).

Return Value :
The FESTIVAL_DATA_ENGINE::Realize_All_Visu function method returns 1 if successful, or 0 otherwise.

Arguments :

Keywords :

5.5/ FESTIVAL_DATA_ENGINE::DIFF_REQUESTED_BY_CLICK

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] Diff_Requested_by_click (Probe_index, Instrument_index, TRUE_FALSE_flag)

Description :
The FESTIVAL_DATA_ENGINE::Diff_Requested_by_click is called when the user clicks in a diff cell of the table of information in a FESTIVAL_VISU_GUI. In this case, the property diff_requested_for_instrument[ ] is updated for the probe & instrument concerned. The new data is then build and the VISU_GUI updated and redrawn.

Return Value :
The FESTIVAL_DATA_ENGINE::Diff_Requested_by_click function method returns 1 if successful, or 0 otherwise.

Arguments :

Probe_index:
The index of the probe associated to this difference data request

Instrument_index:
The index of the instrument associated to this difference data request

TRUE_FALSE_flag:
The flag for the difference request (TRUE means that the difference is requested, FALSE means that the difference is not requested any more).

Keywords :

5.6/ FESTIVAL_DATA_ENGINE::SEARCH_DATA_IN_STACK

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] SEARCH_DATA_IN_STACK (SearchResult, BEST_DATA_FOUND=best_data_found, DIFFERENCE_DATA=difference_data)

Description :
The FESTIVAL_DATA_ENGINE::Search_data_in_stack is called in order to explore the History_stack and look for the data that will best fit the needed one, which is defined by the reference to its SearchResult and all user preferences.
When the needed data is a difference data, the keyword DIFFERENCE_DATA must be set.

Return Value :
The FESTIVAL_DATA_ENGINE::Search_Data_in_Stack function method returns an integer that defines the type of best data found. Possible values are :

- NO_USEFUL_DATA_IN_STACK : there is no data in the stack that can be used to compute the requested one
- ONLY_FILTERED_FOUND : the best data found to compute the requested one is a data where the filtered image can be used (but enhancement and projection and mask can not be used)
- ONLY_ENHANCEDPROJ_BUT_NO_MASK : the best data found to compute the requested one is a data where the enhanced and projected image can be used (but the coordinates of the projected mask can not be used, probably because the masking parameters have changed)
- PERFECT_DATA_FOUND : the exact requested data has been found in HISTORY_STACK.

Arguments :

SearchResult :
The reference to the SearchResult corresponding to the needed data

Keywords :

DIFFERENCE_DATA :
This keyword must be set when the data needed is a difference data.

BEST_DATA_FOUND :
This keyword must be set to a named variable and will contain the reference to the best data found when the function returns, or 0 if no useful data has been found in the stack.

Algorithm :

```
Result = Obj -> [ FESTIVAL_DATA_ENGINE::]SEARCH_DATA_IN_STACK ( SearchResult,
                              BEST_DATA_FOUND=best_data_found,
                              DIFFERENCE=difference)

  best_score = NO_USEFUL_DATA_IN_STACK

  DATA_array = HISTORY_STACK->GET(/ALL,COUNT=Nb_res)

  //starting from Nb_res-1, i.e. the last data used

  FOR i = Nb_res-1, 0, -1 do
  {
    score = COMPUTE_DATA_SIMILARITY (DATA_array[i], SearchResult,
                                     DIFFERENCE=difference)

    if ( score > best_score ) then begin
    {
       best_score = score
       best_data_found = DATA_array[i]
       if score = PREFECT_DATA_FOUND then begin
       {
         // as the data will be re-used, put it on top of the stack
         HISTORY_STACK->MOVE_ON_TOP, i
         return, PERFECT_DATA_FOUND
       }
    } endif

  }
  ENDFOR

  return, best_score
```

5.7/ FESTIVAL_DATA_ENGINE::COMPUTE_DATA_SIMILARITY

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::]COMPUTE_DATA_SIMILARITY (Data_ref, SearchResult, DIFFERENCE_DATA=difference_data)

Description :
The FESTIVAL_DATA_ENGINE::Compute_Data_Similarity is called in order determine if a data referenced by data_ref can be used to compute the SearchResult data respecting the user preferences. When the needed data is a difference data, the keyword DIFFERENCE_DATA must be set.

Return Value :
The FESTIVAL_DATA_ENGINE::Compute_data_similarity function method returns an integer. The better the data fits to the requested searchresult, the higher the integer is. Possible values are (from lowest to highest score) :

- NO_USEFUL_DATA_IN_STACK : there is no data in the stack that can be used to compute the requested one
- ONLY_FILTERED_FOUND : the best data found to compute the requested one is a data where

the filtered image can be used (but enhancement and projection and mask can not be used)

- ONLY_ENHANCEDPROJ_BUT_NO_MASK : the best data found to compute the requested one is a data where the enhanced and projected image can be used (but the coordinates of the projected mask can not be used, probably because the masking parameters have changed),
- PERFECT_DATA_FOUND : the exact requested data has been found in HISTORY_STACK.

Arguments :
Data_ref :
A reference to the existing data (probably already pushed in the HISTORY_STACK) to be compared to the SearchResult data with user preferences.

SearchResult :
The reference to the SearchResult corresponding to the needed data

Keywords :

DIFFERENCE_DATA :
This keyword must be set when the data needed is a difference data.

Algorithm :

```
Result = Obj -> [ FESTIVAL_DATA_ENGINE::]COMPUTE_DATA_SIMILARITY( DATA_ref,
                                               SearchResult, DIFFERENCE=difference)

SearchResult->GetProperty, PROBE=s_probe, INSTRUMENT=s_instrument,DATE=s_date

Data_ref->GetProperty, PROBE=d_probe, INSTRUMENT=d_instrument,DATE=d_date,
   DELTAOFIMAGEDIFF=d_diff

if (keyword_set(difference) NE (abs(d_diff) GT !epsilon) then
    return, NO_USEFUL_DATA_IN_STACK

if (s_probe NE d_probe) OR (s_instrument NE d_instrument)
                        OR (abs(s_date-d_date) GT !epsilon) then
    return, NO_USEFUL_DATA_IN_STACK

// the Data in the stack is for the correct instrument and date, we now need
// to know if the Data fits well to the user preferences.

Data_ref->GetProperty, CalibrationModel = d_CalibrationModel,
                       FilteringSequence = d_FilteringSequence,


oCalibrationGUI->GetProperty, PROBE=s_probe, INSTRUMENT=s_instrument,
      USER_PREFERENCE = s_user_pref

// calibration model = 0 if no calibration is performed, and is >0 otherwise
if NOT SAME_CALIBRATION(s_user_pref,d_CalibrationModel) then
         return, NO_USEFUL_DATA_IN_STACK
```

```
oFilterGUI->GetProperty, PROBE=s_probe, INSTRUMENT=s_instrument,
      USER_PREFERENCE = s_user_pref

if NOT SAME_FILTER(s_user_pref,d_FilteringSequence) then
         return, NO_USEFUL_DATA_IN_STACK

// the Data has same calibration model and same filter

Data_ref->GetProperty, PROJCOORD = d_ProjCoord, ENHANCEMENT = d_Enhancement

oProjCoordGUI->GetProperty, PROBE=s_probe, INSTRUMENT=s_instrument,
      USER_PREFERENCE = s_user_pref

if NOT SAME_PROJCOORD(s_user_pref,d_ProjCoord) then
         return, ONLY_FILTERED_FOUND

oVisEnhancementGUI->GetProperty, PROBE=s_probe, INSTRUMENT=s_instrument,
      USER_PREFERENCE = s_user_pref

if NOT SAME_ENHANCEMENT(s_user_pref,d_Enhancement) then
         return, ONLY_FILTERED_FOUND


// the Data has same calibration model, same filter, same projection and the
// same enhancement

Data_ref->GetProperty, MASKING_PREF = d_Masking_Pref


oMaskingGUI->GetProperty, PROBE=s_probe, INSTRUMENT=s_instrument,
      USER_PREFERENCE = s_user_pref

if NOT SAME_MASKING(s_user_pref,d_Masking_Pref) then
         return, ONLY_ENHANCEDPROJ_BUT_NO_MASK

// the data is exactly the one needed

return, PERFECT_DATA_FOUND
```

5.8/ FESTIVAL_DATA_ENGINE::FIND_OR_BUILD_DATA

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] FIND_OR_BUILD_DATA (SearchResult, DIFFERENCE_DATA=difference_data, NO_PROJECTION_NEEDED=no_projection_needed)

Description :
The FESTIVAL_DATA_ENGINE::Find_or_Build_DATA is called each time a FESTIVAL_DATA is needed. The function first looks at the history stack for the data or an existing DATA that can be used to compute the needed data. If no useful data is found in the history stack, a new data is created and computed. If a new data is created, it is pushed in the HISTORY_STACK.

When the needed data is a difference data, the keyword DIFFERENCE_DATA must be set. In this case, the data corresponding to the previous SearchResult is first searched with the FIND_OR_BUILD_DATA function but with the NO_PROJECTION_NEEDED keyword set because the data that will be used to compute the difference is the filtered image before enhancement and projection.

Return Value :
The FESTIVAL_DATA_ENGINE::Find_or_Build_DATA function method returns a reference to the DATA object found or created if successful, or 0 otherwise.

Arguments :

SearchResult :
The reference to the SearchResult corresponding to the needed data.

Keywords :

DIFFERENCE_DATA :
This keyword must be set when the data needed is a difference data (in this case, the function will find_and_build 3 data : the current data, the previous data and the difference data).

NO_PROJECTION_NEEDED :
This keyword must be set when the content that will be used of the requested data that is only the FilteredImage data part (see above).

Algorithm :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] FIND_OR_BUILD_DATA (SearchResult, DIFFERENCE_DATA=difference_data, NO_PROJECTION_NEEDED=no_projection_needed)

```
DATA=0

info = SEARCH_DATA_IN_STACK ( SearchResult,
                              BEST_DATA_FOUND=best_data_found,
                              DIFFERENCE=difference)

if not [ keyword_set(difference) AND (info EQ NO_USEFUL_DATA_IN_STACK) ] then
{
 ; if no useful data has been found for the difference data (i.e. if the
 ; difference has never been computed, then this part is not run : the execution
 ; goes to the next part.

 switch info of

  NO_USEFUL_DATA_IN_STACK :
  {
    DATA = NEW_OBJECT('FESTIVAL_DATA')
    DATA->DATA_PREP()
    DATA->ROTATE()
    DATA->FILTER()
  }

  ONLY_FILTERED_FOUND :
  {
    IF (DATA EQ 0) THEN DATA = COPY_DATA(best_data_found,/WITH_FILTERED_IMAGE)
    DATA -> ENHANCE
    DATA -> CREATE_PROJECTED_IMAGE(NO_PROJECTION_NEEDED=no_projection_needed)
  }

  ONLY_ENHANCEDPROJ_BUT_NO_MASK :
  {
    IF (DATA EQ 0) then DATA =
       COPY_DATA(best_data_found,/WITH_FILTERED_IMAGE,/WITH_ENHANCED_PROJECT)
    DATA-> BUILD_PROJ_MASK
    DATA-> CLEAN_BEFORE_PUSH
    HISTORY_STACK->PUSH, DATA
    return, DATA
  }
  ELSE :
  {
   //PERFECT_DATA_FOUND, nothing has to be done, needed data already exists :-)
   return, best_data_found
  }

 endswitch

} else ; case where no useful data has been found for the difference data
{
   previous_SearchResult =
                    GLOBAL_RESULT->look_for_previous_searchresult(SearchResult)

   if ( previous_SearchResult NE 0 ) then
      prev_DATA = FIND_OR_BUILD_DATA (previous_SearchResult,/NO_PROJECTION_NEEDED)

   else return, 0

   current_DATA = FIND_OR_BUILD_DATA ( SearchResult, /NO_PROJECTION_NEEDED )

   diff_DATA = MAKE_DIFF ( previous_DATA, current_DATA )

   diff_DATA -> CREATE_PROJECTED_IMAGE

   diff_DATA -> BUILD_PROJ_MASK

   diff_DATA -> CLEAN_BEFORE_PUSH

   HISTORY_STACK -> push, diff_DATA

   return, diff_DATA
}
```

5.9/ FESTIVAL_DATA_ENGINE::NEXT_SET_REQUESTED

Syntax :

Result = Obj -> [FESTIVAL_DATA_ENGINE::] NEXT_SET_REQUESTED (Probe_index)

Description :
The FESTIVAL_DATA_ENGINE::Next_Set_Requested function is called when the user clicks on the NEXT button. It finds the next set of SearchResult and data that is needed by the VISU_GUI that makes the request (or by all VISU_GUI with available data in linked mode)

Return Value :
The FESTIVAL_DATA_ENGINE::Next_Set_Requested function method returns 1 if successful, or 0 otherwise.

Arguments :
Probe_index :
The index of the probe for which the request is made.

Keywords :

Algorithm :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] NEXT_SET_REQUESTED (Probe_index)

```
soon_visualized_SearchResults = visualized_SearchResults

if not linked_mode_is_ON then
{
   if compact_mode_is_ON_for_probe ( Probe_index ) then begin
   {
     new_date = Global_results->get_next_date(Probe_index,
                                              current_compact_date[Probe_index])

     soon_visualized_SearchResults[Probe_index,*] =
     Global_results->get_compact_searchresults(Probe_index, new_date,
                                                    visualized_SearchResults)
     self.current_compact_date[Probe_index] = new_date
   }
   else begin //not in mode compact and not in link mode
   {
     soon_visualized_SearchResults[Probe_index,*] =
     Global_results->get_next_searchresults(visualized_SearchResults,
     visu_gui[probe].instrument_is_Selected)
   }
   endelse
 }
 else begin // linked mode is ON
 {
   if compact_mode_is_ON_for_probe ( Probe_index ) then begin
   {
     new_date = Global_results->get_next_date(Probe_index,
                                              current_compact_date[Probe_index],
                                              /LINK_MODE)

     for (probe_i=0..2) do if probe_is_active[probe_i] then
     {
       soon_visualized_SearchResults[probe_i,*] =
          Global_results->get_compact_searchresults(probe_i, new_date,
                                                    visualized_SearchResults)
       self.current_compact_date[Probe_i] = new_date
     }
   }
   else begin //not in mode compact but in linked mode
   {
     for (probe_i=0..2) do if probe_is_active[probe_i] then
     soon_visualized_SearchResults[probe_i,*] =
     Global_results->get_next_searchresults(visualized_SearchResults,
     visu_gui[probe_i].instrument_is_Selected)
   }
   endelse

   VISUALIZE_CHANGED_RESULTS ( soon_visualized_SearchResults )
```

5.10/ FESTIVAL_DATA_ENGINE::PREVIOUS_SET_REQUESTED

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] PREVIOUS_SET_REQUESTED (Probe_index)

Description :
The FESTIVAL_DATA_ENGINE::Previous_Set_Requested function is called when the user clicks on the PREVIOUS button. It finds the previous set of SearchResult and data that is needed by the VISU_GUI that makes the request (or by all VISU_GUI with available data in link mode).

Return Value :
The FESTIVAL_DATA_ENGINE::Previous_Set_Requested function method returns 1 if successful, or 0 otherwise.

Arguments :

Probe_index :
The index of the probe for which the request is made.

Keywords :

5.11/ FESTIVAL_DATA_ENGINE::VISUALIZE_CHANGED_RESULTS

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] VISUALIZE_CHANGED_RESULTS (soon_visualized_SearchResults, ALL_ACTIVE_PROBES=all_active_probes)

Description :
The FESTIVAL_DATA_ENGINE::Visualize_Changed_Results function is called once a new set of results to be visualized has been determined (by the NEXT/PREVIOUS_SET_REQUESTED, SET_COMPACT_MODE, SET_LINK_MODE, SEARCHRESULT_CLICKED_BY_USER functions, etc).

Return Value :
The FESTIVAL_DATA_ENGINE::Visualize_Changed_Results function method returns 1 if successful, or 0 otherwise.

Arguments :
soon_visualized_SearchResults :
Array [Nb_Probes, MaxNb_of_instruments] of object references to the SearchResults that will be visualized.

Keywords :
ALL_ACTIVE_PROBES :
Set this keyword in order to compute data and redraw all visu_gui. This is used when the user clicks on a “SAVE AND REDRAW” button of the SELECTION_GUI because any user defined parameter may have been changed by the user before and all visu_gui may be concerned even if no searchresult has changed.

Algorithm :
```
Result=Obj->[FESTIVAL_DATA_ENGINE::]
VISUALIZE_CHANGED_RESULTS(soon_visualized_SearchResults,
                                        ALL_ACTIVE_PROBES=all_active_probes  )

   if keyword_set(all_active_probes) then
    { changed_probe = where (visu_gui_is_active) }
   else
   {changed_probe =  “soon_visualized_SearchResults NE visualized_SearchResults”}
   endelse

   nb_operations_for_progressbar = 0

   for each changed_probe
       nb_operations_for_progressbar =  nb_operations_for_progressbar +
        total(obj_valid(soon_visualized_SearchResults[changed_probe]))

   nb_operations_for_progressbar =  4 * nb_operations_for_progressbar

   progressbar -> reinitialise(NB_OPERATIONS_TO_DO=nb_operations_for_progressbar)

   for each changed_probe

      VISU_GUI[probe]->REMOVE_ALL_ELEMENTS

      for each new_result in soon_visualized_SearchResults[probe] (with the order
                                                    defined by the user preferences)

         instrument = new_result->GetProperty(/INSTRUMENT)

         DATA = FIND_OR_BUILD_DATA ( new_result,
                  DIFFERENCE=diff_requested_for_instrument[probe,instrument])

         visualized_data[probe,instrument]=DATA

         new_elt = DATA->CREATE_VISU_ELT()

         VISU_GUI[probe]-> add, new_elt

      endfor

      VISU_GUI[probe]-> “update and redraw”

      visualized_SearchResults[probe] = soon_visualized_SearchResults[probe]

   endfor
```

```
SELECTION_GUI->HIGHLIGHT_VISUALIZED_SEARCHRESULTS( visualized_SearchResults )
```

5.12/ FESTIVAL_DATA_ENGINE::SEARCHRESULT_CLICKED_BY_USER

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] SEARCHRESULT_CLICKED_BY_USER (SearchResult)

Description :
The FESTIVAL_DATA_ENGINE::SearchResult_Clicked_by_User function is called when the user selects any line of SearchResult in the SELECTION_GUI. A new set of SearchResults is build by the function, depending on the mode (compact or not, linked or not) and the results are visualized.

Return Value :
The FESTIVAL_DATA_ENGINE:: SearchResult_Clicked_by_User function method returns 1 if successful, or 0 otherwise.

Arguments :
SearchResult :
object reference to the SearchResults that is requested to be visualized.

Keywords :

Algorithm :

Result=Obj->[FESTIVAL_DATA_ENGINE::]
SEARCHRESULT_CLICKED_BY_USER(SearchResult)

```
soon_visualized_SearchResults = visualized_SearchResults

SearchResult->GetProperty, DATE=date, PROBE=probe

if compact_mode_is_ON_for_probe ( Probe ) then begin
{
  new_date = date

  if linked_mode_is_ON then begin
  {

   for (probe_i=0..2) do if probe_is_active[probe_i] then
   {
    soon_visualized_SearchResults[probe_i,*] =
       Global_results->get_compact_searchresults(probe_i, new_date,
                                                 visualized_SearchResults)
    self.current_compact_date[Probe_i] = new_date
   }


  } else begin
  {

    soon_visualized_SearchResults[probe,*] =
       Global_results->get_compact_searchresults(probe, new_date,
                                                 visualized_SearchResults)
        self.current_compact_date[probe] = new_date
  }
}
else begin
{
   soon_visualized_SearchResults =
       Global_results->get_searchresults_when_selecting_result(
         visualized_SearchResults, selected_Searchresult)

} endelse

VISUALIZE_CHANGED_RESULTS ( soon_visualized_SearchResults )
```

5.13/ FESTIVAL_DATA_ENGINE::SET_COMPACT_MODE

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] Set_Compact_Mode (Probe, SWITCH_OFF=switch_off)

Description :
The FESTIVAL_DATA_ENGINE::Set_Compact_Mode function is called when the user switches the compact mode value by clicking on the “Compact mode” button in the VISU_GUI. A new set of SearchResults is built by the function, depending on the mode (compact or not, linked or not) and the results are visualized. Default is to switch on the compact mode, use the SWITCH_OFF keyword to set it off.

Return Value :
The FESTIVAL_DATA_ENGINE::Set_Compact_Mode function method returns 1 if successful, or 0 otherwise.

Arguments :
Probe :
Index of the probe that has requested to switch the compact mode value.

Keywords :
SWITCH_OFF :
Keyword to be used to exit the compact mode.

Algorithm :

Result=Obj->[FESTIVAL_DATA_ENGINE::]
SET_COMPACT_MODE (Probe, SWITCH_OFF = switch_off)

```
soon_visualized_SearchResults = visualized_SearchResults

if (not compact_mode_is_ON_for_probe ( Probe )) AND not set_keyword(switch_off)
then begin
{
  date = GLOBAL_RESULTS->get_oldest_date_of_selection(
                                            probe, visualized_SearchResults )

  if linked_mode_is_ON then begin
  {
   for (probe_i=0..2) do if probe_is_active[probe_i] then
   {
    soon_visualized_SearchResults[probe_i,*] =
       Global_results->get_compact_searchresults(probe_i, date,
                                                visualized_SearchResults)
    self.current_compact_date[Probe_i] = date
    VISU_GUI[Probe_i]->SetProperty, COMPACT_MODE=TRUE
    compact_mode_is_ON_for_probe (Probe_i)= TRUE
   }
  } else begin
  {

    soon_visualized_SearchResults[probe,*] =
       Global_results->get_compact_searchresults(probe, date,
                                                visualized_SearchResults)
    self.current_compact_date[probe] = date
    VISU_GUI[Probe]->SetProperty, COMPACT_MODE=TRUE
```

```
      compact_mode_is_ON_for_probe (Probe)= TRUE
    }

    // if compact mode is entered, the data will change so update everything:

    VISUALIZE_CHANGED_RESULTS ( soon_visualized_SearchResults )

  }
  else begin
  if compact_mode_is_ON_for_probe ( Probe )) AND set_keyword(switch_off)

 // the user wants to exit the compact mode, this implies no change in the data
 // that is visualized, just the "compact mode" button in the VISU_GUI
  {
    if linked_mode_is_ON then begin
    {
     for (probe_i=0..2) do if probe_is_active[probe_i] then
     {
       VISU_GUI[Probe_i]->SetProperty, COMPACT_MODE=FALSE
       compact_mode_is_ON_for_probe (Probe_i)= FALSE
     }
    } else begin
    {
      VISU_GUI[Probe]->SetProperty, COMPACT_MODE=FALSE
      compact_mode_is_ON_for_probe (Probe)= FALSE
    }

  } endelse
```

5.14/ FESTIVAL_DATA_ENGINE::SET_LINK_MODE

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] Set_Link_Mode (Probe, SWITCH_OFF=switch_off)

Description :
The FESTIVAL_DATA_ENGINE::Set_Link_Mode function is called when the user switches the link mode value by clicking on the "Link mode" button in a VISU_GUI. A new set of SearchResults is build by the function, depending on the mode (compact or not, linked or not) and the results are visualized. Default is to switch on the link mode, use the SWITCH_OFF keyword to set is off.

Return Value :
The FESTIVAL_DATA_ENGINE::Set_Link_Mode function method returns 1 if successful, or 0 otherwise.

Arguments :

Probe :
Index of the probe that has requested to switch the compact mode value.

Keywords :
SWITCH_OFF :
Keyword to be used to exit the link mode.

Algorithm :

```
Result=Obj->[FESTIVAL_DATA_ENGINE::]SET_LINK_MODE (Probe, SWITCH_OFF = switch_off)

   soon_visualized_SearchResults = visualized_SearchResults

   // The only case where new data has to be computed and VISU_GUI will need to be
   // redrawn is the following :

   if not set_keyword(switch_off) and if compact_mode_is_ON_for_probe[Probe]
   then begin
   {
     // the visu_gui that generated the link mode request was in compact mode
     // so that all visu_gui need to switch in compact mode using the current
     // date used by this visu_GUI

     date = self.current_compact_date[Probe]

     for (probe_i=0..2) do if probe_is_active[probe_i] then
     {
       soon_visualized_SearchResults[probe_i,*] =
          Global_results->get_compact_searchresults(probe_i, date,
                                                    visualized_SearchResults)
       self.current_compact_date[Probe_i] = date
       VISU_GUI[Probe_i]->SetProperty, COMPACT_MODE=TRUE, LINK_MODE=TRUE
       compact_mode_is_ON_for_probe (Probe_i)= TRUE
     }
     link_mode_is_ON = TRUE
     VISUALIZE_CHANGED_RESULTS ( soon_visualized_SearchResults )

   }
   else begin
   {
    link_mode_is_ON = NOT( set_keyword(switch_off) )

    for (probe_i=0..2) do if probe_is_active[probe_i] then
     {
       VISU_GUI[Probe_i]->SetProperty, LINK_MODE=link_mode_is_ON
     }

   } endelse
```

5.15/ FESTIVAL_DATA_ENGINE::VISU_GUI_CLOSED_BY_USER

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] Visu_GUI_Closed_by_User (Probe)

Description :
The FESTIVAL_DATA_ENGINE::Visu_GUI_Closed_by_User function is called when the user closes the widget window associated to a VISU_GUI object. The function updates the visualized_Searchresults property and informs the SELECTION_GUI of this change (so that no line of this probe is highlighted).

Return Value :
The FESTIVAL_DATA_ENGINE::Visu_GUI_Closed_by_User function method returns 1 if successful, or 0 otherwise.

Arguments :
Probe :
Index of the probe which VISU_GUI has been closed.

Keywords :

Algorithm :

```
Result=Obj->[FESTIVAL_DATA_ENGINE::]VISU_GUI_CLOSED_BY_USER (Probe)
{
   visualized_SearchResults [Probe, *]= "OBJ_NULL" (=OBJ_NEW())

   SELECTION_GUI->HIGHLIGHT_VISUALIZED_SEARCHRESULTS( visualized_SearchResults )
}
```

5.16/ FESTIVAL_DATA_ENGINE::LATEST_IMAGES_CLICKED

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] Latest_Images_Clicked ( )

Description :
The FESTIVAL_DATA_ENGINE::Latest_Images_Clicked function is called when the user clicks on the "LATEST IMAGES" button in the SELECTION_GUI. The function will call the method GET_LIST_OF_LATEST_IMAGES of the SELECTION_GUI and will compute and visualize the latest data of all instruments and all probes. The visualized data are highlighted in the

SELECTION_GUI. Compact and linked mode is also set.

Return Value :
The FESTIVAL_DATA_ENGINE::Latest_Images_Clicked function method returns 1 if successful, or 0 otherwise.

Arguments :

Keywords :

Algorithm :

```
Result=Obj->[FESTIVAL_DATA_ENGINE::]LATEST_IMAGES_CLICKED ()
{
  SELECTION_GUI-> GET_LIST_OF_LATEST_IMAGES (
            LAST_IMAGE_DATE=last_image_date,
            PROBE_IS_ACTIVE=probe_is_active,
            DATA_FOR_INSTRUMENT_IS_AVAILABLE=data_for_instrument_is_available)

  // As in any situation where a new query has been done, see in the HISTORY STACK
  // what DATA can still be linked to a SearchResult object.

  UPDATE_HISTORY_STACK_AFTER_NEW_QUERY ()

  // Properties of the Data Engine need to be updated (visualized_SearchResult)
  UPDATE_DATA_ENGINE_AFTER_NEW_QUERY ()

  // The available instruments may have changed, so that VISU_GUI widgets have to
  // be updated and images redrawn (some may have disappeared in the selection_gui)
  UPDATE_VISU_GUI_AFTER_NEW_QUERY ()

  link_mode_is_ON = TRUE

  for (probe_i=0..2) do if probe_is_active[probe_i] then
      {
       soon_visualized_SearchResults[probe_i,*] =
          Global_results->get_compact_searchresults(probe_i, last_image_date,
                                                    visualized_SearchResults)
       self.current_compact_date[Probe_i] = date
       VISU_GUI[Probe_i]->SetProperty, COMPACT_MODE=TRUE, LINK_MODE=TRUE
       compact_mode_is_ON_for_probe (Probe_i)= TRUE
      }

  VISUALIZE_CHANGED_RESULTS ( soon_visualized_SearchResults )

}
```

5.17/ FESTIVAL_DATA_ENGINE::NEW_SEARCH_CLICKED

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] New_Search_Clicked (Probe_searched)

Description :
The FESTIVAL_DATA_ENGINE::New_Search_Clicked function is called when the user clicks on the "Search" button in the SELECTION_GUI. The function will call the method MAKE_SEARCH of the SELECTION_GUI and will compute and visualize the data of all instruments and all probes (if autodisplay is activated). The visualized data are highlighted in the SELECTION_GUI.

Return Value :
The FESTIVAL_DATA_ENGINE::New_Search_Clicked function method returns 1 if successful, or 0 otherwise.

Arguments :
Probe_searched :
The index of the probe which "Search Button" has been clicked

Keywords :

Algorithm :

```
Result=Obj->[FESTIVAL_DATA_ENGINE::]NEW_SEARCH_CLICKED (Probe_searched)
{
 SELECTION_GUI-> MAKE_SEARCH (
            FIRST_IMAGE_DATE=first_image_date, //an array of 3 dates (1 per probe)
            PROBE_IS_ACTIVE=probe_is_active,
            DATA_FOR_INSTRUMENT_IS_AVAILABLE=data_for_instrument_is_available,
            FIRST_RESULT_OF_EACH_INSTRUMENT=first_result_of_each_instrument)

  // As in any situation where a new query has been done, see in the HISTORY STACK
  // what DATA can still be linked to a SearchResult object.

 UPDATE_HISTORY_STACK_AFTER_NEW_QUERY ()

  // Properties of the Data Engine need to be updated (visualized_SearchResult)
 UPDATE_DATA_ENGINE_AFTER_NEW_QUERY ()

  // The available instruments may have changed, so that VISU_GUI widgets have to
  // be updated and images redrawn (some may have disappeared in the selection_gui)
 UPDATE_VISU_GUI_AFTER_NEW_QUERY ()

 if (selection_gui.autodisplay_is_ON) then begin
  {
  soon_visualized_SearchResults = visualized_SearchResults
```

```
   if not linked_mode_is_ON then
   {
    for (probe_i=0..2) do {
     if probe_is_active[probe_i] then
     {
       if compact_mode_is_ON_for_probe( probe_i ) then begin
       {
        new_date = first_image_date[probe_i]

        soon_visualized_SearchResults[probe_i,*] =
          Global_results->get_compact_searchresults(probe_i, new_date,
                                                   visualized_SearchResults)
        self.current_compact_date[probe_index] = new_date
       }
       else begin //probe_i is not in mode compact and not in link mode
       {
        soon_visualized_SearchResults[probe_i,*] =
                              first_result_of_each_instrument[probe_i,*]
       }
       endelse
     } // end of the test if probe_is_active[probe_i]
    } endfor
   }
   else begin // linked mode is ON
   {
    if compact_mode_is_ON_for_probe (Probe_searched) then begin
    {
      new_date = min(first_image_date)  // i.e. the smallest of the 3 dates

      for (probe_i=0..2) do if probe_is_active[probe_i] then
      {
        soon_visualized_SearchResults[probe_i,*] =
           Global_results->get_compact_searchresults(probe_i, new_date,
                                                   visualized_SearchResults)
        self.current_compact_date[Probe_i] = new_date
      }
    }
    else begin //not in mode compact but in linked mode
    {
      for (probe_i=0..2) do if probe_is_active[probe_i] then
      soon_visualized_SearchResults[probe_i,*] =
                             first_result_of_each_instrument[probe_i,*]
    }
    endelse
   }
   endelse

   VISUALIZE_CHANGED_RESULTS ( soon_visualized_SearchResults )

 } // end of test if autodisplay_is_ON
```

5.18/ FESTIVAL_DATA_ENGINE::SAVE_AND_REDRAW_REQUESTED

Syntax :
Result = Obj -> [FESTIVAL_DATA_ENGINE::] SAVE_AND_REDRAW_REQUESTED ( )

Description :
The FESTIVAL_DATA_ENGINE::Save_And_Redraw_Requested function is called by the SELECTION_GUI when the user has clicked on the "SAVE AND REDRAW" button of any wizard of the selection_gui (palette wizard, projection wizard, masking wizard, enhancement wizard, filter wizard, order wizard).

Return Value :
The FESTIVAL_DATA_ENGINE::Save_and_Redraw_Requested function method returns 1 if successful, or 0 otherwise.

Arguments :

Keywords :

Algorithm :

```
Result=Obj->[FESTIVAL_DATA_ENGINE::]SAVE_AND_REDRAW_REQUESTED ( )

  VISUALIZE_CHANGED_RESULTS ( visualized_SearchResults, /ALL_ACTIVE_PROBES )
```

5.19/ FESTIVAL_DATA_ENGINE::GETPROPERTY

Syntax :
Result = Obj->[FESTIVAL_DATA_ENGINE::]GetProperty([,PROBE=probe][,INSTRUMENT=instrument][,IMAGE_INDEX=imageIndex][,ABSOLUTE_INDEX=absoluteIndex][,DATE=date][,FILESPECIFICATION=fileSpecification][,CALIBRATION=calibration][,PROJECTION_NAME=projName][,PROJECTION_LIMITS=projLimits][,FILTERING_SEQUENCE=filteringSequence][,ENHANCEMENT=enhancement][,FITSHEADER=fitsHeader])

Description :
The FESTIVAL_DATA_ENGINE::GETPROPERTY procedure method retrieves the value of a property for the FESTIVAL_DATA_ENGINE object.

Arguments :
None

Keywords :

PROBE
Set this keyword to a named variable that will contain the probe name for that image.
INSTRUMENT
Set this keyword to a named variable that will contain the instrument name for that image.
IMAGE_INDEX
Set this keyword to a named variable that will contain the relative image index.
ABSOLUTE_INDEX
Set this keyword to a named variable that will contain the absolute image index.
DATE
Set this keyword to a named variable that will contain the image acquisition date.
FILESPECIFICATION
Set this keyword to a named variable that will contain the full file specification for that image.
CALIBRATION
Returns the name of the calibration algorithm used to calibrate the raw image. Can be “None” if the image was not calibrated.
PROJECTION_NAME
Returns the name of the projection used. Can be “None” if the calibrated image was not projected.
PROJECTION_LIMITS
Returns a vector [latMin, lonMin, latMax, lonMax] giving the limits of the projection. Can be -1 if the calibrated image was not projected.
FILTERING_SEQUENCE
Returns a string array giving the filtering sequence applied to the projected image _pProjectedImage. Can be “None” if no filtering sequence was applied.
ENHANCEMENT
Returns an integer value giving the enhancement algorithm applied to the filtered image. Can be -1 f no enhancement algorithm was applied.
FITSHEADER
Returns a reference to the fits header pointer property _pFITSHeader of the FESTIVAL_DATA_ENGINE object.

5.20/ FESTIVAL_DATA_ENGINE::CLEANUP

Syntax :
OBJ_DESTROY, Obj
or
Obj->[FESTIVAL_DATA_ENGINE::]Cleanup   (In a lifecycle method only.)

Description :
The FESTIVAL_DATA_ENGINE::CLEANUP procedure method performs all cleanup on the object.

Arguments :
None

Keywords :
None