Gui Server Communication

1) LOGIN:-

Each client needs an access token to interact with the GUI server . This access token is typically the process-id of the client. This value is returned by the function

unsigned GUILogIn().

This value is used for future interaction with the server (e.g. message looping and API)

2) Message Looping

Communication with GUI server is two fold . This is achieved via a non-blocking message loop with a call to the function

bool GUIOutput(GMChannel,<hproc>,<struct smt *>);

This function returns 1/true if any output is available and 0/false otherwise.

If any message is received it is loaded in to the structure passed.

SCREEN COMPONENTS

AVAILABLE SCREEN COMPONENTS:

The GUI server has the following components. They are

1. *Window*: A window consists of a title bar , a cross-button and its body .

2. *Canvas*: A rectangular area on which texts, lines, circles, points , bars (in desired colors) can be drawn .

3. *Scrollable Canvas*: This is also a canvas with the above mentioned facilities but only a part of it can be seen at a time. One can navigate the whole canvas by using the arrow keys.This should be used when the canvas size required is greater than the maximum allowable size that can shown on the screen.

4. *EditBox*: This is a data entry box through which one can take text input from the user.

5. *Button*: **Having a text label.**

6. *Label*: **Displays a label.**

7. *CheckBox*: Contains a square area for check/uncheck mark and a text label

8. *Password Box:* This component is used to enter passwords.

9. *Menu*: Implements the standard pulldown menu with hotkeys.

1) Window functions

a) Window Creation:

The following API is used to create a window on the screen.

int Create_Window(unsigned id,unsigned pid ,unsigned short procid, int x,int y,int w,int h,int stat =1);

Parameters:

- id refers to the number identifying the particular window.
- pid refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows.
- procid is the value returned by GUILogin().

- x,y,w,h denote the starting x ,y positions ,width and height receptively.
- stat =1 if a title is required and 0 otherwise

Return Value:

- The function returns SS_SUCCESS on success else SS_FAILURE.

b) Display the Window:

To display the created window which is hidden by default we have the following API.
void Show_Window(unsigned id, unsigned short procid);
parameters:

- **i**d refers to the number identifying the particular window.
- procid is the value returned by GUILogin().

c) Hide the Window:

To hide a window we have the following API.
void Hide_Window (unsigned id, unsigned short procid);
parameters:

- **i**d refers to the number identifying the particular window.
- procid is the value returned by GUILogin()

d) Change the Window Title:

To change the window title which is 'window' by default use the following API

void Set_Window_Title(unsigned id,unsigned pid ,unsigned short procid,char *text);

parameters:

- *id* refers to the number identifying the particular window.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows
- *procid* is the value returned by GUILogin()

e) Deleting the Window:

To delete the window from the server the following API is used for deleting the window.

int Delete_Window(unsigned id , unsigned short procid);

parameters:

- **i**d refers to the number identifying the particular window.
- procid is the value returned by GUILogin()

returns values

SS_SUCCESS or SS_FAILURE.

2) Canvas functions

The canvas is the most versatile of all the components. It is possible to draw lines, text,

bars, circles etc in the desired colors here.

a) Creation:

A canvas is created using the following API.

int Create_Canvas(unsigned id,unsigned pid,unsigned short procid,int x,int y,int w,int h);

parameters:

- *id* refers to the number identifying the particular canvas.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows
- x,y: are the Left and Top screen coordinates of the canvas.
- w,h: are the width and height of the canvas.

Return value: SS_SUCCESS /SS_FAILURE .

b) Shape Drawing:

The following API are used to draw various lines and shapes in a canvas.

parameters:

- *id* refers to the number identifying the particular shape.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows
- *procid* is the value returned by GUILogin().

The coordinates that are used to draw are relative to the area in the Canvas.

The various shapes and their corresponding API are:

i*) Line***:**

To draw a line from the point (x1,y1) to (x2,y2) use:

void Draw_Line(unsigned id,unsigned pic ,unsigned short procid, int x1,int y1,int x2,int y2, int color);

ii) Bar:

To draw a bar from the point (x,y) with a width ‘w’ and height ‘h’ use:

void Draw_Bar(unsigned id,unsigned pic ,unsigned short procid ,int x,int y,int width,int height,int color);

void Draw_Bars(unsigned id,unsigned pid ,unsigned short procid, int tx[],int ty[],int bx[],int by[],int color[],char labels[][30],int count);

The above function is called to draw multiple bars.

iii) Pixel:

To plot a point with color ‘color’ at (x,y) on the canvas

void Draw_Pixel(unsigned id , unsigned pid ,unsigned procid, int x,int y,int color);

iv) Circle:

To draw a circle use:

void Draw_Circle(unsigned id , unsigned pid ,unsigned procid, int x, int y,int radius ,int color);

v)Text:

To draw text on the canvas with selected font we have

void Draw_Text (unsigned id,unsigned pid ,unsigned short procid,

int x,int y ,char *text);

In the above 'shape' functions x,y or(x1,x2,y1,y2) refer to the coordinates relative to the left upper corner of the canvas.

vi)Text Colors:

To change the textcolors (fc = foreground, bc = background) and the current font of the canvas we use

Set_Text_Colors(unsigned id,unsigned pid,unsigned short procid , unsigned fc,unsigned bc,unsigned fno);

The number of fonts available are MAX_FONT_NO = 30 and they are

Font No - 0 to 5 are "times Roman" pts 10, 12, 14, 16, 20, 24
6 to 11 are "times (Italic)" pts 10, 12, 14, 16, 20, 24
12 to 16 are "courier" pts 10, 12, 16, 20, 24
17 t o 20 are " arial (Italic)" pts 10, 14, 16, 24
21 to 24 are "impact" pts 12, 16, 20, 24
25 to 29 are " arial" pts 12, 14, 6, 29, 24

The fonts are identified by numbers 0 through 29, and the text colors are set in the range 1 – 255.

vii)Shifting APIs for canvas:

The following APIs are used to shift the contents of a Canvas by a desired amount to the left, right, top or bottom.

For shifting UP.

void Shift_Up_By (unsigned id, unsigned pid, unsigned short procid, unsigned dy, unsigned color);

For shifting Down

void Shift_Down_By (unsigned id,unsigned pid,unsigned short procid,unsigned dy, unsigned color);

For Shifting Right

void Shift_Right_By (unsigned id,unsigned pid,unsigned short procid,unsigned dx, unsigned color);

For Shifting Left

void Shift_Left_By (unsigned id,unsigned pid,unsigned short procid,unsigned dx, unsigned color);

Here dx, and dy are the amounts by which the canvas contents are to be shifted horizontally and vertically respectively. The cleared portion is filled up by the color value in the parameter 'color'.

viii)To set the keyboard messages on/off (default is off)

The following API enables/disables keyboard messages from the server when focus is on the canvas.

void Set_Canvas_Key_State(unsigned id,unsigned pid, unsigned short procid,int keyState);

keyState = 1 enables while keyState = 0 disables the key flow

3) Scrollable Canvas Function

This component behaves much like a canvas (excepting the font selection which is fixed). However it defines a logical display area which is larger than the one which is available.
The user will have to move around using the arrow keys.

a) Creation:

The scrolling canvas is created by:

int Create_ScrollableCanvas (unsigned id, unsigned pid, unsigned short procid, int x, int y, int w, int h, int vw, int vh);

parameters

- *id* refers to the number identifying the particular canvas.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows
- x,y: are the Left and Top screen coordinates of the canvas.
- w,h: are the width and height of the canvas.
- vw,vh are the virtual width and height of the scrollable canvas.
- procid is the value returned by GUILogin().

b) Drawing.

The various shapes possible are

i) Line:

void Draw_LineSc(unsigned id,unsigned pid ,unsigned short procid, int x1, int y1, int x2, int y2, int color);

ii)Bar:

void Draw_BarSc(unsigned id,unsigned pid ,unsigned short procid, int x, int y,int width,int height,int color);

iii)Point:

void Draw_PixelSc(unsigned id , unsigned pid ,unsigned procid,int x, int y,int color);

iv)Circle:

void Draw_CircleSc(unsigned id , unsigned pid ,unsigned procid, int x, int y,int radius ,int color);

v)Text:

void Draw_TextSc(unsigned id,unsigned pid ,unsigned short procid, intx,int y ,char *text);

In the above 'shape' functions x,y or(x1,x2,y1,y2) refer to the coordinates relative to the left upper corner of the canvas considering from the start of the virtual screen.

c) Text Colors:

short Set_Text_Colors_Sc (unsigned id, unsigned pid, unsigned short procid , unsigned fc, unsigned bc);

used to set the foreground (fc) and background (bc) colors of the font.

d) Navigation:
On selecting the scrollable canvas the arrow keys may be used to move around the scrollable canvas.

e) To set the keyboard messages on/off (default is off)
The following API enables/disables keyboard messages from the server when focus is on the canvas.

void Set_Canvas_Key_State_Sc(unsigned id,unsigned pid, unsigned short procid,int keyState);

keyState = 1 enables while keyState = 0 disables the key flow

4)TableEditor Function

Table Editor is a built in editor that can be used for editing data stored in rows and columns
The API to create the table editor is as follows

a) Creation:

short Create_Table_Editor(unsigned id , unsigned pid, unsigned short procid, int x, int y, char *rcfilename);

parameters

- *id* refers to the number identifying the particular table editor.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows
- x, y: denotes the starting x, y screen coordinates of the table editor in the parent window .

b) Saving:

To save the table editor data in a data file the following API should be used :

short Save_Table_Editor(unsigned id, unsigned pid,unsigned short procid, char *datafilename);

parameters

- *id* refers to the number identifying the particular editor.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows.
- "datafilename" is the string containing the name of the datafile in which table editor data would be stored .

5) Edit Box functions

The Edit Box is used to take in user text information (max 256 characters) .

a) Creation:

To create an Edit Box we use

int Create_EditBox(unsigned id,unsigned pid,unsigned short procid,int x, int y,int w,int h);

parameters

- *id* refers to the number identifying the particular edit box.
- *pid* refers to the parent of the window which should be 0 for normal windows ,and the parent window for modal windows.
- 'x' and 'y' are absolute screen coordinates
- 'w' and 'h' refer the width and height respectively.

Return Value:

The function returns SS_SUCCESS on success and SS_FAILURE otherwise.

b) Setting the Edit Box Text:

It is possible to set the displayed text of the edit box by

int Set_Text_EditBox(unsigned id,unsigned pid,unsigned short procid, char * text);

c) Retrieving the text in the Edit Box:

To get the text from the edit box to the client application use

int Get_Text (unsigned id, unsigned pid, unsigned short procid, char * Buffer);

parameters:

-Here id and pid refer to the identifiers denoting the edit box and its parent window.

- 'text' and 'Buffer' are the buffers (<256 char) from which text is set and kept respectively.

Return values

SS_FAILURE on failure and SS_SUCCESS otherwise

d) Password Box

This component is used to accept passwords from the user through a data entry box on the screen. Only the standard editing facilities are available and the text entered is replaced by asterisks.

i)*Creation:*

To create a Password Box we use the API

int Create_PasswordBox(unsigned id,unsigned pid,unsigned short procid,int x,int y,int w,int h);

parameters:

- 'x' and 'y' are absolute screen coordinates
- 'w' and 'h' refer the width and height respectively.

Return Value:

The function returns SS_SUCCESS on success and SS_FAILURE otherwise.

ii)Setting Text:

To set the password box to an initial value.

int Set_Text_PasswordBox (unsigned id,unsigned pid,unsigned short procid,char * text);

where 'text' refers to the text to be set

e) Retrieving the text in the Password Box:

To get the text from the password box to the client application use

int Get_Text (unsigned id, unsigned pid, unsigned short procid, char * Buffer);

where 'Buffer' refers to the array in which the text is to be kept.

6) Button functions

Buttons are window components whose responses are used to trigger events when they are selected and ENTER is pressed.

a) Creating a Button:

To create a button the following API is used .

short Create_Button(unsigned id,unsigned pid,unsigned short procid , int x,int y ,int w,int h,char *text);

parameters

id : identifier of the Button.
pid :identifier of the window in which the button is to be created.
Procid : the value returned by GUILogin();
x,y,w,h :denote the starting x ,y positions ,width and height repectively.
text : Label of the button (must be < 80 characters).

Return Value:

The API returns SS_SUCCESS on success and SS_FAILURE on Failure

b) Setting the button text:

int Set_Text_Button(unsigned id,unsigned pid,unsigned short procid,char * text);

7) Label functions

A label is used to display a static text on a window.

a) Creation:

To create a label the following function is used.

short Create_Label (unsigned id, unsigned pid, unsigned short procid ,int x, int y , int w, int h,char *text);

parameters

id: identifier of the label
pid: identifier of the parent window
procid: value returned by GUILogin();
text: the string (<80 char) that is to be displayed.

b) Changing the text:

To change the text we use the following API.

int Set_Text_Label (unsigned id, unsigned pid, unsigned short procid, char *text);

c) Set Button Color

To set button color

void Set_Buttob_Color (int color);

8) Check Box functions

For Check Boxes the following API can be used

a) CheckBox creation:

To create a new checkbox at absolute coordinates (x,y) with width 'w' and height 'h' we have the API

short Create_CheckBox(unsigned id,unsigned pid,unsigned short procid , int x,int y , int w, int h,char *text);

b)Getting and Setting the check box status:

To get check box status :-

short Get_CheckBox_Status(unsigned id,unsigned pid,unsigne short *pStatus);

To set check box status :-

short Set_CheckBox(unsigned id,unsigned pid,unsigned short procid, short Status);

Variable description:

id: refers to the identifier of the checkbox.
pid: refers to the identifier of its parent window.
Status: refers to the state 1 for checked 0 for Unchecked states.
Text: refers to label (should be <80 char) associated with the check box

Return values:- SS_FAILURE on failure and SS_SUCCESS otherwise

-

9) Menu Functions

The Menu that is available with the GUI server is like a standard menu supporting hotkeys and various levels of pulldown menus containing text labels .

Each menu is a static, i.e. menu items are not added or deleted dynamically.

It is assumed that the menu-bar is to be placed near the top of the window and there is sufficient space for each popup along the width within each window. This would obviously depend on the text length of the menu-item descriptors.

a) Creating a Menu:

Every menu is defined by passing the name of the file that contains the description of the menu through the following API.

short Create_Menu(unsigned id,unsigned pid,unsigned short procid ,int x,int y , int w, int h,char *filename);

Parameters

- id and pid refer to the identifier values of the menu and its parent window.
- x,y refer to the absolute screen coordinates of the upper left corner of the Menu.
- proid refers to the value returned by GUILogin().
- filename' refers to the file containing the menu definition.

b) To set a particular menuitem active or deactive:

To set particulat menu item active or deactive

void Set_Menu_Item(unsigned id,unsigned pid,unsigned short procid, unsigned Value,short State);

c)To set a particular page active or deactive:

To activate of deactivate a particular page

void Set_Menu_Page(unsigned id,unsigned pid,unsigned short procid,

unsigned Value,short State);

10)Miscellaneous API's

a) void Set_Component_State(unsigned id,unsigned pid,unsigned short procid,short State);

This API is used to set the state of an component in a window (except a menu).

b) short Set_Focus_To(unsigned id,unsigned pid,unsigned short procid);

Sets the focus to the component whose identifier value is id.
Function return SS_SUCCESS/SS_FAILURE .

c) short Delete_Window_Component(unsigned id,unsigned pid,unsigned short procid);

Deletes the component whose identifier value is id.
Function returns SS_SUCCESS/SS_FAILURE

d) unsigned Get_Focussed_Component(unsigned pid,unsigned short procid);

returns the identifier value of the currently focussed component in the window having identifier pid.

e) short Get_Next_Component_Info(unsigned id,unsigned pid,unsigned short procid,unsigned *nextid,unsigned *nexttype);

returns the identifier value of the component next to the component 'id' .The last field reserved.
Function returns SS_SUCCESS/SS_FAILURE

f) void Shutdown_GUI(void);

shuts down the gui server should be used judiciously.

g) void Set_Button_Colors(unsigned id,unsigned pid,unsigned short procid,unsigned FocusColor=0,unsigned ButtonColor=0,unsigned ActiveFontColor=0,unsigned InactiveFontColor=0);

To set the Button Color in respect to whether they have focus, active or deactive

h) void Write_Text(unsigned id,unsigned pid ,unsigned short procid,int x,int y,int fc,int bc,int fno,char *text);
void Write_Texts(unsigned id,unsigned pid ,unsigned short procid,int tx[],int ty[],int fg[],int bg[],int font[],char labels[][60],int count)

To write Text on Gui item(canvas)

parameters fc:- Foreground color
bc:- Background color
x,y starting coordinates of text
fno:- font of text
text:- string to write
tx,ty:- coordinate array
fg:- foreground color array
bg:- background color array

font:-font arry
labels:- Label array
count:- number of labels

i) void Print_Window(unsigned short procid,int mode)

This function is used to send screen pixels to a file , printer or to both.

Mode 1 :- To Printer
2 :- To file
3 :- Both

*__NOTE:- For all GUI commands,maximum response delay shall be less than 50 microseconds__ .