# SWLib Functions

Types of variables:

| | |
|---|---|
| int | The 32-bit integer; |
| double | The 64-bit float; |
| char * | Sequence of 8-bit bytes which are ending by zero byte. |

The functions, usually, returns «0» in case of successful end, and «-1» in case of any error (in common cases "-1" means, that the object has not been created or already removed together with a window). Format of transfer of parametres is stdcall.

## Control the image in a window manually:

1. Press the mouse left button to rotate the image as though it is concluded in sphere.
2. Press SHIFT key + left mouse button to move the image along a window.
3. The mouse wheel works as zoom.

## The basic functions:

### Create object

```
int SWCreateObject ()
```

Returns the object identifier which is more "-1".
It should be the first function for work with library.

### Load the grid and values in grid nodes

```
int SWLoadGrid (char *sFile),
```

Where `sFile` is a name of the file having following format (one figure in each line):
nColumns is a number of grid columns.
X1 is X coordinate of the first grid column.
X2 is X coordinate of the second grid column
...
XnColumns is X coordinate of last column of grid nodes.
nRows is a number of rows of a grid
Y1 is Y coordinate of the first grid row.
Y2 is Y coordinate of the second grid row
...
YnRows is Y coordinate of last grid row
V1,1 is a value in node with coordinates (X1, Y1)
V2,1 is a value in node with coordinates (X2, Y1)
…
VnColumns, 1 is a value in node with coordinates (XnColumns, Y1)
…
VnColumns, nRows is a value in node with coordinates (XnColumns, YnRows)
If instead of grid value there is NAN or not a figure this node is considered not active and where the surface is not shown.
Returns "-1" when cannot open a file.

### Set the grid and values in grid nodes

```
int SWSetGrid (int nColumns, int nRows, double *pdX, double *pdY, double *pdV,
int *pnT),
```

nColumns is a number of grid columns;
nRows is a number of grid rows;
pdX - an array of X coordinates, dimension (nColumns);
pdY – an array of Y coordinates , dimension (nRows);
pdV – an array of  grid values, dimension (nColumns * nRows);
pdT – an array of integer types of grid nodes, dimension (Columns * nRows). If 1 – the node is active, visible or 0 – not active, invisible).

### Load the coordinates and values in points

```
int SWLoadPoints (char *sFile)
```

Where `sFile` – a name of the file having following format (3 values in each line):
X1 Y1 V1
...
Xn Yn Vn
Number of lines is unlimited
Returns “-1” when cannot open a file.

### Set the coordinates and values in points

```
Int SWSetPoints (int nCount, double *pdX, double *pdY, double *pdV)
```

Where
nCount – number of points;
pdX – an array of X coordinates of points, dimension (nCount);
pdY – an array of Y  coordinates of points dimension (nCount);
pdV – an array of grid values in points dimension (nCount).

### Load the bottom picture

```
int SWLoadPicture (char *sFile)
```

Where `sFile` – the name of a file of format GIF, which corners correspond to grid corners.
Returns “-1” when cannot open a file.

### Create and show 3D window

```
int SWShowWindow ()
```

Before this function the object should be created and the grid is entered at least.
At first call this function does not return the control, while the new window will not be created and initialized.
The repeated call of function only shows an existing window.

### To make a window active (to move to foreground)

```
int SWActivateWindow ()
```

### To hide a 3D window

```
int SWHideWindow ()
```

### To change position and the sizes of a window

```
int SWMoveWindow (int x, int y, int nWidth, int nHeight),
```
where x, y are coordinates of the top left corner of a window, `nWidth` – width of a window, `nHeight` – window height.

### To receive position and the sizes of a window

```
int SWGetWindowPosition (int *px, int *py, int *pnWidth, int *pnHeight);
```
where x, y are pointers to integer values for the coordinates of the top left corner of a window, `nWidth` – for width of a window, `nHeight` – for window height.

### Definition callback functions

```
int SWSetCallbackfunction (int (*pFunct) (int n))
```
Callback function will be called at closing or window hiding. For example:
Argument n is 1 if the window was hided, n = 0 if the object was deleted (window `was` closed):

```
int          callbackfunction (int n)
{
             if (n)
             {
                    cout <<"Window is hided";
             }
             else
             {
                    cout <<"Object is deleted";
             }
             return 0;
}
SWSetCallbackfunction (callbackfunction);
```

### Close the window and delete the object

```
int SWClose ()
```
The object will delete as well at manual closing of a window.

## Draw options:

### Change the full window mode

```
int SWSetFullWindow (int n)
```
Where n = 1 – set fast full OpenGL window mode, or n = 0 – the best text quality, but slowly.

To get current setting:
```
int SWGetFullWindow (int *pn)
```

### Redraw objects in window (redraw scene)

```
int SWRedraw ()
```

### To show or not the points in window

```
int SWPointsSetShow (int n)
```
Where n = 1 – to show, or n = 0 – not to show.
The default value is 1, if they have been loaded.

To get current setting:
```
int SWPointsGetShow (int *pn)
```

### To draw the level contours on surface

```
int SWContoursSetShow (int n)
```
Where n = 1 – to draw, or n = 0 – not to draw.
The default value is 1.

To get current setting:
```
int SWContoursGetShow (int *pn)
```

### To show or not coordinate axes

```
int SWAxesSetShow (int n)
```
Where n = 1 – to show, or n = 0 – not to show.
The default value is 1.

To get current setting:
```
int SWAxesGetShow (int *pn)
```

### To draw or not the bottom picture

```
int SWBaseImageSetShow (int n)
```
Where n = 1 – to draw, or n = 0 – not to draw.
The default value is 1, if it has been loaded.

To get current mode:
```
int SWBaseImageGetShow (int *pn)
```

### To set a space of the bottom picture from 3-D surface

```
int SWBaseImageSetOffset (double dO)
```
Where dZ – space factor in shares from большей the parties 3-hmernogo images.
The default value is 0.1.

Get the current value:
```
int SWBaseImageGetOffset (double *pdO)
```

### To rotate the image

```
int SWRotate (double dX, double dY, double dZ)
```
Where dX, dY, dZ are the angles of rotation X, Y, Z axes in degrees. These angles not from current position, but from initial position (0,0,0).
The default values are (0,0,0).

To get the current angles in degree:
```
int SWGetRotate (double *pdX, double *pdY, double *pdZ)
```

### The image zoom

```
int SWSetZoom (double d)
```
Where d is zoom factor. Can be more than 1 or less 1, but it is more than zero.
The default value is 1. It refer to initial approach of the camera to the object calculated so that, it was located in a window.

To get the current zoom factor:
```
int SWGetZoom (double *pd)
```

### To change a mouse wheel zoom step

```
int SWSetZoomFactor (double d)
```
Where d is zoom factor step. Can be more than 1, and it is less 1, but it is more than zero.
The default value is 1 (one wheel division scale image for 1/100 its lengths).

To get the current zoom step:
```
int SWGetZoomFactor (double *pd)
```

### To move the surface along the screen

```
int SWTranslate (int nXFrom, int nYFrom, int nXTo, int nYTo)
```
Analogue move the surface by mouse from a window point (`nXFrom, nYFrom`) to a point (`nXTo, nYTo`)

### To move the image to new position

```
int SWSetTranslation (double dX, double dY);
```
Move the surface center from an initial position along vector (`dX, dY`) in units of XY coordinates.

Get the current value:
```
int SWGetTranslation (double *pdX, double *pdY)
```

### Return to zero position

```
int SWSetInitPosition ()
```

### To establish background color of image

```
int SWSetBackgroundColor (int r, int g, int b);
```
Where r, g, b are red, green and blue color components. They can change from 0 to 255. The default value is white color - `(255, 255, 255)`

Get the current background color:
```
int SWGetBackgroundColor (int *pr, int *pg, int *pb);
```

### To set Z stretch factor

```
int SWSetZStretch (double dZ)
```
Where dZ is Z stretch factor.
This factor defines the rate of XY coordinate units Z value units.
The default value is set that the visual rate of height to length as 1/2.

Get the current Z stretch factor:
```
int SWGetZStretch (double *pdZ)
```
where pdZ is the pointer to double value where the Z stretch factor will stored.

### To set extension factor on Y

```
int SWSetYStretch (double dY)
```
Where dY is Y stretch factor.

Get the current Y stretch factor:
```
int SWGetYStretch (double *pdY)
```

### To set a thickness and color of contours on 3-D surfaces

```
int SWSetContours (double d, int r, int g, int b)
```
Where d is a thickness of contour lines from 1 (by default) to 10. r, g, b are red, green and blue color components. They can change from 0 to 255. The default value is the black color.

Get the current values:
```
int SWGetContours (double *pd, int *pr, int *pg, int *pb)
```

### To set a thickness and color of point lines

```
int SWPointsSetLines (double d, int r, int g, int b)
```
Where d is a thickness of contour lines from 1 (by default) to 10. r, g, b are red, green and blue color components. They can change from 0 to 255. The default value is the black color.

Get the current values:
```
int SWPointsGetLines (double *pd, int *pr, int *pg, int *pb)
```

### Change projection

```
int SWSetPerspective (int n)
```
Where n = 1 is a perspective projection, or n = 0 – orthogonal.
The default mode is perspective.

Get the current mode:
```
int SWGetPerspective (int *pn)
```

### To change the perspective projection angle

```
int SWSetPerspectiveAngle (double d);
```
Where d is angle from 8 to 75 degrees.
The default values 15 degrees.

Get the current value:
```
int SWGetPerspectiveAngle (double *pd);
```

### To change cursors

```
int SWSetCursor (int n, HCURSOR cur)
```

Where n = 1 – the cursor for moving, or n = 0 – for rotation. To return to cursors by default, it is enough to set cur as NULL (Zero value).

## Save image:

### To define the sizes of a saved picture

```
int SWImageSetSize (int nX, int nY)
```
Where nX and nY are the sizes of saved picture in pixels. The default size is 800 x 800 pixels. The saved image size is indepent from the window size, but image proportions do not vary.

### Save 3-D image in BMP file

```
int SWImageSave (char *sFile)
```
Where `sFile` is BMP file name (24 bits pixel format).

## The colour scale properties:

After changing color bar properties it is necessary call `SWColorBarRedraw()` command to update the image.

### To show or hide the colour bar

```
int SWColorBarSetShow (int n)
```
Where n = 1 is to show, or n = 0 is to hide.
The default value is 1.

Get the current value:
```
int SWColorBarGetShow (int *pn)
```

### To set number of colors level in the colour bar

```
int SWColorBarSetColorsCount (int n)
```
Where n is number of colors (levels).
The default value is about 10, it is calculated from values of a grid.

Get the current value:
```
int SWColorBarGetColorsCount ()
```

### To define the color for minimum level in the color bar

```
int SWColorBarSetMinColor (int r, int g, int b);
```
Where r, g, b are red, green and blue color components. They can change from 0 to 255. The default values are `(255, 255, 0)`

Get the current color for min value:
```
int SWColorBarGetMinColor (int *pr, int *pg, int *pb)
```

### To define the color for a maximum level in the colour bar

```
int SWColorBarSetMaxColor (int r, int g, int b)
```
Where r, g, b are red, green and blue color components. They can change from 0 to 255. The default values are `(255, 125, 0)`

Get the current color for max value:
```
int SWColorBarGetMaxColor (int *pr, int *pg, int *pb)
```

### Set color for a range of levels in the colour bar

```
int SWColorBarSetColors (int n1, int r1, int g1, int b1, int n2, int r2, int g2,
int b2)
```
Where r1, g1, b1 are red, green and blue color components for n1-th level, r2, g2, b2 are red, green and blue color components for n2-th level. Each component can vary from 0 to 255. If n1 and n2 is the equal, color for one level is set. If n1 < n2 colors are interpolated in this range.

Get the current color for n-th color bar level:
```
int SWColorBarSetColors (int n, int *pr, int *pg, int *pb)
```

### Set the font and its size for digital marks in the color bar

```
int SWColorBarSetFont (int nSize, char *sFontName)
```
Where `sFontName` is a font name, `nSize` is its size.
In Windows system the font order is made through structure `LOGFONT`. This function set the some members of structure the specified values (**`lfHeight =` `nSize`**, **`lfFaceName =` `sFontName`**). Windows selects fonts, from what are in presence**.** It is recommended to use TrueType fonts.
The default values are (**`lfHeight = 60`**, **`lfFaceName = "Arial"`**).
Also, it is possible to transfer the pointer to prepared structure `LOGFONT`
```
int SWColorBarSetLOGFONT (LOGFONT *plf)
```

To copy current structure LOGFONT to allocated structure:
```
int SWColorBarGetLOGFONT (LOGFONT *plf)
```

### Set the number of decimal space after a point for digital marks in the colour bar

```
int SWColorBarSetDecimalPlaces (int n)
```
Where n is number of spaces.
The default value is 2 spaces after a point.

Get the current value:
```
int SWColorBarGetDecimalPlaces (int *pn)
```

### Set the fixed range of values in the color bar

```
int SWColorBarSetMinMax (int n, double dMin, double dMax);
```
Where n is 1 or 0.
By default the range is calculated from values in grid nodes. At n = 1, it is set new range from `dMin` to `dMax`. At n = 0, return to the calculated values. The surface out of this range is not displayed.

Get the current values:

```
int SWColorBarGetMinMax (int *pn, double *pdMin, double *pdMax);
```

### Set the top and bottom margins for the colour bar

```
int SWColorBarSetVerticalMargins (double dUp, double dDown);
```
Where `dUp` and `dDown` are top and the bottom margins which define a vertical bar size. Value of this parameters is set in fractions of height of a window or the image.
The default values are `dUp` = 0.4, `dDown` = 0.15.

Get the current values:
```
int SWColorBarGetVerticalMargins (double *pdUp, double *pdDown);
```

### Set the color of cell borders in the color bar

```
int SWColorBarSetCellColor (int n, int r, int g, int b)
```
If n is 1, then to draw borders of color cells with color (r, g, b), if n is 0, then to draw cells without borders.
The default value is 1.

Get the current values:
```
int SWColorBarGetCellColor (int *pn, int *pr, int *pg, int *pb)
```

### Set the background color for the color bar

```
int SWColorBarSetBackgroundColor (int r, int g, int b)
```
Where r, g, b are red, green and blue color components. They can change from 0 to 255. The default values are `(255, 255, 255)`.

Get the current color:
```
int SWColorBarGetBackgroundColor (int *pr, int *pg, int *pb)
```

### To set an order of values in the color bar

```
int SWColorBarSetUpDown (int n)
```
If n is 0, then the maximum value is located at top, if n is 1 – at bottom.
The default value is 0

Get the current value:
```
int SWColorBarGetUpDown (int *pn)
```

### To set the left or right location of the colour bar

```
int SWColorBarSetPosition (int n);
```
Where if n is 0, the location is at left from surface, if n is 1 – at right. The default value is 1.

Get the current value:
```
int SWColorBarGetPosition (int *pn)
```

### Set the title for the color bar

```
int SWColorBarSetTitle (char *sName)
```
Where `sName` is the title.
The default title is empty.

To copy the current title to string:
```
int SWColorBarGetTitle (char *sName)
```
If the text has not been set, return “-1”.

### To draw or not title for the colour bar

```
int SWColorBarSetTitleShow (int n)
```
Where n is 1, then to draw, if n is 0, then not to draw.
The default value is 1, if the title has been set.

Get the current value:
```
int SWColorBarGetTitleShow (int *pn)
```

### Set the font and its size for color bar title

```
int SWColorBarSetTitleFont (int nSize, char *sFontName)
```
Where `sFontName` is a font name, `nSize` is its size.
In Windows system the font order is made through structure `LOGFONT`. This function set the some members of structure the specified values (**lfHeight =** `nSize`, **lfFaceName =** `sFontName`). Windows selects fonts, from what are in presence. It is recommended to use TrueType fonts.
The default values are (**lfHeight = 60**, **lfFaceName = “Arial”**).
Also, it is possible to transfer the pointer to prepared structure `LOGFONT`
```
int SWColorBarSetTitleLOGFONT (LOGFONT *plf)
```

To copy current structure LOGFONT in plf:
```
int SWColorBarGetTitleLOGFONT (LOGFONT *plf)
```
If the font has not been set obviously, возвращет “-1”

### Set the range levels of a color bar to show on 3-D surface

```
int SWColorBarSetLevelsLimit (int nZ1, int nZ2)
```
Where nZ1 < nZ2 are the indices of levels in color bar which should be drawn, out of it range – not to draw.

Get the current values:
```
int SWColorBarGetLevelsLimit (int *pnZ1, int *pnZ2)
```

### Redraw the 3-D surface and the color bar after it was modified.

```
int SWColorBarRedraw ()
```

## Options for image title:

After changing title properties it is necessary call `SWTitleRedraw ()` command to update the title in window.

### Set the image title

```
int SWTitleSetText (int nLine, char *sText);
```
Where `nLine` is the title line (from 0 to 2), `sText` is the text.

To copy the current text of the specified line:
```
int SWTitleGetText (int nLine, char *sText);
```

### To set the font for title

```
int SWTitleSetFont (int nLine, int nSize, char *sFontName);
```
Where `nLine` is the title line (from 0 to 2), `sFontName is the` font name, `nSize` is its size.
In Windows system the font order is made through structure `LOGFONT`. This function set the some members of structure the specified values (**`lfHeight =` `nSize`**, **`lfFaceName =` `sFontName`**). Windows selects fonts, from what are in presence. It is recommended to use TrueType fonts.
The default values are for the first line (**`lfHeight = 120`**, **`lfFaceName = "Arial"`**), for the second and third line (**`lfHeight = 60`**, **`lfFaceName = "Arial"`**).
Also, it is possible to transfer the pointer to prepared structure `LOGFONT`
```
int SWTitleSetLOGFONT (int nLine, LOGFONT *plf)
```

To copy current structure LOGFONT in plf:
```
int SWTitleGetLOGFONT (int nLine, LOGFONT *plf)
```
Returns "-1" if the font for this line has not been obviously set.

### Set the background color for the image title

```
int SWTitleSetBackgroundColor (int r, int g, int b);
```
Where r, g, b are red, green and blue color components. They can change from 0 to 255. The default values are `(255, 255, 255)`.

To receive current color:
```
int SWTitleGetBackgroundColor (int *pr, int *pg, int *pb);
```

### To show or not the image title

```
int SWTitleSetShow (int n)
```
If n is 1, then draw, if n is 0, then not draw.
The default value is 1, if the title has been set.

Get the current value:
```
int SWTitleGetShow (int *pn)
```

### Redraw the image title

```
int SWTitleRedraw ()
```

## Options for the grid:

### Get the grid size

```
int SWGridGetSize (int *pnX, int *pnY)
```
Where `pnX`, `pnY` are the pointers to variables where the sizes of a grid will locate.

### Get the minimum and maximum grid values

```
int SWGridGetMinMaxValues (double *pdMin, double *pdMax)
```
Where `pdMin`, `pdMax` are the pointers to variables where minumum and maximum values will locate.

### Set the column and row grid ranges to show in window (slices)

```
int SWGridSetSlices (int nX1, int nX2, int nY1, int nY2)
```
Where nX1 < nX2 are the columns indices of the grid should be drawn, out of it range should not drawn;
nY1 < nY2 are rows indices of the grid which should be drawn.
Function `SWRedraw ()` should be call to redraw scene.

Get the current values:
```
int SWGridGetSlices (int *pnX1, int *pnX2, int *pnY1, int *pnY2);
```

## Options for a box around 3-D surface:

### To draw or not the box

```
int SWBoxSetShow (int n)
```
If n is 1 then to draw, else not to draw.
The default value is 0.

Get the current value:
```
int SWBoxGetShow (int *pn)
```

### To draw or not the digital marks on box

```
int SWBoxTicksSetShow (int n)
```
If n is 1 then to draw, else not to draw.
The default value is 0.

To receive current status:
```
int SWBoxTicksПetShow (int *pn)
```

### Set color for the numbers on the box

```
Int SWBoxSetTextColor (int nAxe, int r, int g, int b)
```
Where nAxe is 0 for axe X or 1 for axe Y or 2 for axe Z.. r, g, b are red, green and blue color components.
They can change from 0 to 255.

Get current color for `nAxe-th` axe:
```
Int SWBoxGetTextColor (int nAxe, int *pr, int *pg, int *pb)
```

### Set the number of decimal spaces after point

```
int SWBoxSetDecimalPlaces (int nAxe, int n)
```
Where nAxe is 0 for axe X or 1 for axe Y or 2 for axe Z,  n is number of spaces.
The default value is 2 spaces after a point.

To receive the current parameter for `nAxe-th` axe:
```
int SWBoxGetDecimalPlaces (int nAxe, int *pn)
```

### Set the number of digital marks

```
int SWBoxSetCount (int nAxe, int n)
```
Where nAxe is 0 for axe X or 1 for axe Y or 2 for axe Z, n is the number of marks.
The default value is 3 marks.

Get the current marks count for `nAxe-th` axe:
```
int SWBoxGetCount (int nAxe, int *pn)
```

### Set the font size for digital marks

```
int SWBoxSetFontSize (double d)
```
Where d is font size factor. For marks it is used constant 3D font, but its size can be increased (d> 1) or to reduce (d <1)

Get the current value:
```
int SWBoxGetFontSize (double *pd)
```