Qick Overview of the SGS (Simple Graphic System)

Developer's Manual

Igor Grešovnik

Cachan, March 2004

Contents:

1 INTRODUCTION

SGS (Simple Graphic System) is an auxiliary library that enables to draw basic things. I develop this library for quick visual verification and representation of the things that I am developing in C. The basic characteristics of SGS is that functions for drawing various graphs in 2D or 3D can be programmed very quickly. You are mainly concerned with designing functions that create graphic objects that represent what you do. This is usually simple because you compose this of basic primitives such as lines, surfaces or text strings. Different objects that you create can be easily combined and put together. The functions of the system are then used to display these objects, view them from different angles, apply light effects, and print obtained images in vector formats such as Encapsulated PostScript (.eps) so that you can further process or import them in documents while retaining the high quality.

With SGS, you can do two distinct things: plot directly on a window or to a graphic file, or display graphic objects without explicitly using the functions for plotting. Functions for displaying graphical objects make implicit use of the plotting functions. You can combine both kinds of utilities, for example to make a graph that is additionally decorated, for example by a title or a legend. There are functions that you use with both kinds of utilities, e.g. for opening or closing windows (or equivalently, opening and initializing or finalizing and closing output graphic files)

2 QUICK HOW-TO GUIDE

2.1 Preparing for Graphical Output

In order to plot on the screen, you must first *open a graphical* window. Similarly, if you want to plot to a graphical file (e.g. in a vector format such as PostScript), you must first *open a file* and equip it with a head required by a given format. In IGS, both are done by the **sg_openwindow()** function, but what this function actually does depends on which plotting interface is used.

The current plotting interface is set or changed by the **sg_interface()** function. This function must be called at least once before any other SGS function is called, while its integer argument specifies which plotting interface is used. The default is 0 (or SG_PLOT_DEFAULT), which normally installs one of the available interfaces for plotting on the screen. Other values are for example SG_PLOT_PS (plotting into files in Encapsulated PostScript) or SG_PLOT_TCL (plotting to files that can be interpreted by the Tcl/Tk applications such as wish).

Before a window is open, its properties such as width, height, position, title and other properties can be set. These settings remain valid until they are modified, therefore several windows (or graphic files) with the same properties can easily be created. Several windows can be opened at the same time. The **sg_openwindow()** returns an identification number for each newly open window (or file). This ID can be used in the **sg_setwindow()** call, which sets one window as a current (or active), such that that all subsequent plotting operations refer to this window.

When we don't want a current window to appear on the screen or we want to finish plotting into a current graphic file, we close it with the **sg_closewindow()** command. To close a window that is not an active one, we must first activate it by the **sg_setwindow()** call and then close it by the **sg_closewindow().**

The following code illustrates handling of graphical output:

```
<#include stdio.h>
<#include sg_plot.h>
int main(void) {
     int id1,id2,ch;
     sg_interface(0);  /* activate a screen plotting interface */
     /* Prepare settings for newly open windows (dimensions and title): */
     sg_setwindowheight(0.4); sg_setwindowwidth(0.4);
     sg_setwindowtitle("Application graphic window")
     id1=sg_openwindow();  /* open a graphic window */
     …  /* plot something in the first window*/
     /* Modify settings for newly open windows: */
     sg_setwindowheight(0.2); sg_setwindowxpos(0.5);
     id2=sg_openwindow();  /* open (and activate) another window */
     …  /* plot something in the second window */
     sg_setwindow(id1);  /* re-activate the first window */
     …  /* plot again in the first window */
     sg_interface(SG_PLOT_PS);  /* activate plotting to PostScript files */
     /* Set the name of the next graphic file that will be open by the
              sg_openwindow() command: */
     sg_setwindowfile("out.eps");
     /* Modify some "window" settings: */
```

```
        sg_setwindowheight(0.9); sg_setwindowwidth(0.9);
        id_ps1=sg_openwindow();  /* open and initialize the PostScript file */
        … /* plot into PostScript file */
        sg_closewindow();  /* finalize and close the current output file */
        /* Open a new PostScript output file: */
        sg_setwindowfile("out1.eps"); id_ps2=sg_openwindow();
        …  /* Plot something to this file */
        ch=getchar();  /* Wait until user presses <Enter> */
        sg_closewindow();  /* finalize the output file */
        /* Switch to screen drawing interface: window identified by id1 is
                  active in this interface! */
        sg_interface(0);
        … /* plot again to the first window on the screen */
        sg_closewindow();  /* close the first window */
        sg_setewindow(id2);  sg_slosewindow();  /* close the second window */
}
```

2.2 Plotting Directly

You can plot lines, text strings, framed or filled triangles and rectangles, circles, ellipses, circle and ellipse arcs, markers, arrows, etc. on graphic windows and in graphic files. The concept is similar as with opening windows: you first set attributes for certain groups of primitives and then plot.

Attributes (or settings) remain valid until they are changed. Attributes are set separately for line drawing, filling and text drawing. For any of these you can set three components of RGB color (floating point numbers from 0 to 1). For lines you can set thickness and type, while for text and markers you can in addition set the type (which is the font ID for text) and the size.

Usually only the coordinates are specified when calling the plotting functions. Other attributes are set separately as described above. These are specified as window relative co-ordinates running from 0 to 1 (the values can be out of range).

Please note that although the direct plotting ability can be very useful in certain situations, SGS has not been primarily designed for this kind of use. You will find fare stronger and more useful the SGS capabilities of composition of 2D or 3D graphic objects (such as variety of graphs) and their graphical representation.

```
<#include stdio.h>
<#include sg_plot.h>
int main(void) {
     int id1,id2,ch;
     sg_interface(0);  /* activate a screen plotting interface */
     /* Prepare settings and open a window: */
     sg_setwindowheight(0.4); sg_setwindowwidth(0.4);
     sg_setwindowtitle("Window 1")
     id1=sg_openwindow();
     /* Prepare settings (red thin lines): */
     sg_setlinecolor(1,0,0); sg_setlinewidth(0.005);
```

```
      /* Draw some lines: */
      sg_line(0,0,1,1); sg_line(0,0.5,1,0.5);
      /* Change the line color and draw more lines: */
      sg_setlinecolor(0,0.5,0.5);
      sg_line(0.9,0.2,0.2,0.2); sg_line(0.3,0.3,1,0.4);
      sg_circle(0.5,0.5,0.2); sg_circle(0.5,0.5,0.4);
      /* Open a smaller window and draw something there: */
      sg_setwindowheight(0.2); sg_setwindowxpos(0.5);
      sg_setwindowtitle("Win 2"); id2=sg_openwindow();
      /* Draw some shapes in this window: */
      sg_setfillcolor(1,0.8,0);
      sg_fillellipse(0.2,0.3,0.5,0.15); sg_fillcircle(0.6,0.7,0.5);
      sg_setlinewidth(0.1);
      sg_line(0.2,0.3,0.6,0.7);
      sg_setlinecolor(0,0.8,0);
      sg_line(0,0,1,1); sg_line(0,1,1,0);
      /* Wait for user response and close the windows: */
      ch=getchar();
      sg_closewindow();
      sg_setwindow(id1);
      sg_closewindow();
}
```

```
<#include stdio.h>
<#include sg_plot.h>
int main(void) {
      int id1,id2,ch;
      sg_interface(0);  /* activate a screen plotting interface */
      /* Prepare settings and open a window: */
      sg_setwindowheight(0.4); sg_setwindowwidth(0.4);
      sg_setwindowtitle("Window 1")
      id1=sg_openwindow();
      /* Prepare settings (red thin lines): */
      sg_setlinecolor(1,0,0); sg_setlinewidth(0.005);
      /* Draw some lines: */
      sg_line(0,0,1,1); sg_line(0,0.5,1,0.5);
      /* Change the line color and draw more lines: */
      sg_setlinecolor(0,0.5,0.5);
      sg_line(0.9,0.2,0.2,0.2); sg_line(0.3,0.3,1,0.4);
      sg_circle(0.5,0.5,0.2); sg_circle(0.5,0.5,0.4);
      /* Open a smaller window and draw something there: */
      sg_setwindowheight(0.2); sg_setwindowxpos(0.5);
      sg_setwindowtitle("Win 2"); id2=sg_openwindow();
      /* Draw some shapes in this window: */
      sg_setfillcolor(1,0.8,0);
      sg_fillellipse(0.2,0.3,0.5,0.15); sg_fillcircle(0.6,0.7,0.5);
      sg_setlinewidth(0.1);
      sg_line(0.2,0.3,0.6,0.7);
      sg_setlinecolor(0,0.8,0);
      sg_line(0,0,1,1); sg_line(0,1,1,0);
      /* Wait for user response and close the windows: */
      ch=getchar();
```

```
        sg_closewindow();
        sg_setwindow(id1);
        sg_closewindow();
    }
```

```
1.    <#include stdio.h>
2.    <#include sg_plot.h>
3.    int main(void) {
4.         int id1,id2,ch;
5.         sg_interface(0);  /* activate a screen plotting interface */
6.         /* Prepare settings and open a window: */
7.         sg_setwindowheight(0.4); sg_setwindowwidth(0.4);
8.         sg_setwindowtitle("Window 1")
9.         id1=sg_openwindow();
10.        /* Prepare settings (red thin lines): */
11.        sg_setlinecolor(1,0,0); sg_setlinewidth(0.005);
12.        /* Draw some lines: */
13.        sg_line(0,0,1,1); sg_line(0,0.5,1,0.5);
14.        /* Change the line color and draw more lines: */
15.        sg_setlinecolor(0,0.5,0.5);
16.        sg_line(0.9,0.2,0.2,0.2); sg_line(0.3,0.3,1,0.4);
17.        sg_circle(0.5,0.5,0.2); sg_circle(0.5,0.5,0.4);
18.        /* Open a smaller window and draw something there: */
19.        sg_setwindowheight(0.2); sg_setwindowxpos(0.5);
20.        sg_setwindowtitle("Win 2"); id2=sg_openwindow();
21.        /* Draw some shapes in this window: */
22.        sg_setfillcolor(1,0.8,0);
23.        sg_fillellipse(0.2,0.3,0.5,0.15); sg_fillcircle(0.6,0.7,0.5);
24.        sg_setlinewidth(0.1);
25.        sg_line(0.2,0.3,0.6,0.7);
26.        sg_setlinecolor(0,0.8,0);
27.        sg_line(0,0,1,1); sg_line(0,1,1,0);
28.        /* Wait for user response and close the windows: */
29.        ch=getchar();
30.        sg_closewindow();
31.        sg_setwindow(id1);
32.        sg_closewindow();
33.   }
```

```
1.
2.
3.
4.
5.
```

2.3 Plotting Graphical Objects

3 DEVELOPER INFORMATION

Plotting functions are not divided anymore to those for putting output on the screen and into graphic files. Instead, there is one common interface for both kinds of functions, while the functions in use are switched by the appropriate command.

Different groups of plotting functions share the variables that store current settings, i.e. current line, fill and text style and color, and window settings for opening new windows.

When plotting to graphic files, files are treated as windows when plotting on the screen, i.e. they can be open, switched or closed by the sg_openwindow(), sg_setwindow() and sg_closewindow() functions. Addition functions are provided by file format modules for specifying the output file, and these must be called in the same manner as for example functions for specifying the dimensions of the newly open windows. For file output, the display size can be also specified specially (while plotting to the screen, the display size is normally obtained by querying the actual size of the display in pixels).

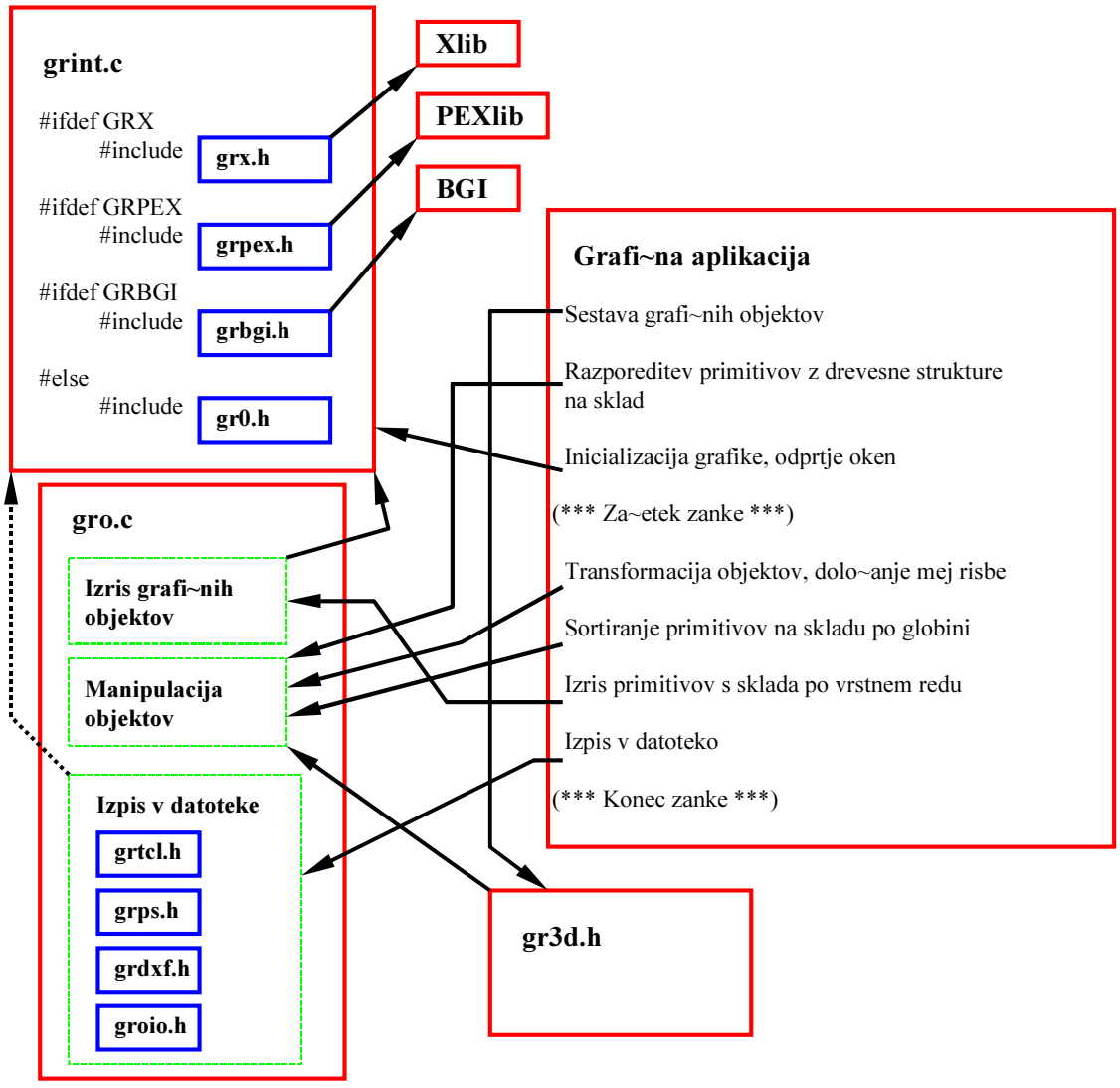
Window IDs have different space for each plotting interface. The current window is maintained locally about each interface. This means that you do not need to set the active window after you switch to another interface, since the last active window before you left the interface will be restored. On the contrary, when you activate a window in one plotting interface, this does not have any connection with the window that is active in other plotting interfaces, if they are used. With other words, windows must be switched for each plotting interface separately.

It is not yet decided whether fonts will be set and installed separately for each plotting interface. This remains an unsolved issue.

4 STRUCTURE OF THE SGS

Table 1: files of SGS.

File name	Old name	Purpose
sg_draw.c	grint.c	Bottom-most drawing interface
sg_itk.c	gritk.g	Drawing by ITK


grint.c
#ifdef GRX
#include
grx.h
#ifdef GRPEX
#include
grpex.h
#ifdef GRBGI
#include
grbgi.h
#else
#include
gr0.h
Xlib
PEXlib
BGI
Grafi~na aplikacija
Sestava grafi~nih objektov
Razporeditev primitivov z drevesne strukture na sklad
Inicializacija grafike, odprtje oken
(*** Za~etek zanke ***)
Transformacija objektov, dolo~anje mej risbe
Sortiranje primitivov na skladu po globini
Izris primitivov s sklada po vrstnem redu
Izpis v datoteko
(*** Konec zanke ***)
gro.c
Izris grafi~nih objektov
Manipulacija objektov
Izpis v datoteke
grtcl.h
grps.h
grdxf.h
groio.h
gr3d.h

5 IGS TEXT

IGS is the predecessor of the SGS. Below follows the text that describes IGS.

6 VRSTE GRAFIČNIH PRIMITIVOV

Grafični primitivi (tip *goprimitive*) so različnih vrst. Vrsto primitiva določajo polja *(...)->i1*, *(...)->i2* in *(...)->i3* primitiva. V razpredelnici Tab. 6.1.

Tab. 6.1: Vrste grafičnih primitivov. V 1. Stolpcu tabele so identifikacijaka polja, ki dolocajo vrsto primitiva, v 2. so imena, ki jih uporabljamo v funkcijah, ki delajo s posameznimi primitivi različnih vrst, v 3. Pa je kratek opis primitivov različnih vrst.

Identifikacijski znaki vrste primitiva			**ime uporabljeno v funkcijah**	**opis**
t	**0**	**0**	**text**	navaden tekst
l	**0**	**0**	**line**	črta
3	**0**	**0**	**triangle**	trikotnik iz črt
3	**f**	**0**	**filltriangle**	pobarvan trikotnik
3	**b**	**0**	**bordtriangle**	pobarvan obrobljen trikotnik
3	**p**	**0**	**partbordtriangle**	pobarvan delno obrobljen trikotnik
4	**0**	**0**	**fourangle**	štirikotnik iz črt
4	**f**	**0**	**fillfourangle**	pobarvan štirikotnik
4	**b**	**0**	**bordfourangle**	pobarvan obrobljen štirikotnik
4	**p**	**0**	**partbordfourangle**	pobarvan delno obrobljen štirikotnik
m	**0**	**0**	**marker**	marker za označevanje točk

Tab. 6.2: Vrste lastnosti, na katere lahko kažejo kazalci *(...)->props1 … (...)->props3* grafičnega primitiva, glede na vrsto primitiva.

Identifikacijski znaki vrste	**Pomen kazalcev (…)->props1, …, (…)->props3**

primitiva				
t	**0**	**0**	**props1**	Tekstovni niz (char *)
			props2	Nastavitve za tekst (gotextsettings)
l	**0**	**0**	**props1**	Nastavitve za linije (golinesettings)
3	**0**	**0**	**props1**	Nastavitve za linije (golinesettings)
3	**f**	**0**	**props1**	Nastavitve za ploskve (gofillsettings)
3	**b**	**0**	**props1** **props2**	Nastavitve za ploskve (gofillsettings) Nastavitve za linije (golinesettings)
3	**p**	**0**	**props1** **props2** **props3**	Nastavitve za ploskve (gofillsettings) Nastavitve za linije (golinesettings) Zastavice za aktivne robove (char[3])
4	**0**	**0**	**props1**	Nastavitve za linije (golinesettings)
4	**f**	**0**	**props1**	Nastavitve za ploskve (gofillsettings)
4	**b**	**0**	**props1** **props2**	Nastavitve za ploskve (gofillsettings) Nastavitve za linije (golinesettings)
4	**p**	**0**	**props1** **props2** **props3**	Nastavitve za ploskve (gofillsettings) Nastavitve za linije (golinesettings) Zastavice za aktivne robove (char[3])
m	**0**	**0**	**props1** **props2** **props3** **props4**	Nastavitve za ploskve (gofillsettings) Nastavitve za linije (golinesettings) Velikost (double[1]) Vrsta (int[1])

7 MODUL *GRO*

Modul ***gro*** vsebuje funkcije za delo z grafičnimi objekti in funkcije za izris grafičnih objektov na zaslon.

7.1 Opis funkcij v gro.c in njihovega delovanja

V tem poglavju so opisani delovanje funkcij iz gro.c, njihove povezave z lokalnimi spremenljivkami tega modula in soodvisnosti funkcij in spremenljivk.

Funkcije so opisane približno v takšnem vrstnem redu, kot se pojavljajo v datoteki gro.c.

7.1.1 Alokacija in brisanje objektov različnih tipov

Na začetku modula gro.c so funkcije za brisanje in alokacijo objektov osnovnih grafičnih tipov. To so funkciji za alokacijo in brisanje grafičnih primitivov *newgoprimitive()* in *dispgoprimitive()*, funkciji za alokacijo skupin grafičnih objektov *newgogroup()* in *newgogroupst()* ter funkcija za brisanje skupin grafičnih objektov *dispgogroup()*.

Funkcija <*goprimitive newgoprimitive(void)*> alocira prostor za objekt tipa *goprimitive* in vse kazalce objekta postavi na null. Polja *i1*, *i2* in *i3*, ki določajo vrsto primitiva, postavi na znak "0". Vrne kazalec na alociran prostor.

Funkcija <*void dispgoprimitive(goprimitive *gp)*> najprej izvede funkcijo, na katero kaze polje *(*gp)->clear*, ce je to polje različno od NULL. Po dogovoru to polje kaže na funkcijo za brisanje posebnih delov grafičnega primitiva, na katere ne kažejo direktno kazalci, ki so polja na primitivu *gp*, ampak morebitni kazalci na strukturah, na katere kažejo polja **gp*. Funkcija potem sp klicem *free(...)* sprosti vse strukture, na katere kažejo polja **gp*, seveda, če so ta polja različna od NULL. Potem sprosti še **gp* in ga postavi na NULL.

Funkcija <*gogroup newgogroup(void)*> alocira prostor za objekt tipa *gogroup* in vse kazalce objekta postavi na null. Polje *id*, ki predstavlja identifikacijsko številko skupine, postavi na 0. Tudi polji *limitsset* in *transflimitsset* postavi na 0. Vrne kazalec na alociran prostor.

Funkcija <*gogroup newgogroup(int ngroups,int nprimitives,int nextraprimitives)*> deluje podobno kot funkcija *newgogroup()*, le da alocira tudi sklade *(...)->groups, (...)->primitives* in *(...)->primitives* skupine, ki jo vrne.

Funkcija <*void dispgoprimitive(gogroup *gg)*> najprej izvede funkcijo, na katero kaze polje *gg->clear*, ce je to polje različno od NULL. Po dogovoru to polje kaže na funkcijo za brisanje posebnih delov grafične skupine, na katere ne kažejo direktno kazalci, ki so polja na skupini *gg*, ampak morebitni kazalci na strukturah, na katere kažejo polja *gg*. Funkcija potem sp klicem *free(...)* sprosti vse strukture, na katere kažejo polja **gg*, seveda, če so ta polja različna od NULL. Potem sprosti še **gg* in ga postavi na NULL.

7.1.2 Pretvarjanje koordinat objekta v okenske koordinate

Pretvarjanje koordinat objekta (naravnih ali transformiranih) v okenske koordinate vrši <*static void (*gpwindowcoord) (coord3d point, double *x, double *y)*>, ki je kazalec na eno izmed funkcij, namenjenih za to opravilo. V *point* damo koordinate točke objekta in funkcija nam v *x* in *y* vrne izračunane okenske koordinate te točke. Ob inicializaciji grafičnega sistema je *gpwindowcoord* funkcija *baswindowcoord()*.

Možna je uporaba naslednjih funkcij za pretvorbo:

<*static void baswindowcoord(coord3d point,double *x,double *y)*> povzročI risanje v pravokorni projekciji, vrne skalirani koordinati x in y iz točke *point*.

<*static void eyewindowcoord(coord3d point,double *x,double *y)*> povzročI risanje v perspektivi, vrne skalirani koordinati x in y točke *point*, vendar njuno oddaljenost še pomnoži s faktorjem, ki je obratno sorazmeren oddaljenosti točke *point* v smeri osi z. Pri tem igra pomembno vlogo lokalna spremenljivka *distancefactor*, ki predstavlja razmerje med oddaljenostjo opazovalca od središča grafa in diagonalo grafa.

7.1.3 Transformacije koordinat grafičnih objektov

7.1.3.1 Transformacije koordinat

Za transformaciijo koordinat se uporabljajo funkcije *tipa <static void (...) (coord3d original, coord3d transformed)>*, kjer so *original* originalne koordinate, v *transformed* pa se zapišejo transformirane koordinate. Prostor za transformirane koordinate mora biti v celoti alociran. Vsi parametri transformacij so v ustreznih lokalnih spremenljivkah.

*<static void (*gotransfcoord) (coord3d,coord3d)>* je kazalec na funkcijo, ki se v danem trenutku uporablja za transformacijo koordinat.

<static void transfcoordsimp(coord3d original, coord3d transformed)> je funkcija za enostavno transformacijo koordinat, kjer imamo podan azimut in polarni kot pogleda na grafične objekte. Funkcija uporablja parametre transformacije, ki so v lokalni spremenljivki *trsimp*.

<static void transfcoordsimpscale(coord3d original, coord3d transformed)> transformira podobno kot *transfcoordsimp()*, le da hkrati opravi še skaliranje koordinat v smereh osi x, u in z za faktorje *xscale*, *yscale* in *zscale*, ki so lokalne spremenljivke modula.

*<static void (*gotransfcoord) (coord3d,coord3d)>* je kazalec na funkcijo, ki se trenutno uporablja za transformacijo koordinat. Na začetku je postavljen na *transfcoordsimp()*.

<static void transformgoprimitive(goprimitive gp)> je funkcija, ki pretransformira koordinate grafičnega primitiva *gp* s pomočjo funkcije, na katero kaže *gotransfcoord*. Funkcija sama alocira prostor za transformirane koordinate primitiva, če je to potrebno. Po transformaciji funkcija postavi vrednost *gp->dp* na povprečno vrednost koordinat z vseh točk primitiva (po potrebi pred tem alocira prostor za **gp->dp*).

*<static void (*gotransfprimitive) (goprimitive)>* je kazalec na funkcijo, ki se uporablja za transformacijo koordinat grafičnih primitivov. Na začetku kaže na *transformgoprimitive()*, ki je zaenkrat tako edina funkcija za transformacijo koordinat grafičnega primitiva.

<void gotransfgroup(gogroup gg)> pretransformira koordinate vseh grafičnih primitivov, ki so vsebovani v skupini *gg* in njenih podskupinah, s pomočjo funkcije, na katero kaže *gotransfprimitive*.

<void gotransfstack(stack st)> pretransformira koordinate vseh grafičnih primitivov, ki so na skladu *st*, s pomočjo funkcije, na katero kaže *gotransfprimitive*.

7.1.3.2 Nastavitev načina pretvorbe koordinat

Pretvorba naravnih koordinat objektov v okenske poteka prek dveh stopenj. Najprej se izračunajo transformirane koordinate objekta, za kar se uporabi funkcija, na katero kaže *gotransfcoord*. Nato se transformirane koordinate pretvorijo v okenske. Pretvorba prvotnih

koordinat v transformirane in iz transformiranih v okenske je odvisna od funkcij, ki se za to uporabljajo, in od lokalnih spremenljivk, ki določajo parametre pretvorbe.

Funkcije za nastavitev parametrov, ki vplivajo na pretvorbo naravnih koordinat grafičnih objektov v okenske, so oblike <*void gosettransf...(...)*>. Zadnji del imena določa vrsto pretvorb naravnih koordinat v transformirane in the v okenske, argumenti pa so parametri transformacije iz naravnih koordinat v transformirane. Parametri, ki doočajo pretvorbo transformiranih koordinat v okenske, niso podani kot argumenti the funkcij, ampak se nastavijo posebej. V vseh primerih so to meje grafičnih objektov v koordinatah, v katerih se bodo izrisali (naravne ali transformirane) in meje območja v oknu, kamor naj se izriše vse iz grafa. Poleg the so lahko še drugi parametri, na primer razdalja od središča grafa do opazovalca.

Funkcije za nastavitev pretvorb koordinat so:

<*void gosettransfsimp(double fi, double theta)*>: Funkcija za transformacijo koordinat postane *transfcoordsimp()*. *fi* in *theta* sta azimut in polarni kot položaja opazovalca glede na središče grafa. Lokalna spremenljivka, ki določa transformacijo, je *trsimp*. Funkcija za pretvorbo v okenske koordinate postane *baswindowcoord()*.

<*void gosettransfsimpscale(double fi, double theta*)>: Vse je podobno kot pri funkciji *gosettransfsimp()*, le da se koordinate pri transformaciji v različnih smereh še skalirajo glede na vrednosti lokalnih spremenljivk *xscalingfactor*, *yscalingfactor* in *zscalingfactor*. Zato se za transformacijo koordinat uporablja funkcija *transfcoordsimpscale()*.

<*void gosettransfeyesimp(double fi, double theta)*>: Podobno kot pri funkciji *gosettransfsimp()*, le da se koordinate objekta pretvorijo v okenske koordinate tako, da se objekt vidi v perspektivi. Za transformacijo koordinat se uporablja funkcija *transfcoordsimp()*, za pretvorbo v okenske koordinate pa funkcija <*eyewindowcoord()*>.

<*gosettransfeyesimpscale*(double fi, double theta)>: Pri transformaciji se koordinate različno skalirajo v različnih smereh, pri pretvorbi v okenske koordinate pa se uporabi perspektiva. Za transformacijo koordinat se uporablja funkcija *transfcoordsimpscale()*, za pretvorbo v okenske koordinate pa funkcija <*eyewindowcoord()*>.

7.1.3.2.1 Lokalne spremenljivke

<*static _transfsimptype trsimp*> Vsebuje nastavitve za enostavno transformacijo koordinat.

<*static double xscalingfactor=1,yscalingfactor=1,zscalingfactor=0.6*> so faktorji, s katerimi se množijo koordinate pri skaliranju. Uporabljajo se v funkcijah za transformacijo koordinat, ki vključujejo tudi skaliranje.

7.1.4 Funkcije za risanje grafičnih objektov

7.1.4.1 Funkcije za nastavitev lastnosti osnovnih grafičnih objektov

Te funkcije služijo za hkratno nastavitev vseh lastnosti za določeno skupino tipov osnovnih grafičnih objektov. Mišljeno je osnovnih na nivoju modula **intfc.c**, ne osnovnih na nivoju modula

gro.c, kjer so osnovni grafični objekti vsi tisti, ki jih predstavimo z grafičnimi primitivi. Takšni objekti se pri risanju lahko izrišejo iz več osnovnih objektov na nivoju modula **intfc.c**. Tako je na primer primitiv "zapolnjen obrobljen štirikotnik", ki je na nivoju gro.c osnovni objekt, pri risanju (ki se dogaja na nivoju intfc.c) sestavljen iz ene štirikotne ploskve in enega štirikotnika iz črt.

Omenjene funkcije nastavijo lastnosti grafičnih objektov glede na posevej za to namenjene strukture, katerih tip je odvisen od skupine tipovosnovnih objektov. Nastavitve se odražajo v sistemu **grint.c** in ostanejo v veljavi, dokler niso povožene z drugačnimi nastavitvami. Zaenkrat obstajajo tri skupine osnovnih objektov, za katere lahko vse lastnosti grupiramo v takšne strukture, in sicer črte (tem ustreza tip *golinesettings*), ploskve (ustrezen tip je *gofillsettings*) in tekst (ustrezen tip je *gotextsettings*).

Dotične funkcije so:

- <*static void setlinesettings(golinesettings ls)*> nastavi nastavitve za risanje črt, kot so v *ls*.
- <*static void setfillsettings(gofillsettings fs)*> nastavi nastavitve za risanje ploskev, kot so v fs.
- <*static void settextsettings(gotextsettings ts)*> nastavi nastavitve za risanje teksta, kot so v ts.

7.1.4.2 Funkcije za osvetljevanje objektov

v IGS imamo možnost osvetljevanja grafičnih objektov z lučmi, ki so razporejene po prostoru. Zaenkrat je implementirano osvetljevanje z difuzno svetlobo in z neskončno oddaljenimi svetili.

7.1.4.2.1 Spremenljivke

7.1.4.2.1.1 Difuzna svetloba

<*static _truecolor intdifuselight*> - intenziteta difuzne sbetlobe.

7.1.4.2.1.2 Neskončno oddaljena svetila

<*static stack intfarlights*> - sklad, ki vsebuje intenzitete neskončno oddaljenih svetil.
<*static stack dirfarlights*> - sklad, na katerem so smeri neskončno oddaljenih svetil.

7.1.4.2.1.3 Splošne spremenljivke

<*static double powfarlighting*> - poteca, s katero se množijo kosinusi vpadnih kotov. Z večanjem te potence udarimo efekt zatemnjevanja zaradi poševnega vpada svetlobe na ploskve.

<*static double lightfactor*> Faktor, s katerim se intenziteta odbite svetlobe površine.

<*static char dolighting*> - Indicira, ali naj se računajo barve grafičnih primitivov glede na definirana svetila (če je enak 1) ali naj se objekti izrišejo v svoji naravni barvi (če je enak 0).

7.1.4.2.2 Funkcije

7.1.4.2.2.1 Funkcije za kontrolo skupne intenzitete svetlobe svetil in kontrasta

<*void godolighting(char dl)*> Preklaplja med načinom, pri katerem se upoštevajo svetila (*dl*=1) in načinom, pri katerem se ne (*dl*=0). <*double gogetlightfactor(void)*> vrne vrednost lokalne spremenljivke *lightfactor*, s katero se množi izračunana intenziteta odbite svetlobe pri risanju. <*void*

gosetlightfactor(double factor)> postavi lokalno spremenljivko *lightfactor* na vrdnost *factor*. <*void goupdatelightfactor(double factor)*> Najprej postavi lokalno spremenljivko *lightfactor* na recipročno vrednost največje vsote ene komponente barve po intenzitetah vseh svetil, nato pa jo še pomnoži z vrednostjo *factor*.

<*double gogetpowfarlighting(void)*> vrne vrednost lokalne spremenljivke *powfarlighting*, ki določa potenco, s katero se potencirajo kosinusi vpadnih kotov pri računanju osvetljenosti objektov. Če je vrednost te sspremenljivke večja, je bolj izrazit efekt senčenja. Vrednost te spremenljivke nastavimo s funkcijo S povečanjem te spremenljivke poudarimo efekt senčenja. To storimo s funkcijo <*void gosetpowfarlighting(double pw)*>.

7.1.4.2.2.2 Dodajanje luči in računanje barve odbite svetlobe

S funkcijo <*void gosetdifuselight(double red,double green,double blue)*> dodamo difuzno svetlobo (to je razpršena svetloba, ki prihaja enakomerno z vseh strani) zi intenzitetamo komponenr *red*, *green* in *blue*. S funkcijo <*void gosetfarlight(double red,double green,double blue,double xdir,double ydir,double zdir)*> dodamo neskončno oddaljen vir svetlobe v smeti (*xdir,ydir,zdir*) z jakostjo komponent *red*, *green* in *blue*.

Funkcija <*static calclightintensity(stack coord,truecolor original,truecolor calc)*> izračuna barvo svetlobe, odbite nagrafičnem objektu, glede na definirane luči. *coord* je sklad koordinat objekta, *original* je naravna barva objekta, v *calc* pa se zapiše izračunana barva svetlobe, ki se odbije od objekta.

Pri ploskvah (če sta na skladu *coord* več kot 2 koordinati) se izračuna intenziteta komponent odbite svetlobe po formuli

$$I' = I^0\left(I^{difuse} + \sum_{i=1}^{n\_far} I_i^{far}\left(\cos\left(\phi_i^{far}\right)\right)^{powfarlighting}\right),$$

kjer je I^0 komponenta naravne barve objekta, I^{difuse} ustrezna komponenta difuzne svetlobe, n_{far} je število neskončno oddaljenih svetil, I_i^{far} je ustrezna komponenta svetlobe z *i*-tega neskončno oddaljenega svetila in ϕ_i^{far} je kot, pod katerim pada svetloba na objekt. V funkciji *calclightintensity()* se najprej izračuna sinus tega kota kot

$$\sin\left(\phi_i^{far}\right) = \mathbf{n} \times \mathbf{s}_i^{far} = \frac{\left(\mathbf{p}_2 - \mathbf{p}_1\right) \times \left(\mathbf{p}_3 - \mathbf{p}_1\right)}{\left\|\left(\mathbf{p}_2 - \mathbf{p}_1\right) \times \left(\mathbf{p}_3 - \mathbf{p}_1\right)\right\|} \times \mathbf{s}_i^{far},$$

kjer je $\mathbf{s}_i^{far}$ normirana smer žarka i. oddaljenega svetila, $\mathbf{p}_1$, $\mathbf{p}_2$ in $\mathbf{p}_3$ pa so prve tri koordinate na skladu *coord*. Iz sinusa kota se potem izračuna njegov kosinus kot

$$\cos\left(\phi_i^{far}\right) = \sqrt{1 - \left(\sin\left(\phi_i^{far}\right)\right)^2}\ .$$

Pri črtah (če sta na skladu *coord* 2 koordinati) se intenziteta izračuna posobno, le da se namesto normale ploskve $\mathbf{n}$ vzame smerni vektor črte ter je zato

$$\sin\left(\phi_i^{far}\right) = \mathbf{n} \times \mathbf{s}_i^{far} = \frac{\left(\mathbf{p}_2 - \mathbf{p}_1\right)}{\left\|\left(\mathbf{p}_2 - \mathbf{p}_1\right)\right\|} \times \mathbf{s}_i^{far}.$$

7.1.4.3 Izris grafičnih primitivov

Vsaka vrsta grafičnega primitiva ima svoje funkcije za izris. V resnici za izris skrbijo kazalci na te funkcije. Definirani so kot <*static void (* gpdraw…) (goprimitive gp)*>, kjer je … ime vrste grafičnega primitiva, na primer <*static void (* gpdrawline) (goprimitive gp)*> je kazalec na funkcijo, ki izriše primitiv tipa (“l”,”0”,”0”) oz. črto. Argument *gp* je primitiv, ki naj se izriše.

Imena, ki se uporabljajo v funkcijah za izris grafičnih primitivov, so razvidna iz Tab. 6.1.

Definirani so naslednji kazalci na funkcije za izris:

static void (gpdrawline) (goprimitive gp)=basgpline;*
static void (gpdrawtriangle) (goprimitive gp)=basgptriangle;*
static void (gpdrawfilltriangle) (goprimitive gp)=basgpfilltriangle;*
static void (gpdrawbordtriangle) (goprimitive gp)=basgpbordtriangle;*
static void (gpdrawpartbordtriangle) (goprimitive gp)=basgppartbordtriangle;*
static void (gpdrawfourangle) (goprimitive gp)=basgpfourangle;*
static void (gpdrawfillfourangle) (goprimitive gp)=basgpfillfourangle;*
static void (gpdrawbordfourangle) (goprimitive gp)=basgpbordfourangle;*
static void (gpdrawpartbordfourangle) (goprimitive gp)=basgppartbordfourangle;*
static void (gpdrawmarker) (goprimitive gp)=basgpmarker;*
static void (gpdrawtext) (goprimitive gp)=basgptext;*

Zaenkrat je za vsak primitiv definirana le po ena funkcija za izris, in sicer <*static void basgp…(goprimitive gp)*>. Delovanje takšnih funkcij je naslednje:

7.1.4.3.1 Delovanje funkcij za izris posameznih grafičnih primitivov različnih vrst

Kot že rečeno, so imena the funkcij oblike <*static void basgp…(goprimitive gp)*>, kjer je “…” zamenjava za ime vrste primitiva (2. Stolpec v Tab. 6.1). Takšne funkcije zahtevajo kot edini argument grafični primitiv (kazalec tipa *goprimitive*), ki ga izrišejo.

V vsaki takšni funkciji se najprej najdejo koordinate primitiva. Če je statična spremenljivka *drawtransf* različna od 0, se vzamejo transformirane koordinate *gp->transfcoord*, seveda, če obstajajo, torej, če je ta kazalec različen od NULL in če je na skladu, na katerega kaže, vsaj toliko koordinat, kot jih potrebujemo za primitiv danega tipa. V nasprotnem primeru se za koordinate vzamejo netransformirane koordinate.

Ko imamo koordinate, se dobijo in nastavijo vse nastavitve za risanje osnovnih objektov glede na modul **intfc.c**. Za vse nastavitve se preveri, če obstajajo na samem primitivu (torej, če so ustrezni kazalci *gp->props1* … *gp->props4* različni od NULL. Od vrste primitiva je odvisno, na kakšne lastnosti naj bi kazal kateri od the kazalcev (glej Tab. 6.2). Če torej nastavitve za dano vrsto grafičnih objektov obstajajo na samem primitivu, se nastavijo te nastavitve s klicem ustreznih funkcij modula **grint.c**. V nasprotnem primeru se poskusijo nastavitve najti na skupini grafičnih objektov, ki vsebuje naš primitiv (kazalec na skupino, ki vsebuje primitiv *gp*, je *gp->grp*). Iz vrste primitiva *gp* je seveda razvidno, kakšne vrste lastnosti potrebujemo. Na skupinah so za lastnosti objektov namenjeni kazalci *(...)->ls1* in *(...)->ls2* (nastavitve za risanje črt), *(...)->fs1* in *(...)->fs2* (lastnosti za risanje ploskev) ter *(...)->ts1* in *(...)->ts2* (lastnosti za risanje teksta). Če je lokalna spremenljivka *dolighting* različna od 0, se barve objektov izračunajo glede na definirane luči, drugače pa se objekti izrišejo v naravih (lastnih) barvah.

Ko so nastavljene vse nastavitve za risanje objektov, se opravi preslikava koordinat primitiva v okenske koordinate, nakar se primitiv izreše s klicem funkcij iz modula **grint.c**.

7.1.4.3.2 Funkcija za izris grafičnega primitiva

<*void godrawprimitive(goprimitive gp)*> Izriše grafični primitiv *gp*. Glede na identifikacijske znake *gp->i1*, *gp->i2* in *gp->i3* kliče funkcije za izris primitivov dane vrste. Razvejitev je zaenkrat izvedena z *if*, v prihodnosti naj bi se preslo na *switch*, ki je hitrejši.

7.1.4.3.3 Izris sklada grafičnih primitivov

<*void godrawstack(stack st)*> po vrsti izriše vse primitive, ki so naloženi na skladu *st*. Za izris kliče

7.1.4.3.4 *gosetdrawtransf()*

<*void gosetdrawtransf(char dt)*> postavi lokalno spremenljivko *drawtransf* na vrednost *dt*. Če je *drawtransf* enak 0, se objekti ne izrisujejo v transformiranih, ampak v originalnih koordinatah.

7.1.4.4 Izris grafičnih objektov v različnih formatih

Na tem mestu so vrinjeni celi moduli za izris grafičnih objektov v različnih formatih (PostScript, DXF in Tcl). Vsebovani so z direktivami #include.

7.1.5 Priprava na risanje

7.1.5.1 Razporeditev grafičnih primitivov na sklad

Pri risanju trodimenzionalnih grafov je pomemben vrstni red, v katerem se izrišejo grafični primitivi pri risanju sestavljenih grafičnih objektov. Primitivi, ki so globlje, semorajo izrisati prej, da jih primitivi bliže mestu opazovalca prekrijejo. To se doseže tako, da se vsi primitivi, ki sestavljajo graf, najprej razvrstijo na sklad, nato sortirajo po globini in se izrišejo po vrstnem redu, v kakršnem so na skladu.

<*void goloadtostack(gogroup gg,stack st)*> potisne vse grafične primitive, ki so vsebovani na skladih *(...)->primitives* skupine *gg* in njenih podskupin v vseh nivojih, na sklad *st*.

<*static int gpcomparedp(void *p1,void *p2)*> je funkcija, primerja grafična primitiva *p1* in *p2* po polju *(...)->dp*. To polje kaže na število tipa double in predstavlja pri grafičnih primitivih povprečno višino (koordinato z) primitiva. Funkcija vrne -1, če je *p1->dp<p2->dp*, drugače pa 1.

<*static int (*gpcompare) (void *,void *) = gpcomparedp*> je kazalec na funkcijo za primerjavo dveh grafičnih primitivov. Funkcija, na katero kaže ta kazalec, se v resnici uporabi pri sortiranju primitivov na skladu s funkcijo *gsortstack()*. *Gpcompare* najprej kaže na *gpcomparedp*.

<*void gosortstack(stack st)*> sortira grafične primitive na skladu *st* glede na funkcijo, na katero trenutno kaže *gpcompare*.

7.1.5.2 Priprava geometrijskih parametrov

<*void preparegraph3dbas(frame3d limits,frame3d frame)*> pripravi geometrijske parametre, ki vplivajo na izris grafičnih objektov, na pretvorbo iz koordinat objekta v okenske koordinate. V *limits* so geometrijske meje tega, kar naj se izriše. V *frame* je okvir v oknu, v katerega naj se izrišejo stvari, ki so v koordinatah objekta znotraj *limits*.

7.1.6 Konstrukcija grafičnih objektov

7.1.6.1 Funkcija goupdatelimits()

Funkcija <*void goupdatelimits(coord3d p,gogroup gg)*> po potrebi razširi meje na *gg->min* in *gg->max* tako, da je točka s koordinatami *p* znotraj the mej.

7.1.6.2 Funkcije za dodajanje grafičnih primitivov

Funkcije vrste <*goadd…(…,coord3d p1,…,stack st,gogroup grp)*> so namenjene za lazje programiranje sestave različnih grafičnih objektov. Vsaka takšna funkcija doda nov primitiv dane vrste (ki je odvisna od imena funkcije) na sklad *st*, njegovo matično skupino pa postavi na *grp*. 1. serija argumentov se nanaša na lastnosti, ki so za določeno vrsto primitiva obvezne (na primer znakovni niz pri tekstu), 2. serija pa na koordinate primitiva. Da je zadoščeno pravilom grafičnega sistema IGS, mora biti pri klicu takšnih funkcij *st* enak ali *gr->primitives* ali *gr->extraprimitives*.

Primer:

<*goprimitive goaddmarker(coord3d p1,double size,int kind,stack st,gogroup grp)*> naredi grafični primitiv, ki predstavlja marker, s koordinato *p1*, velikosti *size*, vrste *kind* in potisne njegov kazalec na sklad *st*, njegovo polje *(…)->grp* pa postavi na *grp*. Pri klicu mora biti *grp* skupina, ki vsebuje sklat *st* (*st* je lahko *grp->primitives* ali *grp->extraprimitives*).