```c
/*
MathGL samples w/ IupMglPlot
Based on: MathGL documentation (v. 1.10), Cap. 9
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define USE_IM 1
#ifdef USE_IM
#include "im.h"
#include "im_image.h"
#include "im_convert.h"
#include "im_process.h"
#endif

#include "iup.h"
#ifdef USE_IM
#include "iupim.h"
#endif
#include "iupcontrols.h"
#include "iup_mglplot.h"
#include "iupkey.h"

static Ihandle *plot,           /* plot */
               *minmaxY_dial, *minmaxX_dial,  /* dials for zooming */
               *autoscaleY_tgg, *autoscaleX_tgg,    /* auto scale on|off toggles */
               *grid_tgg,           /* grid show|hide toggles */
               *legend_tgg,           /* legend show|hide toggle */
               *box_tgg,           /* box show|hide toggle */
               *aa_tgg,           /* antialias enable|disable toggle */
               *transp_tgg,           /* transparent enable|disable toggle */
               *light_tgg,           /* light enable|disable toggle */
               *opengl_tgg;          /* opengl enable|disable toggle */

static char filenameSVG[300], filenameEPS[300], filenamePNG[300];

static void ResetClear(void)
{
  IupSetAttribute(plot, "RESET", NULL);
  IupSetAttribute(plot, "CLEAR", NULL);

  // Some defaults in MathGL are different in IupMglPlot
  IupSetAttribute(plot, "AXS_X", "NO");
  IupSetAttribute(plot, "AXS_Y", "NO");
  IupSetAttribute(plot, "AXS_Z", "NO");

  IupSetAttribute(plot, "FONT", "Helvetica, 8");
//  IupSetAttribute(plot, "FONT", "Courier, 10");
//  IupSetAttribute(plot, "FONT", "Heros, 10");
//  IupSetAttribute(plot, "FONT", "Termes, 12");
//  IupSetAttribute(plot, "FONT", "XXXX, 10");

  IupSetCallback(plot, "POSTDRAW_CB", NULL);
}

static void UpdateFlags(void)
{
  char *value;

  /* auto scaling Y axis */
  if (IupGetInt(plot, "AXS_YAUTOMIN") && IupGetInt(plot, "AXS_YAUTOMAX"))
  {
    IupSetAttribute(autoscaleY_tgg, "VALUE", "ON");
    IupSetAttribute(minmaxY_dial, "ACTIVE", "NO");
  }
```

```
  else
  {
    IupSetAttribute(autoscaleY_tgg, "VALUE", "OFF");
    IupSetAttribute(minmaxY_dial, "ACTIVE", "YES");
  }

  /* auto scaling X axis */
  if (IupGetInt(plot, "AXS_XAUTOMIN") && IupGetInt(plot, "AXS_XAUTOMAX"))
  {
    IupSetAttribute(autoscaleX_tgg, "VALUE", "ON");
    IupSetAttribute(minmaxX_dial, "ACTIVE", "NO");
  }
  else
  {
    IupSetAttribute(autoscaleX_tgg, "VALUE", "OFF");
    IupSetAttribute(minmaxX_dial, "ACTIVE", "YES");
  }

  /* grid */
  value = IupGetAttribute(plot, "GRID");
  if (value && strstr(value, "XYZ"))
    IupSetAttribute(grid_tgg, "VALUE", "ON");
  else
    IupSetAttribute(grid_tgg, "VALUE", "OFF");

  /* legend */
  if (IupGetInt(plot, "LEGEND"))
    IupSetAttribute(legend_tgg, "VALUE", "ON");
  else
    IupSetAttribute(legend_tgg, "VALUE", "OFF");

  /* box */
  if (IupGetInt(plot, "BOX"))
    IupSetAttribute(box_tgg, "VALUE", "ON");
  else
    IupSetAttribute(box_tgg, "VALUE", "OFF");

  /* antialias */
  if (IupGetInt(plot, "ANTIALIAS"))
    IupSetAttribute(aa_tgg, "VALUE", "ON");
  else
    IupSetAttribute(aa_tgg, "VALUE", "OFF");

  /* transparent */
  if (IupGetInt(plot, "TRANSPARENT"))
    IupSetAttribute(transp_tgg, "VALUE", "ON");
  else
    IupSetAttribute(transp_tgg, "VALUE", "OFF");

  /* light */
  if (IupGetInt(plot, "LIGHT"))
    IupSetAttribute(light_tgg, "VALUE", "ON");
  else
    IupSetAttribute(light_tgg, "VALUE", "OFF");

  /* opengl */
  if (IupGetInt(plot, "OPENGL"))
    IupSetAttribute(opengl_tgg, "VALUE", "ON");
   else
    IupSetAttribute(opengl_tgg, "VALUE", "OFF");
}

static void SampleVolume(const char* ds_mode)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "-2*((2*x-1)^2 + (2*y-1)^2 + (2*z-1)^4 - (2*z-1)^2 - 0.1)", 60, 50,
40);
```

```
  IupSetAttribute(plot, "DS_MODE", ds_mode);

  IupSetAttribute(plot, "ROTATE", "40:0:60");
  IupSetAttribute(plot, "LIGHT", "YES");
  IupSetAttribute(plot, "TRANSPARENT", "YES");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleDensityProjectVolume(void)
{
  SampleVolume("VOLUME_DENSITY");
  IupSetAttribute(plot, "PROJECT", "YES");
  IupSetAttribute(plot, "PROJECTVALUEX", "-1");
  IupSetAttribute(plot, "PROJECTVALUEY", "1");
  IupSetAttribute(plot, "PROJECTVALUEZ", "-1");
  IupSetAttribute(plot, "TRANSPARENT", "NO");
  IupSetAttribute(plot, "LIGHT", "NO");
}

static void SampleContourProjectVolume(void)
{
  SampleVolume("VOLUME_CONTOUR");
  IupSetAttribute(plot, "PROJECT", "YES");
  IupSetAttribute(plot, "PROJECTVALUEX", "-1");
  IupSetAttribute(plot, "PROJECTVALUEY", "1");
  IupSetAttribute(plot, "PROJECTVALUEZ", "-1");
  IupSetAttribute(plot, "TRANSPARENT", "NO");
//  IupSetAttribute(plot, "LIGHT", "NO");
}

static void SampleContourFilledProjectVolume(void)
{
  SampleVolume("VOLUME_CONTOUR");
  IupSetAttribute(plot, "PROJECT", "YES");
  IupSetAttribute(plot, "PROJECTVALUEX", "-1");
  IupSetAttribute(plot, "PROJECTVALUEY", "1");
  IupSetAttribute(plot, "PROJECTVALUEZ", "-1");
  IupSetAttribute(plot, "CONTOURFILLED", "YES");
  IupSetAttribute(plot, "TRANSPARENT", "NO");
//  IupSetAttribute(plot, "LIGHT", "NO");
}

static void SampleContourFilledVolume(void)
{
  SampleVolume("VOLUME_CONTOUR");
  IupSetAttribute(plot, "CONTOURFILLED", "YES");
  IupSetAttribute(plot, "TRANSPARENT", "NO");
}

static void SampleContourVolume(void)
{
  SampleVolume("VOLUME_CONTOUR");
  IupSetAttribute(plot, "TRANSPARENT", "NO");
//  IupSetAttribute(plot, "LIGHT", "NO");
}

static void SampleDensityVolume(void)
{
  SampleVolume("VOLUME_DENSITY");
  IupSetAttribute(plot, "AXS_XORIGIN", "0");
  IupSetAttribute(plot, "AXS_YORIGIN", "0");
  IupSetAttribute(plot, "AXS_ZORIGIN", "0");
  IupSetAttribute(plot, "AXS_X", "Yes");
  IupSetAttribute(plot, "AXS_Y", "Yes");
  IupSetAttribute(plot, "AXS_Z", "Yes");
  IupSetAttribute(plot, "LIGHT", "NO");
}
```

```c
static void SampleCloudVolume(void)
{
  SampleVolume("VOLUME_CLOUD");
  IupSetAttribute(plot, "COLORSCHEME", "wyrRk");
  IupSetAttribute(plot, "CLOUDCUBES", "NO");
  IupSetAttribute(plot, "LIGHT", "NO");
}

static void SampleCloudCubesVolume(void)
{
  SampleVolume("VOLUME_CLOUD");
  IupSetAttribute(plot, "COLORSCHEME", "wyrRk");
  IupSetAttribute(plot, "LIGHT", "NO");
}

static void SampleIsoSurfaceVolume(void)
{
  SampleVolume("VOLUME_ISOSURFACE");
}

static void SamplePlanar(const char* ds_mode)
{
  int ds = IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, ds, "0.6*sin(2*pi*x)*sin(3*pi*y) + 0.4*cos(3*pi*(x*y))", 50, 40, 1);
  IupSetAttribute(plot, "DS_MODE", ds_mode);

  IupSetAttribute(plot, "ROTATE", "40:0:60");
  IupSetAttribute(plot, "LIGHT", "YES");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleGradientLinesPlanar(void)
{
  SamplePlanar("PLANAR_GRADIENTLINES");
  SamplePlanar("PLANAR_DENSITY");
  IupSetAttribute(plot, "TRANSPARENT", "YES");
  IupSetAttribute(plot, "ROTATE", NULL);
}

static void SampleAxialContourPlanar(void)
{
  SamplePlanar("PLANAR_AXIALCONTOUR");
  IupSetAttribute(plot, "TRANSPARENT", "YES");
}

static void SampleContourFilledPlanar(void)
{
  SamplePlanar("PLANAR_CONTOUR");
  IupSetAttribute(plot, "CONTOURFILLED", "Yes");
}

static void SampleContourPlanar(void)
{
  SamplePlanar("PLANAR_CONTOUR");
  IupSetAttribute(plot, "CONTOURLABELS", "BELLOW");
}

static void SampleDensityPlanar(void)
{
  SamplePlanar("PLANAR_DENSITY");
  IupSetAttribute(plot, "COLORBAR", "Yes");
  IupSetAttribute(plot, "ROTATE", NULL);
}

static void SampleBoxesPlanar(void)
{
```

```
  SamplePlanar("PLANAR_BOXES");
  IupSetAttribute(plot, "AXS_XORIGIN", "0");
  IupSetAttribute(plot, "AXS_YORIGIN", "0");
  IupSetAttribute(plot, "AXS_ZORIGIN", "0");
}

static void SampleTilePlanar(void)
{
  SamplePlanar("PLANAR_TILE");
}

static void SampleBeltPlanar(void)
{
  SamplePlanar("PLANAR_BELT");
}

static void SampleFallPlanar(void)
{
  SamplePlanar("PLANAR_FALL");
}

static void SampleMeshPlanar(void)
{
  SamplePlanar("PLANAR_MESH");
}

static void SampleSurfaceColorsPlanar(void)
{
  SamplePlanar("PLANAR_SURFACE");
  IupSetAttribute(plot, "COLORSCHEME", "BbcyrR|");
}

static void SampleSurfacePlanar(void)
{
  SamplePlanar("PLANAR_SURFACE");
}

static void SampleSurfaceContourPlanar(void)
{
  SamplePlanar("PLANAR_SURFACE");
  SamplePlanar("PLANAR_CONTOUR");
}

static void SampleDotsLinear3D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotLoadData(plot, 0, "../test/hotdogs.pts", 0, 0, 0);
  IupSetAttribute(plot, "DS_MODE", "DOTS");
  IupSetAttribute(plot, "DS_REARRANGE", NULL);

  IupSetAttribute(plot, "ROTATE", "40:0:60");
  IupSetAttribute(plot, "LIGHT", "YES");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleCrustLinear3D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotLoadData(plot, 0, "../test/hotdogs.pts", 0, 0, 0);
  IupSetAttribute(plot, "DS_MODE", "CRUST");
  IupSetAttribute(plot, "DS_REARRANGE", NULL);

  IupSetAttribute(plot, "ROTATE", "40:0:60");
  IupSetAttribute(plot, "LIGHT", "YES");
  IupSetAttribute(plot, "BOX", "YES");
}
```

```c
static void SamplePieLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "rnd+0.1", 7, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "CHART");

  IupSetAttribute(plot, "COLORSCHEME", "bgr cmy");
  IupSetAttribute(plot, "PIECHART", "Yes");
  IupSetAttribute(plot, "DATAGRID", "Yes");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleChartLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "rnd+0.1", 7, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "CHART");
  IupSetAttribute(plot, "DATAGRID", "Yes");

  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleStemLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "0.7*sin(2*pi*x) + 0.5*cos(3*pi*x) + 0.2*sin(pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "STEM");
  IupSetAttribute(plot, "DS_MARKSTYLE", "HOLLOW_CIRCLE");
  IupSetAttribute(plot, "DS_SHOWMARKS", "Yes");

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 1, "sin(2*pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "STEM");
  IupSetAttribute(plot, "DS_MARKSTYLE", "HOLLOW_CIRCLE");
  IupSetAttribute(plot, "DS_SHOWMARKS", "Yes");

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 2, "cos(2*pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "STEM");
  IupSetAttribute(plot, "DS_MARKSTYLE", "HOLLOW_CIRCLE");
  IupSetAttribute(plot, "DS_SHOWMARKS", "Yes");

  IupSetAttribute(plot, "AXS_XORIGIN", "0");
  IupSetAttribute(plot, "AXS_YORIGIN", "0");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleStepLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "0.7*sin(2*pi*x) + 0.5*cos(3*pi*x) + 0.2*sin(pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "STEP");

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 1, "sin(2*pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "STEP");

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 2, "cos(2*pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "STEP");

  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleBarhLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
```

```
  IupMglPlotSetFromFormula(plot, 0, "0.8*sin(pi*(2*x+y/2))+0.2*rnd", 10, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "BARHORIZONTAL");

  IupSetAttribute(plot, "AXS_XORIGIN", "0");
  IupSetAttribute(plot, "AXS_YORIGIN", "0");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleBarsLinear3D(void)
{
  IupMglPlotNewDataSet(plot, 3);
  IupMglPlotSetFormula(plot, 0, "cos(pi*2*x-pi)", "sin(pi*(2*x-1))", "2*x-1", 50);
  IupSetAttribute(plot, "DS_MODE", "BAR");
  IupSetAttribute(plot, "DS_LINESTYLE", "SMALLDASH_DOT");

  IupSetAttribute(plot, "ROTATE", "40:0:60");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleBarsLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "0.8*sin(pi*(2*x+y/2))+0.2*rnd", 10, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "BAR");

  //TODO: allow combination of several datasets into one bar plot
  //TODO: allow bars "above" sample

  IupSetAttribute(plot, "AXS_XORIGIN", "0");
  IupSetAttribute(plot, "AXS_YORIGIN", "0");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleAreaLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "0.7*sin(2*pi*x) + 0.5*cos(3*pi*x) + 0.2*sin(pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "AREA");

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 1, "sin(2*pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "AREA");

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 2, "cos(2*pi*x)", 50, 1, 1);
  IupSetAttribute(plot, "DS_MODE", "AREA");

  IupSetAttribute(plot, "AXS_XORIGIN", "0");
  IupSetAttribute(plot, "AXS_YORIGIN", "0");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleRadarLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "0.4*sin(pi*(2*x+y/2))+0.1*rnd", 10, 3, 1);
  IupSetAttribute(plot, "DS_MODE", "RADAR");
  IupSetAttribute(plot, "DS_SPLIT", NULL);

  IupSetAttribute(plot, "CURRENT", "1");
  IupSetAttribute(plot, "DS_MODE", "RADAR");

  IupSetAttribute(plot, "CURRENT", "2");
  IupSetAttribute(plot, "DS_MODE", "RADAR");

  IupSetAttribute(plot, "RADARSHIFT", "0.4");  // So all datasets will use the same radarshift
  IupSetAttribute(plot, "DATAGRID", "Yes");
```

```
  IupSetAttribute(plot, "BOX", "YES");
}

static void SamplePlotLinear3D(void)
{
  IupMglPlotNewDataSet(plot, 3);
  IupMglPlotSetFormula(plot, 0, "cos(pi*2*x-pi)", "sin(pi*(2*x-1))", "2*x-1", 50);
  IupSetAttribute(plot, "DS_MODE", "LINE");

  IupSetAttribute(plot, "ROTATE", "40:0:60");
  IupSetAttribute(plot, "BOX", "YES");
}

static void SamplePlotLinear1D(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 0, "0.7*sin(2*pi*x)+0.5*cos(3*pi*x)+0.2*sin(pi*x)", 50, 1, 1);

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 1, "sin(2*pi*x)", 50, 1, 1);

  IupMglPlotNewDataSet(plot, 1);
  IupMglPlotSetFromFormula(plot, 2, "cos(2*pi*x)", 50, 1, 1);

  IupSetAttribute(plot, "BOX", "YES");
}

static int postdraw_cb(Ihandle* ih)
{
  IupMglPlotDrawText(ih, "It can be \\wire{wire}, \\big{big} or #r{colored}", 0, 1.0f, 0);
  IupMglPlotDrawText(ih, "One can change style in string: " "\\b{bold}, \\i{italic, \\b{both}}", 0, 0.6f,
0);
  IupMglPlotDrawText(ih, "Easy to \\a{overline} or \\u{underline}", 0, 0.2f, 0);
  IupMglPlotDrawText(ih, "Easy to change indexes ^{up} _{down} @{center}", 0, -0.2f, 0);
  IupMglPlotDrawText(ih, "It parse TeX: \\int \\alpha \\sqrt{sin(\\pi x)^2 + \\gamma_{i_k}} dx", 0,
-0.6f, 0);
//  IupMglPlotDrawText(ih, "It parse TeX: \\int \\alpha \\cdot \\sqrt3{sin(\\pi x)^2 + \\gamma_{i_k}}
dx", 0, -0.6f, 0);
  IupMglPlotDrawText(ih, "And more TeX: \\sqrt{\\frac{\\alpha^{\\gamma^2}+" "\\overset 1{\\big\\infty}}{\
\sqrt{2+b}}}", 0, -1.0f, 0);
//  IupMglPlotDrawText(ih, "And more TeX: \\sqrt{\\frac{\\alpha^{\\gamma^2}+" "\\overset 1{\\big\\infty}}
{\\sqrt3{2+b}}}", 0, -1.0f, 0);
  return IUP_DEFAULT;
}

static void SampleText(void)
{
  IupSetCallback(plot, "POSTDRAW_CB", (Icallback)postdraw_cb);
}

static void SampleLegend(void)
{
  IupMglPlotNewDataSet(plot, 1);
  IupSetAttribute(plot, "DS_LEGEND", "sin(\\pi {x^2})");
  IupMglPlotSetFromFormula(plot, 0, "sin(2*pi*x*x)", 50, 1, 1);

  IupMglPlotNewDataSet(plot, 1);
  IupSetAttribute(plot, "DS_LEGEND", "sin(\\pi x)");
  IupMglPlotSetFromFormula(plot, 1, "sin(2*pi*x)", 50, 1, 1);

  IupMglPlotNewDataSet(plot, 1);
  IupSetAttribute(plot, "DS_LEGEND", "sin(\\pi \\sqrt{\\a x})");
  IupMglPlotSetFromFormula(plot, 2, "sin(2*pi*sqrt(x))", 50, 1, 1);

  IupSetAttribute(plot, "LEGEND", "YES");
  IupSetAttribute(plot, "AXS_X", "Yes");
  IupSetAttribute(plot, "AXS_Y", "Yes");
```

```
  IupSetAttribute(plot, "BOX", "YES");
}

static void SampleSemiLog(void)
{
  IupMglPlotNewDataSet(plot, 2);
  IupMglPlotSetFormula(plot, 0, "0.01/(x+10^(-5))", "sin(1/x)", NULL, 2000);
  IupSetAttribute(plot, "DS_COLOR", "0 0 255");
  IupSetAttribute(plot, "DS_LINEWIDTH", "2");

  IupSetAttribute(plot, "AXS_XSCALE", "LOG10");
  IupSetAttribute(plot, "AXS_X", "Yes");
  IupSetAttribute(plot, "AXS_Y", "Yes");
  IupSetAttribute(plot, "AXS_XLABEL", "x");
  IupSetAttribute(plot, "AXS_YLABEL", "y = sin 1/x");
  IupSetAttribute(plot, "BOX", "YES");
  IupSetAttribute(plot, "GRID", "YES");
  IupSetAttribute(plot, "GRIDCOLOR", "0 255 0");
}

static void SampleLogLog(void)
{
  IupMglPlotNewDataSet(plot, 2);
  IupMglPlotSetFormula(plot, 0, "pow(10,6*x-3)", "sqrt(1+x^2)", NULL, 100);
  IupSetAttribute(plot, "DS_COLOR", "0 0 255");
  IupSetAttribute(plot, "DS_LINEWIDTH", "2");

  IupSetAttribute(plot, "AXS_XSCALE", "LOG10");
  IupSetAttribute(plot, "AXS_YSCALE", "LOG10");
  IupSetAttribute(plot, "AXS_X", "Yes");
  IupSetAttribute(plot, "AXS_Y", "Yes");
  IupSetAttribute(plot, "AXS_XLABEL", "x");
  IupSetAttribute(plot, "AXS_YLABEL", "y=\\sqrt{1+x^2}");
  IupSetAttribute(plot, "BOX", "YES");
  IupSetAttribute(plot, "GRID", "YES");
  IupSetAttribute(plot, "GRIDCOLOR", "0 255 0");
  IupSetAttribute(plot, "GRIDLINESTYLE", "DASHED");
}
#if 0
Semi-Log
gr->Axis(mglPoint(0.01,-1),mglPoint(1000,1),mglPoint(0.01,-1));
Modify Y is relative to V and uses X

Log-Log
gr->Axis(mglPoint(0.001,0.1),mglPoint(1000,1000),mglPoint(0.001,0.1));
Modify Y is relative to V and uses X
#endif

static void Dummy(void)
{
}

typedef struct _TestItems{
  char* title;
  void (*func)(void);
}TestItems;

static TestItems test_list[] = {
  {"Plot (Linear 1D)", SamplePlotLinear1D},
  {"Plot (Linear 3D)", SamplePlotLinear3D},
  {"Radar (Linear 1D)", SampleRadarLinear1D},
  {"Area (Linear 1D)", SampleAreaLinear1D},
  {"Bars (Linear 1D)", SampleBarsLinear1D},
  {"Bars (Linear 3D)", SampleBarsLinear3D},
  {"Barh (Linear 1D)", SampleBarhLinear1D},
  {"Step (Linear 1D)", SampleStepLinear1D},
  {"Stem (Linear 1D)", SampleStemLinear1D},
```

```
  {"Chart (Linear 1D)", SampleChartLinear1D},
  {"Pie (Linear 1D)", SamplePieLinear1D},
  {"Dots (Linear 3D)", SampleDotsLinear3D},
  {"Crust (Linear 3D)", SampleCrustLinear3D},
  {"----------", Dummy},
  {"Surface (Planar)", SampleSurfacePlanar},
  {"Surface Colors (Planar)", SampleSurfaceColorsPlanar},
  {"Surface Contour (Planar)", SampleSurfaceContourPlanar},
  {"Mesh (Planar)", SampleMeshPlanar},
  {"Fall (Planar)", SampleFallPlanar},
  {"Belt (Planar)", SampleBeltPlanar},
  {"Tile (Planar)", SampleTilePlanar},
  {"Boxes (Planar)", SampleBoxesPlanar},
  {"Density (Planar)", SampleDensityPlanar},
  {"Contour (Planar)", SampleContourPlanar},
  {"Contour Filled (Planar)", SampleContourFilledPlanar},
  {"Axial Contour (Planar)", SampleAxialContourPlanar},
  {"GradientLines (Planar)", SampleGradientLinesPlanar},
  {"----------", Dummy},
  {"Iso Surface (Volume)", SampleIsoSurfaceVolume},
  {"CloudCubes (Volume)", SampleCloudCubesVolume},
  {"Cloud (Volume)", SampleCloudVolume},
  {"Density (Volume)", SampleDensityVolume},
  {"Contour (Volume)", SampleContourVolume},
  {"ContourFilled (Volume)", SampleContourFilledVolume},
  {"ContourProject (Volume)", SampleContourProjectVolume},
  {"ContourFilledProject (Volume)", SampleContourFilledProjectVolume},
  {"DensityProject (Volume)", SampleDensityProjectVolume},
  {"----------", Dummy},
  {"Text Styles", SampleText},
  {"Legend", SampleLegend},
  {"Semi-log", SampleSemiLog},
  {"Log-log", SampleLogLog},
};

static void ChangePlot(int item)
{
  ResetClear();
  test_list[item].func();
  UpdateFlags();
  sprintf(filenameSVG, "../%s.svg", test_list[item].title);
  sprintf(filenameEPS, "../%s.eps", test_list[item].title);
  sprintf(filenamePNG, "../%s.png", test_list[item].title);
//  sprintf(filenamePNG, "../html/en/ctrl/images/iupmglplot_%s.png", IupGetAttribute(plot, "DS_MODE"));
//  iupStrLower(filenamePNG, filenamePNG);
  IupSetAttribute(plot, "REDRAW", NULL);

  {
    char* errmsg = IupGetAttribute(plot, "ERRORMESSAGE");
    if (errmsg)
      IupMessage("Error", errmsg);
  }
}

static int k_enter_cb(Ihandle*ih)
{
  int pos = IupGetInt(ih, "VALUE");
  if (pos > 0)
    ChangePlot(pos-1);
  return IUP_DEFAULT;
}

static int action_cb(Ihandle *ih, char *text, int item, int state)
{
  (void)text;
  (void)ih;
```

```
  if (state==1)
    ChangePlot(item-1);

  return IUP_DEFAULT;
}

static int close_cb(Ihandle *ih)
{
  (void)ih;
  return IUP_CLOSE;
}

/* Y zoom */
static int minmaxY_dial_btndown_cb(Ihandle *self, double angle)
{
  (void)angle;
  (void)self;

  IupStoreAttribute(plot, "OLD_YMIN", IupGetAttribute(plot, "AXS_YMIN"));
  IupStoreAttribute(plot, "OLD_YMAX", IupGetAttribute(plot, "AXS_YMAX"));

  return IUP_DEFAULT;
}

static int minmaxY_dial_btnup_cb(Ihandle *self, double angle)
{
  double x1, x2, xm;
  char *ss;
  (void)self;

  x1 = IupGetFloat(plot, "OLD_YMIN");
  x2 = IupGetFloat(plot, "OLD_YMAX");

  ss = IupGetAttribute(plot, "AXS_YMODE");
  if ( ss && ss[3]=='2' )
  {
    /* LOG2:  one circle will zoom 2 times */
    xm = 4.0 * fabs(angle) / 3.141592;
    if (angle>0.0) { x2 /= xm; x1 *= xm; }
    else { x2 *= xm; x1 /= xm; }
  }

  if ( ss && ss[3]=='1' )
  {
    /* LOG10:  one circle will zoom 10 times */
    xm = 10.0 * fabs(angle) / 3.141592;
    if (angle>0.0) { x2 /= xm; x1 *= xm; }
    else { x2 *= xm; x1 /= xm; }
  }
  else
  {
    /* LIN: one circle will zoom 2 times */
    xm = (x1 + x2) / 2.0;
    x1 = xm - (xm - x1)*(1.0-angle*1.0/3.141592);
    x2 = xm + (x2 - xm)*(1.0-angle*1.0/3.141592);
  }

  if (x1<x2)
  {
    IupSetfAttribute(plot, "AXS_YMIN", "%g", x1);
    IupSetfAttribute(plot, "AXS_YMAX", "%g", x2);
  }

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}
```

```c
/* X zoom */
static int minmaxX_dial_btndown_cb(Ihandle *self, double angle)
{
  (void)angle;
  (void)self;

  IupStoreAttribute(plot, "OLD_XMIN", IupGetAttribute(plot, "AXS_XMIN"));
  IupStoreAttribute(plot, "OLD_XMAX", IupGetAttribute(plot, "AXS_XMAX"));

  return IUP_DEFAULT;
}

static int minmaxX_dial_btnup_cb(Ihandle *self, double angle)
{
  double x1, x2, xm;
  (void)self;

  x1 = IupGetFloat(plot, "OLD_XMIN");
  x2 = IupGetFloat(plot, "OLD_XMAX");

  xm = (x1 + x2) / 2.0;

  x1 = xm - (xm - x1)*(1.0-angle*1.0/3.141592); /* one circle will zoom 2 times */
  x2 = xm + (x2 - xm)*(1.0-angle*1.0/3.141592);

  IupSetfAttribute(plot, "AXS_XMIN", "%g", x1);
  IupSetfAttribute(plot, "AXS_XMAX", "%g", x2);

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* auto scale Y */
static int autoscaleY_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
  {
    IupSetAttribute(minmaxY_dial, "ACTIVE", "NO");
    IupSetAttribute(plot, "AXS_YAUTOMIN", "YES");
    IupSetAttribute(plot, "AXS_YAUTOMAX", "YES");
  }
  else
  {
    IupSetAttribute(minmaxY_dial, "ACTIVE", "YES");
    IupSetAttribute(plot, "AXS_YAUTOMIN", "NO");
    IupSetAttribute(plot, "AXS_YAUTOMAX", "NO");
  }

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* auto scale X */
static int autoscaleX_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
  {
    IupSetAttribute(minmaxX_dial, "ACTIVE", "NO");
    IupSetAttribute(plot, "AXS_XAUTOMIN", "YES");
    IupSetAttribute(plot, "AXS_XAUTOMAX", "YES");
```

```
  }
  else
  {
    IupSetAttribute(minmaxX_dial, "ACTIVE", "YES");
    IupSetAttribute(plot, "AXS_XAUTOMIN", "NO");
    IupSetAttribute(plot, "AXS_XAUTOMAX", "NO");
  }

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* show/hide grid */
static int grid_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "GRID", "YES");
  else
    IupSetAttribute(plot, "GRID", "NO");

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* show/hide legend */
static int legend_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "LEGEND", "YES");
  else
    IupSetAttribute(plot, "LEGEND", "NO");

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* show/hide box */
static int box_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "BOX", "YES");
  else
    IupSetAttribute(plot, "BOX", "NO");

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* enable/disable antialias */
static int aa_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "ANTIALIAS", "YES");
  else
    IupSetAttribute(plot, "ANTIALIAS", "NO");
```

```
  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* enable/disable transparent */
static int transp_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "TRANSPARENT", "YES");
  else
    IupSetAttribute(plot, "TRANSPARENT", "NO");

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* enable/disable light */
static int light_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "LIGHT", "YES");
  else
    IupSetAttribute(plot, "LIGHT", "NO");

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

/* enable/disable opengl */
static int opengl_tgg_cb(Ihandle *self, int v)
{
  (void)self;

  if (v)
    IupSetAttribute(plot, "OPENGL", "YES");
  else
  {
    IupSetAttribute(aa_tgg, "VALUE", "OFF");
    IupSetAttribute(plot, "ANTIALIAS", "NO");
    IupSetAttribute(plot, "OPENGL", "NO");
  }

  IupSetAttribute(plot, "REDRAW", NULL);

  return IUP_DEFAULT;
}

static int bt1_cb(Ihandle *self)
{
  (void)self;
  IupMglPlotPaintTo(plot, "SVG", 0, 0, 0, filenameSVG);
  return IUP_DEFAULT;
}

static int bt2_cb(Ihandle *self)
{
  (void)self;
  IupMglPlotPaintTo(plot, "EPS", 0, 0, 0, filenameEPS);
  return IUP_DEFAULT;
```

```
}

#ifdef USE_IM
static int bt3_cb(Ihandle* self)
{
  imImage* image;
  Ihandle* clipboard = IupClipboard();
  int w, h;
  void* gldata;
  IupGetIntInt(plot, "DRAWSIZE", &w, &h);
  gldata = malloc(w*h*3);
  image = imImageCreate(w, h, IM_RGB, IM_BYTE);
  IupMglPlotPaintTo(plot, "RGB", w, h, 0, gldata);
  imConvertPacking(gldata, image->data[0], w, h, 3, 3, IM_BYTE, 1);
  imProcessFlip(image, image);
  IupSetAttribute(clipboard, "NATIVEIMAGE", IupGetImageNativeHandle(image));
  IupDestroy(clipboard);
  free(gldata);
  imImageDestroy(image);
  (void)self;
  return IUP_DEFAULT;
}

static int bt4_cb(Ihandle* self)
{
  imImage* image;
  int w, h;
  void* gldata;
  IupGetIntInt(plot, "DRAWSIZE", &w, &h);
  gldata = malloc(w*h*3);
  image = imImageCreate(w, h, IM_RGB, IM_BYTE);
  IupMglPlotPaintTo(plot, "RGB", w, h, 0, gldata);
  imConvertPacking(gldata, image->data[0], w, h, 3, 3, IM_BYTE, 1);
  imProcessFlip(image, image);
  imFileImageSave(filenamePNG, "PNG", image);
  free(gldata);
  imImageDestroy(image);
  (void)self;
  return IUP_DEFAULT;
}
#endif

Ihandle* controlPanel(void)
{
  Ihandle *vbox1, *vbox2, *vbox3, *lbl1, *lbl2, *lbl3, *lbl4,
    *bt1, *bt2, *bt3=NULL, *bt4=NULL,
    *boxminmaxY_dial, *boxminmaxX_dial, *f1, *f2;

  /* left panel: plot control
     Y zooming               */
  lbl1 = IupLabel("+");
  lbl2 = IupLabel("-");

  minmaxY_dial = IupDial("VERTICAL");
  IupSetAttribute(minmaxY_dial, "ACTIVE", "NO");
  IupSetAttribute(minmaxY_dial, "SIZE", "20x52");
  IupSetCallback(minmaxY_dial, "BUTTON_PRESS_CB", (Icallback)minmaxY_dial_btndown_cb);
  IupSetCallback(minmaxY_dial, "MOUSEMOVE_CB", (Icallback)minmaxY_dial_btnup_cb);
  IupSetCallback(minmaxY_dial, "BUTTON_RELEASE_CB", (Icallback)minmaxY_dial_btnup_cb);

  boxminmaxY_dial = IupVbox(lbl1, minmaxY_dial, lbl2, NULL);
  IupSetAttribute(boxminmaxY_dial, "ALIGNMENT", "ACENTER");
  IupSetAttribute(boxminmaxY_dial, "GAP", "2");
  IupSetAttribute(boxminmaxY_dial, "MARGIN", "2");

  autoscaleY_tgg = IupToggle("Y Autoscale", NULL);
  IupSetCallback(autoscaleY_tgg, "ACTION", (Icallback)autoscaleY_tgg_cb);
```

```
  IupSetAttribute(autoscaleY_tgg, "VALUE", "ON");

  f1 = IupFrame( IupVbox(boxminmaxY_dial, autoscaleY_tgg, NULL) );
  IupSetAttribute(f1, "TITLE", "Y Zoom");
  IupSetAttribute(f1, "GAP", "0");
  IupSetAttribute(f1, "MARGIN", "5x5");

  /* X zooming */
  lbl1 = IupLabel("-");
  lbl2 = IupLabel("+");

  minmaxX_dial = IupDial("HORIZONTAL");
  IupSetAttribute(minmaxX_dial, "ACTIVE", "NO");
  IupSetAttribute(minmaxX_dial, "SIZE", "64x16");
  IupSetCallback(minmaxX_dial, "BUTTON_PRESS_CB", (Icallback)minmaxX_dial_btndown_cb);
  IupSetCallback(minmaxX_dial, "MOUSEMOVE_CB", (Icallback)minmaxX_dial_btnup_cb);
  IupSetCallback(minmaxX_dial, "BUTTON_RELEASE_CB", (Icallback)minmaxX_dial_btnup_cb);

  boxminmaxX_dial = IupHbox(lbl1, minmaxX_dial, lbl2, NULL);
  IupSetAttribute(boxminmaxX_dial, "ALIGNMENT", "ACENTER");
  IupSetAttribute(boxminmaxX_dial, "GAP", "2");
  IupSetAttribute(boxminmaxX_dial, "MARGIN", "2");

  autoscaleX_tgg = IupToggle("X Autoscale", NULL);
  IupSetCallback(autoscaleX_tgg, "ACTION", (Icallback)autoscaleX_tgg_cb);

  f2 = IupFrame( IupVbox(boxminmaxX_dial, autoscaleX_tgg, NULL) );
  IupSetAttribute(f2, "TITLE", "X Zoom");
  IupSetAttribute(f2, "GAP", "0");
  IupSetAttribute(f2, "MARGIN", "5x5");

  grid_tgg = IupToggle("Grid", NULL);
  IupSetCallback(grid_tgg, "ACTION", (Icallback)grid_tgg_cb);

  box_tgg = IupToggle("Box", NULL);
  IupSetCallback(box_tgg, "ACTION", (Icallback)box_tgg_cb);

  legend_tgg = IupToggle("Legend", NULL);
  IupSetCallback(legend_tgg, "ACTION", (Icallback)legend_tgg_cb);

  lbl3 = IupLabel("");
  IupSetAttribute(lbl3, "SEPARATOR", "HORIZONTAL");
  IupSetAttribute(lbl3, "EXPAND", "NO");
  IupSetAttribute(lbl3, "RASTERSIZE", "160x2");

  transp_tgg = IupToggle("Transparent", NULL);
  IupSetCallback(transp_tgg, "ACTION", (Icallback)transp_tgg_cb);

  light_tgg = IupToggle("Light", NULL);
  IupSetCallback(light_tgg, "ACTION", (Icallback)light_tgg_cb);

  lbl4 = IupLabel("");
  IupSetAttribute(lbl4, "SEPARATOR", "HORIZONTAL");
  IupSetAttribute(lbl4, "EXPAND", "NO");
  IupSetAttribute(lbl4, "RASTERSIZE", "160x2");

  aa_tgg = IupToggle("Antialias", NULL);
  IupSetCallback(aa_tgg, "ACTION", (Icallback)aa_tgg_cb);

  opengl_tgg = IupToggle("OpenGL", NULL);
  IupSetCallback(opengl_tgg, "ACTION", (Icallback)opengl_tgg_cb);

  bt1 = IupButton("Export SVG", NULL);
  IupSetCallback(bt1, "ACTION", (Icallback)bt1_cb);

  bt2 = IupButton("Export EPS", NULL);
  IupSetCallback(bt2, "ACTION", (Icallback)bt2_cb);
```

```
#ifdef USE_IM
  bt3 = IupButton("Copy RGB", NULL);
  IupSetCallback(bt3, "ACTION", (Icallback)bt3_cb);

  bt4 = IupButton("Save RGB", NULL);
  IupSetCallback(bt4, "ACTION", (Icallback)bt4_cb);
#endif

  vbox1 = IupFrame(IupVbox(f1,
                           f2,
                           grid_tgg,
                           box_tgg,
                           legend_tgg,
                           lbl3,
                           transp_tgg,
                           light_tgg,
                           NULL));
  IupSetAttribute(vbox1, "GAP", "4");
  IupSetAttribute(vbox1, "MARGIN", "5x5");

  vbox2 = IupVbox(aa_tgg, opengl_tgg,
                  lbl4, IupVbox(bt1, bt2,
#ifdef USE_IM
                  bt3, bt4,
#endif
                  NULL),
                  NULL);
  IupSetAttribute(vbox2, "GAP", "4");
  IupSetAttribute(vbox2, "MARGIN", "5x0");

  vbox3 = IupVbox(vbox1, vbox2, NULL);
  IupSetAttribute(vbox3, "GAP", "7");
  IupSetAttribute(vbox3, "MARGIN", "0x0");

  return vbox3;
}

int main(int argc, char* argv[])
{
  int i, count = sizeof(test_list)/sizeof(TestItems);
  char str[50];
  Ihandle *dlg, *list, *panel;

  IupOpen(&argc, &argv);
  IupControlsOpen();
  IupMglPlotOpen();     /* init IupMGLPlot library */

  IupSetGlobal("MGLFONTS", "../etc/mglfonts");

  list = IupList(NULL);
  plot = IupMglPlot();
  panel = controlPanel();

  dlg = IupDialog(IupHbox(list, panel, IupVbox(IupFill(), plot, IupFill(), NULL), NULL));
  IupSetAttribute(dlg, "MARGIN", "10x10");
  IupSetAttribute(dlg, "GAP", "5");
  IupSetAttribute(dlg, "TITLE", "MathGL samples w/ IupMglPlot");
  IupSetCallback(dlg, "CLOSE_CB", close_cb);

  IupSetAttribute(plot, "RASTERSIZE", "700x500");  // Minimum initial size
//  IupSetAttribute(plot, "RASTERSIZE", "350x250");
//  IupSetAttribute(plot, "RASTERSIZE", "460x280");
//  IupSetAttribute(plot, "EXPAND", "NO");

  IupSetAttribute(list, "EXPAND", "VERTICAL");
  IupSetAttribute(list, "VISIBLELINES", "15");  // Not all, because the dialog will be too big
```

```
  IupSetAttribute(list, "VISIBLECOLUMNS", "15");
  IupSetCallback(list, "ACTION", (Icallback)action_cb);

  for (i=0; i<count; i++)
  {
    sprintf(str, "%d", i+1);
    IupSetAttribute(list, str, test_list[i].title);
  }

  IupSetAttribute(list, "VALUE", "1");

  IupShowXY(dlg, 100, IUP_CENTER);

  IupSetAttribute(plot, "RASTERSIZE", NULL);  // Clear initial size

  ChangePlot(0);

  IupMainLoop();

  IupClose();

  return EXIT_SUCCESS;
}
```