MPPG
C ++ library for graphical interface development at numerical modeling of physical processes

Pruuel Edward Reinovich, pru@hydro.nsc.ru

10th November 2006

Contents

It has appeared, it should to appear

Introduction

MPPG (Modeling Physical Phenomena Graphical User Interface) is a cross-platform C++ GUI library for UNIX/Linux (X11), Microsoft Windows. MPPG provides GUI for numeric calculations (windows with plots, line input/output for variable editing). Mppg is easy to start, functional enough and allows to attract attention to numeric calculation and don't spend much effort to GUI development.It is good decision for schoolboys, students and scientist, for all who develops and tuns numeric algorithms. It is currently maintained by Edward Pruuel (pru@hydro.nsc.ru).

MPPG dont provides any numercs algorithms or methods, it is only GUI developmet tool.

Features

- Uniform interface: canvas for graphics plotting, table with eitable spaces for input/output values of program variables.
- Easy add editing spaces for input/output values of program variables.
- Kit of graphics primitives for drawing with physicle scales.
- Save canvas to varies graphics formatd (tif, png , eps).

On 10.08.05 version 2.9.4 are available. (ftp://ancient.hydro.nsc.ru/local/mppg, user=ftp).

Supported platforms: X (UNIX), and Win32.

Supported compilers: GCC (GCC, CYGWIN, MINGW), Microsoft Visual C++.

MPPG—C++ library of classes and functions which helps to create graphical user interface for numeric calculation.

MPPG obviously and implicitly uses FLTK (the Fast Light Tool Kit), www.fltk.org.

1 Overview

1.1 Library compilation and installation

1.2 Dependencies

For mppg building you need fltk library. Unfortunate, at present we use unofficial fltk-1.2 brunch. Src files you can download from original site www.fltk.org or
ftp://ancient.hydro.nsc.ru/local/home_page/fltk. We recommend last way.

Compile fltk with next command lines:

```
cd ./fltk-1.2
 ./configure --enable-threads --prefix=/home/kkk
 make
 make install
```

After, create directory /usr/local/share/fltk-1.2 and copy there FL directory and all libraries from /home/kk/lib. Don't forgot delete /home/kkk directory.

For grabbing canvas to different graphics format files you need install Ghostscript utility (http://www.cs.wisc.edu/ ghost).

Now you can compile mppg library.

1.2.1 GNU utilities

You can use GNU utilities for library ant test compilation.

```
./configure [options]
make
make install
```

.

To start your own mppg project you can use template in user directory. There you cat find two files makefile and mppg_hello.cpp. Try compile it.

1.2.2 Microsoft Visual C++

Unfortunately this part of document out of date. At present no Microsoft Visual C++ support. (Volunteers wanted)

Start the project msvs/mpp_lib.dsw, compile library and tests, independently copy heading files and library to directories of the compiler.

1.3 Naming

All open parts MPPG are placed in names mpp.

- Names of classes begin with a capital letter: Table, Mpp_window.
- Names of functions are written by waste letters, parts the compound name are joined through underlining: add_item(...);
- All heading files are in the catalogue ¡mpp/(...)¿.

1.4 Example of program

Let's present an simple example of MPPG application.

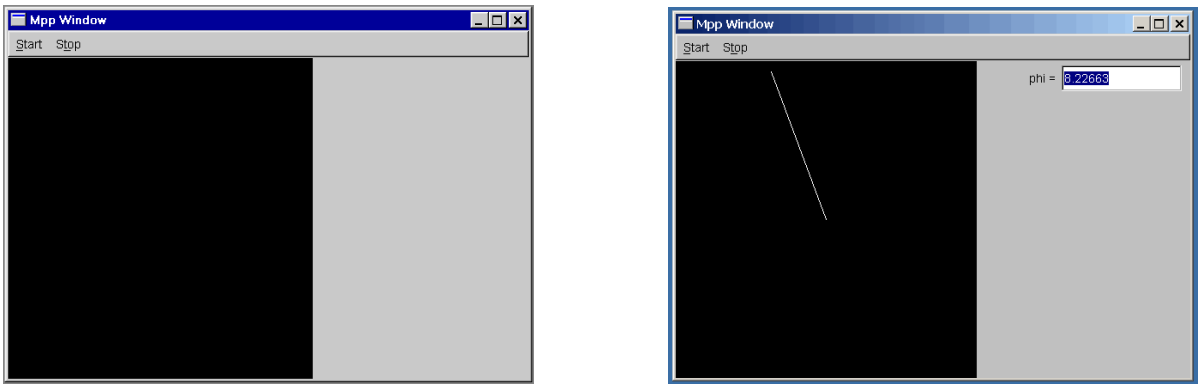
```
#include <mpp/mpp_window.h>
using namespace mpp;
int main () {
   Mpp_window win;  // create window
   return run ();   // start main events loop
}
```

Its appearance is represented on fig. 1.

1.5 MPPG parts interaction

Example of more realistic application.

```
#include <cmath>
#include <mpp/mpp_window.h>
#include <mpp/draw.h>
using namespace mpp;
Scale scale (-1, 1,-1, 1);
double phi=0, dphi=1e-6;
```



a b

Figure 1: Examples of MPPG application.

```
 void ud () {                          // that we shall draw
     set_scale (scale);
     set_color (FL_WHITE);
     line (0,0, cos (phi), sin (phi)); // draw a line
 }
 void f () {                    // calc function
     while (may_i_repeat()) {
          phi + = dphi;
          ask_update ();        // we ask to redraw all screens and update all tables
 } // and to update the table
 }
 int main () {
     Mpp_window win;        // we create basic window with the table and area for drawi
     win.canvas.add_shape (new User_draw (ud)); // add function for drawing
     win.table.add_item (" phi = ", phi); // add variable editor
     set_work (f);                 // set function f to start in separate thread
     start ();                     // start settlement function
     return run ();                // start main events loop
}
```

MPPG application can consist of unlimited amount of windows (objects such as Mpp_window, Table and Canvas).

MPPG application are possible only two conditions: calculation and editing. Any other conditions cannot be. Switching it is carried out by means of buttons "Start" and "Stop" or from programs functions start() and stop().

In a condition calculation it is started settlement functions, in which are basic computing actions occur. On the timer all objects (tables, canvases are updated...), marked as demanding updating. To what or from objects can mark, as demanding updating having caused his method ask_update(). It is possible to mark all objects in the program having caused function ask_update().

For redrawing a canvas method redraw() are called, for each shapes in the canvas list. Thus, each of figures redrawing corresponding with the new data. MPPG provides a set of useful graphic objects. If this set not satisfies, it is possible to use User_draw. This class allows to draw directly by graphic primitives of MPPG and fltk.

1.6 Peculiarity and alternatives

MPPG provides to way to set calculation function for application. Calculation function work in separate thread with small priority (set_work (f)). Calculation function periodically executes is in the same thread with main event loop (set_work2 (f)). Each of ways has advantages and disadvantages, in different cases you may use one, other or both.

If function is executed in the same thread, it removes problem of synchronization, but restrict the period of her work; while it is executed the application does not react on events. It is supposed, that all used by function variables have global visibility. It not the most convenient decision, its advantage—simplicity interaction of calc functions with the application. Calculation function can call any methods, in that number Table::ask_update(), anvas::ask_update(), ask_update().

The case when calc function executed in separate thread is more convenient. Main function cycle work while may_i_repeat(), otherwise calc function must finished, having left all global data used by her in valid state. In a case of multithreading the responsibility for synchronization lays on the user of library.

2 Classes and functions MPPG

2.1 Classes

Hierarchy of classes

- Widget
  - Group
    - Window
      - Canvas (Fl_Window)
      - Table (Fl_Window)
      - Mpp_Window (Fl_Window)
  - Input (Fl_Input)
- Manager (Fl_Window)
- Scale
- Shape
  - Bitmap
  - Circle
  - Grid
  - Hist
  - Orbit
  - Plot
  - Plot_a
  - Plot_v
  - Plot_l
  - Roll
  - User_draw

2.1.1 Base classes

Widget #include <mpp/mpp.h>
Base class for all MPPG widgets. All inherited classes call its constructor, in which is caused a registration, it allows to handle all objects.

Widget ()
virtual ~Widget()
virtual void ask_update()
virtual void update()
virtual void set_value()

virtual void ask_update()
Marks widget, as needy updating.
virtual void update()
Updates widget in corresponding with the current values of the data in the program.
virtual void set_value()
Reads out value from a field of input also updates a condition of a corresponding variable in to the program.

Group #include <mpp/mpp.h>
Group of widgets with which can be manipulated as one the whole. By a call of methods ask_update(), update() and set_value() recursively action is transferred to all members of group.

Group ()
void add(Widget *)
void mpp_begin()
void mpp_end()

Window #include <mpp/mpp.h>
A base class for all MPPG window, at it constructing the index on it is given the manager of the application, that allows to manipulate in further them through manager.

2.1.2 Basic graphic widgets

Canvas #include <mpp/canvas.h>
Widget providing space for drawing.

Canvas
~Canvas
add_shape
redraw

Canvas(const char* name)
Canvas(int w=500, int h=500, const char* name = "Canvas")
Canvas(int x, int y, int w=500, int h=500, const char * name = "anvas")
A canvas is created as separate window with the name *name*, through this name occurs identification windows by the manager; default constructor; canvas is created as a part of current window.
~Canvas ()
void add_shape(Shape *sh)
void add_shape(Shape & sh)
Adds the shape to the list of displayed objects. Nothing redraw. It is a unique method which determine what to be drawn.

Table #include <MPP/table.h>
Menu with edited fields of input (table).

Table
add_item
update
set

Table (const char * name)

Table (int w=250, int h=300, const char * name = "Table")
Table (int x, int y, int w, int h, const char * name=0)
Create an empty table.
template <class T> void add_item (const char * name, T* var, int deact=0)
template <class T> void add_item (const char * name, T & var, int deact=0)
Adds in the table an edited line with label variable to the address *var*. The flag *deact* specifies on creation of not edited field. A template instance for all types having operators of reading and record in a stream.
void update()
Updates lines in the table, reading out values from variables to the sting in table.
void set_value()
Updates global variables, reading out their values from the string in table.

Mpp_window #include <mpp/mpp_window.h>
Basic window of application. Contains area for drawing (Canvas) and a table.

Mpp_window	Canvas canvas
~Mpp_window	Table table

Mpp_window (int w=600, int h=400, const char *name = " Mpp Window ")
Mpp_window (const char *name)
Mpp_window:: ~Mpp_window ()

Manager #include <MPP/manager.h>
Window for application management. Allows to display others windows to start and stop the calculation. Automatically it is created at performance of function mpp::run().
Manager (int w=300, int h=200, const char * name = " MPP manager ")

2.1.3 Graphic objects

The library provides a set of useful figures for displays of the calculation data. All of them are derivated from abstract class Shape, at redrawing canvas runs throw list of figures also causes in everyone a method draw which makes necessary drawing. New graphic object are added to the canvas list by a calling canvas methods: add_shape(Shape *), add_shape (Shape &).

Shape #include<MPP/draw.h>
The abstract class for drawing be relative complex graphic objects (*hist*, *orbit*, ...). It is supposed to use them by means of *canvas.add_shape*.

Bitmap #include <MPP/draw.h>
Draw bidimentional array by shades grey.
Bitmap (const double *, int w, int h, double C1, double C2, const Scale *)
Bitmap (const double *, int w, int h, double C1, double C2, const Scale &)

Circle #include <MPP/draw.h>
The circle with corresponding parameters.
Circle (Float x, Float y, Float R, Fl_Color, Scale *)
Circle (Float x, Float y, Float R, Fl_Color, Scale &)

Grid #include <MPP/draw.h>
The rectangular grid from xmin with step dx while inside is drawn areas limited in the scale. Similarly on y.
Grid (double xmin, double dx, double ymin, double dy, Scale & scale)

Hist #include <MPP/draw.h> The histogram.
Hist (unsigned int nc, double xmin, double xmax, Fl_Color color1, Scale * scale, int legs=0)
Hist (unsigned int nc, double xmin, double xmax, Fl_Color color1, Scale & scale, int legs=0)
Builds the histogram from *nc* channels on an interval from *xmin* up to *xmax*.
void set (const double *x, unsigned int sz)
Allocates values from a array *x* lengths *sz* on channels of the histogram.
void add(const double *x, unsigned int sz)
it is similar *set*, but adds new events to old values in channels.

Orbit #include <MPP/draw.h>
Draws a trajectory by points.
Orbit (int sz_all, int sz_head, Fl_Color color_head, Fl_Color color_tail, Scale *)
Orbit (int sz_all, int sz_head, Fl_Color color_head, Fl_Color color_tail, Scale &)
void add (double x, double y)
Adds a new point to a trajectory.

Plot #include <MPP/draw.h>
Draws one iterator concerning another.
template ¡class T¿ class Plot (T x1, T x2, T y1, Fl_Color color, const Scale & s)
template ¡class T¿ class Plot (T x1, T x2, T y1, Fl_Color color, const Scale *s)

Plot_a #include <MPP/draw.h>
Draws one array versus another.
Plot_a(const double *x, const double *y, int sz, Fl_Color color, const Scale & scale)

Plot_v #include <MPP/draw.h>
Draws one vector versus another.
Plot_v (const std:: vector ¡double¿ &x, const std:: vector ¡double¿ &y, Fl _Color color, const Scale scale)

Plot_l #include <MPP/draw.h>
Draws the list of pairs points.
Plot_l (const std::list ¡std::pair ¡mpp::Float, mpp::Float¿¿ & pointl, Fl_Color color, const Scale & scale)

Roll #include <MPP/draw.h>
Set of points shifted to the left.
Roll (Fl_Color color1, Scale *s)
Roll (Fl_Color color1, Scale &s)
void add (double x, double y)
the new point is added.

User_draw #include <MPP/draw.h>
the User himself, with the help graphic , defines*determines* in Functions *ff* that will be drawn.
User_draw (void (*ff) ())

2.1.4 Stuff

Scale #include<MPP/scale.h>
A class allowing to draw in canvas in physical coordinates. The scale does not belong to any window, but adapts work of graphic primitives for any window with the help of functions void set_scale(const Scale *), void set_scale(const Scale &).

For transformation of coordinates and intervals from physical units in window functions are used: int transx(double x), int transy(double y), int scalex(double x), int scaley(double y).
Scale (Float x1, Float x2, Float y1, Float y2, bool xy=false)
Designs scale with corresponding material coordinates In a window. The argument xy establishes the identical sanction on Horizontals and verticals.
void set (Float x1, Float x2, Float y1, Float y2, bool xy=false)
Establishes new parameters for scale.

2.2 Functions

file_diaolog	scalex	set_work	transx
grabbing	scaley	start	transy
plot	set_scale	stop	
run	set_update_time	set _work	

2.2.1 Graphic primitives

A set of graphic primitives for drawing, which can be used only in User_draw. Before to use primitives, follows to expose attributes of drawing: set_color(Fl_Color), set_scale(const Scale *). Without the last scales can not work correctly.

plot #include¡MPP/draw.h¿
void point (Float x, Float y)
void point (Float x, Float y, Fl_Color color)
void pointb (Float x, Float y)
void pointb (Float x, Float y, Fl_Color color)
void line (Float x, Float y, Float x1, Float y1)
void line (Float x, Float y, Float x1, Float y1, Fl_Color color)
void circle (Float x, Float y, Float r)
void circle (Float x, Float y, Float r, Fl_Color color)

Function point finishs in the size 1+2 point_size, pointb puts Point in the size 1+2 point_size_b.
extern int point_size=0
extern int point_size_b=1

The sizes of points by default. If necessary they can be changed.
template ¡class T¿ void plot (T x1, T x2, T y1)
template ¡class T¿ void plot (const T & f, double xl, double xr, int n)
void plot (double (*) (double), double xl, double xr, int n)

x1, x2— on the beginning and the end of the container with x In coordinates, y1— by the beginning of the container with values y. The object function (f (double)) on an interval (xl, xr) on n is drawn To points. Function (f (double)) on an interval (xl, xr) on n is drawn To points.

2.2.2 Work with scales

A set of functions for transformation of coordinates and intervals from physical units to screen units. Correctly work only after establishments of attribute set_scale.

scalex #include<MPP/scale.h>
int scalex (Float x)
int scaley (Float y)

transx #include<MPP/scale.h>
int transx (Float x)
int transy (Float y)

set_scale #include<MPP/scale.h>
void set_scale (const Scale *)
void set_scale (const Scale & s)

2.2.3 Project management

set_work #include<MPP/mpp.h>
void set_work(void(*f)(void))
void set_work2(void*(*f)(void))
Establishes function f to start in a separate thread. Establishes function f for periodic start in the same thread.
int run()
the main events loop is started. Function finished when last window of the project is closed.
void start()
Set the application to "calculation" mode. If the mode "editing" was, takes away values from the table in corresponding variables. Starts updating the table on timer. Starts settlement functions. If the mode "calculation" was, is preliminary caused stop.
void stop ()
Set the application to "editing" mode. If the window was in a "calculation" mode any more does not start settlement function if it is started function in a separate thread exposes a flag askedit=true and waits end of a stream. After that updates the table and ceases to update it on the timer. If a window in a mode "editing" — makes nothing.
void ask_update()
Marks all widgets as demanding updating (for example redrawing for canvas).
void update ()
Updates all widgets demanding updating. Usually it is caused automatically by the timer.
void set_update_time(double time)
Establishes the period of check of all objects on necessity of updating in seconds. By default the size of period makes 0.25 .

2.2.4 Screen grabbing

unsigned char* grabbing (Fl_Window* , int x, int y, int dx, int dy)
unsigned char* grabbing (Fl_Window*)
void grabbing (Fl_Window* , int x, int y, int dx, int dy, const char* fname, int compr=COMPRESSION_DEFLATE)
inline void grabbing (Fl_Window*, const char* fname, int compr=COMPRESSION_NONE)

inline void grabbing (Fl_Window & win, const char* fname, int compr=COMPRESSION_NO
Copies a window to the buffer or file. For copying to tiff file format is supported some compressions COMPRESSION_NONE - without compression, COMPRESSION_DEFLATE - zip a compression without loss of quality (gives the best result on compression and quality of the image), COMPRESSION_JPEG - jpg a compression with loss of quality (it is recommended to apply to compression density filled images).

2.2.5 File choose dialog

char* file_dialog(const char * dialog_title=0, const char *pattern="All Fies (*.*)"0*.*"0", const char *fname="")
char * file_dialog_f(const char * message=0, const char *pattern=0, const char *fname=0)
Functions cause dialogue of a choice of a file and return a line with a name of a chosen file, If the user has refused a choice of a file– Comes back 0. file_ dialog - win32 Dialogue, file_dialog_f - fltk dialogue.

Acknowledgement and wishes

Many thanks to people who participating in the project. Golubenko D.J. for fruitful discussions, to developers of FLTK library, for convenient toolkit, Medvedev D.A. For some "Shapes" and useful discussions.

It is interesting to receive opinions on use MPPG, wishes about new features, error messages (in a code and documentation).

pru@hydro.nsc.ru

Pruuel E.R.