

C2Gnuplot visualization library

Zoltan Kuscsik

30.Jan.2007

To include the header file

```
#include <c2gnuplot.h>
```

The gnuplot session can be initialized by the type *gnuplot_window*. For example

```
gnuplot_window Window;
```

will create a gnuplot session with name *Window*. Data or commands sent to this object will be forwarded to gnuplot. Gnuplot commands can be passed by the function *command()*. Lines

```
Window.command("set size square");  
Window.command("set title 'My function'");
```

will tell to gnuplot to set the window size square and the graph title to 'My function'.

Functions can be used similarly as *printf()*. For example

```
Window.command("plot %d*x+%d",3,2);
```

sends the command *plot 3*x+2*.

Because gnuplot can read data only from files, data can not be sent by the *command* function. For data sending use the *plot_data()*, *splot_data()*, *data()* and *flush* functions, which use a temporary FIFO file for this purpose. For example, if you want to draw a data point at coordinates (4.3,2.5) with size 3

```
gnuplot_window Window;  
float a=4.3;  
float b=2.5  
Window.plot_data("u 1:2 w p ps 3");  
Window.data("%f, %f",a,b);  
Window.flush();
```

Functions *plot_data()* and *splot_data()* are used to open a 2D or 3D data plot with specific option. This must be done before you write the data by the *data()* function. The graph is plotted directly after flushing the FIFO buffer with *flush()*. There are several known bugs in piping to gnuplot. Low delay between *flush()* commands will cause the crash of gnuplot. It is recommended to include a time delay between two plots

```
Window.flush();  
usleep(100000); //delay between graphs
```

To set the gnuplot terminal to video output write

```
Window.mpeg_out("video.mpg");
```

which will create a mpg video with name *video.mpg* and resolution 640×480 .
To finish the video sequence just set the terminal after next flush to another value

```
Window.command("set term x11");
```

To compile your source code

```
g++ -I/usr/local/include <source_file> -o <binary_file>
```