Trabajando con gráficos vectoriales en C++ (Cairo-Graphics 2-D)

Ivan Cruz Aceves
Centro de Investigación en Matemáticas

¿Qué es Cairo?
https://www.cairographics.org/

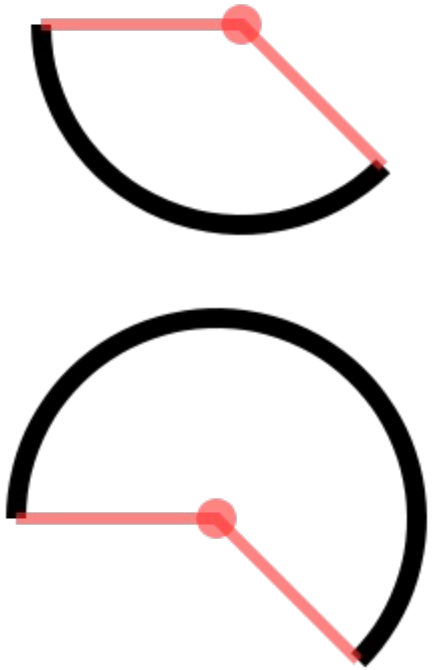
- Cairo es una libreria para generar gráficos vectoriales en 2-D
- Free Software (GNU LGPL)
- Es multiplataforma
- Múltiples dispositivos de salida
- Image buffers, Postscript, PDF, SVG
- Puede ser embebido en ambientes gráficos como wxWidgets
- Es utilizada por Gnome, Firefox, Dia, entre otros...

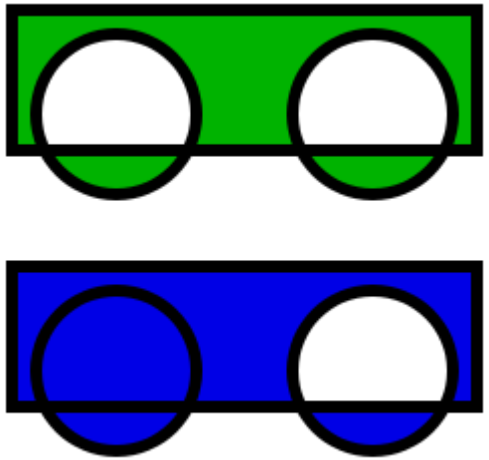
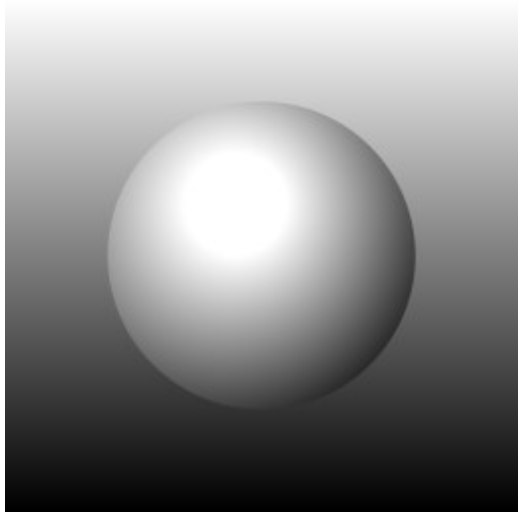
¿Qué es Cairo?
https://www.cairographics.org/

- Con Cairo es posible la creación de diversos gráficos como:
  - Gráficas de barras
  - Gráficas circulares
  - Boxplot
  - Gráficas de dispersión
  - Gráficas de lineas y puntos
  - Diagramas de todo tipo
  - Manipulación de imágenes

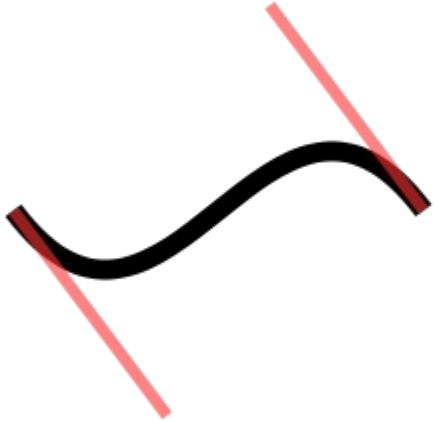
¿Qué es Cairo?

https://www.cairographics.org/samples/



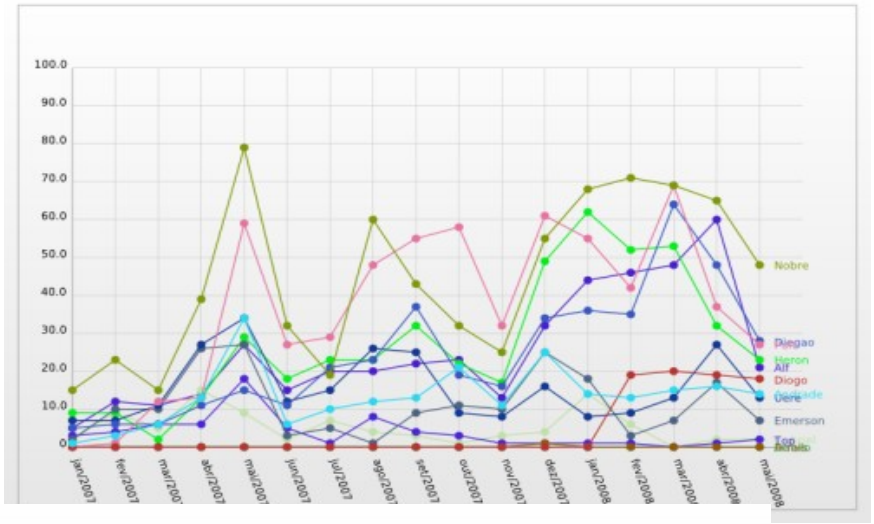


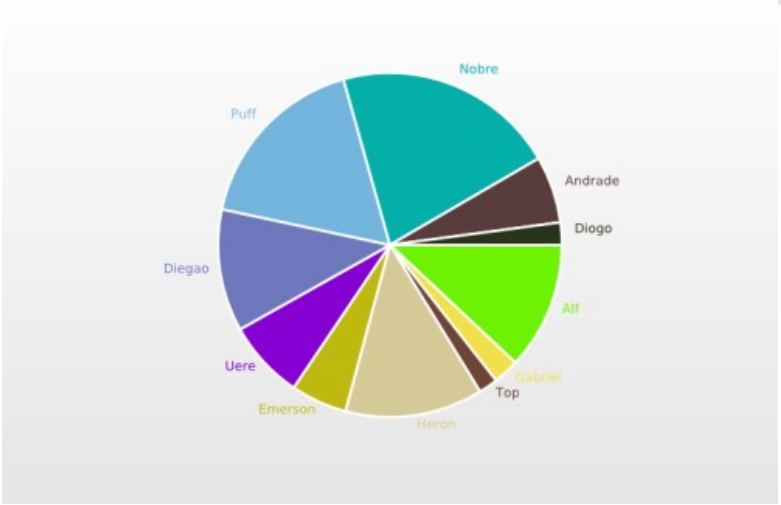



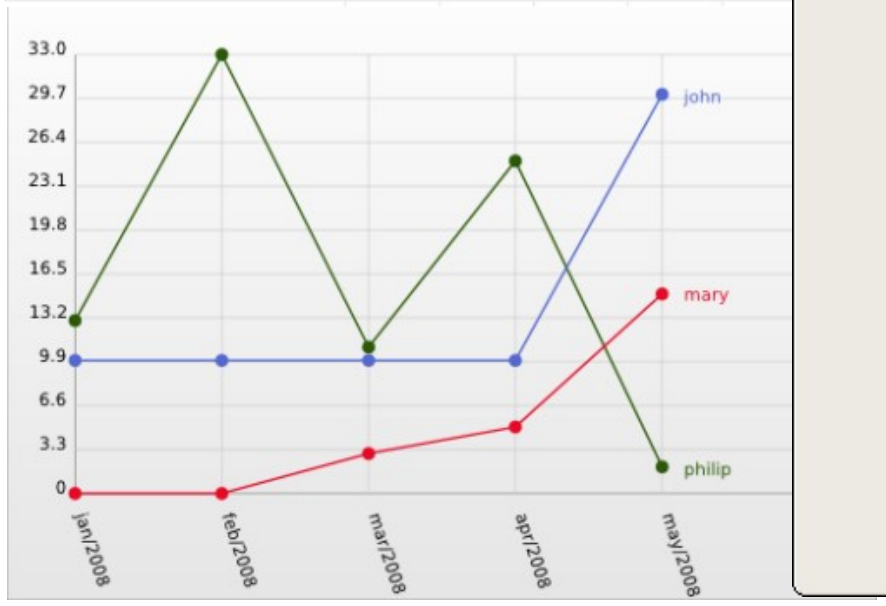
Ejemplos de uso con python (pycairo)

https://linil.wordpress.com/2008/06/14/cairoplot-plotting-graphics-using-python-and-cairo/

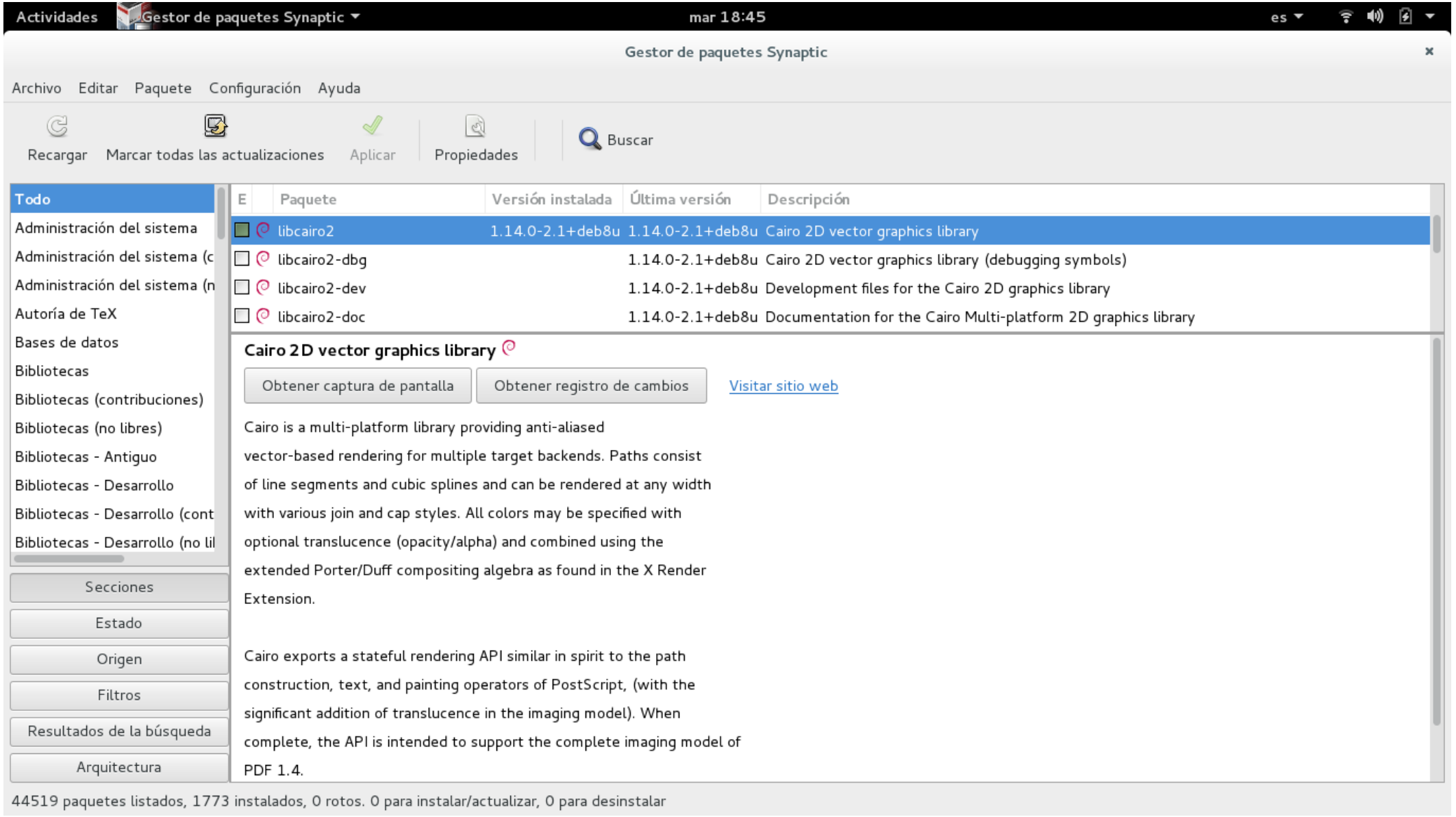
https://setanta.wordpress.com/2007/02/17/aventuras-no-cairo/



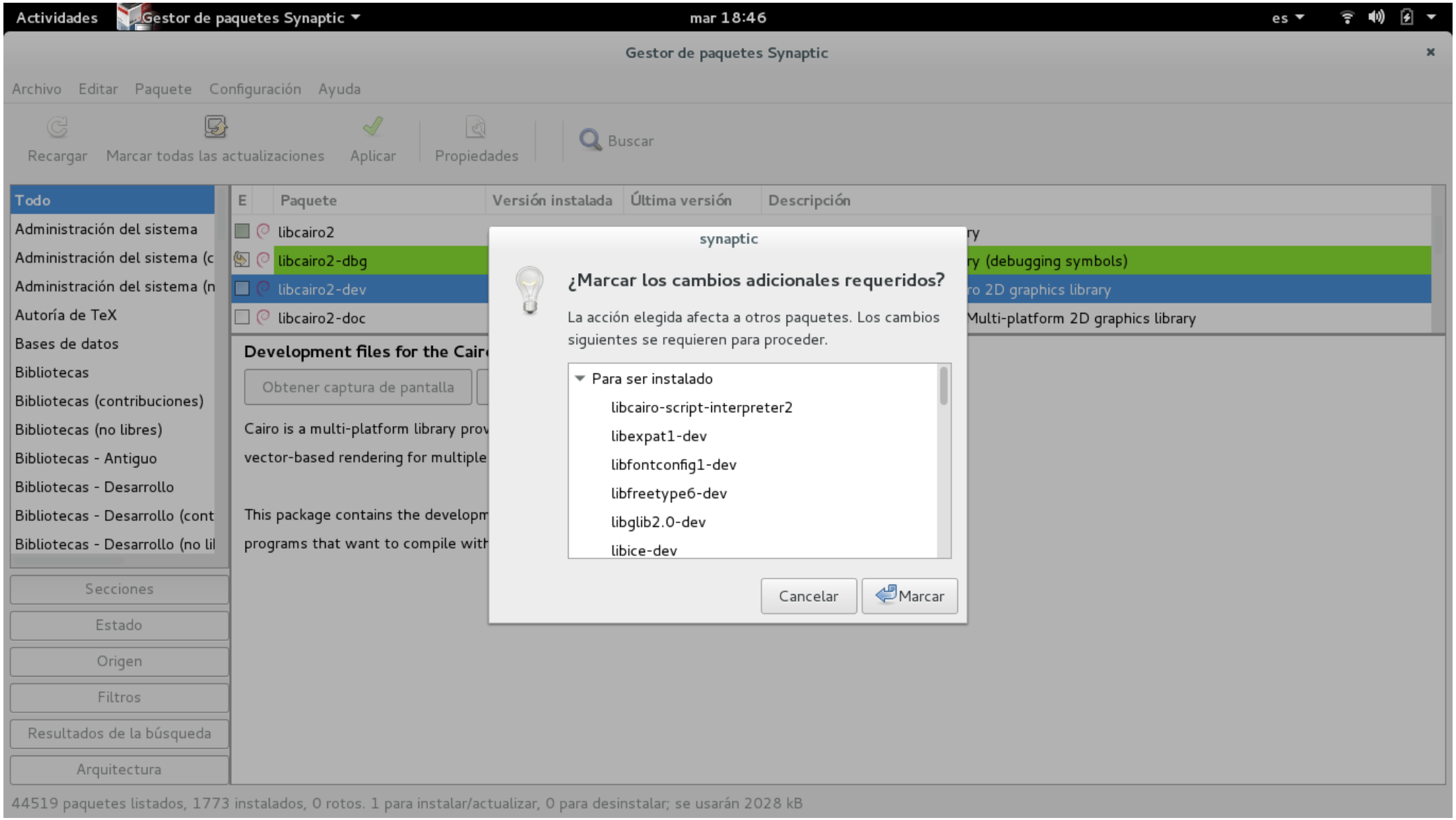




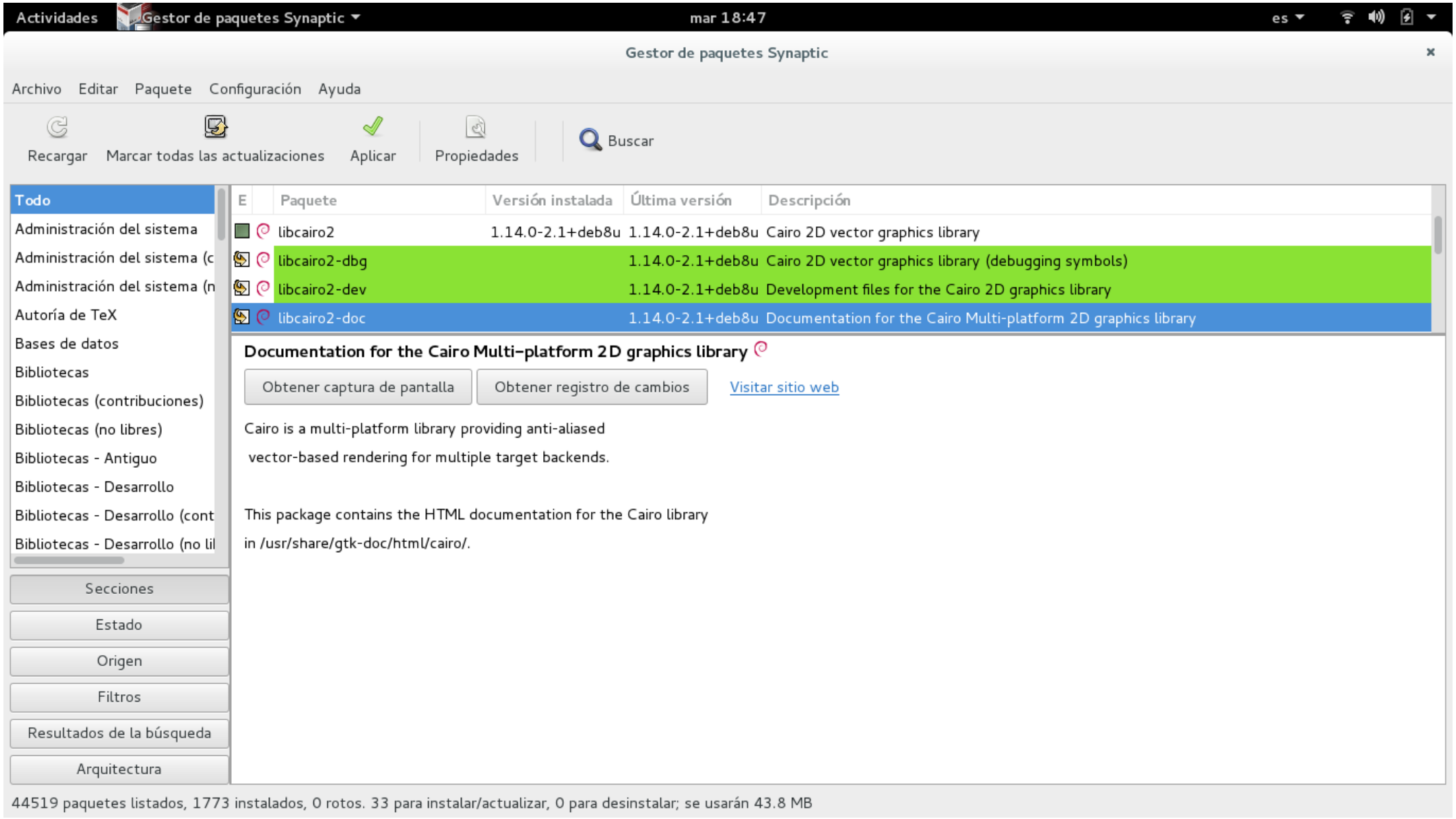
Instalación de cairo en Sistemas Linux



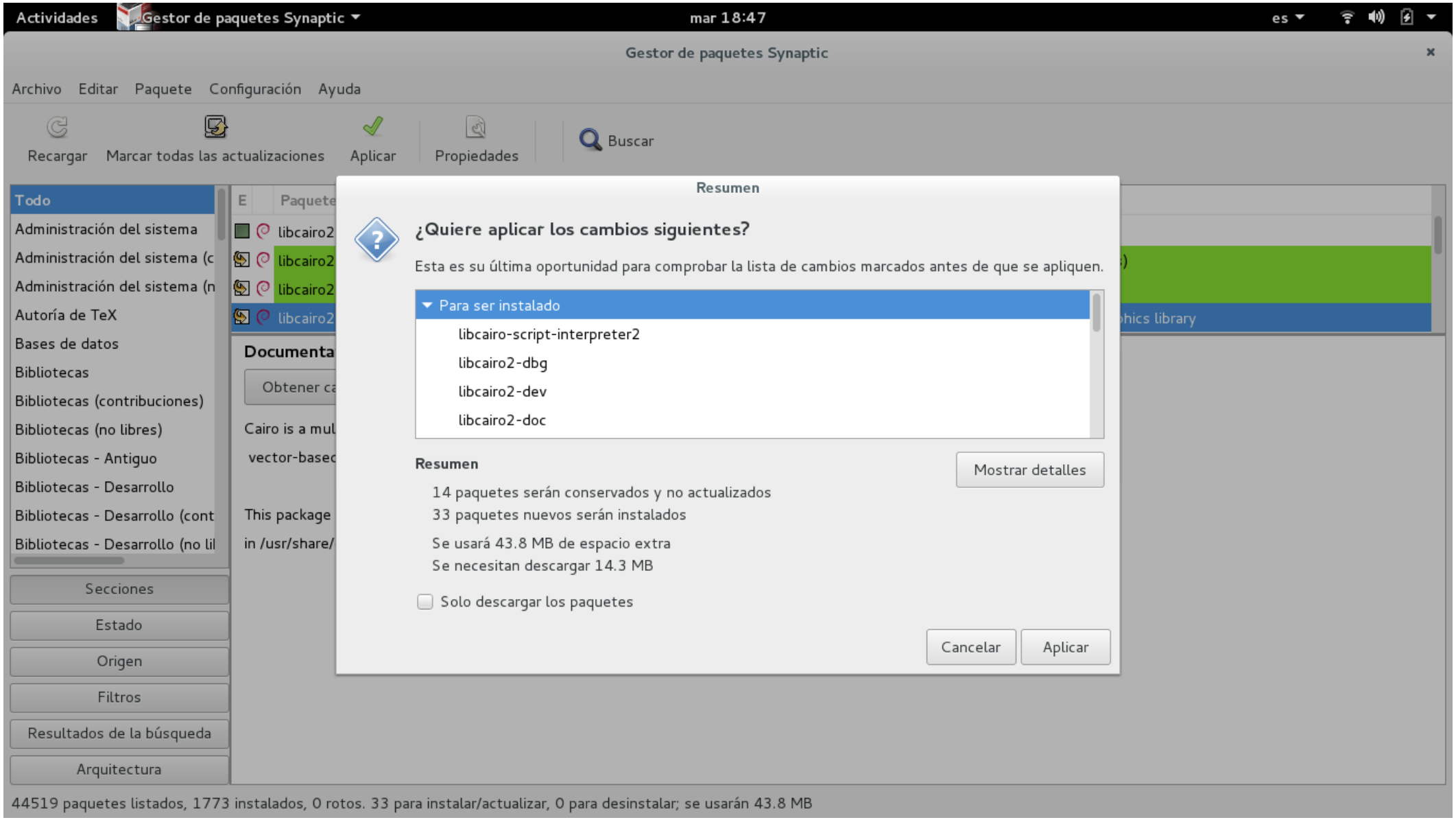
Instalación de cairo en Sistemas Linux



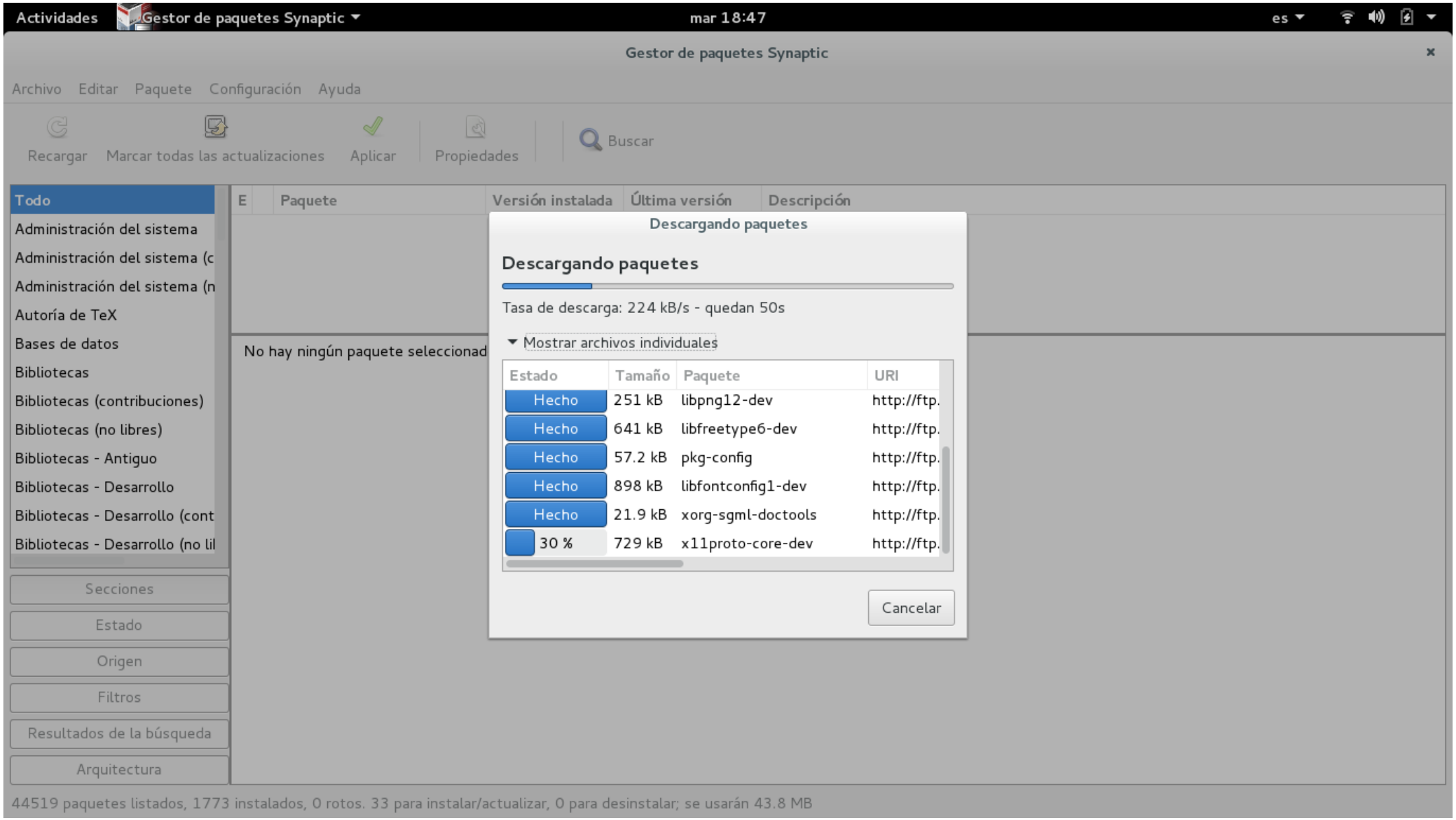
Instalación de cairo en Sistemas Linux



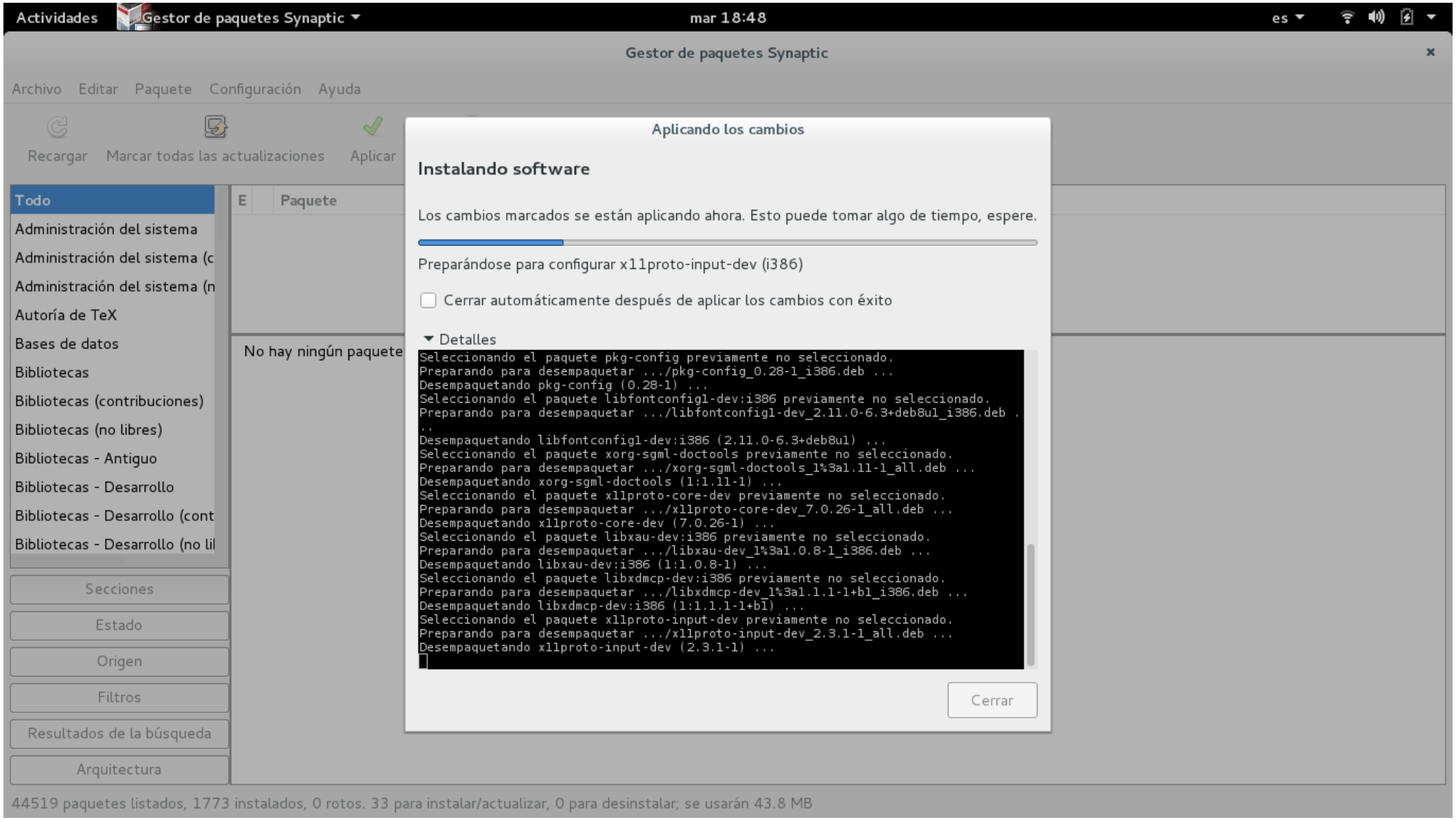
Instalación de cairo en Sistemas Linux



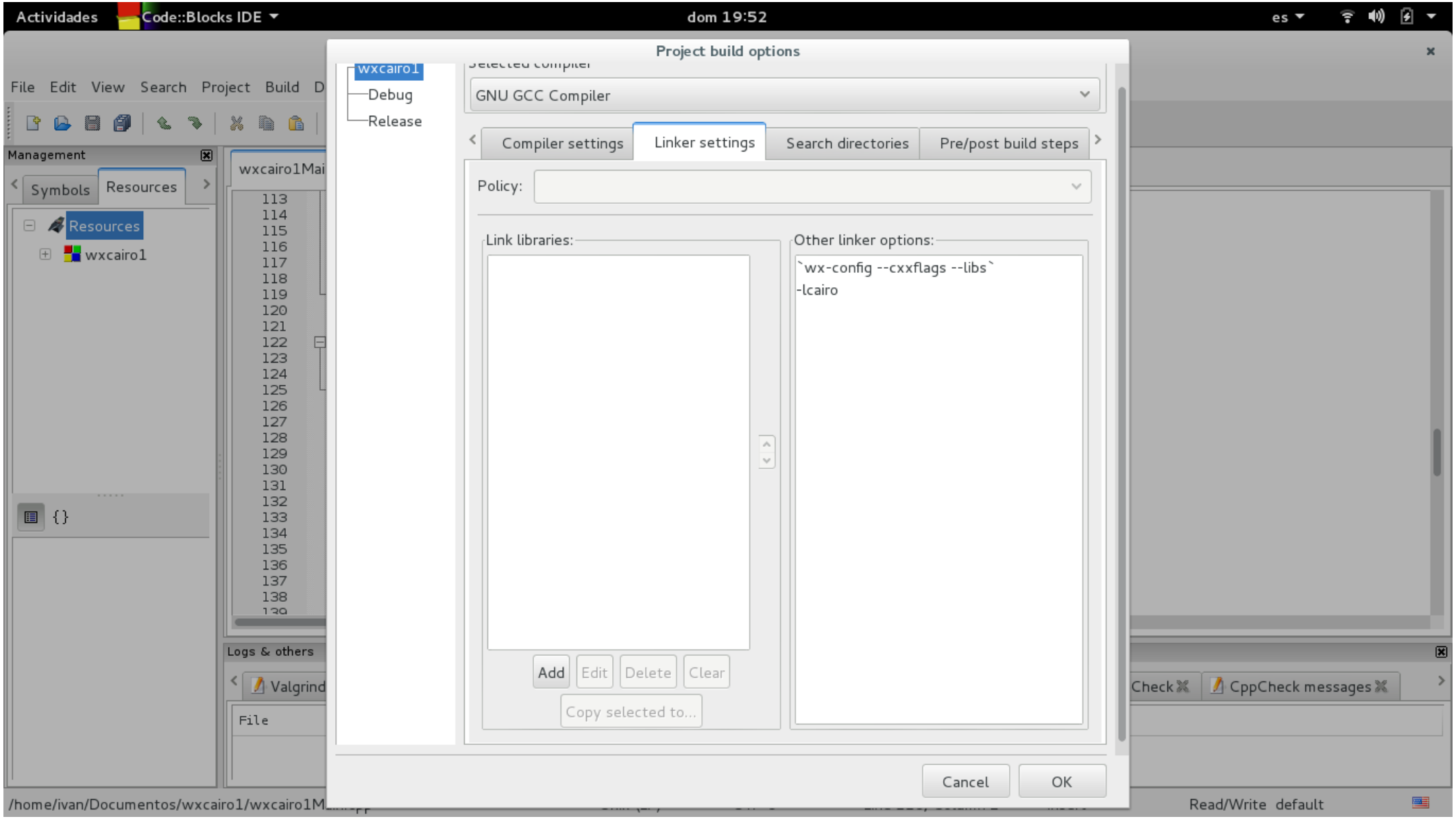
Instalación de cairo en Sistemas Linux



Instalación de cairo en Sistemas Linux



Trabajando cairo en CodeBlocks #include <cairo/cairo.h>



Instalando Cairo-graphics en C++ con CodeBlocks en MSW

Ivan Cruz Aceves
Centro de Investigación en Matemáticas

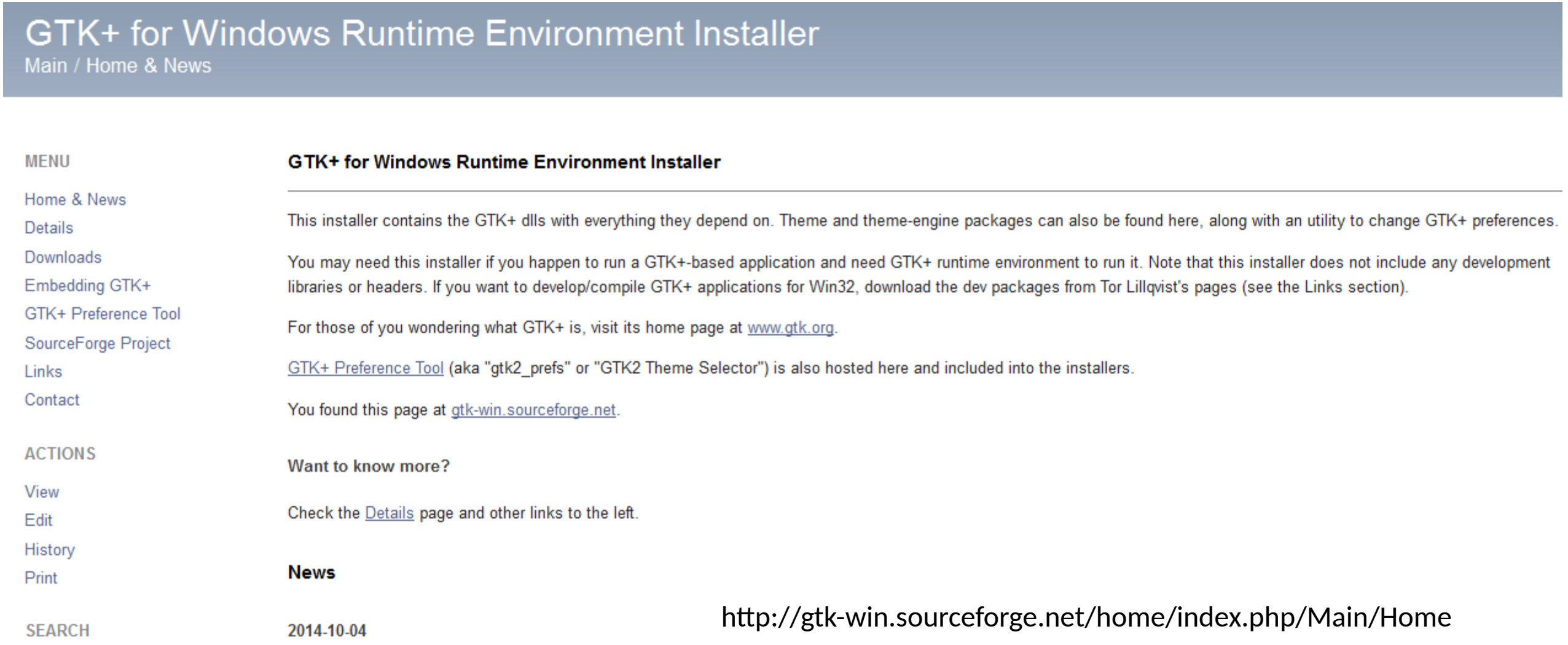
Instalación de cairo sobre MS-Windows

1. Descargar GTK para windows
2. Configuración de Codeblocks para trabajar con Cairo-graphics
3. Linker/ Compiler
4. Ejemplo basado en la pagina de Cairo

Instalación de cairo sobre Windows

https://www.gtk.org/

Instalación de cairo sobre Windows



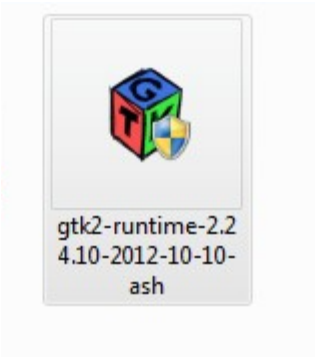
http://gtk-win.sourceforge.net/home/index.php/Main/Home

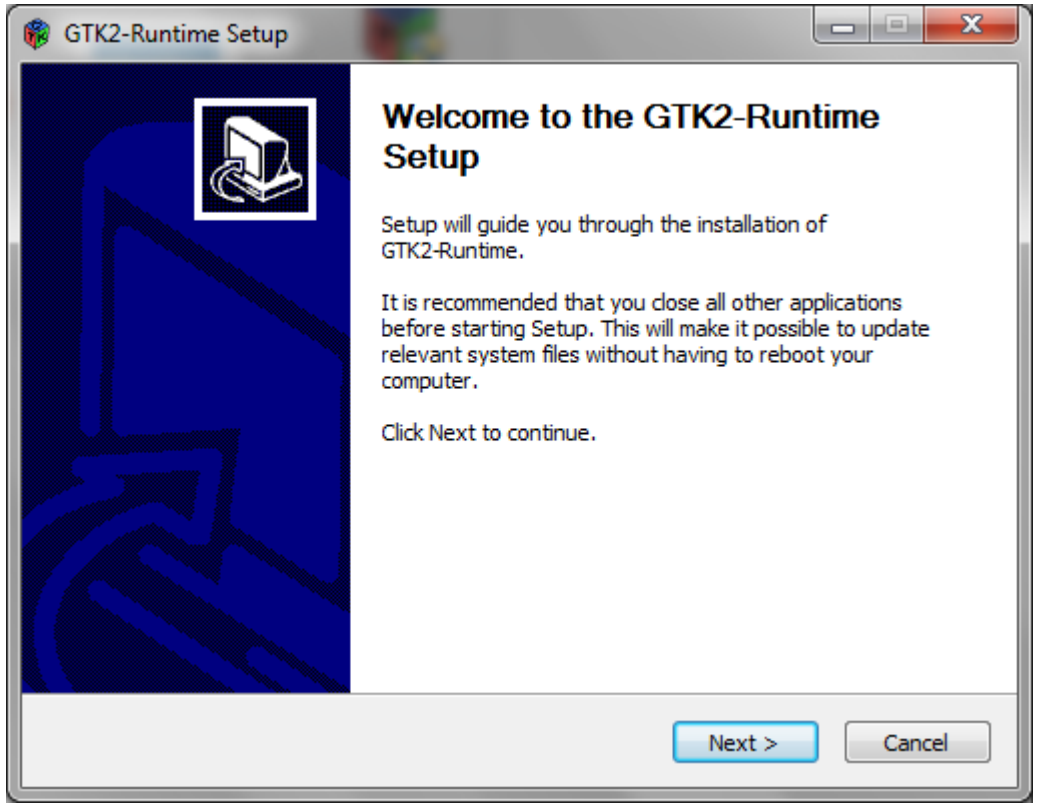
Instalación de cairo sobre Windows

https://sourceforge.net/projects/gtk-win/

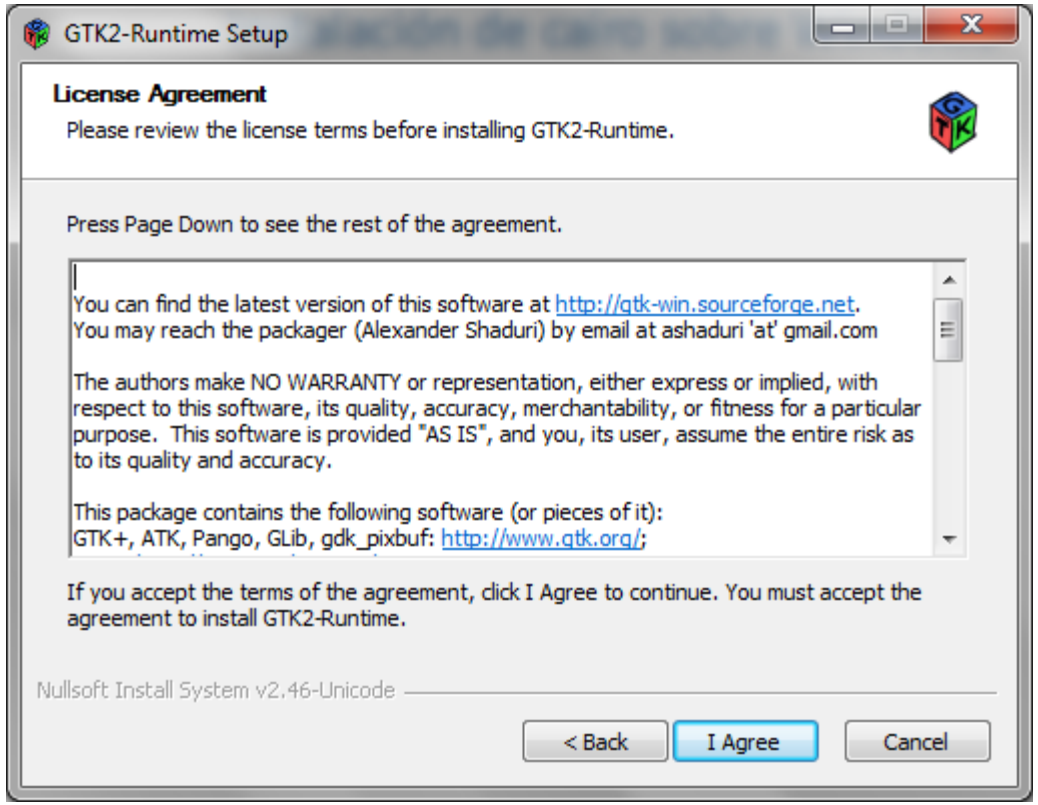
Instalación de cairo sobre Windows

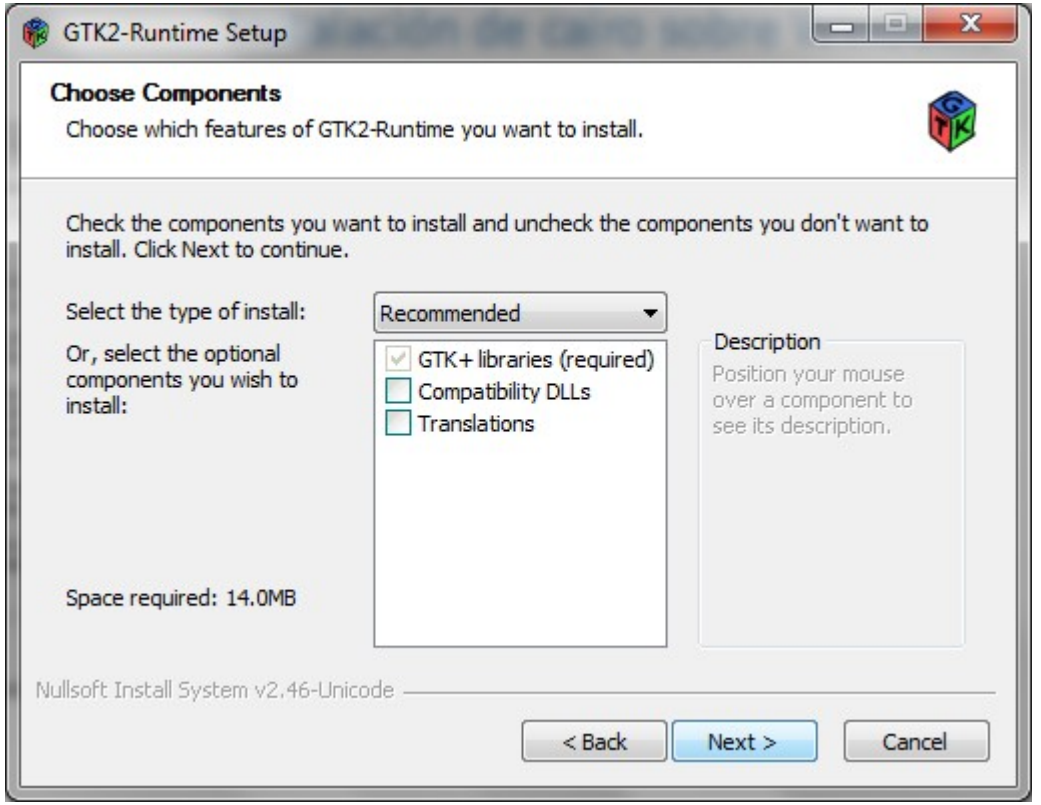
gtk2-runtime-2.24.10-2012-10-10-ash



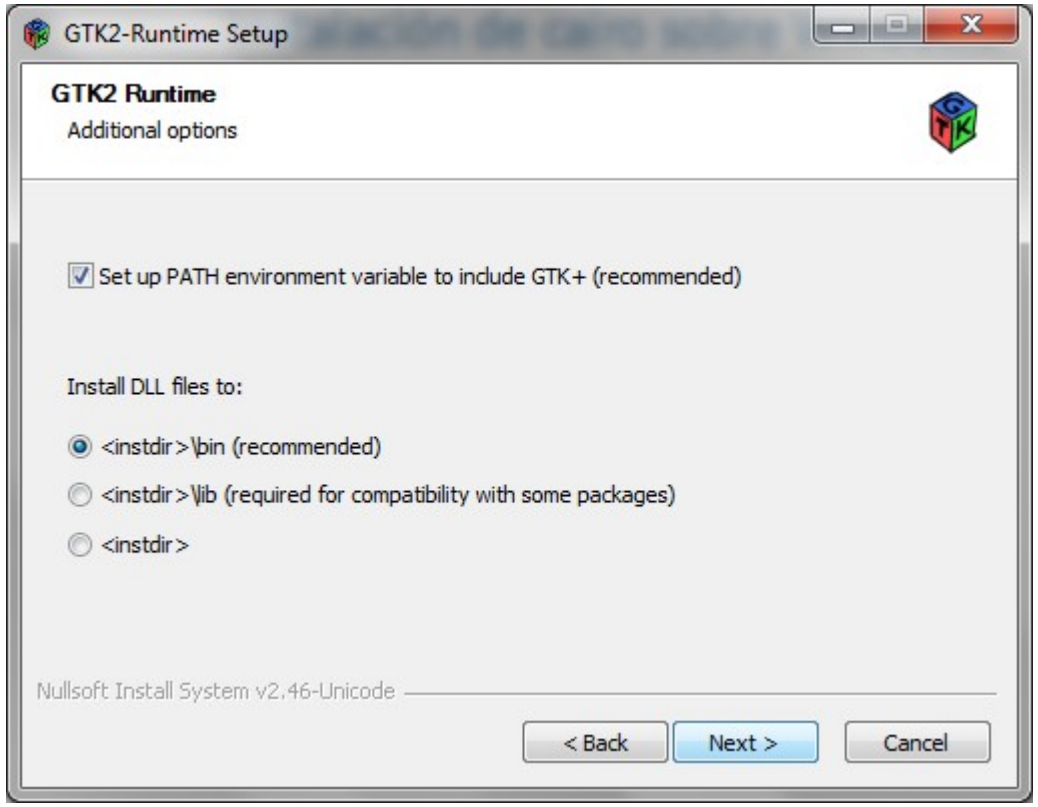


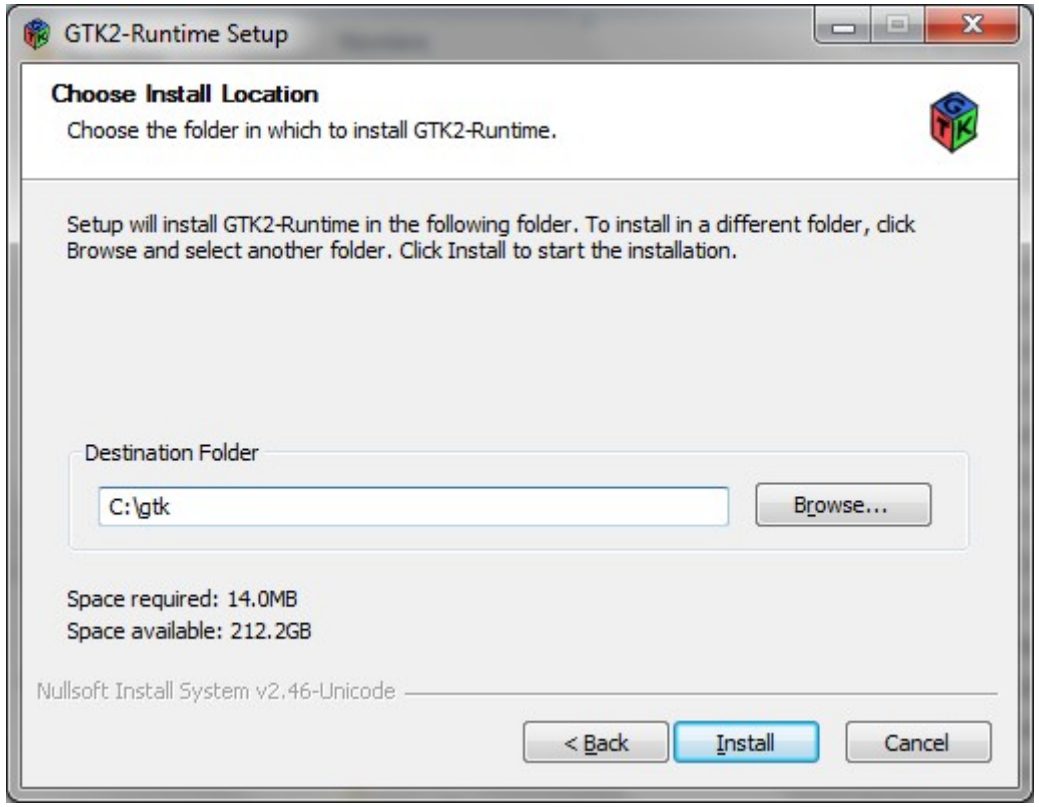
Instalación de cairo sobre Windows



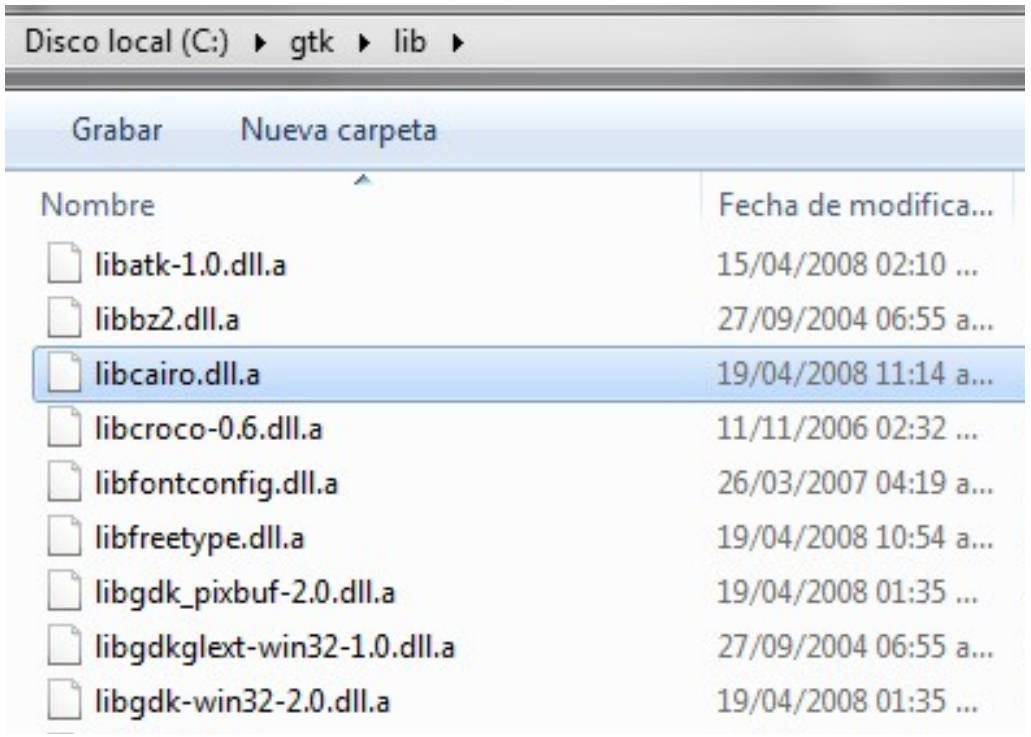


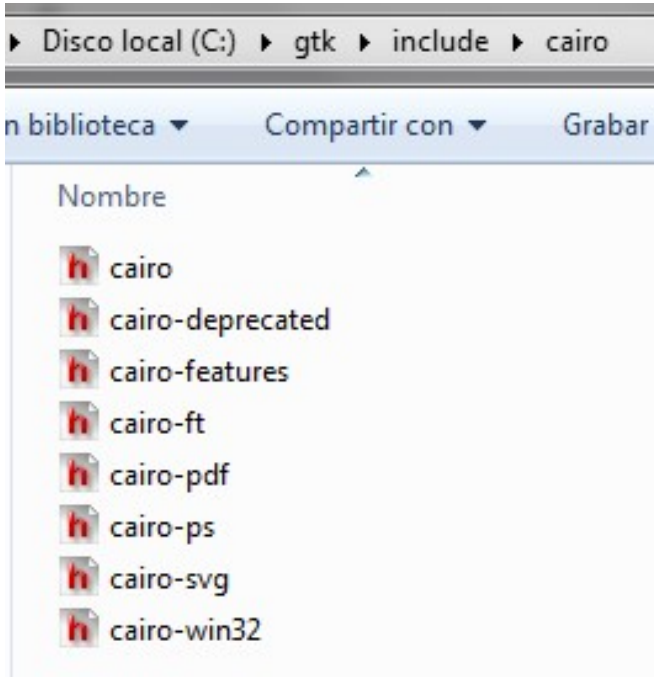
Instalación de cairo sobre Windows

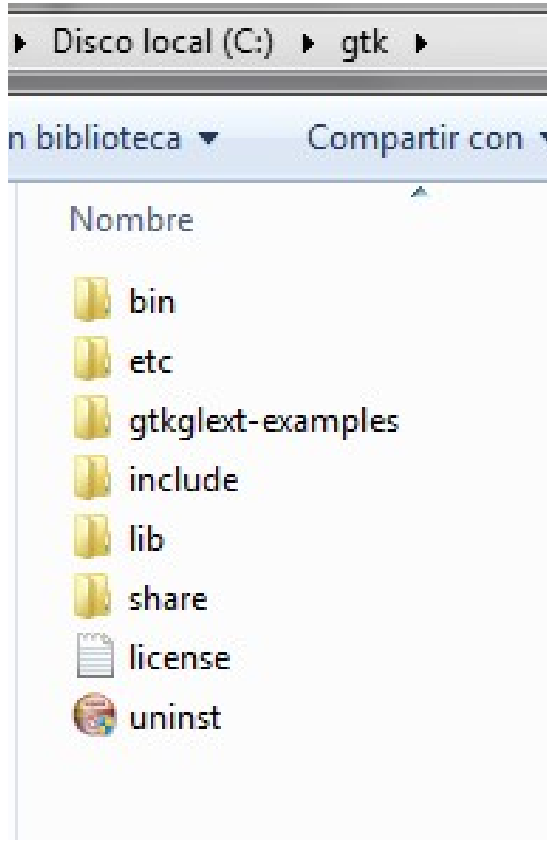




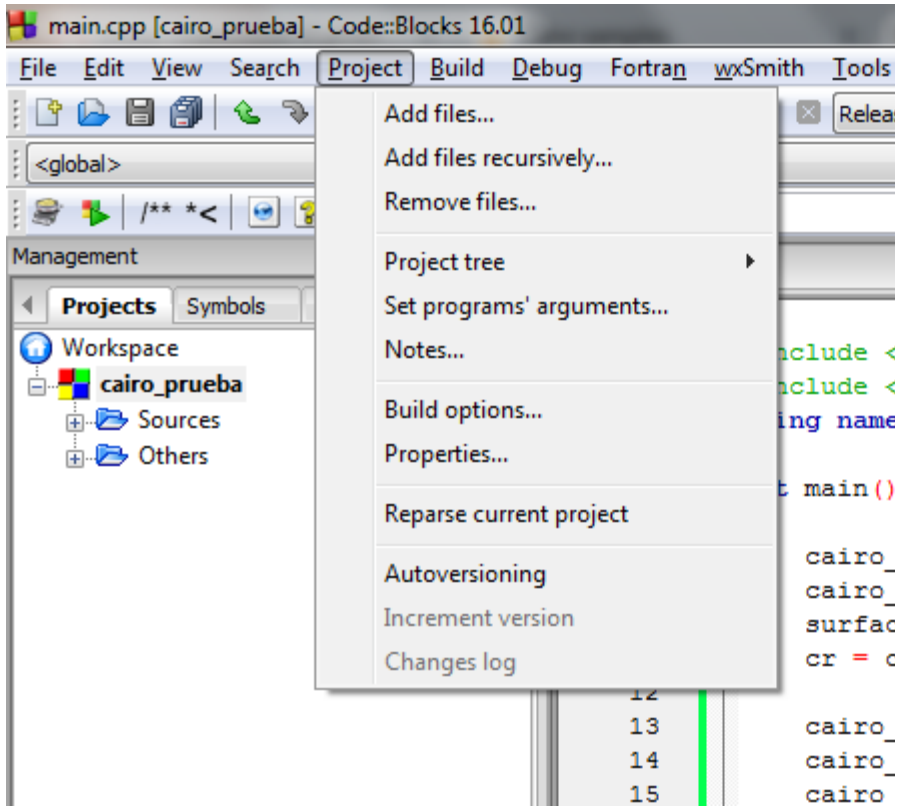
Instalación de cairo sobre Windows

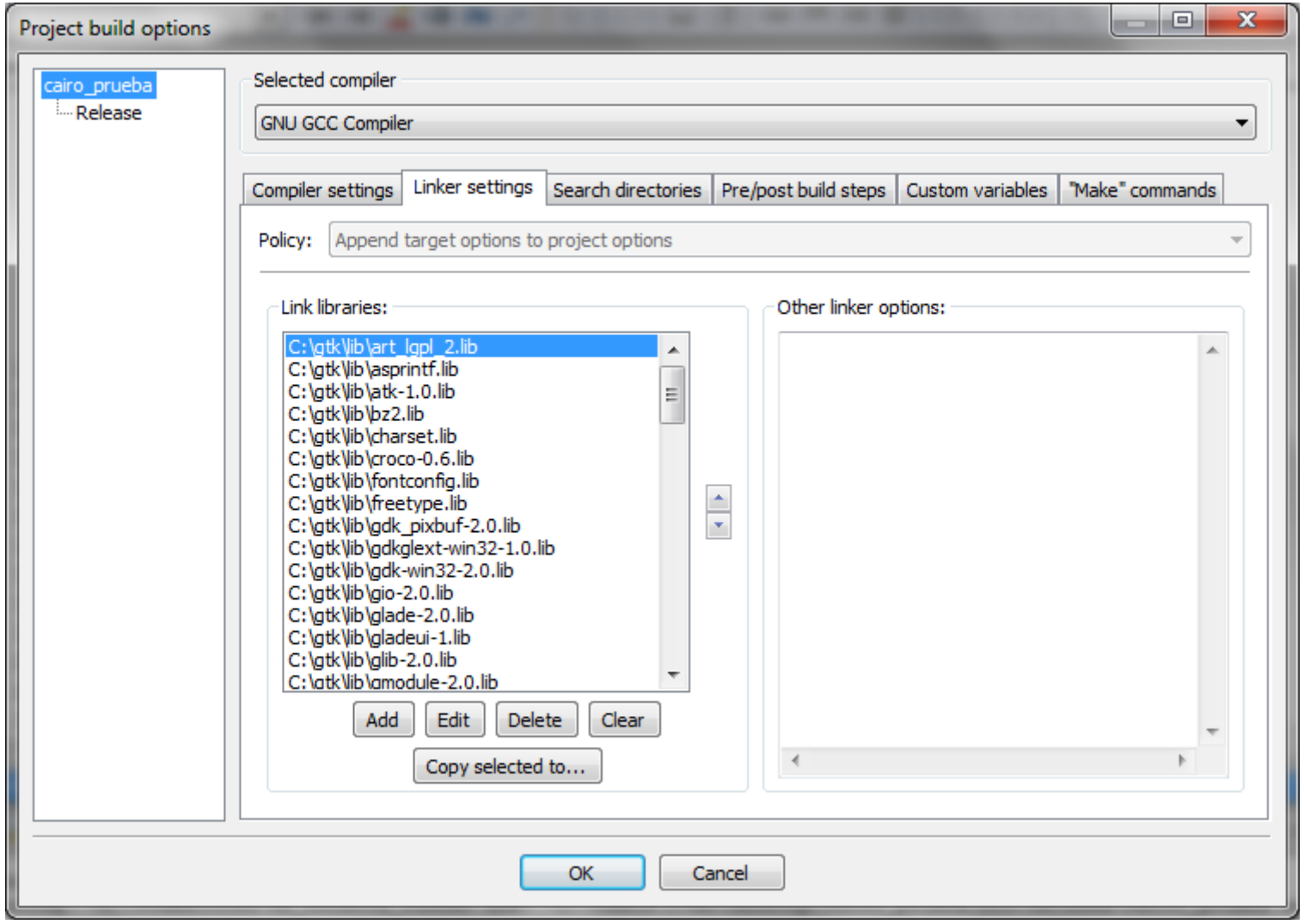






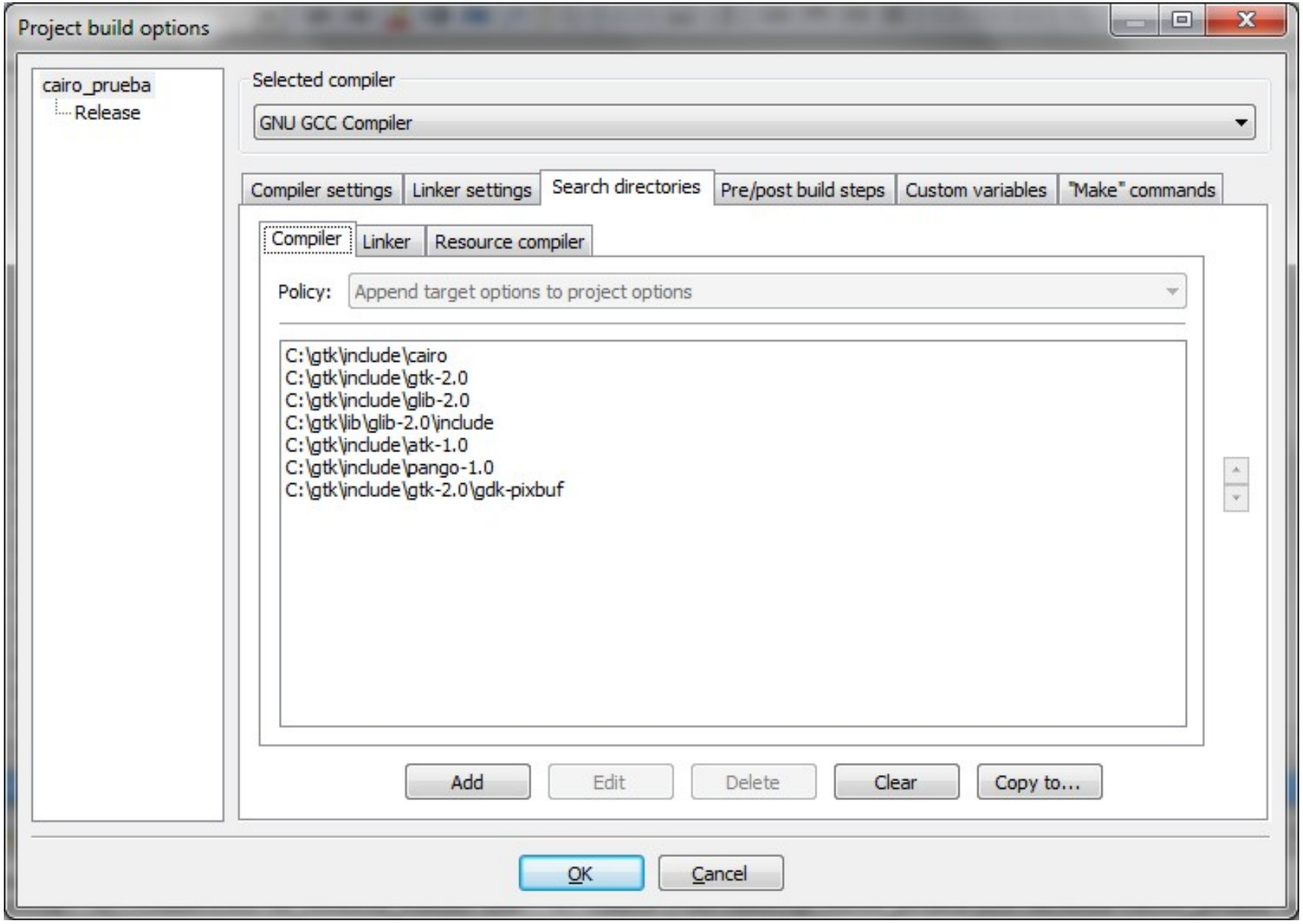
Configurando CodeBlocks para usar cairo sobre Windows





Copiar todas las librerias de-> C:\gtk\lib

Configurando CodeBlocks para usar cairo sobre Windows



Traza al Compilar aplicación en CodeBlocks / cairo

Logs & others

Search results | Cccc | Build log | Build messages | CppCheck | CppCheck messages | Cscope | Debugger | DoxyBlocks

```
-------------- Build: Release in cairo_prueba (compiler: GNU GCC Compiler)---------------

mingw32-g++.exe -Wall -fexceptions -O2 -IC:\gtk\include\cairo -IC:\gtk\include\gtk-2.0 -IC:\gtk\include\glib-2.0 -IC:\gtk\lib\glib-2.0\include -IC:\gtk
\include\atk-1.0 -IC:\gtk\include\pango-1.0 -IC:\gtk\include\gtk-2.0\gdk-pixbuf -IC:\gtk\include\cairo -IC:\gtk\include\gtk-2.0 -IC:\gtk\include\glib-
2.0 -IC:\gtk\lib\glib-2.0\include -IC:\gtk\include\atk-1.0 -IC:\gtk\include\pango-1.0 -IC:\gtk\include\gtk-2.0\gdk-pixbuf -c C:\Users\Ivan\Desktop
\cairo_prueba\main.cpp -o obj\Release\main.o
mingw32-g++.exe  -o bin\Release\cairo_prueba.exe obj\Release\main.o  -s  C:\gtk\lib\art_lgpl_2.lib C:\gtk\lib\asprintf.lib C:\gtk\lib\atk-1.0.lib C:\gtk
\lib\bz2.lib C:\gtk\lib\charset.lib C:\gtk\lib\croco-0.6.lib C:\gtk\lib\fontconfig.lib C:\gtk\lib\freetype.lib C:\gtk\lib\gdk_pixbuf-2.0.lib C:\gtk\lib
\gdkglext-win32-1.0.lib C:\gtk\lib\gdk-win32-2.0.lib C:\gtk\lib\gio-2.0.lib C:\gtk\lib\glade-2.0.lib C:\gtk\lib\gladeui-1.lib C:\gtk\lib\glib-2.0.lib C:
\gtk\lib\gmodule-2.0.lib C:\gtk\lib\gobject-2.0.lib C:\gtk\lib\gsf-1.lib C:\gtk\lib\gsf-win32-1.lib C:\gtk\lib\gthread-2.0.lib C:\gtk\lib\gtkglext-
win32-1.0.lib C:\gtk\lib\gtk-win32-2.0.lib C:\gtk\lib\iconv.lib C:\gtk\lib\intl.lib C:\gtk\lib\jpeg.lib C:\gtk\lib\libart_lgpl_2.dll.a C:\gtk\lib
\libasprintf.dll.a C:\gtk\lib\libatk-1.0.dll.a C:\gtk\lib\libbz2.dll.a C:\gtk\lib\libcairo.dll.a C:\gtk\lib\libcroco-0.6.dll.a C:\gtk\lib
\libfontconfig.dll.a C:\gtk\lib\libfreetype.dll.a C:\gtk\lib\libgdk_pixbuf-2.0.dll.a C:\gtk\lib\libgdkglext-win32-1.0.dll.a C:\gtk\lib\libgdk-win32-
2.0.dll.a C:\gtk\lib\libgio-2.0.dll.a C:\gtk\lib\libglade-2.0.dll.a C:\gtk\lib\libgladeui-1.dll.a C:\gtk\lib\libglib-2.0.dll.a C:\gtk\lib\libgmodule-
2.0.dll.a C:\gtk\lib\libgobject-2.0.dll.a C:\gtk\lib\libgsf-1.dll.a C:\gtk\lib\libgsf-win32-1.dll.a C:\gtk\lib\libgthread-2.0.dll.a C:\gtk\lib
\libgtkglext-win32-1.0.dll.a C:\gtk\lib\libgtk-win32-2.0.dll.a C:\gtk\lib\libgw32c.a C:\gtk\lib\libiconv.dll.a C:\gtk\lib\libintl.dll.a C:\gtk\lib
\libjpeg.dll.a C:\gtk\lib\libpango-1.0.dll.a C:\gtk\lib\libpangocairo-1.0.dll.a C:\gtk\lib\libpangoft2-1.0.dll.a C:\gtk\lib\libpangowin32-1.0.dll.a C:
\gtk\lib\libpng.dll.a C:\gtk\lib\libpng12.dll.a C:\gtk\lib\libpopt.dll.a C:\gtk\lib\librsvg-2.dll.a C:\gtk\lib\libtiff.dll.a C:\gtk\lib\libxml2.dll.a C:
\gtk\lib\libz.dll.a C:\gtk\lib\pango-1.0.lib C:\gtk\lib\pangocairo-1.0.lib C:\gtk\lib\pangoft2-1.0.lib C:\gtk\lib\pangowin32-1.0.lib C:\gtk\lib\png.lib
C:\gtk\lib\popt.lib C:\gtk\lib\rsvg-2.lib C:\gtk\lib\tiff.lib C:\gtk\lib\xml2.lib C:\gtk\lib\z.lib C:\gtk\lib\art_lgpl_2.lib C:\gtk\lib\asprintf.lib C:
\gtk\lib\atk-1.0.lib C:\gtk\lib\bz2.lib C:\gtk\lib\charset.lib C:\gtk\lib\croco-0.6.lib C:\gtk\lib\fontconfig.lib C:\gtk\lib\freetype.lib C:\gtk\lib
\gdk_pixbuf-2.0.lib C:\gtk\lib\gdkglext-win32-1.0.lib C:\gtk\lib\gdk-win32-2.0.lib C:\gtk\lib\gio-2.0.lib C:\gtk\lib\glade-2.0.lib C:\gtk\lib\gladeui-
1.lib C:\gtk\lib\glib-2.0.lib C:\gtk\lib\gmodule-2.0.lib C:\gtk\lib\gobject-2.0.lib C:\gtk\lib\gsf-1.lib C:\gtk\lib\gsf-win32-1.lib C:\gtk\lib\gthread-
2.0.lib C:\gtk\lib\gtkglext-win32-1.0.lib C:\gtk\lib\gtk-win32-2.0.lib C:\gtk\lib\iconv.lib C:\gtk\lib\intl.lib C:\gtk\lib\jpeg.lib C:\gtk\lib
\libart_lgpl_2.dll.a C:\gtk\lib\libasprintf.dll.a C:\gtk\lib\libatk-1.0.dll.a C:\gtk\lib\libbz2.dll.a C:\gtk\lib\libcairo.dll.a C:\gtk\lib\libcroco-
0.6.dll.a C:\gtk\lib\libfontconfig.dll.a C:\gtk\lib\libfreetype.dll.a C:\gtk\lib\libgdk_pixbuf-2.0.dll.a C:\gtk\lib\libgdkglext-win32-1.0.dll.a C:\gtk
\lib\libgdk-win32-2.0.dll.a C:\gtk\lib\libgio-2.0.dll.a C:\gtk\lib\libglade-2.0.dll.a C:\gtk\lib\libgladeui-1.dll.a C:\gtk\lib\libglib-2.0.dll.a C:\gtk
\lib\libgmodule-2.0.dll.a C:\gtk\lib\libgobject-2.0.dll.a C:\gtk\lib\libgsf-1.dll.a C:\gtk\lib\libgsf-win32-1.dll.a C:\gtk\lib\libgthread-2.0.dll.a C:
\gtk\lib\libgtkglext-win32-1.0.dll.a C:\gtk\lib\libgtk-win32-2.0.dll.a C:\gtk\lib\libgw32c.a C:\gtk\lib\libiconv.dll.a C:\gtk\lib\libintl.dll.a C:\gtk
\lib\libjpeg.dll.a C:\gtk\lib\libpango-1.0.dll.a C:\gtk\lib\libpangocairo-1.0.dll.a C:\gtk\lib\libpangoft2-1.0.dll.a C:\gtk\lib\libpangowin32-1.0.dll.a
C:\gtk\lib\libpng.dll.a C:\gtk\lib\libpng12.dll.a C:\gtk\lib\libpopt.dll.a C:\gtk\lib\librsvg-2.dll.a C:\gtk\lib\libtiff.dll.a C:\gtk\lib\libxml2.dll.a
C:\gtk\lib\libz.dll.a C:\gtk\lib\pango-1.0.lib C:\gtk\lib\pangocairo-1.0.lib C:\gtk\lib\pangoft2-1.0.lib C:\gtk\lib\pangowin32-1.0.lib C:\gtk\lib
\png.lib C:\gtk\lib\popt.lib C:\gtk\lib\rsvg-2.lib C:\gtk\lib\tiff.lib C:\gtk\lib\xml2.lib C:\gtk\lib\z.lib
Output file is bin\Release\cairo_prueba.exe with size 543.00 KB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
```

Funciones para utilizar en Cairo
https://www.cairographics.org/manual/

cairo_surface_t	Clase base para generacion de superficies
cairo_t	Tipo de dato definido para generar un contexto
cairo_image_surface_create	Crea un espacio de almacenamiento (formato,tamaño)
cairo_create	Asignacion de surface al contexto definido
cairo_select_font_face	Seleccion del tipo de fuente
cairo_set_font_size	Seleccion del tamaño de fuente
cairo_set_source_rgb	Asignación del color
cairo_move_to	Posicionamiento del puntero sobre la surface
cairo_show_text	Desplegado de texto
cairo_destroy	Liberación de memoria para contexto
cairo_surface_write_to_png	Salida del gráfico como imagen tipo "png"
cairo_surface_destroy	Liberación de memoria para Surface
cairo_stroke	Liberación de lienzo de un objeto pintado

Ejemplo de Cairo en C++

```
#include <iostream>
#include <cairo.h>
using namespace std;

int main()
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32,500,200);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr,0,1,0);
    cairo_rectangle(cr,20,20,120,80);
    cairo_fill(cr);

    cairo_surface_write_to_png(surface,"Cuadro.png");
    cairo_destroy(cr);
    cairo_surface_destroy(surface);
    return 0;
}
```

Inicialización de:
- Superficie de pintado
- Lienzo a pintar

Ejemplo de Cairo en C++

```
#include <iostream>
#include <cairo.h>
using namespace std;

int main()
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32,500,200);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr,0,1,0);
    cairo_rectangle(cr,20,20,120,80);
    cairo_fill(cr);

    cairo_surface_write_to_png(surface,"Cuadro.png");
    cairo_destroy(cr);
    cairo_surface_destroy(surface);
    return 0;
}
```

Inicialización de:
- Superficie de pintado
- Lienzo a pintar

- Escritura de objeto gráfico
- Liberación de memoria

Ejemplo de Cairo en C++

```
#include <iostream>
#include <cairo.h>
using namespace std;

int main()
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32,500,200);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr,0,1,0);
    cairo_rectangle(cr,20,20,120,80);
    cairo_fill(cr);

    cairo_surface_write_to_png(surface,"Cuadro.png");
    cairo_destroy(cr);
    cairo_surface_destroy(surface);
    return 0;
}
```

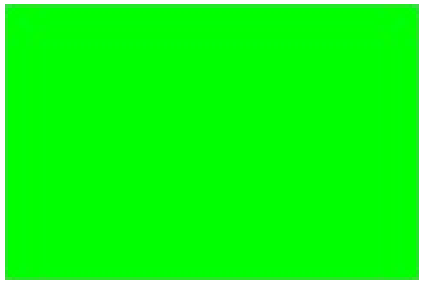
Inicialización de:
- Superficie de pintado
- Lienzo a pintar

- Declaramos color verde (0-1-0)
- Creamos un rectángulo (x,y,L,W)
- Coloreamos rectángulo

- Escritura de objeto gráfico
- Liberación de memoria

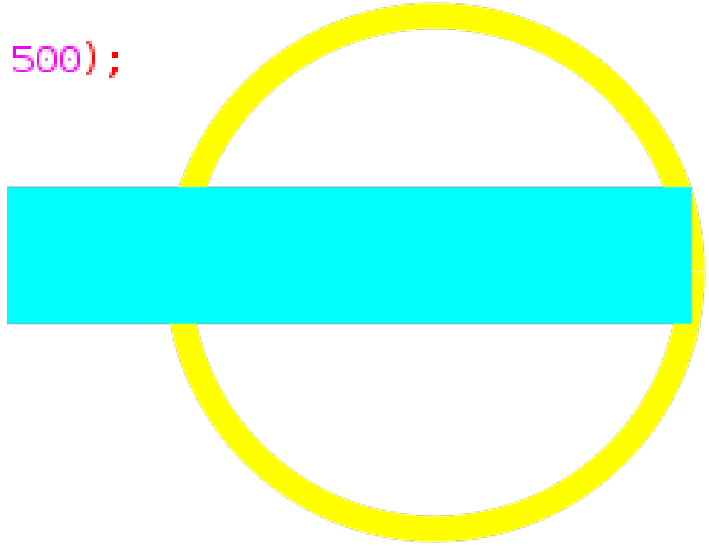
Ejemplo de Cairo en C++

```
#include <iostream>
#include <cairo.h>
using namespace std;

int main()
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32,500,200);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr,0,1,0);
    cairo_rectangle(cr,20,20,120,80);
    cairo_fill(cr);

    cairo_surface_write_to_png(surface,"Cuadro.png");
    cairo_destroy(cr);
    cairo_surface_destroy(surface);
    return 0;
}
```



Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;

    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 500, 500);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr, 1.0, 1.0, 0.0);
    cairo_set_line_width(cr,15);
    cairo_arc(cr,250,250,150,0,6.28);
    cairo_stroke(cr);
    cairo_set_source_rgb (cr, 0.0, 1.0, 1.0);
    cairo_rectangle(cr,0,200,400,80);
    cairo_fill(cr);
    cairo_surface_write_to_png (surface, "Anuncio.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);


}
```



Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;

    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 250, 100);
    cr = cairo_create (surface);

    cairo_select_font_face (cr, "serif", CAIRO_FONT_SLANT_NORMAL, CAIRO_FONT_WEIGHT_BOLD);
    cairo_set_font_size (cr, 12.0);
    cairo_set_source_rgb (cr, 0.5, 0.1, 1.0);
    cairo_move_to (cr, 10.0, 10.0);
    cairo_show_text (cr, "Programando en Cairo");
    cairo_destroy (cr);
    cairo_surface_write_to_png (surface, "Imagen1.png");
    cairo_surface_destroy (surface);


}
```

Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 500, 500);
    cr = cairo_create (surface);
    cairo_fill(cr);
    cairo_set_source_rgb (cr, 0.0, 1.0, 0.0);
    cairo_set_line_width(cr,25);
    cairo_arc(cr,200,200,100,0,6.28);
    cairo_stroke(cr);
    cairo_set_source_rgb (cr, 0.0, 1.0, 1.0);
    cairo_rectangle(cr,0,160,400,50);
    cairo_fill(cr);
    cairo_select_font_face (cr, "sans", CAIRO_FONT_SLANT_NORMAL, CAIRO_FONT_WEIGHT_NORMAL);
    cairo_set_font_size (cr, 24.0);
    cairo_set_source_rgb (cr, 0.5, 0.1, 1.0);
    cairo_move_to (cr, 200.0, 180.0);
    cairo_show_text (cr, "Cairo");
    cairo_surface_write_to_png (surface, "Anuncio1.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);


}
```

Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 500, 200);
    cr = cairo_create (surface);

    cairo_move_to (cr, 20, 20);
    cairo_curve_to(cr,320,200,330,110,450,40);
    cairo_stroke(cr);

    cairo_surface_write_to_png (surface, "Bezier.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);


}
```

http://zetcode.com/gfx/pycairo/basicdrawing/

Trabajando cairo

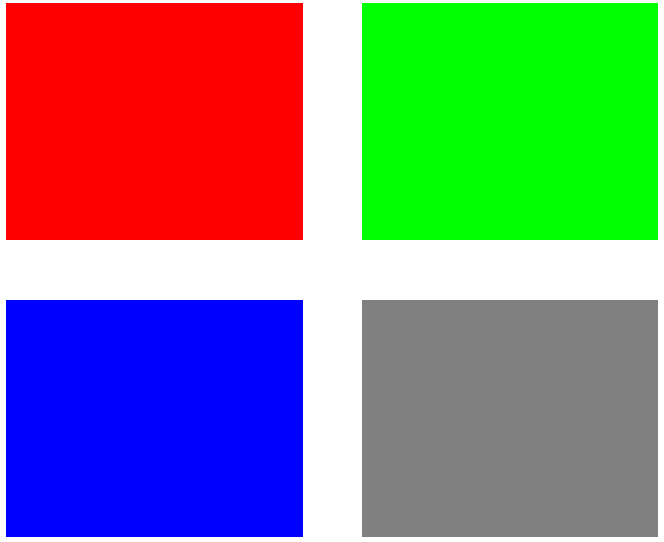
```
void wxcairolFrame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 500, 200);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr, 0.0, 1.0, 0.0);
    cairo_rectangle(cr,20, 20, 120, 80);
    cairo_rectangle(cr,180, 20, 80, 80);
    cairo_fill(cr);

    cairo_surface_write_to_png (surface, "Path.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);
}
```

Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 500, 200);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr, 0.0, 1.0, 0.0);
    cairo_rectangle(cr,20, 20, 120, 80);
    cairo_rectangle(cr,180, 20, 80, 80);
    cairo_fill(cr);
    cairo_arc(cr,350, 70, 30, 0, 2* 3.1416);
    cairo_fill(cr);

    cairo_surface_write_to_png (surface, "Path.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);
}
```

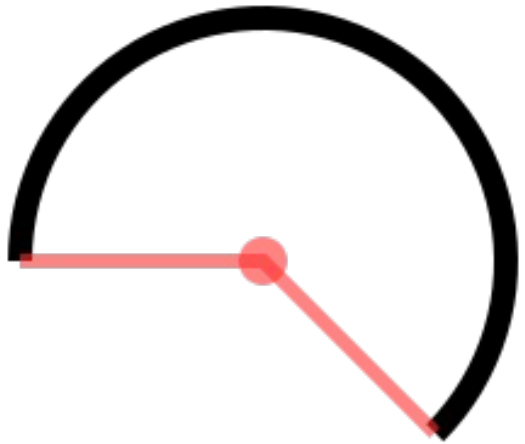
Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 300, 300);
    cr = cairo_create (surface);

    cairo_set_source_rgb (cr, 1.0, 0.0, 0.0);
    cairo_rectangle(cr,20, 30, 100, 80);
    cairo_fill(cr);

    cairo_set_source_rgb (cr, 0.0, 1.0, 0.0);
    cairo_rectangle(cr,140, 30, 100, 80);
    cairo_fill(cr);

    cairo_set_source_rgb (cr, 0.0, 0.0, 1.0);
    cairo_rectangle(cr,20, 130, 100, 80);
    cairo_fill(cr);

    cairo_set_source_rgb (cr, 0.5, 0.5, 0.5);
    cairo_rectangle(cr,140, 130, 100, 80);
    cairo_fill(cr);

    cairo_surface_write_to_png (surface, "Recs.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);
}
```



Trabajando cairo

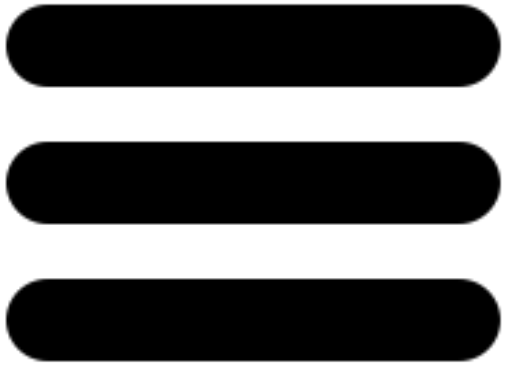
```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 300, 300);
    cr = cairo_create (surface);
    double xc = 128.0;
    double yc = 128.0;
    double radius = 100.0;
    double angle1 = 45.0  * (M_PI/180.0);  /* angles are specified */
    double angle2 = 180.0 * (M_PI/180.0);  /* in radians           */
    cairo_set_line_width (cr, 10.0);
    cairo_arc_negative (cr, xc, yc, radius, angle1, angle2);
    cairo_stroke (cr);
    /* draw helping lines */
    cairo_set_source_rgba (cr, 1, 0.2, 0.2, 0.6);
    cairo_set_line_width (cr, 6.0);
    cairo_arc (cr, xc, yc, 10.0, 0, 2*M_PI);
    cairo_fill (cr);
    cairo_arc (cr, xc, yc, radius, angle1, angle1);
    cairo_line_to (cr, xc, yc);
    cairo_arc (cr, xc, yc, radius, angle2, angle2);
    cairo_line_to (cr, xc, yc);
    cairo_stroke (cr);
    cairo_surface_write_to_png (surface, "CairoEjemplo.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);
}
```



https://www.cairographics.org/samples/

Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    cairo_surface_t *surface;
    cairo_t *cr;
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 300, 300);
    cr = cairo_create (surface);

    cairo_move_to (cr, 50.0, 75.0);
    cairo_line_to (cr, 200.0, 75.0);

    cairo_move_to (cr, 50.0, 125.0);
    cairo_line_to (cr, 200.0, 125.0);

    cairo_move_to (cr, 50.0, 175.0);
    cairo_line_to (cr, 200.0, 175.0);

    cairo_set_line_width (cr, 30.0);
    cairo_set_line_cap (cr, CAIRO_LINE_CAP_ROUND);
    cairo_stroke (cr);

    cairo_surface_write_to_png (surface, "Lines.png");
    cairo_destroy (cr);
    cairo_surface_destroy (surface);
}
```



https://www.cairographics.org/samples/

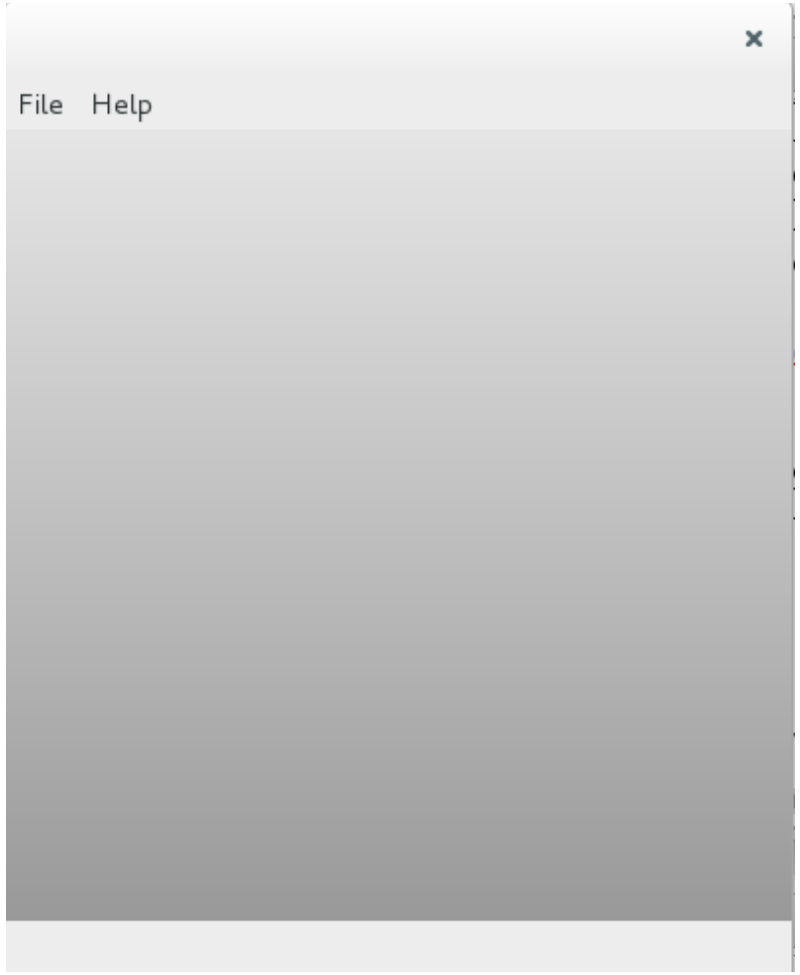
Trabajando cairo

```
void wxcairo1Frame::OnAbout(wxCommandEvent& event)
{
    int w, h;
    cairo_surface_t *image;

    image = cairo_image_surface_create_from_png ("lobo.png");
    w = cairo_image_surface_get_width (image);
    h = cairo_image_surface_get_height (image);

    cairo_surface_write_to_png (image, "lobo_cairo.png");
    cairo_surface_destroy (image);
}
```

Generando un Sistema Cartesiano

Generando un graficador 2-D (Fondo)

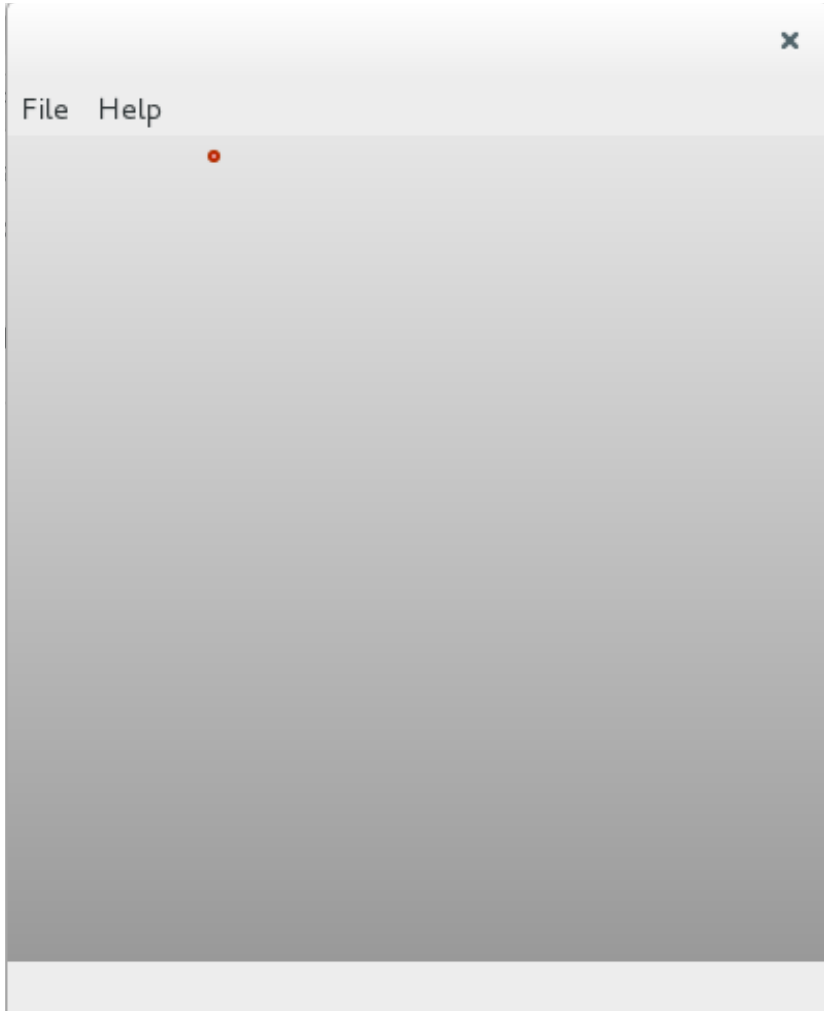
```
cairo_pattern_t *pat;
pat = cairo_pattern_create_linear (width/2, 0.0,  width/2, height);
cairo_pattern_add_color_stop_rgb (pat, 0.0, 0.9, 0.9, 0.9);
cairo_pattern_add_color_stop_rgb (pat, 1.0, 0.6, 0.6, 0.6);
cairo_rectangle (cr, 0, 0, width, height);
cairo_set_source (cr, pat);
cairo_fill (cr);
```



Generando un graficador 2-D (Data Points)

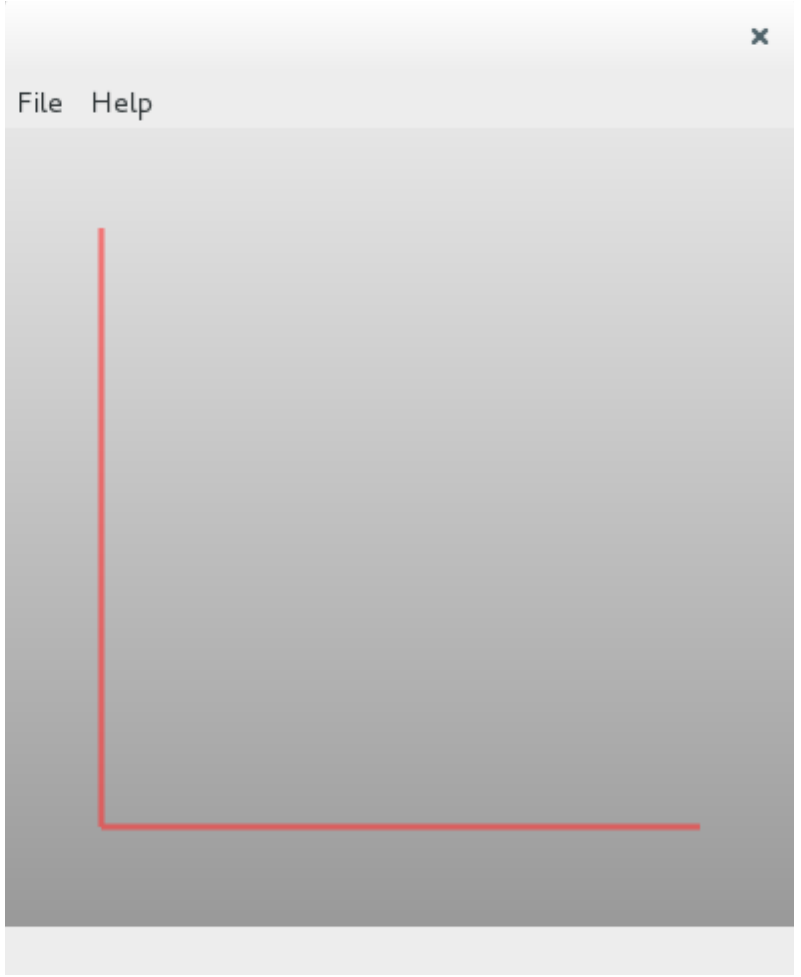
```
cairo_set_source_rgb(cr, 0.69, 0.19, 0);
cairo_arc(cr, 100, 10,2.1, 0, 2 * M_PI);
cairo_stroke_preserve(cr);
cairo_set_source_rgba(cr, 0.9, 0.0, 0.0,0.3);
cairo_fill(cr);
```

Contexto
Pos_x
Pos_y
Radio
[0,2pi]



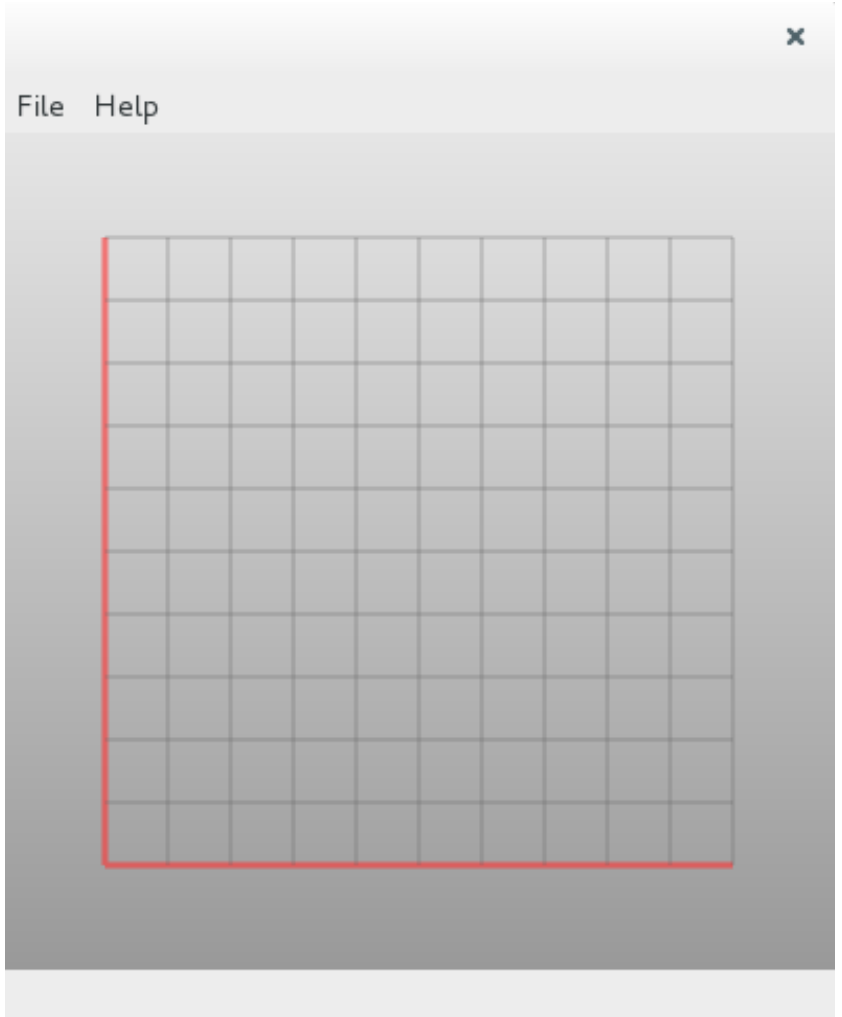
Generando un graficador 2-D (Eje)

```
//1.- Obtener el max y min de (x,y)
double xmax = Max(vectorx,tam);double xmin = Min(vectorx,tam);
double ymax = Max(vectory,tam);double ymin = Min(vectory,tam);
//2.-Ubicar el plano cartesiano a) origen b) eje x c) eje y
double ejex = width - (width/8);//proporcion del 1/8 sobre el canvas
double ejey = height/8;//proporcion del 1/8 sobre el canvas
double origenx = width/8;//proporcion del 1/8 sobre el canvas
double origeny = height - (height/8);//proporcion del 1/8 sobre el canvas
//pintado de los ejes x,y
cairo_set_source_rgba (cr, 1, 0.2, 0.2, 0.6);
cairo_set_line_width (cr, 3.0);
cairo_move_to (cr,origenx,origeny); cairo_line_to (cr,ejex,origeny);//eje x
cairo_move_to (cr,origenx,origeny); cairo_line_to (cr,origenx,ejey);//eje y
```



Generando un graficador 2-D (Grid)

```
//pintado del grid
double metrica_ejex = (ejex - origenx)/10.0;//proporcion de longit
double metrica_ejey = (origeny - ejey)/10.0;//proporcion de longit
for (int indice=1; indice < 11; indice++)
{
    cairo_set_source_rgb (cr,0.4, 0.4, 0.4);
    cairo_set_line_width (cr, 0.5);
    //vertical(solo cambia la posicion en x)
    cairo_move_to (cr,(origenx+(metrica_ejex*indice)),origeny);
    cairo_line_to (cr,(origenx+(metrica_ejex*indice)),ejey);
    //horizontal(solo cambia la posicion en x)
    cairo_move_to (cr,origenx,(ejey+(metrica_ejey*(indice-1))));
    cairo_line_to (cr,ejex,   (ejey+(metrica_ejey*(indice-1))));
    cairo_stroke(cr);
}
```

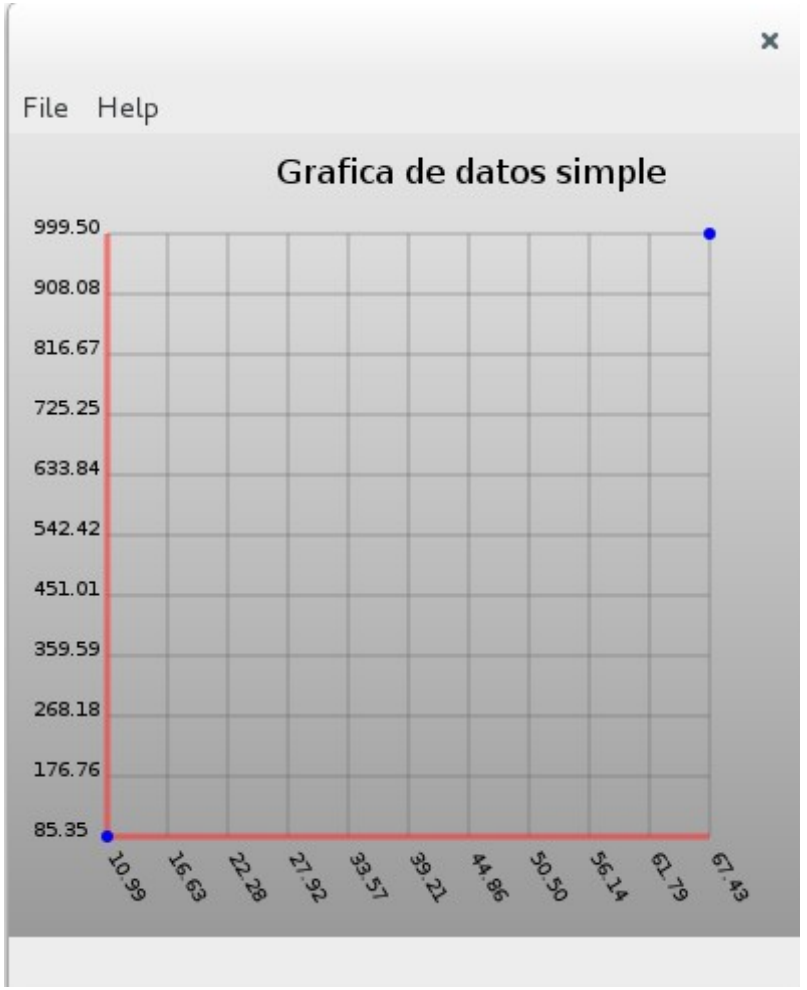


Generando un graficador 2-D (Labels)

```
//1.-Labels horizontales falta
for (int i=0; i < 11; i++)
{
    cairo_set_source_rgb(cr, 0.0, 0.0, 0.0);
    cairo_select_font_face (cr, "Sans",CAIRO_FONT_SLANT_NORMAL,C
    cairo_set_font_size (cr, 10.0);
    cairo_move_to (cr,(origenx+(metrica_ejex*i)),(origeny+10));
    cairo_save(cr);
    cairo_rotate(cr,45.0);
    cairo_show_text (cr,metricax[i]);
    cairo_restore(cr);
}
//2.- Nombre de grafica
cairo_set_source_rgb(cr, 0.0, 0.0, 0.0);
cairo_select_font_face (cr, "Sans", CAIRO_FONT_SLANT_NORMAL,CAIR
cairo_set_font_size (cr, (ejey/3.0));
cairo_move_to (cr, width/3.0,ejey/2.0);
cairo_show_text (cr, nombre);
```

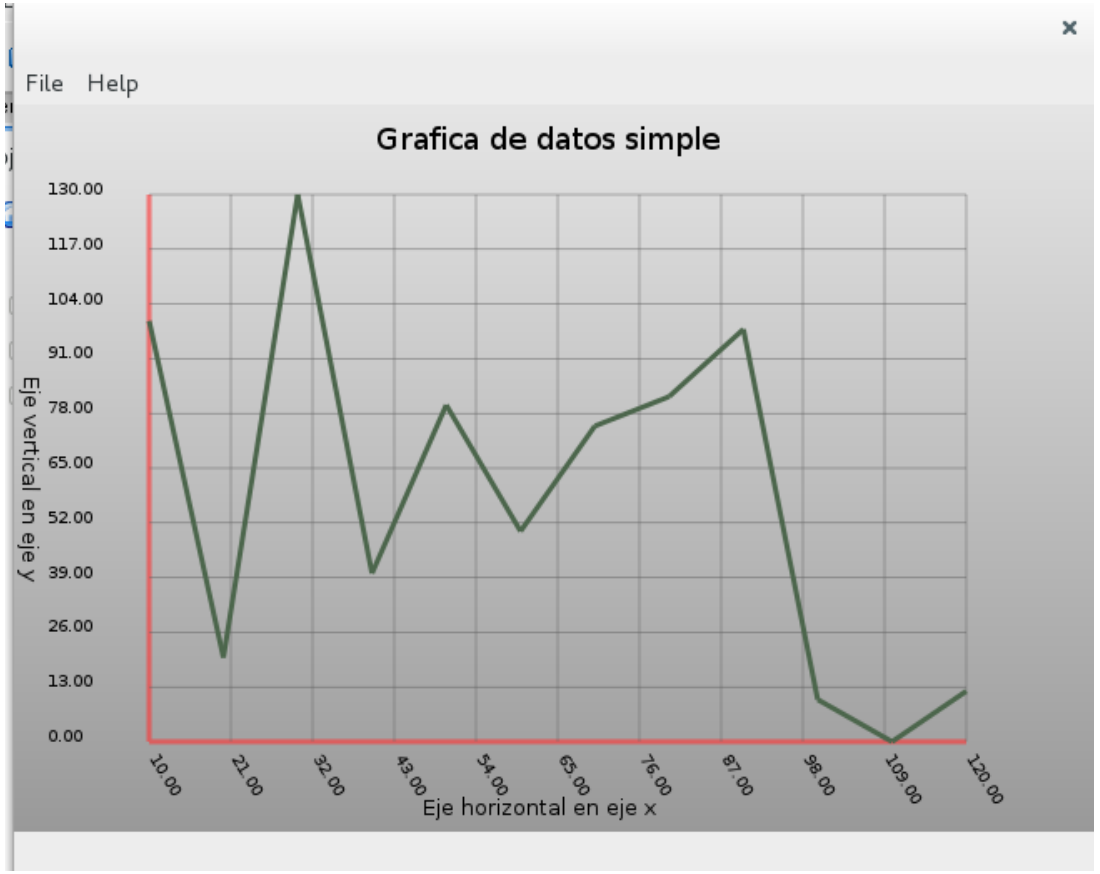
Generando un graficador 2-D (Scatter Plot)

```
//pintado de los datapoints
    for(int i=0;i<tam;i++)
    {
        double valx = Ecuacion_recta(ejex,origenx,xmax,xmin,vectorx[i]);
        double valy = Ecuacion_recta(ejey,origeny,ymax,ymin,vectory[i]);
        cairo_set_source_rgb(cr, 0.0, 0.0, 1.0);
        cairo_arc(cr, valx, valy,3,0,2*M_PI);
        cairo_fill(cr);
        cairo_stroke(cr);
    }
```

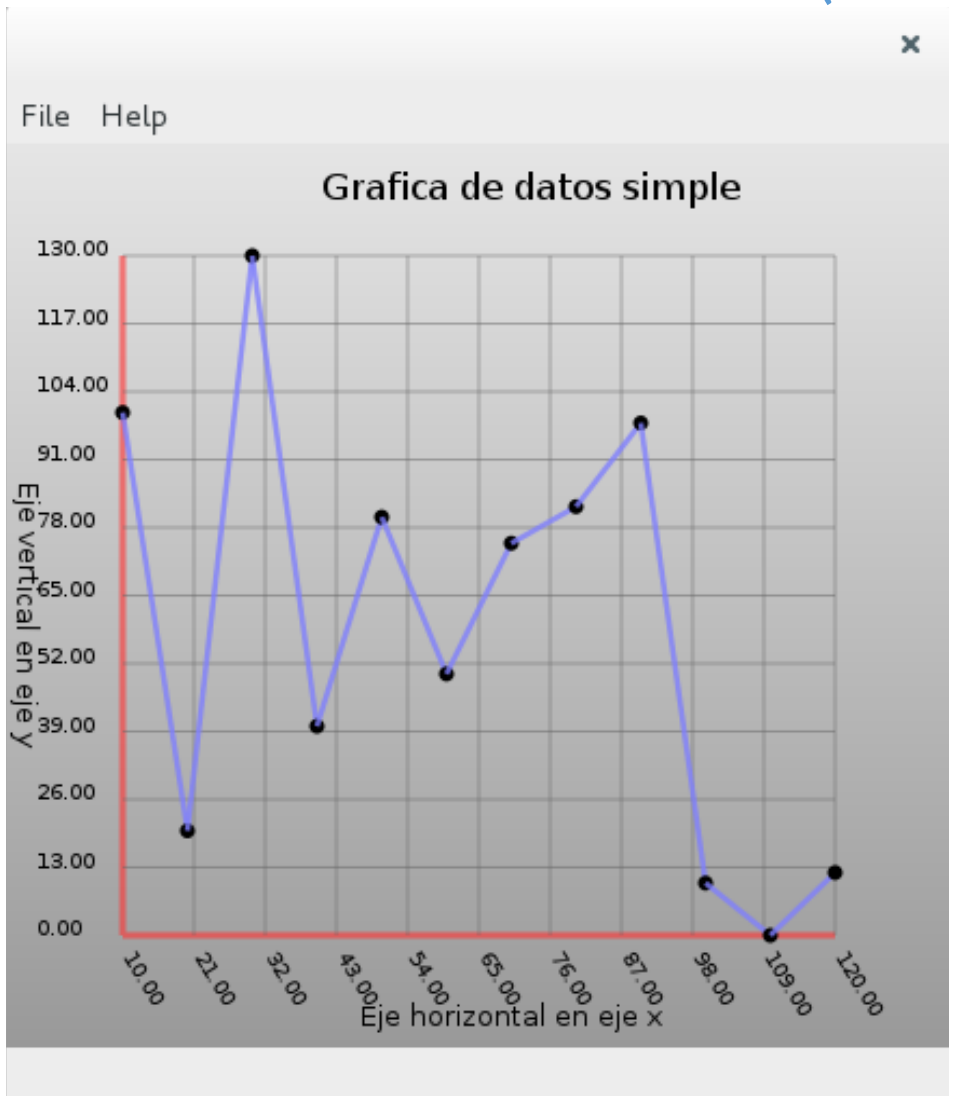


Generando un graficador 2-D (Gráfica lineal)

```
for(int i=0;i<tam;i++)
{
    double valx = Ecuacion_recta(ejex,origenx,xmax,xmin,vectorx[i]);
    double valy = Ecuacion_recta(ejey,origeny,ymax,ymin,vectory[i]);
    if(i >= 1)
    {
        cairo_set_source_rgba (cr,0.3, 0.4, 0.3, 1.0);
        cairo_set_line_width (cr, 3);
        cairo_move_to (cr,xaux,yaux);
        cairo_line_to (cr,valx,valy);
        cairo_stroke(cr);
    }
    xaux = valx;
    yaux = valy;
}
```

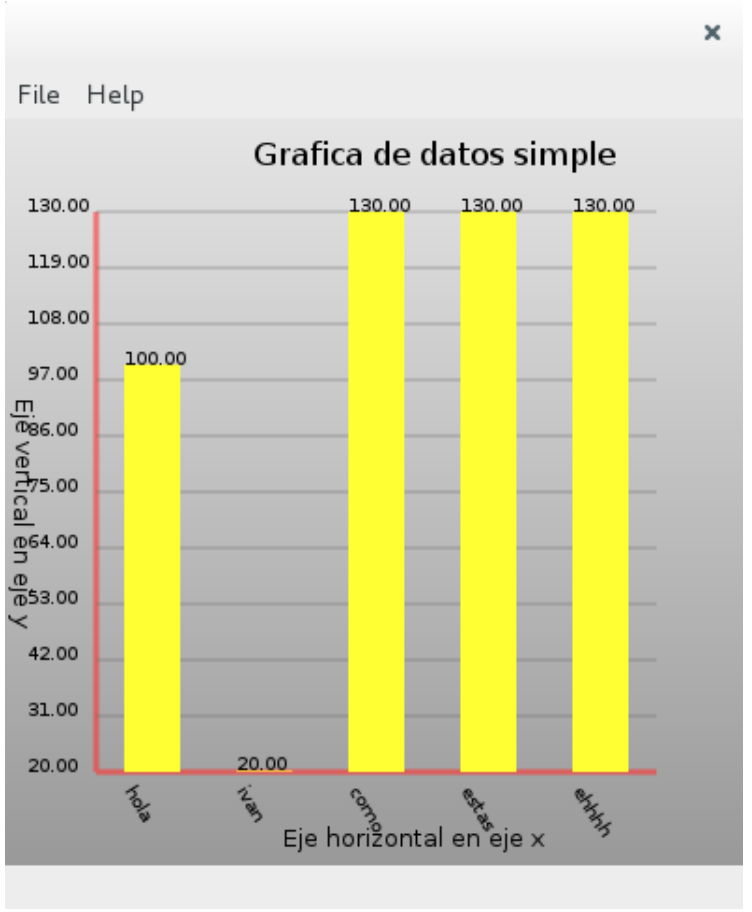


Generando un graficador 2-D
(Otras gráficas)



Generando un graficador 2-D (Bar plot)

```
if (corners == false)
{
    for(int i=0;i<tam;i++)
    {
        double valy = Ecuacion_recta(ejey,origeny,ymax,ymin,vec_vals[i]);
        cairo_set_source_rgba (cr,1.0, 1.0, 0.2, 1.0);
        if (valy == origeny)
            valy -=((origeny-ejey)/500.0);
        cairo_rectangle(cr,(origenx+(metrica_ejex*i)+15),valy,(metrica_ejex/2.0),(origeny-valy)
        cairo_fill(cr);
        cairo_stroke(cr);
        cairo_set_source_rgb(cr, 0.0, 0.0, 0.0);
        cairo_select_font_face (cr, "Sans",CAIRO_FONT_SLANT_NORMAL,CAIRO_FONT_WEIGHT_NORMAL);
        cairo_set_font_size (cr, 10.0);
```



Generando un graficador 2-D (Bar plots)