/\*

u8g\_ellipse.c

Utility to draw empty and filled ellipses.

Universal 8bit Graphics Library

Copyright (c) 2011, bjthom@gmail.com

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,

are permitted provided that the following conditions are met:

\* Redistributions of source code must retain the above copyright notice, this list

of conditions and the following disclaimer.

\* Redistributions in binary form must reproduce the above copyright notice, this

list of conditions and the following disclaimer in the documentation and/or other

materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND

CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,

INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR

CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,

STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF

ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Addition to the U8G Library as of 02/29/12

Adapted from Bresenham's Algorithm and the following websites:

http://free.pages.at/easyfilter/bresenham.html

http://homepage.smc.edu/kennedy\_john/belipse.pdf

\*/

#include "u8g.h"

#ifdef WORK\_IN\_PROGRESS

void u8g\_DrawEllipseRect(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t x1, u8g\_uint\_t y1)

{

int a = abs(x1 - x0);

int b = abs(y1 - y0); //get diameters

int b1 = b&1;

long dx = 4\*(1-a)\*b\*b;

long dy = 4\*(b1+1)\*a\*a;

long err = dx+dy+b1\*a\*a;

long e2;

if (x0 > x1) { x0 = x1; x1 += a; }

if (y0 > y1) { y0 = y1; }

y0 += (b+1)/2;

y1 = y0-b1;

a \*= 8\*a;

b1 = 8\*b\*b;

do {

u8g\_DrawPixel(u8g, x1, y0);

u8g\_DrawPixel(u8g, x0, y0);

u8g\_DrawPixel(u8g, x0, y1);

u8g\_DrawPixel(u8g, x1, y1);

e2 = 2\*err;

if (e2 >= dx) {

x0++;

x1--;

err += dx += b1;

}

if (e2 <= dy) {

y0++;

y1--;

err += dy += a;

}

} while (x0 <= x1);

while (y0-y1 < b) {

u8g\_DrawPixel(u8g, x0-1, y0);

u8g\_DrawPixel(u8g, x1+1, y0++);

u8g\_DrawPixel(u8g, x0-1, y1);

u8g\_DrawPixel(u8g, x1+1, y1--);

}

}

void u8g\_DrawEllipse(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t xr, u8g\_uint\_t yr)

{

u8g\_DrawPixel(u8g, x0, y0+yr);

u8g\_DrawPixel(u8g, x0, y0-yr);

u8g\_DrawPixel(u8g, x0+xr, y0);

u8g\_DrawPixel(u8g, x0-xr, y0);

}

#endif

#if defined(U8G\_16BIT)

typedef int32\_t u8g\_long\_t;

#else

typedef int16\_t u8g\_long\_t;

#endif

/\*

Source:

ftp://pc.fk0.name/pub/books/programming/bezier-ellipse.pdf

Foley, Computer Graphics, p 90

\*/

static void u8g\_draw\_ellipse\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option) U8G\_NOINLINE;

static void u8g\_draw\_ellipse\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option)

{

/\* upper right \*/

if ( option & U8G\_DRAW\_UPPER\_RIGHT )

{

u8g\_DrawPixel(u8g, x0 + x, y0 - y);

}

/\* upper left \*/

if ( option & U8G\_DRAW\_UPPER\_LEFT )

{

u8g\_DrawPixel(u8g, x0 - x, y0 - y);

}

/\* lower right \*/

if ( option & U8G\_DRAW\_LOWER\_RIGHT )

{

u8g\_DrawPixel(u8g, x0 + x, y0 + y);

}

/\* lower left \*/

if ( option & U8G\_DRAW\_LOWER\_LEFT )

{

u8g\_DrawPixel(u8g, x0 - x, y0 + y);

}

}

void u8g\_draw\_ellipse(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rx, u8g\_uint\_t ry, uint8\_t option)

{

u8g\_uint\_t x, y;

u8g\_long\_t xchg, ychg;

u8g\_long\_t err;

u8g\_long\_t rxrx2;

u8g\_long\_t ryry2;

u8g\_long\_t stopx, stopy;

rxrx2 = rx;

rxrx2 \*= rx;

rxrx2 \*= 2;

ryry2 = ry;

ryry2 \*= ry;

ryry2 \*= 2;

x = rx;

y = 0;

xchg = 1;

xchg -= rx;

xchg -= rx;

xchg \*= ry;

xchg \*= ry;

ychg = rx;

ychg \*= rx;

err = 0;

stopx = ryry2;

stopx \*= rx;

stopy = 0;

while( stopx >= stopy )

{

u8g\_draw\_ellipse\_section(u8g, x, y, x0, y0, option);

y++;

stopy += rxrx2;

err += ychg;

ychg += rxrx2;

if ( 2\*err+xchg > 0 )

{

x--;

stopx -= ryry2;

err += xchg;

xchg += ryry2;

}

}

x = 0;

y = ry;

xchg = ry;

xchg \*= ry;

ychg = 1;

ychg -= ry;

ychg -= ry;

ychg \*= rx;

ychg \*= rx;

err = 0;

stopx = 0;

stopy = rxrx2;

stopy \*= ry;

while( stopx <= stopy )

{

u8g\_draw\_ellipse\_section(u8g, x, y, x0, y0, option);

x++;

stopx += ryry2;

err += xchg;

xchg += ryry2;

if ( 2\*err+ychg > 0 )

{

y--;

stopy -= rxrx2;

err += ychg;

ychg += rxrx2;

}

}

}

void u8g\_DrawEllipse(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rx, u8g\_uint\_t ry, uint8\_t option)

{

/\* check for bounding box \*/

{

u8g\_uint\_t rxp, rxp2;

u8g\_uint\_t ryp, ryp2;

rxp = rx;

rxp++;

rxp2 = rxp;

rxp2 \*= 2;

ryp = ry;

ryp++;

ryp2 = ryp;

ryp2 \*= 2;

if ( u8g\_IsBBXIntersection(u8g, x0-rxp, y0-ryp, rxp2, ryp2) == 0)

return;

}

u8g\_draw\_ellipse(u8g, x0, y0, rx, ry, option);

}

static void u8g\_draw\_filled\_ellipse\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option) U8G\_NOINLINE;

static void u8g\_draw\_filled\_ellipse\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option)

{

/\* upper right \*/

if ( option & U8G\_DRAW\_UPPER\_RIGHT )

{

u8g\_DrawVLine(u8g, x0+x, y0-y, y+1);

}

/\* upper left \*/

if ( option & U8G\_DRAW\_UPPER\_LEFT )

{

u8g\_DrawVLine(u8g, x0-x, y0-y, y+1);

}

/\* lower right \*/

if ( option & U8G\_DRAW\_LOWER\_RIGHT )

{

u8g\_DrawVLine(u8g, x0+x, y0, y+1);

}

/\* lower left \*/

if ( option & U8G\_DRAW\_LOWER\_LEFT )

{

u8g\_DrawVLine(u8g, x0-x, y0, y+1);

}

}

void u8g\_draw\_filled\_ellipse(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rx, u8g\_uint\_t ry, uint8\_t option)

{

u8g\_uint\_t x, y;

u8g\_long\_t xchg, ychg;

u8g\_long\_t err;

u8g\_long\_t rxrx2;

u8g\_long\_t ryry2;

u8g\_long\_t stopx, stopy;

rxrx2 = rx;

rxrx2 \*= rx;

rxrx2 \*= 2;

ryry2 = ry;

ryry2 \*= ry;

ryry2 \*= 2;

x = rx;

y = 0;

xchg = 1;

xchg -= rx;

xchg -= rx;

xchg \*= ry;

xchg \*= ry;

ychg = rx;

ychg \*= rx;

err = 0;

stopx = ryry2;

stopx \*= rx;

stopy = 0;

while( stopx >= stopy )

{

u8g\_draw\_filled\_ellipse\_section(u8g, x, y, x0, y0, option);

y++;

stopy += rxrx2;

err += ychg;

ychg += rxrx2;

if ( 2\*err+xchg > 0 )

{

x--;

stopx -= ryry2;

err += xchg;

xchg += ryry2;

}

}

x = 0;

y = ry;

xchg = ry;

xchg \*= ry;

ychg = 1;

ychg -= ry;

ychg -= ry;

ychg \*= rx;

ychg \*= rx;

err = 0;

stopx = 0;

stopy = rxrx2;

stopy \*= ry;

while( stopx <= stopy )

{

u8g\_draw\_filled\_ellipse\_section(u8g, x, y, x0, y0, option);

x++;

stopx += ryry2;

err += xchg;

xchg += ryry2;

if ( 2\*err+ychg > 0 )

{

y--;

stopy -= rxrx2;

err += ychg;

ychg += rxrx2;

}

}

}

void u8g\_DrawFilledEllipse(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rx, u8g\_uint\_t ry, uint8\_t option)

{

/\* check for bounding box \*/

{

u8g\_uint\_t rxp, rxp2;

u8g\_uint\_t ryp, ryp2;

rxp = rx;

rxp++;

rxp2 = rxp;

rxp2 \*= 2;

ryp = ry;

ryp++;

ryp2 = ryp;

ryp2 \*= 2;

if ( u8g\_IsBBXIntersection(u8g, x0-rxp, y0-ryp, rxp2, ryp2) == 0)

return;

}

u8g\_draw\_filled\_ellipse(u8g, x0, y0, rx, ry, option);

}