DrawArea

COLLABORATORS			
	TITLE : DrawArea		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		September 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

DrawArea

1.1 Implementation notes

```
                          The A++ DrawArea classes
                         ------------------------------
                         ($Date: 1994/05/09 21:24:54 $)


                        The A++ DrawArea class - an introduction

                        The A++ AutoDrawArea class

                        The A++ Canvas class

                        The A++ TextView class
                          ------------------------------------------------------------
 -> Back to the root menu..
```

1.2 drawarea_1

```
                       The DrawArea class - an introduction
                     ----------------------------------------
```

The DrawArea class supplies all Graphics-Rastport draw functions clipped in a rectangular bounding box.
A DrawArea gets its dimensions from a RectObject, like a GraphicObject. Therefore, all draw methods take coordinates that are relative to the RectObject's inner bounding box.

First, set up pen color, mode and pattern according to the Graphics library. Note, that on method entry you must not assume any color, mode or pattern being active. So, set up all your parameters before drawing:

```
   void setAPen(UBYTE);    // foreground pen
   void setBPen(UBYTE);    // background pen
   void setOPen(BYTE);     // area outline pen
   void setDrMd(UBYTE);    // set drawmode
```

```
   void setDrPt(UWORD);    // 16 bit line draw pattern
```

Then, set up clipping (which is strongly recommended to do):

1. If you do not know after a window resizing, wether the new dimensions had been applied to the clip region or not, use:

```
   // adjust the clipping to the present values of the RectObject
   void adjustStdClip();
```

   If you like to set up your personal clip regions, you can specify rectangles that shall be included or erased from the standard clip:

```
   // use the following methods to create your own clipping areas
   void andRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
   void orRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
   void xorRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
   void clearRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
```

2. Then, activate the DrawArea's clip region:

```
   // installs standard clipping on the inner boundaries of the RectObject
   void setStdClip();
```

3. Draw...
   (You may alter the clip region with some of the methods above during the draw period. The change takes effect immediately when setStdClip() has been called once before under (2.). adjustStdClip() always resets to the RectObject's inner bounding box.)

4. Deactivate clip region and reset to the previously active region:

```
   // reinstall previous clip region
   void resetStdClip();
```

Now, the draw methods follow. They work in exactly the same way they work in the Graphics library, except for the advantage that they can be clipped to the RectObject's inner bounding box:

```
   void draw(XYVAL x,XYVAL y);
   // draw a GadTools bevel box
   void drawBevelBox(XYVAL xmin,XYVAL ymin,WHVAL width,WHVAL height,BOOL ↩
      recessed);
   void drawEllipse(XYVAL x,XYVAL y,WHVAL hr,WHVAL vr);

   void move(XYVAL x,XYVAL y);
   void moveTx(XYVAL x,XYVAL y);
   // places graphic cursor to upper left edge of text render box for
   // subsequent text() calls; considers baseline of the current font
   // set with setFont()

   // give a table of RectObject-relative coordinates
   void polyDraw(LONG count,WORD *polyTable);

   // max values being smaller than min values are catched and can no longer
   // cause GURU MEDITATION
```

```
   void rectFill(XYVAL xmin,XYVAL ymin,XYVAL xmax,XYVAL ymax);
   void scrollRaster(LONG dx,LONG dy,XYVAL xmin,XYVAL ymin,XYVAL xmax,XYVAL  ↩
       ymax);

   // set the font to use in subsequent text() calls
   void setFont(FontC& font);
   // if you leave out textLength the string will be measured by the method.
   void text(UBYTE *textString,UWORD textLength=0);

If you like to set up your personal clip regions,
   // use the following methods to create your own clipping areas
   void andRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
   void orRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
   void xorRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
   void clearRectRegion(XYVAL MinX,XYVAL MinY,XYVAL MaxX,XYVAL MaxY);
```

1.3 drawarea_2

```
                              The AutoDrawArea class
             ----------------------------

The AutoDrawArea class combines a Standard boolean Gadget with a DrawArea.
The DrawArea is adjusted in the Gadget's RectObject inner bounding box.

The class declares a virtual method

   virtual void drawSelf();

You can overwrite this method with your own method that uses
             DrawArea methods
                to draw into the inner box. On method entry, the standard  ↩
                   clipping
region is already set up, which confines drawing to the inner box.
```

1.4 canvas_main

```
                              The Canvas class
            ------------------------

The Canvas class displays a rectangular view that shows a portion of a
virtual draw area. The user can specify a callback method where drawing into
the virtual bitmap can be done. The canvas clips the drawing that goes
outside the visible view. The view window can be moved all over the canvas
draw area. Scroller gadgets may easily be assigned to move the view.
.

               The Canvas Attribute Tags
                 ------------------------------------------------------

               -> Back to the root menu..
```

1.5 canvas_1

The canvas plane can be imagined as being covered by a lattice that divides it into equal rectangles, each rectangle having a width of 'GranularityX' and a height if 'GranularityY' pixels. The rectangles count from 0 to the number of rectangles fitting the width/heigth of the canvas.

The view window can be moved in steps of 'GranularityX'/'GranularityY' pixels only, but all drawing methods still expect coordinates and dimensions given in RastPort pixels!

Use the following tags with the get/setAttributes() method:

1. Dimension of the canvas (set/get)

   CNV_Width,(UWORD) : the width of the canvas in GranularityX units

   CNV_Height,(UWORD): the height of the canvas in GranularityY units

2. Position of the view window over the canvas (set/get):

   CNV_ViewX,(UWORD) : the lowest visible lattice rectangle
   (0-Width/GranularityX)
   CNV_ViewY,(UWORD) : the lowest visible lattice rectangle
   (0-Height/GranularityY)

3. The granularity in which the view window can be moved (set/get):

   CNV_GranularityX,(UWORD) : the width of a lattice rectangle in pixels

   CNV_GranularityY,(UWORD) : the height of a lattice rectangle in pixels

4. The dimension of the view window (get only, since it depends on the size of the Canvas object):

   CNV_VisibleX,(UWORD) : horizontal lattice rectangles totally visible

   CNV_VisibleY,(UWORD) : vertical lattice rectangles totally visible

5. Specify, wether you want any only partly visible lattice rectangles on the right and lower side of the view window get scrolled with the view or not (set/get):

   CNV_ScrollGratX,(BOOL) : TRUE for including the border when scrolling

   CNV_ScrollGratY,(BOOL) : TRUE for including the border when scrolling

1.6 canvas_2

1.7 textview_main

The TextView class

The TextView utilizes the Canvas class to display lines of strings.
It is totally customizeable to the point of where to get the line strings
from and how to format the output.
All proportional and non-proportional fonts can be used.
A cursor is available for moving through the text. It can be moved by method
call or by the user's mouse actions.
Furthermore a portion of text can be marked with the mouse.
Except for mouse events user input is not considered in the TextView class.
Connection to further input processing can be achieved easily.

How to create TextView objects
How to access an TextView

-> Back to the root menu..