```c
/*
 * texticlass.c Copyright (C) 1991 Commodore-Amiga, Inc. All Rights Reserved Worldwide
 * Written by David N. Junod
 *
 * Compiled with SAS/C 5.10a LC -cfist -ms -v (must be linked with classface.o and
 * hookface.o)
 *
 * The Image structure as used by this class:
 *
 * struct Image {
 *
 * SHORT    LeftEdge;        <----Offset relative to the container
 * SHORT    TopEdge;
 *
 * SHORT    Width;           <----Contains the text extent of the string
 * SHORT    Height;
 *
 * SHORT    Depth;           <----Maintained by boopsi (must be set to CUSTOMIMAGEDEPTH).
 *
 * USHORT   *ImageData;      <----Pointer to a NULL terminated text string
 *
 * UBYTE    PlanePick;       <----We use this for the foreground color
 *
 * UBYTE    PlaneOnOff;      <----We use this for the background color
 *
 * struct Image *NextImage;  <----Pointer to the next image.  Handled by DrawImage(). };
 */

#include <exec/types.h>
#include <exec/memory.h>
#include <exec/libraries.h>
#include <intuition/intuition.h>
#include <intuition/classes.h>
#include <intuition/classusr.h>
#include <intuition/cghooks.h>
#include <intuition/gadgetclass.h>
#include <intuition/imageclass.h>
#include <intuition/icclass.h>
#include <intuition/screens.h>
#include <graphics/gfx.h>
#include <graphics/gfxmacros.h>
#include <libraries/gadtools.h>
#include <utility/tagitem.h>
#include <clib/macros.h>
#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/graphics_protos.h>
#include <clib/utility_protos.h>
#include <string.h>

extern struct Library *SysBase, *DOSBase;
extern struct Library *IntuitionBase, *GfxBase, *UtilityBase;

/*
 * Because we are dealing with imageclass objects, the data structure that makes up the
 * object is an intuition Image structure.
 */
#define IM(o)   ((struct Image *)(o))

#define MYCLASSID       NULL
#define SUPERCLASSID    (IMAGECLASS)

Class          *initmyTextLabelClass(VOID);
ULONG           freemyTextLabelClass(Class * cl);
ULONG __saveds  dispatchmyTextLabel(Class * cl, Object * o, Msg msg);
ULONG           setmyTextLabelAttrs(Class * cl, Object * o, struct opSet * msg);
ULONG           getmyTextLabelAttr(Class * cl, Object * o, struct opGet * msg);
ULONG           drawmyTextLabel(Class * cl, Object * o, struct impDraw * msg);
WORD            aTextExtent(struct RastPort *, STRPTR, LONG, struct TextExtent *);
UWORD           GetLabelKeystroke(STRPTR label);
static VOID     getContentsExtent(Class * cl, Object * o, struct DrawInfo * drinfo);

/* prototypes of functions from classface.o */
ULONG           DoMethod(Object * o, ULONG methodID,...);
ULONG           DoSuperMethod(Class * cl, Object * o, ULONG methodID,...);
ULONG           CoerceMethod(Class * cl, Object * o, ULONG methodID,...);
ULONG           CM(Class * cl, Object * o, Msg msg);
ULONG           DM(Object * o, Msg msg);
ULONG           DSM(Class * cl, Object * o, Msg msg);
ULONG           SetSuperAttrs(Class * cl, Object * o, ULONG data,...);
```

```c
struct localObjData
{
    /* Font to use */
    struct TextFont *lod_Font;

    /* The key that is underlined */
    UWORD           lod_Key;

    /* DrawMode */
    UBYTE           lod_Mode;
};


Class          *
initmyTextLabelClass(VOID)
{
    ULONG           hookEntry(); /* from hookface.o */
    Class          *cl;

    if (cl = MakeClass(MYCLASSID,
                       SUPERCLASSID, NULL,
                       sizeof(struct localObjData), 0))
    {
        /* Fill in the callback hook */
        cl->cl_Dispatcher.h_Entry = hookEntry;
        cl->cl_Dispatcher.h_SubEntry = dispatchmyTextLabel;
    }
    /* Return a pointer to the class */
    return (cl);
}


ULONG
freemyTextLabelClass(Class * cl)
{

    /* Try to free the public class */
    return ((ULONG) FreeClass(cl));
}


/*
 * The SAS "__saveds" flag tells the SAS compiler to put
 * the data storage address + 32766 into A4.
 */
ULONG           __saveds
dispatchmyTextLabel(Class * cl, Object * o, Msg msg)
{
    struct localObjData *lod;
    Object         *newobj;
    ULONG           retval;

    switch (msg->MethodID)
    {
    case OM_NEW:
        /* Pass up to the superclass... */
        if (newobj = (Object *) DSM(cl, o, msg))
        {
            struct TagItem *attrs = ((struct opSet *) msg)->ops_AttrList;
            struct DrawInfo *drinfo;

            /* Get the DrawInfo */
            drinfo = (struct DrawInfo *) GetTagData(SYSIA_DrawInfo, NULL, attrs);

            /* Get the instance data */
            lod = INST_DATA(cl, newobj);

            /* Establish defaults */
            IM(newobj)->PlanePick = 1;
            lod->lod_Mode = JAM1;

            /* Set the attributes */
            setmyTextLabelAttrs(cl, newobj, (struct opSet *) msg);

            /* Get the bounding rectangle of the label */
            getContentsExtent(cl, newobj, drinfo);
        }
        retval = (ULONG) newobj;
        break;
```

```
    case OM_GET:
        retval = getmyTextLabelAttr(cl, o, (struct opGet *) msg);
        break;

    case OM_UPDATE:
    case OM_SET:
        /* Do the superclass first */
        retval = DSM(cl, o, msg);

        /* Call our set routines */
        retval += setmyTextLabelAttrs(cl, o, (struct opSet *) msg);
        break;

    case IM_DRAW:               /* draw the label */
    case IM_DRAWFRAME:          /* drawmyTextLabel() will take care of
                                   extra framing info */
        retval = drawmyTextLabel(cl, o, (struct impDraw *) msg);
        break;

    /* Let the superclass handle everything else */
    default:
        retval = (ULONG) DSM(cl, o, msg);
        break;
    }

    return (retval);
}


/* Set attributes of an object */
ULONG
setmyTextLabelAttrs(Class * cl, Object * o, struct opSet * msg)
{
    struct localObjData *lod = INST_DATA(cl, o);
    struct TagItem *tags = msg->ops_AttrList;
    struct TagItem *tstate;
    struct TagItem *tag;
    ULONG           tidata;

    /* process rest */
    tstate = tags;
    while (tag = NextTagItem(&tstate))
    {
        tidata = tag->ti_Data;
        switch (tag->ti_Tag)
        {
        case IA_FGPen:
            IM(o)->PlanePick = (UBYTE) tidata;
            break;

        case IA_BGPen:
            IM(o)->PlaneOnOff = (UBYTE) tidata;
            break;

            /* Must be a TextFont pointer. */
        case IA_Font:
            /* Set the font */
            lod->lod_Font = (struct TextFont *) tidata;
            break;

            /* Drawing mode to use */
        case IA_Mode:
            lod->lod_Mode = (UBYTE) tidata;
            break;

        case IA_Data:
            IM(o)->ImageData = (USHORT *) tidata;
            lod->lod_Key = GetLabelKeystroke((STRPTR) tidata);
            break;
        }
    }

    return (1L);
}
```

```
/* Inquire attributes of an object */
ULONG
getmyTextLabelAttr(Class * cl, Object * o, struct opGet * msg)
{
    struct localObjData *lod = INST_DATA(cl, o);

    switch (msg->opg_AttrID)
    {
    case IA_Font:
        *msg->opg_Storage = (ULONG) lod->lod_Font;
        break;

    case IA_Mode:
        *msg->opg_Storage = (ULONG) lod->lod_Mode;
        break;

        /* Let the superclass try */
    default:
        return ((ULONG) DSM(cl, o, msg));
    }

    return (1L);
}


ULONG
drawmyTextLabel(Class * cl, Object * o, struct impDraw * msg)
{
    struct localObjData *lod = INST_DATA(cl, o);
    STRPTR          label = (STRPTR) IM(o)->ImageData;
    struct DrawInfo *di = msg->imp_DrInfo;
    struct RastPort *rp = msg->imp_RPort;
    struct TextFont *tf = NULL;
    WORD            len = strlen(label);
    WORD            left, top;
    WORD            height = 0;
    WORD            width = 0;
    WORD            i;

    /* Clear the key */
    lod->lod_Key = NULL;

    /* Get a pointer to the font to use */
    if (!(tf = lod->lod_Font) && di)
    {
        tf = di->dri_Font;
    }

    /* Make sure we have font pointer */
    if (tf)
    {
        /* Set the font */
        SetFont(rp, tf);
    }
    /* Figure out our coordinates */
    top = msg->imp_Offset.Y + IM(o)->TopEdge + rp->TxBaseline;
    left = msg->imp_Offset.X + IM(o)->LeftEdge;

    /* See if we have frame information. */
    if (msg->MethodID == IM_DRAWFRAME)
    {
        /* Center the text inside the frame. */
        width = msg->imp_Dimensions.Width;
        height = msg->imp_Dimensions.Height;
        top += ((height - IM(o)->Height) > 0) ? ((height - IM(o)->Height) / 2) : 0;
        left += ((width - IM(o)->Width) > 0) ? ((width - IM(o)->Width) / 2) : 0;
    }

    /* Set the colors */
    SetAPen(rp, IM(o)->PlanePick);
    SetBPen(rp, IM(o)->PlaneOnOff);

    /* Set the drawing mode */
    SetDrMd(rp, lod->lod_Mode);

    /* Move to the start */
    Move(rp, left, top);
```

```
    /* Step through string */
    for (i = 0; i < (len - 1); i++)
    {
        /* Is this an '_' ? */
        if (label[i] == '_')
        {
            WORD            bot = (top + rp->TxHeight - rp->TxBaseline);
            WORD            mark;

            /* Draw the first part of the string */
            Text(rp, label, i);

            /* Remember where we are in the string */
            mark = rp->cp_x;

            /* Draw the underscore */
            Move(rp, mark, bot);
            Draw(rp, (mark + TextLength(rp, &label[(i + 1)], 1L) - 2), bot);

            /* Return to where we were */
            Move(rp, mark, top);

            /*
             * Draw the rest of the string.  This one is done last so that the cursor
             * could be positioned after the text.
             */
            Text(rp, &label[(i + 1)], (len - i - 1));

            /* Return the underlined character */
            lod->lod_Key = (UWORD) label[i];
        }
    }

    /* Do we have an underscore? */
    if (!lod->lod_Key)
    {
        /* Didn't find an '_' sign */
        Text(rp, label, len);
    }
    return (1L);
}


UWORD
GetLabelKeystroke(STRPTR label)
{
    LONG            count = (label) ? strlen(label) : 0L;
    LONG            i;

    /* Search for an _ sign */
    for (i = 0; i < (count - 1); i++)
    {
        /* Did we find an _ sign? */
        if (label[i] == '_')
        {
            return ((UWORD) label[(i + 1)]);
        }
    }

    return (0);
}


/* TextExtent that honors the '_' as being a non-printable character (once) */
WORD
aTextExtent(struct RastPort * rp, STRPTR string, LONG count, struct TextExtent * te)
{
    WORD retval = FALSE;
    STRPTR buffer;
    LONG i;

    /* Allocate a temporary buffer */
    if (buffer = AllocVec ((count + 1), MEMF_CLEAR))
    {
       /* Step through string */
       for (i = 0; i < count; i++)
       {
            /* Is this an '_' sign? */
            if (string[i] == '_')
            {
              /* Add the rest of the label to the buffer */
              strcat (buffer, &string[(i + 1)]);

              /* Adjust the length of the string. */
              count--;
              break;
           }
           else
           {
              /* Copy each character over, until we reach the _ mark */
              buffer[i] = string[i];
           }
        }

        /* Get the extent */
        TextExtent(rp, buffer, count, te);

       /* Free the temporary buffer */
       FreeVec (buffer);

       /* Show that we were successful */
       retval = TRUE;
    }

    /* Return whatever textextent returned */
    return (retval);
}

static          VOID
getContentsExtent(Class * cl, Object * o, struct DrawInfo * drinfo)
{
    struct localObjData *lod = INST_DATA(cl, o);
    struct TextExtent te =
    {NULL};
    struct RastPort rp;
    STRPTR          label;

    /* maybe look at some flags to handle other types of text someday */
    if (label = (STRPTR) IM(o)->ImageData)
    {
        /* Initialize the RastPort */
        InitRastPort(&rp);

        if (lod->lod_Font)
        {
            SetFont(&rp, lod->lod_Font);
        }
        else if (drinfo && drinfo->dri_Font)
        {
            SetFont(&rp, drinfo->dri_Font);
        }
        /* Get the rectangle for the label */
        aTextExtent(&rp, label, strlen(label), &te);

        /* Set the image structure */
        IM(o)->Width = te.te_Width;
        IM(o)->Height = te.te_Height;
    }
    else
    {
        IM(o)->Width = IM(o)->Height = 0;
    }
}
```

```
/*
 * example.c Copyright (C) 1991 Commodore-Amiga, Inc. All Rights Reserved Worldwide
 * Written by David N. Junod
 *
 * Compiled with SAS/C 5.10a LC -cfist -ms -v (must be linked with
 * texticlass.o, classface.o and hookface.o)
 *
 */

#include <exec/types.h>
#include <exec/libraries.h>
#include <intuition/intuition.h>
#include <intuition/classes.h>
#include <intuition/classusr.h>
#include <intuition/cghooks.h>
#include <intuition/gadgetclass.h>
#include <intuition/imageclass.h>
#include <graphics/gfx.h>
#include <graphics/gfxmacros.h>
#include <libraries/gadtools.h>
#include <utility/tagitem.h>
#include <clib/macros.h>
#include <clib/exec_protos.h>
#include <clib/dos_protos.h>
#include <clib/intuition_protos.h>
#include <clib/graphics_protos.h>
#include <clib/utility_protos.h>
#include <string.h>

extern struct Library *SysBase, *DOSBase;
struct Library *IntuitionBase, *GfxBase, *UtilityBase;

Class          *initmyTextLabelClass(VOID);
ULONG           freemyTextLabelClass(Class * cl);

VOID
main(VOID)
{
    Class          *cl;
    struct Image   *im;
    struct Window  *win;
    struct RastPort *rp;
    UWORD           top, left, height;

    /* Make sure we're at least using Version 2.0 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36))
    {
        GfxBase = OpenLibrary("graphics.library", 36);
        UtilityBase = OpenLibrary("utility.library", 36);

        /* Open a window, without system gadgets or IDCMP events */
        if (win = OpenWindowTags(NULL,
                                 WA_Left, 10,
                                 WA_Top, 10,
                                 WA_Width, 320,
                                 WA_Height, 100,
                                 TAG_DONE))
        {
            /* Cache the pointer to the RastPort */
            rp = win->RPort;

            /* Cache the upper-left coordinates of the window */
            top = win->BorderTop + INTERHEIGHT;
            left = win->BorderRight + INTERWIDTH;

            /* Cache the height of the font */
            height = rp->TxHeight + INTERHEIGHT;

            /* Initialize the custom image class. */
            if (cl = initmyTextLabelClass())
            {
                /* Create a new image structure, using the given string. */
                if (im = NewObject(cl, NULL,
                                   IA_Data, (ULONG) "Line _1",
                                   TAG_DONE))
                {
                    /* Paint using the provided text string. */
                    DrawImageState(rp, im, left, top,
                                   IDS_NORMAL, NULL);

                    /* Replace the text string, and paint it. */
                    im->ImageData = (USHORT *) "Line _2";
                    DrawImageState(rp, im, left, top + height,
                                   IDS_NORMAL, NULL);

                    /* Replace the text string, and paint it. */
                    im->ImageData = (USHORT *) "Line _3";
                    DrawImageState(rp, im, left, top + (height * 2),
                                   IDS_NORMAL, NULL);

                    /* Free the image. */
                    DisposeObject(im);
                }

                /* Free the image class. */
                freemyTextLabelClass(cl);
            }

            Delay(250);
            CloseWindow(win);
        }

        CloseLibrary(UtilityBase);
        CloseLibrary(GfxBase);
        CloseLibrary(IntuitionBase);
    }
}
```

❖