```
;/* Jumpy.c - Execute me to compile me with Lattice 5.10a
LC -b1 -cfistq -v -y -j73 Jumpy.c
Blink FROM LIB:c.o,Jumpy.o TO Jumpy LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <graphics/text.h>
#include <libraries/gadtools.h>

#ifdef LATTICE
#include <string.h>
#include <clib/alib_protos.h>
#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/gadtools_protos.h>
#include <clib/graphics_protos.h>
/* disable SAS/C CTRL-C handing */
int             CXBRK(void)
{
    return (0);
}
int             chkabort(void)
{
    return (0);
}

#endif

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Library *GadToolsBase;
struct Library *IconBase;
struct Library *CxBase;

LONG            main(LONG, UBYTE **);

LONG
main(LONG argc, UBYTE ** argv)
{
    struct Window       *window;
    struct IntuiMessage *imsg;
    struct Gadget       *gadgetcontext;
    struct Gadget       *gadget, *nextscreengadget;
    struct NewGadget     ng;
    struct TextExtent    textextent;
    UWORD                left, top;
    void                *visualinfo;
    UBYTE               *startupname;
    UBYTE                namebuffer[MAXPUBSCREENNAME];
    UBYTE              **tooltypes;
    BOOL                 ABORT = FALSE;

    if (IntuitionBase = OpenLibrary("intuition.library", 37))
    {
        /* Open GfxBase to use TextExtent() so we can handle proportional fonts */
        if (GfxBase = OpenLibrary("graphics.library", 37))
        {
            if (GadToolsBase = OpenLibrary("gadtools.library", 37))
            {

                /*
                 * Open commodities & icon.library so we can use ArgArray
                 * functions
                 */
                if (CxBase = OpenLibrary("commodities.library", 37))
                {
                    if (IconBase = OpenLibrary("icon.library", 37))
                    {
                        left = 50;
                        top = 50;               /* Initial offset */
                        /* Note that these are functions in amiga.lib */
                        if (tooltypes = ArgArrayInit(argc, argv))
                        {
                            startupname =
                                ArgString(tooltypes, "PUBSCREEN", "Workbench");
                            strcpy(namebuffer, startupname);
                            ArgArrayDone();
                        }
                        else
                            strcpy(namebuffer, "Workbench");
                        do
                        {
                            /* open a window with tags */
                            /* no NewWindow structure, tags only */
                            if (window = OpenWindowTags(NULL,
                                        /* Open at far left corner */
                                        WA_Left,  left,
                                        WA_Top,   top,
                                        WA_Width, 150,
                                        WA_Height, 80,
                                        WA_Title, (LONG) "jumpy",
                                        WA_PubScreenName, (LONG) namebuffer,
                                        /* if no pubscreen with this name exists... */
                                        WA_PubScreenFallBack, TRUE,
                                        /* ...fall back on default pubscreen */
                                        WA_Flags, WFLG_DRAGBAR | WFLG_DEPTHGADGET |
                                            WFLG_CLOSEGADGET | WFLG_ACTIVATE |
                                            WFLG_SMART_REFRESH | WFLG_NOCAREREFRESH,
                                        WA_IDCMP, IDCMP_CLOSEWINDOW | IDCMP_GADGETUP,
                                        TAG_DONE))
                            {

                                /*
                                 * Get the visual info gadtools needs for the
                                 * screen we opened on
                                 */
                                if (visualinfo = GetVisualInfoA(window->WScreen, NULL))
                                {

                                    /*
                                     * Create a simple gadtools button and sort
                                     * of lay it out. Note this doesn't do any
                                     * checking for legal (window) dimensions.
                                     */
                                    if (gadget = CreateContext(&gadgetcontext))
                                    {

                                        /*
                                         * Use TextExtent to handle
                                         * proportional fonts
                                         */
                                        TextExtent(&(window->WScreen->RastPort),
                                                   "Jump", 4, &textextent);
                                        ng.ng_Width = textextent.te_Width + 8;
                                        ng.ng_LeftEdge = (window->Width / 2) -
                                                         (ng.ng_Width / 2);
                                        ng.ng_Height = textextent.te_Height + 4;
                                        ng.ng_TopEdge = (
                                            (window->Height - window->BorderTop -
                                                window->BorderBottom) / 2) +
                                            (ng.ng_Height / 2);
                                        ng.ng_TextAttr = window->WScreen->Font;
                                        ng.ng_GadgetText = "Jump";
                                        ng.ng_VisualInfo = visualinfo;
                                        ng.ng_GadgetID = 1;
                                        ng.ng_Flags = PLACETEXT_IN;
                                        nextscreengadget = gadget =
                                            CreateGadget(BUTTON_KIND, gadget, &ng,
                                                         TAG_END);
                                        AddGList(window, gadget, -1, -1, NULL);
                                        RefreshGList(gadget, window, NULL, -1);
                                        GT_RefreshWindow(window, NULL);

                                        WaitPort(window->UserPort);
                                        while (imsg = (struct IntuiMessage *)
                                            GetMsg(window->UserPort))
                                        {
                                            if (imsg->Class == IDCMP_CLOSEWINDOW)
                                                ABORT = TRUE;
                                            else if (imsg->Class = IDCMP_GADGETUP)
                                                NextPubScreen(window->WScreen,
                                                              namebuffer);
```

```
                                        ReplyMsg((struct Message *) imsg);
                                    }
                                    RemoveGadget(window, nextscreengadget);
                                    FreeGadgets(gadgetcontext);
                                }
                                FreeVisualInfo(visualinfo);
                            }
                            left = window->LeftEdge;
                            top = window->TopEdge;
                            CloseWindow(window);
                        }
                    } while (ABORT == FALSE);

                    CloseLibrary(IconBase);
                }
                CloseLibrary(CxBase);
            }
            CloseLibrary(GadToolsBase);
        }
        CloseLibrary(GfxBase);
    }
    CloseLibrary(IntuitionBase);
}
    return (0);
}
```

```
;/* Zoom.c - Execute me to compile me with Lattice 5.10a
LC -b1 -cfistq -v -y -j73 Zoom.c
Blink FROM LIB:c.o,Zoom.o TO Zoom LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/
#include <intuition/intuition.h>
#include <intuition/screens.h>

#ifdef LATTICE
#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
/* disable SAS/C CTRL-C handing */

int             CXBRK(void)
{
    return (0);
}

int             chkabort(void)
{
    return (0);
}
#endif

struct IntuitionBase *IntuitionBase;

LONG            main(void);

LONG main(void)
{
    struct IBox     ibox;    /* The structure we'll use to specify the zoom'ed
                              * dimension. */
    struct Screen  *wbscreen;
    struct Window  *window;
    struct Message *msg;     /* Make this an IntuiMessage when you want to use
                              * it */


    if (IntuitionBase = OpenLibrary("intuition.library", 37))
    {

        /* Lock workbench screen so we can watch it closely */
        if (wbscreen = LockPubScreen("Workbench"))
        {

            /*
             * Generate a nice position for the zoom'ed window. Note that this
             * specifies the INITIAL position of the window. Since we don't
             * have a size gadget, the user can't change the height and width,
             * but the offset is changed as the window is dragged.
             */
            ibox.Left = wbscreen->Width - 180;   /* far right corner */
            ibox.Top = wbscreen->BarHeight + 1;  /* Just below screen bar */
            ibox.Width = 180;
            /* ght of the window topborder */
            ibox.Height = wbscreen->WBorTop + wbscreen->Font->ta_YSize + 1;

            /* open a window with tags */
            if (window = OpenWindowTags(NULL,    /* no NewWindow structure,
                                                  * tags only */
                                        WA_Left, 0,      /* Open at far left
                                                          * corner */
                                        WA_Top, wbscreen->BarHeight + 1,
                                        WA_Width, 200,
                                        WA_Height, 100,
                                        WA_Title, (LONG) "A simple window",
                WA_Flags, WFLG_DRAGBAR | WFLG_DEPTHGADGET | WFLG_CLOSEGADGET |
                     WFLG_ACTIVE | WFLG_SIMPLE_REFRESH | WFLG_NOCAREREFRESH,

                                        /* Only interested in closewindow      */
                                        WA_IDCMP, IDCMP_CLOSEWINDOW,

                                        /* pass the alternate zoom dimension   */
                                        WA_Zoom, (LONG) & ibox,

                                        TAG_DONE))
            {
```

```
                /* And just wait for windowclose */
                WaitPort(window->UserPort);
                /* clear the message port */
                while (msg = GetMsg(window->UserPort))
                    ReplyMsg(msg);

                CloseWindow(window);
            }
            UnlockPubScreen(NULL, wbscreen);
        }
        CloseLibrary(IntuitionBase);
    }
}
```

```
/*
 * Hide.h
*/

UWORD chip      ImageI1Data[] =
{
/* Plane 0 */
    0x0000, 0x0000, 0x0000, 0x0000, 0x8208, 0x0000, 0x0000,
    0x8208, 0x7FFF, 0xFFFF, 0xFEF8, 0x4000, 0x0000, 0x00A8, 0x4000,
    0x0000, 0x0048, 0x4000, 0x0000, 0x00A8, 0x4000, 0x0000, 0x0048,
    0x4000, 0x0000, 0x00A8, 0x4000, 0x0000, 0x0008, 0x4000, 0x0000,
    0x0008, 0x4000, 0x0000, 0x00E8, 0x4000, 0x0000, 0x0008, 0xFFFF,
    0xFFFF, 0xFFF8,
/* Plane 1 */
    0xFFFF, 0xFFFF, 0xFFF8, 0x8000, 0x0000, 0x4100, 0x8000, 0x0000,
    0x4100, 0x8000, 0x0000, 0x0100, 0xC000, 0x0000, 0x0110, 0xC000,
    0x0000, 0x0110, 0xC000, 0x0000, 0x0110, 0xC000, 0x0000, 0x0110,
    0xC000, 0x0000, 0x0110, 0xC000, 0x0000, 0x0110, 0xC000, 0x0000,
    0x0110, 0xC000, 0x0000, 0x0110, 0xFFFF, 0xFFFF, 0xFFF0, 0x0000,
    0x0000, 0x0000,
};

struct Image    ImageI1 =
{
    0, 0,                   /* Upper left corner */
    45, 14, 2,              /* Width, Height, Depth */
    ImageI1Data,            /* Image data */
    0x0003, 0x0000,         /* PlanePick, PlaneOnOff */
    NULL                    /* Next image */
};

UWORD chip      ImageI2Data[] =
{
/* Plane 0 */
    0x0000, 0x0000, 0x0000, 0x7FFF, 0xFFFF, 0xBEF8, 0x7FFF, 0xFFFF,
    0xBEF8, 0x7FFF, 0xFFFF, 0xFEF8, 0x4000, 0x0000, 0x00E8, 0x4000,
    0x0000, 0x00E8, 0x4000, 0x0000, 0x00E8, 0x4000, 0x0000, 0x00E8,
    0x4000, 0x0000, 0x00E8, 0x4000, 0x0000, 0x00A8, 0x4000, 0x0000,
    0x00A8, 0x4000, 0x0000, 0x00E8, 0x4000, 0x0000, 0x0008, 0xFFFF,
    0xFFFF, 0xFFF8,
/* Plane 1 */
    0xFFFF, 0xFFFF, 0xFFF8, 0xFFFF, 0xFFFF, 0x7DF0, 0xFFFF, 0xFFFF,
    0x7DF0, 0x8000, 0x0000, 0x0100, 0xC000, 0x0000, 0x0150, 0xC000,
    0x0000, 0x01B0, 0xC000, 0x0000, 0x0150, 0xC000, 0x0000, 0x01B0,
    0xC000, 0x0000, 0x0150, 0xC000, 0x0000, 0x01F0, 0xC000, 0x0000,
    0x01F0, 0xC000, 0x0000, 0x0110, 0xFFFF, 0xFFFF, 0xFFF0, 0x0000,
    0x0000, 0x0000,
};

struct Image    ImageI2 =
{
    0, 0,                   /* Upper left corner */
    45, 14, 2,              /* Width, Height, Depth */
    ImageI2Data,            /* Image data */
    0x0003, 0x0000,         /* PlanePick, PlaneOnOff */
    NULL                    /* Next image */
};

struct DiskObject AppIconDObj =
{
    0,
    0,
    {                       /* Embedded Gadget Structure */
        NULL,               /* Next Gadget Pointer */
        0, 0, 45, 15,       /* Left,Top,Width,Height */
        GFLG_GADGHIMAGE,
        0,                  /* Activation Flags */
        0,                  /* Gadget Type */
        (APTR) & ImageI1,   /* Render Image */
        (APTR) & ImageI2,   /* Select Image */
        NULL,               /* Gadget Text */
        NULL,               /* Mutual Exclude */
        NULL,               /* Special Info */
        0,                  /* Gadget ID */
        NULL,               /* User Data */
    },
    0,                      /* Icon Type */
```

```
    NULL,                   /* Default Tool */
    NULL,                   /* Tool Type Array */
    NO_ICON_POSITION,       /* Current X */
    NO_ICON_POSITION,       /* Current Y */
    NULL,                   /* Drawer Structure */
    NULL,                   /* Tool Window */
    0                       /* Stack Size */
};




;/* Hide.c - Execute me to compile me with Lattice 5.10a
LC -b1 -cfistq -v -y -j73 Hide.c
Blink FROM LIB:c.o,Hide.o TO Hide LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <graphics/text.h>
#include <libraries/gadtools.h>
#include <workbench/startup.h>
#include <workbench/workbench.h>
#include "hide.h"

#ifdef LATTICE
#include <clib/alib_protos.h>
#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/gadtools_protos.h>
#include <clib/graphics_protos.h>
#include <clib/wb_protos.h>

/* disable SAS/C CTRL-C handing */
int             CXBRK(void)
{
    return (0);
}
int             chkabort(void)
{
    return (0);
}

#endif

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct WorkbenchBase *WorkbenchBase;
struct Library *GadToolsBase;

LONG            main(void);

LONG
main(void)
{
    struct Window  *window;
    struct IntuiMessage *imsg;
    struct Gadget  *gadgetcontext;
    struct Gadget  *gadget, *hidegadget;
    struct MsgPort *appport;
    struct NewGadget ng;
    struct TextExtent textextent;
    struct AppIcon *appicon = NULL;
    struct AppMessage *appmsg;
    UWORD           left, top;
    void           *visualinfo;
    ULONG           signal, windowsignal, waitmask;
    BOOL            ABORT = FALSE;
    BOOL            CONTINUE, ICONIFY;

    if (IntuitionBase = OpenLibrary("intuition.library", 37))
    {
        /* Open GfxBase to use TextExtent() so we can handle proportional fonts */
        if (GfxBase = OpenLibrary("graphics.library", 37))
        {
            /* Open gadtools for that lonely gadget */
            if (GadToolsBase = OpenLibrary("gadtools.library", 37))
            {
                if (WorkbenchBase = OpenLibrary("workbench.library", 37))
                {
                    /* Message to receive appmessage on */
                    if (appport = CreateMsgPort())
                    {
                        /* open a window with tags */
                        left = top = 50;
                        do
                        {
                            /* no NewWindow structure, tags only */
                            if (window = OpenWindowTags(NULL,
                                        /* Open at far left corner */
                                        WA_Left, left,
                                        WA_Top, top,
                                        WA_Width, 150,
                                        WA_Height, 80,
                                        WA_Title, (LONG) "hide",
                                        WA_Flags, WFLG_DRAGBAR |
                                            WFLG_DEPTHGADGET | WFLG_CLOSEGADGET |
                                            WFLG_ACTIVATE | WFLG_SMART_REFRESH |
                                            WFLG_NOCAREREFRESH,
                                        WA_IDCMP,
                                            IDCMP_CLOSEWINDOW | IDCMP_GADGETUP,
                                        TAG_DONE))
                            {
                                windowsignal = 1L << window->UserPort->mp_SigBit;
                                /*
                                 * Get the visual info gadtools needs for the
                                 * screen we opened on
                                 */
                                if (visualinfo = GetVisualInfoA(window->WScreen, NULL))
                                {

                                    /*
                                     * Create a simple gadtools button and sort
                                     * of lay it out
                                     */
                                    if (gadget = CreateContext(&gadgetcontext))
                                    {

                                        /*
                                         * Use TextExtent to handle
                                         * proportional fonts
                                         */
                                        TextExtent(&(window->WScreen->RastPort),
                                                   "Hide", 4, &textextent);
                                        ng.ng_Width = textextent.te_Width + 8;
                                        ng.ng_LeftEdge = (window->Width / 2)
                                                                - (ng.ng_Width / 2);
                                        ng.ng_Height = textextent.te_Height + 4;
                                        ng.ng_TopEdge = (
                                            (window->Height - window->BorderTop
                                                - window->BorderBottom) / 2)
                                            + (ng.ng_Height / 2);
                                        ng.ng_TextAttr = window->WScreen->Font;
                                        ng.ng_GadgetText = "Hide";
                                        ng.ng_VisualInfo = visualinfo;
                                        ng.ng_GadgetID = 1;
                                        ng.ng_Flags = PLACETEXT_IN;
                                        hidegadget = gadget =
                                            CreateGadget(BUTTON_KIND, gadget,
                                                         &ng, TAG_END);
                                        AddGList(window, gadget, -1, -1, NULL);
                                        RefreshGList(gadget, window, NULL, -1);
                                        GT_RefreshWindow(window, NULL);

                                        CONTINUE = TRUE;
                                        waitmask = windowsignal|
                                                      1L << appport->mp_SigBit;
                                        do
                                        {
                                            signal = Wait(waitmask);
```

```
                        if (signal & windowsignal)
                          {
                              while (imsg = (struct IntuiMessage *)
                                  GetMsg(window->UserPort))
                              {
                                  if (imsg->Class ==
                                          IDCMP_CLOSEWINDOW)
                                  {
                                      ABORT = TRUE;
                                      CONTINUE = FALSE;
                                      ICONIFY = FALSE;
                                  }
                                  else
                                  if (imsg->Class == IDCMP_GADGETUP)
                                      ICONIFY = TRUE;
                                  ReplyMsg((struct Message *) imsg);
                              }
                          }
                          if (signal & (1L << appport->mp_SigBit))
                          {
                              while (appmsg = (struct AppMessage *)
                                  GetMsg(appport))
                              {

                                  /*
                                   * If am->NumArgs is zero
                                   * the user double-clicked
                                   * on our icon, otherwise
                                   * one or more icons were
                                   * dropped on top of it.
                                   */
                                  if (appmsg->am_NumArgs == 0)
                                  {
                                      RemoveAppIcon(appicon);
                                      CONTINUE = FALSE;
                                  }
                                  ReplyMsg(
                                      (struct Message *) appmsg);
                              }
                          }
                          if (ICONIFY)
                          {

                              /*
                               * Add appicon, close window if
                               * succesful
                               */
                              appicon = AddAppIcon(1, NULL, "Hide",
                                  appport, NULL, &AppIconDObj, NULL);
                              if (appicon == NULL)
                              {
                                  DisplayBeep(window->WScreen);
                              }
                              else
                              {
                                  RemoveGadget(window, hidegadget);
                                  left = window->LeftEdge;
                                  top = window->TopEdge;
                                  CloseWindow(window);
                                  window = NULL;
                                  /* there is no window
                                   * message port anymore */
                                  waitmask =
                                      1L << appport->mp_SigBit;
                              }
                              ICONIFY = FALSE;
                          }
                      } while (CONTINUE == TRUE);
                      if (window)
                          RemoveGadget(window, hidegadget);
                      FreeGadgets(gadgetcontext);
                  }
                  FreeVisualInfo(visualinfo);
              }
```

```
                              if (window)
                              {
                                  left = window->LeftEdge;
                                  top = window->TopEdge;
                                  CloseWindow(window);
                              }
                          }
                      } while (ABORT == FALSE);
                      DeleteMsgPort(appport);
                  }
                  CloseLibrary(WorkbenchBase);
              }
              CloseLibrary(GadToolsBase);
          }
          CloseLibrary(GfxBase);
      }
      CloseLibrary(IntuitionBase);
  }
  return (0);
}
```

❖