Naming Graphics Display Modes

by Martin Taillefer

The graphics.library database contains a list of all available display modes on any given Amiga. Each mode has a given set of attributes, including possibly a name. Only a subset of the available modes in the system have names. The other modes are less significant and don't have names directly associated with them.

When showing a list of available modes to the user, unnamed modes become a problem. How should they be presented to the user? A simple solution is to dynamically construct names for unnamed modes. The name construction can be based on a mode's attributes, resulting in a descriptive name for the mode.

The NameMode() routine in the program below accepts a graphics.library mode id, and a string buffer, and fills-in the string buffer with the name of the given mode. If the mode has a real name entry in the graphics database, that name is returned. If there is no real name entry, a name is constructed for the mode based on the mode's properties. A routine very similar to this is present in the ASL screen mode requester, and in the ScreenMode prefs editor.

The following program outputs the names of all current graphics database modes to the console.

```
#include <exec/types.h>
#include <graphics/displayinfo.h>
#include <stdio.h>
#include <string.h>

#include <clib/graphics_protos.h>
#include <clib/utility_protos.h>

/*****************************************************************************/

#define MONITOR_PART(id) ((id) & MONITOR_ID_MASK)

/*****************************************************************************/

BOOL NameMode (ULONG modeID, STRPTR result)
{
  struct NameInfo nameInfo;
  struct DisplayInfo dispInfo;
  struct DimensionInfo dimInfo;
  struct MonitorInfo monInfo;
  char buffer[DISPLAYNAMELEN + 1];
  UWORD len;
  DisplayInfoHandle dh;

  result[0] = 0;

  dh = FindDisplayInfo (modeID);
  if (GetDisplayInfoData (dh, (APTR) & dispInfo, sizeof (struct DisplayInfo),
                          DTAG_DISP, INVALID_ID), INVALID_ID)
  {
    if (!dispInfo.NotAvailable)
    {
      if (GetDisplayInfoData (dh, (APTR) & dimInfo, sizeof (struct DimensionInfo),
                              DTAG_DIMS, INVALID_ID))
      {
     /* Get name or make one if no name available */
     if (GetDisplayInfoData (dh, (APTR) & nameInfo, sizeof (struct NameInfo),
                             DTAG_NAME, INVALID_ID))
     {
       strcpy (result, nameInfo.Name);
       return (TRUE);
     }
     else
     {
       if (GetDisplayInfoData (dh, (APTR) & monInfo, sizeof (struct MonitorInfo),
                               DTAG_MNTR, INVALID_ID))
       {
         if ((monInfo.Mspc) && (monInfo.Mspc->ms_Node.xln_Name))
         {
           strcpy (buffer, monInfo.Mspc->ms_Node.xln_Name);
           len = strlen (buffer);
           if ((len > 8) && (Strnicmp (&buffer[len - 8], ".monitor", len - 8)
                                                                     == 0))
           {
          buffer[len - 8] = 0;
          len -= 8;
           }

           while (len > 0)
          buffer[--len] = ToUpper (buffer[len]);
         }
       }

       sprintf (result, "%s:%lu x %lu %s%s%s",
             buffer,
             (dimInfo.Nominal.MaxX - dimInfo.Nominal.MinX + 1),
             (dimInfo.Nominal.MaxY - dimInfo.Nominal.MinY + 1),
             (dispInfo.PropertyFlags & DIPF_IS_HAM) ? "HAM " :
             (dispInfo.PropertyFlags & DIPF_IS_EXTRAHALFBRITE) ? "EHB " : "",
             (dispInfo.PropertyFlags & DIPF_IS_PF2PRI) ? "DPF2 " :
             (dispInfo.PropertyFlags & DIPF_IS_DUALPF) ? "DPF " : "",
             (dispInfo.PropertyFlags & DIPF_IS_LACE) ? "Laced " : "", "");

       return (TRUE);
     }
      }
    }
  }

  return (FALSE);
}

/******************************************************************************/

void main (void)
{
  ULONG modeID;
  char name[64];

  modeID = INVALID_ID;
  while ((modeID = NextDisplayInfo (modeID)) != INVALID_ID)
  {
    if (MONITOR_PART (modeID))/* ignore "default" monitor */
    {
      if (NameMode (modeID, name))
      {
     printf ("%s\n", name);
      }
    }
  }
}
```