The user interface is even readable on TV sets (this has been done with CDTV and composite/RF output in mind).

Features of the Page Lay-out System

- Opening and saving of structured documents. The page lay-out is described using 1001's proprietary PageTalk language.
- Sizeable windows (Mac-style with 8 handles).
- Cut, Copy and Paste facility for windows and their contents.
- Import of IFF pictures and screens with the option to resize to fit in window. The screen import function can be used to 'grab' the screen of a running application (e.g a paint program or a video digitizer).
- Thumbnail preview of pictures in the file requester.
- Resolution and color independent color palette requester at bottom of screen (i.e. 20 virtual colors in hires, 4 for the user interface and 16 for the wells).
- Unlimited switching between resolutions and depths. The page is resized accordingly.
- Interlace independent user interface. The floating palettes, requesters and dialog boxes are sized as the program runs.
- The text editor allows one typeface per window. The height and color can be changed per line, the style per character.
- Buttons can be created on which the user can click and they can be used to create 'local events'. This enables the user to create interactive presentations.
- 10 favourite applications can be stored in a menu.
- 7 running applications can be brought upfront by selecting their name from a menu.
- Support for PAL, NTSC, standard and severe overscan screens.
- Output to WorkBench™ printers and PostScript™ devices.
- Supported screen modes are lores, hires, extra halfbrite, HAM, interlace, productivity and super-hires.
- Support for DCTV, ColorMaster 12/24, IV24 etc.

Both the 10 favourite applications and the 7 running programs can be accessed in the Page Lay-out System and the Script Editor. Both screens share these two menus as well as the Project and File menu.

Features of the Script Editor

- Opening and saving of scripts. The scripts are described using 1001's proprietary ScriptTalk language.
- The Script Editor has a script window and a tools window. The tools window contains objects to show pictures, to play music and samples, to control external devices and to execute DOS and ARexx scripts.
- Standard objects are: picture/page shower, ANIM player (for animations stored in RAM but also for larger-than-memory animations with real time loading from harddisk), SMUS player, 8SVX player (for samples stored in RAM but also for larger-than-memory samples with real time loading from harddisk), SoundTracker/NoiseTracker/MOD player, MIDI player, DOS command/script executer, ARexx script executer and a driver for the Canon™ Still Video player.
- Cut, Copy and Paste facility for objects or a collection of objects.

- Thumbnail preview of pictures in the file requester.
- Interlace independent user interface. The floating palettes, requesters and dialog boxes are sized as the program runs.
- The script can contain 'serial' and 'parallel' events. This allows the user to nest scripts within scripts and to start several events at the same time or at very accurately timed moments.
- Labels can be attached to objects. Global events and local events can trigger a jump to such labels. Global events are caused by pressing a key on the keyboard or by programs or ARexx scripts which send 'fake' key events into the system's input handler. Local events are events created by clicking with the mouse pointer over a button. Events can be used to create interactive presentations.
- Third party developers can add objects (so called XaPPs or plug-ins) to MediaLink with little programming effort. This is accomplished by using the MediaLink library (stored in the libs: directory) which facilitates the creation of a MediaLink user interface. The library and its associated linkable modules can be used to create objects which can be dragged into MediaLink scripts. XaPPs are two part programs. The first part comes in action when launched from within MediaLink. It presents the user with a window in which e.g. a LaserDisc can be controlled and programmed. All the functions of the LaserDisc can be accessed via this window and the actions are recorded to a disk file. When the same object is executed by the Script Driver as part of a presentation then the second part of the program comes into action and uses the data file to play back the actions of the user. To conserve memory, the MediaLink library needn't to be opened in the latter case. The Canon™ Still Video player object is an example of a XaPP created by 1001.
- Transferring of scripts, XaPPs and data files is done using the Z modem protocol or the AmigaTalk local area network software. Sending in the background while other tasks are running and scheduled transmissions are possible. Scripts can be uploaded as well as download from and to several Script Drivers.

ScriptTalk and PageTalk

The script and the pages are stored in a readable format. This means that experienced users can write their own scripts and pages just like writing a computer program.

The ScriptTalk language has 21 commands, they are:

```
ANIM, AREXX, BINARY, DOS, DURATION, EFFECT, END, ENDSER, ENDPAR, GLOBALEVENT,
ILBM, LABEL, MAIL, PAGE, PROGRAM, SCRIPTTALK, SOUND, START, STARTSER, STARTPAR,
USERAPPLIC.
```

The PageTalk language, which is a subset of the ScriptTalk language has 8 commands, they are:

```
CLIP, FONT, LOCALEVENT, PAGETALK, PALETTE, SCREEN, TEXT, WINDOW.
```

An 'empty' ScriptTalk script looks like this:

```
SCRIPTTALK 1,0
STARTSER
   STARTSER
   ENDSER
ENDSER
```

The first line tells the parser of this script which version of the language is used. All the events in the script must be enclosed in a STARTSER/ENDSER block, and an empty script always contains one empty event. Therefore an empty script contains two serial event blocks.

A script which shows an IFF picture might look like this:

```
SCRIPTTALK 1,0
STARTSER
   STARTSER
     START
        ILBM 'work:pictures/test', NOCYCLE
     END
   ENDSER
ENDSER
```

The ILBM event can be extended with a few parameters like:

```
START
   ILBM 'work:pictures/test', NOCYCLE
   DURATION 5
   EFFECT 12
   PROGRAM SU|MO|TU|WE|TH|FR|SA
END
```

The PageTalk language is used to describe the page lay-out. A typical page might look like this:

```
PAGETALK 1,0
OBJECTSTART
   SCREEN 320,512,4096,4
   PALETTE 0x000, 0xfff, 0x523, 0xfe5, 0x4d2, 0x4a2, 0x9ff, 0xe53,
           0xdd3, 0xcc5, 0xaad, 0xd0d, 0x235, 0x778, 0x112, 0x999
   LOCALEVENT -1, "MainMenu", 20, 30, 80, 100, COMPLEMENT, OFF
   WINDOW 0, 0, 320, 512, 1, 1, 2, 2, 5, 0, TRANSP
   CLIP "work:pictures/test", 10, 10, 100, 100, OFF
   FONT "work:fonts/Grace_Hi/80"
   TEXT "work:text/test.asc"
OBJECTEND
```

Of course, the average user won't ever see ScriptTalk or PageTalk.

The menus

Amiga-style menus are used to access various functions. The favourite application menu, the Project menu, the Edit menu and the Screens menu (the rightmost menu) are shared by the Page Lay-out System and the Script Editor. The figures 1 and 2 show these menus:

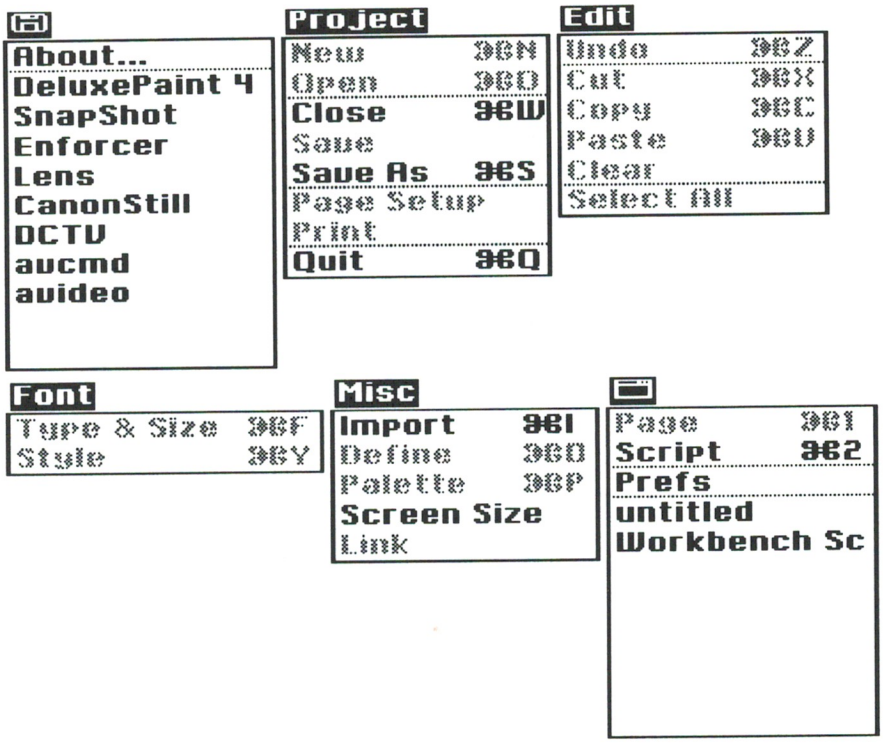

Fig. 1

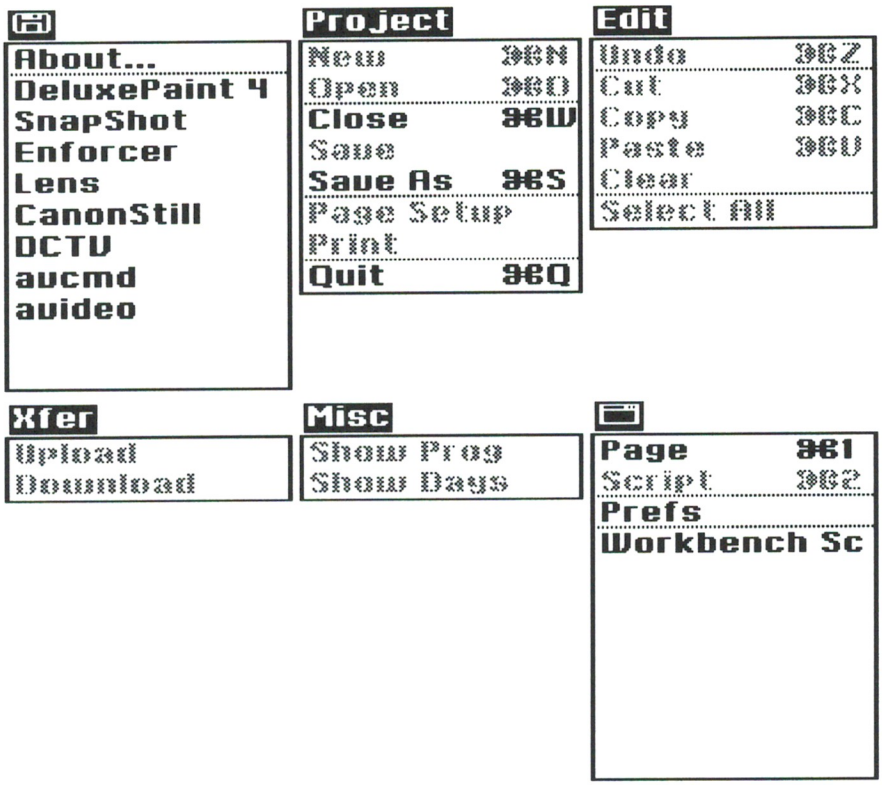

Fig. 2

As you can see the user can enter which frame or sequence of frames should be shown. A preview option is also available (this is very important: the user should not have to run the script in order to preview his actions, direct previewing inside the XaPP must be available). The programmed actions must be saved to a data file else the Script Driver cannot play back the actions. Therefore, a name must be entered which will be used to save the data file. If the user wants to use a previously stored data file then this file may be selected using the file requester. When a script is send to a remote location e.g. by modem all the objects of the script, including XaPPs and their related data files are transferred.

The MediaLink library

The MediaLink library contains functions for opening windows and screens, to draw user interface elements and various routines which support the communication between the running MediaLink application and the XaPP. In all there are 34 library functions and some extra 15 functions in linkable modules. They are:

```
UA_CheckGadget();
UA_CheckGadgetList();
UA_ClearButton();
UA_CloseScreen();
UA_CloseWindow();
UA_DisableButton();
UA_DoubleGadgetDimensions();
UA_DrawGadget();
UA_DrawGadgetList();
UA_DrawSpecialGadgetText();
UA_DrawStringText();
UA_DrawText();
UA_EnableButton();
UA_GetDepthOfWindowScreen();
UA_GetKeys();
UA_GetRadioGadgetVal();
UA_HalveGadgetDimensions();
UA_HiliteButton();
UA_HiliteRadioButton();
UA_HostScreenPresent();
UA_InitStruct();
UA_InvertButton();
UA_InvertRadioButton();
UA_IsUAScreenLaced();
UA_IsWindowOnLacedScreen();
UA_OpenScreen();
UA_OpenWindow();
UA_ProcessCycleGadget();
UA_ProcessRadioGadget();
UA_ProcessStringGadget();
UA_SetCycleGadgetToVal();
UA_SetStringGadgetToVal();
UA_SetValToCycleGadgetVal();
UA_SetValToStringGadgetVal();
```

Beside the library, the XaPP development kit also includes linkable modules which include routines for a file requester, routines to parse data files (the same routines which interpret the PageTalk and ScriptTalk language) and controlling of serial and parallel devices.

How the functions are used and how little programming effort it takes to create an window and gadgets is shown in the following example. How the final application looks can be seen in figure 5.

The include file with the gadget and image definitions:

```
/*****************************************/
/**** test.h                            ****/
/*****************************************/

/* Canon Still Video (CSV) gadgets */

struct StringRecord CSV_single_SR= { 2, "00" };
struct StringRecord CSV_from_SR  = { 2, "00" };
struct StringRecord CSV_to_SR    = { 2, "00" };
struct StringRecord CSV_duration_SR  = { 2, "00" };
struct StringRecord CSV_datafile_SR  = { 20, "                    " };

struct GadgetRecord CSV_GR[] =
{
  0,   0, 411, 188, NULL,              DIMENSIONS, NULL,
  0,   0, 410, 187, NULL,              DBL_BORDER_REGION, NULL,
 21,  46,  37,  54, "Show frame",      RADIO_GADGET, NULL,
 21,  60,  37,  68, "Show from frame", RADIO_GADGET, NULL,
147,  44, 177,  57, NULL,              INTEGER_GADGET,
   (struct GadgetRecord *)&CSV_single_SR,
192,  58, 222,  71, NULL,              INTEGER_GADGET,
   (struct GadgetRecord *)&CSV_from_SR,
245,  58, 275,  71, "to",              INTEGER_GADGET,
   (struct GadgetRecord *)&CSV_to_SR,
186,  82, 216,  95, "Duration per frame", INTEGER_GADGET,
   (struct GadgetRecord *)&CSV_duration_SR,
 21, 110,  37, 118, "Blank screen after showing", CHECK_GADGET, NULL,
302, 110, 318, 118, "Counter",         CHECK_GADGET, NULL,
 95, 133, 292, 146, NULL,              LOBOX_REGION, NULL,
299, 133, 388, 146, "Directory",       BUTTON_GADGET, NULL,
  9, 171,  90, 183, "OK",              BUTTON_GADGET, NULL,
100, 171, 181, 183, "Preview",         BUTTON_GADGET, NULL,
320, 171, 401, 183, "Cancel",          BUTTON_GADGET, NULL,
 20, 134,  90, 144, "Data File",       STRING_GADGET,
   (struct GadgetRecord *)&CSV_datafile_SR,
142,   5, 401,  17, "Canon Still Video Player RV-321", TEXT_REGION,
   NULL,
142,  21, 401,  33, "MediaLink™ XaPP™", TEXT_REGION, NULL,
 21,  76, 388,  76, NULL,              LO_LINE, NULL,
 21, 101, 388, 101, NULL,              LO_LINE, NULL,
 21, 127, 388, 127, NULL,              LO_LINE, NULL,
-1
};
```

```
/* Image data for Canon logo */

USHORT chip ImageDataCanon[] = {
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
 0x8000,0xFFFF,0xDFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,

  etc. etc. etc.

 0x3FCF,0xF8FF,0x00C0,0xC00F,0xF1FF,0x0000,0x0000,0x0000,
 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
};

struct Image ImageCanon = {
  0,0, 129,29, ImageDataCanon, 0x0007,0x0000, NULL
};
```

The example application (fragments from the Canon still video player XaPP):

```
/*****************************************/
/**** test.c                          ****/
/*****************************************/

/* Functions which start with 'UA_' are in           */
/* the medialink.library. Other functions reside in  */
/* linkable modules or are OS functions.             */

void main(int argc, char **argv)
{
struct UserApplicInfo UAI;

  /**** initialize the User Application Info structure ***/

  UA_InitStruct(&UAI);

  /**** open and load the medialink font ***/

  OpenUserApplicationFont(&UAI, "fonts:medialink.font", 12);

  /**** open a window ****/

  if (UA_HostScreenPresent(&UAI))
    UAI.windowModes = 1;  /* open on the MediaLink screen */
  else
    UAI.windowModes = 3;  /* open on the first (frontmost) screen */
                          /* (e.g. the WorkBench screen)          */

  /* double the dimensions of gadgets etc. if screen is laced */
  /* the CSV_GR structure can be found in test.h              */

  if (UA_IsUAScreenLaced(&UAI))
    UA_DoubleGadgetDimensions(CSV_GR);

  UAI.windowX      = -1;  /* -1 means center on screen */
  UAI.windowY      = -1;  /* -1 means center on screen */
  UAI.windowWidth  = CSV_GR[0].x2;
  UAI.windowHeight = CSV_GR[0].y2;
  UA_OpenWindow(&UAI);
```

```
/**** render logo ****/

if (UAI.windowModes == 1)
 DrawImage(UAI.userWindow->RPort,(struct Image *)&ImageCanon,10,5);

/**** render all gadgets ****/

UA_DrawGadgetList(UAI.userWindow, CSV_GR);

/**** handle events (not covered) here ****/

etc. etc. etc.

/**** close the window ****/

UA_CloseWindow(&UAI);

/**** close the medialink font ****/

CloseUserApplicationFont(&UAI);
```

The event handling is also supported by the library:

```
signalMask = (1L << win->UserPort->mp_SigBit);
loop=TRUE;
while(loop)
{
  signals = Wait(signalMask);
  if (signals & signalMask)
  {
    while(message=(struct IntuiMessage *)GetMsg(win->UserPort)))
    {
      ED.Class     = message->Class;
      ED.Code      = message->Code;
      ED.Qualifier = message->Qualifier;
      ED.MouseX    = message->MouseX;
      ED.MouseY    = message->MouseY;
      ReplyMsg((struct Message *)message);

      if (ED.Class==MOUSEBUTTONS && ED.Code==SELECTDOWN)
      {
        ID = UA_CheckGadgetList(win, CSV_GR, &ED);
        switch(ID)
        {
          case 2:     /* ID contains gadget number */

etc. etc. etc.

        }
      }
    }
  }
}
```

There are also functions to highlight or invert gadgets, to process integer and string gadgets, cycle gadgets, check gadgets, radio buttons.

Inverting a gadget is done as follows:

```
UA_InvertButton(win, &CSV_GR[2]);
```

Processing a string gadget is done as follows:

```
UA_ProcessStringGadget(win,CSV_GR,&CSV_GR[4],&ED);
```

The entered string can be found in the string record structure.

ScriptTalk and PageTalk functions

Source code guidelines:

- Tabs, spaces and linefeeds may be used to structure the script. Commas may be used to separate parameters or be left out.
- Strings must be enclosed by single quotes. Double quotes may only be included in strings e.g. the DOS command might look like DOS 'say "hello, world" ', DEFER.

How this document should be used:

- Commands enclosed between braces {} may be repeated several times or be omitted completely.
- The REGISTER part of most functions describes the entries found in the ScriptNodeRecord structure if not indicated otherwise. After a script is parsed, a list of nodes (which may point to other lists) is created consisting of 'n' ScriptNodeRecord nodes. A ScriptInfoRecord is also created and when GLOBALEVENTS are present an array of ScriptEventRecords is created as well.
- Only objects with the **(in list)** indication show up in a list.
- If the REGISTER part is not listed, the object does not show up in a parsed list. These objects are only used to structure the script.