VisualArts

Danny Y. Wong

COLLABORATORS			
	TITLE : VisualArts		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Danny Y. Wong	December 7, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 VisualArts **1**

1.1 VISUAL ARTS DOCUMENTATIONS . . . 1
1.2 Visual Arts Feature List . . . 1
1.3 Layout Editor . . . 2
1.4 Environment Setup . . . 2
1.5 Introduction . . . 2
1.6 create gadtools . . . 3
1.7 Screen Mode Setup . . . 4
1.8 Layout Editor . . . 4
1.9 Arrow . . . 5
1.10 creating text . . . 6
1.11 creating circles . . . 6
1.12 creating rectangles . . . 7
1.13 creating line . . . 7
1.14 creating a GadTool object . . . 7
1.15 deleting an object . . . 8
1.16 undelete an object . . . 8
1.17 create lists . . . 8
1.18 create menus . . . 9
1.19 creating window . . . 10
1.20 creating grid . . . 12
1.21 Object Justifications . . . 12
1.22 Layout Editor Short Cuts . . . 12
1.23 Screen Setup . . . 13
1.24 Visual Arts Preference Setting . . . 14
1.25 Object Master . . . 15
1.26 Scripting . . . 16
1.27 Generation C Source Codes . . . 19
1.28 General Attribute Requester . . . 22
1.29 Future Ehancements . . . 23

1.30 Demos . . . 23
1.31 Bugs! Bugs! and more Bugs! This really Bugs Me! . . . 23
1.32 History . . . 24
1.33 Registered Users . . . 25
1.34 User Registeration . . . 26
1.35 Disclaimer . . . 26
1.36 Copyright Notice . . . 27
1.37 Arexx Setup . . . 27
1.38 Custom Images . . . 27
1.39 Setting a Color Map . . . 28
1.40 Why Another GUI Designer? . . . 28
1.41 Visual Arts Installation . . . 29
1.42 What's New In Version 2.11 . . . 30
1.43 Context Sensitive . . . 34
1.44 Requester and ASL Functions . . . 36
1.45 Speech . . . 36
1.46 Adding a Console Window . . . 37
1.47 Console and Serial Notes . . . 37
1.48 Adding a Serial Handler . . . 38
1.49 AppWindwo Example Codes . . . 38
1.50 AppWindow Example Codes -- Function File . . . 46
1.51 Visual Arts Built-In Functions -- Quick Help . . . 47
1.52 Clip Board . . . 56

Chapter 1

VisualArts

1.1 VISUAL ARTS DOCUMENTATIONS

Introduction - what is Visual Arts?

Installation - how to install Visual Arts

Disclaimer - read before you use it

Copyright Notice - some technical stuff

Registration Info - registeration and contact info

Registered Users - info for new upgrade

History - history of the project

Bugs - know bugs

Demos - stuff on demos

Future Ehancements - future to do list

Why Another GUI - why I built Visual Arts

Whats New in 2.x - new features in this release

Environment Setup - Visual Arts envrionment setup

Layout Editor - Visual Arts layout editor

Feature List - Visual Arts feature list

Home 403-274-9348 (6:00pm to 11:59pm MST) Internet: danwong@cadvision.com

danwong@oanet.com

1.2 Visual Arts Feature List

These features are built into the Visual Arts library. See Built-In

functions topic for more details.

Speech - add speech to the project

Clip Board - standard clip board support

Console - attach a console to the project

Serial - add a serial handler to project
Requesters and ASL - standard requster and file and font ASL
Built-In Functions - listing of VA built-in functions

1.3 Layout Editor

Layout Editor - display Visual Arts editor
Screen Setup - set screen attributes
Scripts - add script to active objects
Object Master - manage objects
C Codes - generate C codes
ARexx - setup ARexx for the project
Custom Images - add custom images to the project
Color Map - modify project colors
Context Sensitive - setup context layout for project

1.4 Environment Setup

Screen Mode Setup - setup screen mode for project
Preference Setup - preference setup for project

1.5 Introduction

Visual Arts Introduction
Visual Arts is an Graphical User Interface designer based on the Gadget Tools library for Workbench 2.0 or higher. Visual Arts takes a different approach in creating professional GUIs that will improve your development time.
Visual Arts has some unique features such as creating primitive graphic drawing objects such as Circles, Lines and Rectangles with options of filled and custom patterns; a Menu Manager to manage all menus, a List Manager to manage lists, and Scripts can be attached to any active object or menu items and many other unique features such as PopUp menus, AREXX interface, custom images, custom images for buttons, palette picker, Context Sensitve layout, console and serial handlers, speech and standard requester and ASL libraries.
What sets Visual Arts apart from other GUI designers is its ease of use and object oriented approach. Any objects created such as GadTools or Texts can be manipulated such as sized, change its attributes such color,

font or positioning; everything you need to do is all visually on
screen, no need to dig through multi-level menus or requesters to perform
an operation. Unlimited windows can be opened even with the same layout.
With Visual Arts simple user interface, prototyping and screen designs can
be constructed in minutes with instant C source code and
dynamic link to other applications thru AREXX.

1.6 create gadtools

Creating Gadget Tools General Requster(IFF)
To create a gadget tool select the Object tool type from the tool
palette and draw the outline of the button. Every object requires a name.
Once the object is named select a Gadget Tool from the left or by selecting
the 'Edit' button to edit the default gadget tool.
The positioning of any gadget tools or objects can be done using
the co-ordinates in the middle or by dragging the object in the layout.
To edit any gadget tools simple double click on the object and the
attribute requester will be shown with the gadget tool type selected for
this Gadget Tool. GadTool Requester(IFF)
Majority of the Gadget Tools you can assign a keyboard equivelent. To
assign a keyboard short cut, check the UnderScore box and enter the key
short cut. The key short cut must exist in the Gadget Tool name. For
example if button gadget is created and name 'My Button' and assigned a key
short cut of 'a', the 'a' key must be in 'My Button' which in this example
is not. Note: do not use duplicate assignments such assigning the same
key value to two different objects. Uppercase 'A' and lowercase 'a' are
not the same key value. If your object is assigned an 'A' you will need
to press shift-'a' in order to select or activated the object.
Also each Gadget Tool may have its own font, size and style. If you are
going to distribute your application make sure that your users have the
same type of fonts or simply include the fonts with your distribution.
MX (Radio), Cycle and ListView Gadgets
When these gadgets are created and when setting up their attributes, a
label name is required for each GadTool except for ListView gadgets this is
optional. The label name is taken from the List Manager and in order
to assign a lable you must first create a list using the List Manager.
IMPORTANT: When creating a large application with many sub-programs, you
must name each object different and thus they must be unique otherwise
the compiler will compliant.

1.7 Screen Mode Setup

Screen Mode Setup Screen Setup (IFF)
When starting up Visual Arts you can select any of the available screen modes, changes it size or depth. The reason to include all the available types for your environment is that you may want to design in lo-re, hi-res, super-hires or a custom screen size. Visual Arts will handle any types of screen that your system supports under WB2.0+ including AGA modes.
In start up you can also load a project, Visual Arts will use this project as its default screen. You can not change the screen type in the design mode. Any projects loaded will be defaulted to the screen mode selected at start up.
For example. If you loaded a project that has depth of 2 maximum four colors, and then opened a new project that has a depth of 3, it will not use the new projects screen type.
Visual Arts uses Hires 640 x 200 screens therefore when you open a Lores screen without virtual screen, all requester will be chopped off because of the size different.
Note: If you set the screens to a virtual screen, it may crash your system on some of the modes.
Visual Arts now open as a PUBLIC screen. Any application can now open its window in Visual Arts screen. The screen name is VArts.
Hint: You can now double click on the name of the screen mode. This will select the screen mode and will open the screen to the selection.

1.8 Layout Editor

Layout Editor Editor (IFF)
Objects are created, edited or deleted in layout editor. Objects created in the layout editor may not represent the final actual representation. However, Visual Arts produces a close approximation of each object. Because the actual Intuition GadTools are not used in the layout, it does not give the true WYSIWYG but the final output is very close.
When creating an object, you are required to enter the objects name, this name is for identification for non-GadTool objects, and this name will be the actual function name for GadTools.
Layout Editor has a Tool Bar (IFF) attached to it and any objects you create, you must use the tool bar. The tool bar consists of:

Key Short Cuts - description of all key short cuts

Arrow - to manipulate an object

Text - to create a text object

Circle - to create an unfilled circle object

Filled Circle - to create an filled circle object

Rectangle - to create unfilled rectangle object

Filled Rectangle - to create unfilled rectangle object

Line - to create a line object

Object - to create a GadTool object

DEL - to delete selected objects

UND - to undelete deleted objects

LIST - to create lists

MENU - to create menus

GRID - to turn on/off grids

WINDOW - to customize your window

Justification - to justify objects

1.9 Arrow

Layout Editor : Arrow Tool Bar (IFF)

When the Arrow tool is selected, you can select any objects on the layout by moving the pointer to the object and press the left button mouse. Once an object is selected, four handles are drawn on each corner of the object. To size the object move the pointer to any handle and press and hold the left mouse button and drag the mouse in any direction to size the object. Once satified with the result, release the mouse button the object is now deselected.

To move an object, move the pointer to the object and press and hold the left mouse button then drag the object to the desire position and release.

To edit the objects attributes select the object by double clicking the object. This will bring up the object attribute requester.

To deselect an selected object move the pointer to a none object and press the left button mouse.

Multiple objects can also be selected by drawing an rectangle box around the all the objects. An object will be selected if it is in the drawn box.

You can delete, copy, cut or move the selected objects.

Moving and Copying Multiple Objects

To select multiple objects hold down the LEFT shift key and start selecting objects by clicking on them. When done release the shift key.

To move multiple objects, select all the objects, now select an object and hold down the left mouse button while the LEFT shift key is still down.
Another way is to draw an rectangle box around all the objects to be selected using the mouse.
To copy multiple objects, select the objects and select the Copy edit menu option. To select all the objects, select the ALL menu item in the Edit menu. You can move, delete and copy with the ALL option.

1.10 creating text

Layout Editor : Text Tool Bar (IFF)
The text tool lets you create text objects on the layout. To create a text object, select the 'A' object next the the Arrow tool. Once text tool is selected, move the pointer to the desired position on the layout and press the left mouse button; this will bring the up the text attribute requester. The text attribute requester require that you enter the objects name, the text to be printed on the layout, optional font name, text color, text background color and text box. The text box draws a box around the text. To size the box, select the object and size the handle.

1.11 creating circles

Layout Editor : Circles Tool Bar (IFF)
By selecting this tool it will create a filled or unfilled circle on the layout with optional pattern. To create a circle object, select the either circle tool and position the pointer on the layout and press and hold the left mouse button and drag the pointer to size the object to a desire size. Once satified release the mouse button, this will bring up the general object attribute requester.
The circle object may have the following attributes set. Object name, border color, fill color, line width and pattern.
The position box on the middle shows the actual position of the object on the layout, the object position may be changed by editing the objects co-oridinates. Selecting the OK button will accept the object and selecting the CANCEL button will abort the creation if this is a new object.

1.12 creating rectangles

Layout Editor : Rectangles Tool Bar (IFF)
By selecting this tool it will create a filled or unfilled rectangle on the layout with optional pattern. To create a rectangle object, select the either rectangle tool and position the pointer on the layout and press and hold the left mouse button and drag the pointer to size the object to a desire size. Once satified release the mouse button, this will bring up the general object attribute requester.
The rectangle object may have the following attributes set. Object name, border color, fill color, line width and pattern.
The position box on the middle shows the actual position of the object on the layout, the object position may be changed by editing the objects co-oridinates. Selecting the OK button will accept the object and selecting the CANCEL button will abort the creation if this is a new object.

1.13 creating line

Layout Editor : Line Tool Bar (IFF)
By selecting this tool it will create a line object on the layout. To create a line object, select the line tool and position the pointer on the layout and press and hold the left mouse button and drag the pointer to size the object to a desire size. Once satified release the mouse button, this will bring up the general object attribute requester.
The line object may have the following attributes set. Object name, border color and line width.
The position box on the middle shows the actual position of the object on the layout, the object position may be changed by editing the objects co-oridinates. Selecting the OK button will accept the object and selecting the CANCEL button will abort the creation if this is a new object.

1.14 creating a GadTool object

Button
Check Box
Cycle
Integer/String
ListView
MX
Numeric/Text
Palette
Slider
Scroller

1.15 deleting an object

Layout Editor : Deleting an Object Tool Bar (IFF)
Selecting this tool will delete the selected object. To delete an object
move the pointer to the object to be deleted and select it by press the
left button mouse. Once the object is selected, select the 'DEL' tool from
the tool palette. This will delete the object from the layout. You will
not be warn when deleting an object. To undelete the deleted object select
the 'UNO' tool from the tool palette. Keyboard short cut is
the 'DEL' key next to Help.

1.16 undelete an object

Layout Editor : Undelete an Object Tool Bar (IFF)
Selecting this tool will undelete the deleted object. Undelete will only
work if no other operation has been done since the object is deleted such
as selecting another object or tool. To undelete an delete object select
the 'UDO' tool.

1.17 create lists

Layout Editor : List Manager Tool Bar (IFF)
One of the unique feature in Visual Arts is the List Manager and the
Menu Manager. List (IFF)The List Manager enables you to create
unlimited number of lists and list items.
To create a list select the menu tool below the 'DEL' tool this
will bring up the list manager window. The left side of window shows all
the list name that has been created for this project, and on right side it
shows the list items that associated with the each list name.
To enter a new list name, type in the list name and press the TAB key or
by selecting the 'New' button this will erase the current input buffer.
Once a new list name has been entered you can now add new items to the list
by typing in the item on the right labeled 'List Items'.
To delete an existing list name, select the list name from the 'List Names'
listview and select the 'Del' button. To delete a list item, select the
item from the 'List Items' listview and select the 'Delete' button on the
List Items side. To add new items to an existing list name, simply select
the list name and starting entering the new list items.
The List Manager includes sorting in Adscending and Descending features.

To sort the selected List to Adscending select 'Sort A' and to sort the
List to Descending select 'Sort D'. If you preferr to custom pick the
ordering of the list, you can select the 'Pick' button. This will put you
into the Pick List requester.
Custom Ordering Pick (IFF)
In the Pick List requester, the left side contains all the list items,
and on the right is the ordering. To reorder the list items, simply
select an item from the left and select the '>>' button, this will move
the item to the right. To move the item back to the item list, simply
select the item from the right and select the '<<' button. When all the
items have been reordered, the OK button will be enabled. Select the OK
button to accept the ordering or CANCEL to revert any changes.
Note: All the lists and list routines will be generated during source
code output.

1.18 create menus

Layout Editor : Menu Manager Tool Bar (IFF)
One of the unique feature in Visual Arts is the Menu Manager and the
List Manager. Menu Manager (IFF) The Menu Manager enables you
to create unlimited number of menu bars. To create a new menu bar select
the menu tool next to the List tool this will bring up the Menu Manager.
The Menu Manager manages each menu bar separately. To create a new menu
bar, select the 'New' button, to delete a menu bar first select an existing
menu bar from the menus listview and select the 'Delete' button, to edit
an menu bar first select an existing menu bar from the menus listview and
select the 'Edit' button, and to exit Menu Manager select the 'Cancle'
button or the CloseBox'.
Creating/Editing Menu Bar Menu Bar (IFF)
When editing or creating a new menu bar Menu Manager opens another
window. The menu window is logically placed so that the Menu Title are
on the left, Menu Item in the middle and Sub Items on the right. Each
listview contains the 'New', 'Bar' (except for Menu Title), and 'Delete'
buttons and an text entry gadgets. To add a new menu select 'New' from the
'Menu Name' listview and enter the new menu name and press RETURN. This
will bring you to the Menu Item listview area. To add a new menu item,
enter the new menu item name and press RETURN, this will bring you to the
Sub Item listview. To add a new sub item enter the new sub item name and
press RETURN. To place a separator bar between Menu Items or Sub Items

select the ’Bar’ button.
Each Menu Item and/or Sub Item may contain a short cut key equivlent or
check mark. To assign a Menu Item or Sub item a short cut, select the Menu
Item or Sub Item name from the listview and select the Keyboard, Checkit,
Checked, Disabled or MenuToggle flags. Note while assigning a short cut
key, you must press enter after assign the key.
To change the default Topaz 8 font for the window, select the ’Font’
button and select a font. Note that it only displays the font name when
you select it and doesn’t display the size and flags.
To test the menu in real time, double click the right mouse and hold on
to the right mouse. This will let you preview the current menu bar.
Scripts for Menu Items
Custom codes can be inserted in for each menu item. Select the menu item
and click on the Script button, this will open the editor. Make sure the
editor path is setup in the Preference menu.

1.19 creating window

Layout Editor : Window Setup Tool Bar (IFF)
Window Type (IFF)Selecting this option will open the window setup window. Here
you specify what type of window you want, window flags, window IDCMPs and
window attributes such as window co-ordinates and window title. To select
a window type click in the window showing. The default window has a
CloseGadget and DepthGadget. Currently there are only 12 types will you
can choose from. To change the window flags select the ’Flag’ button.
This will let you select custom flags for you window.
To change the IDCMP flags, select the ’IDCMP’ flag this will let you
select custom IDCMP flags. Note that you must select the IDCMP_VANILLAKEY
check box if you want short key cut codes generated for you.
To change the window attributes select the ’Attr’ button. WindAttr (IFF)
The left, top, width and height values are taken from the physical size of
the layout in design mode. To change the size of the window you can change
it here or by sizing and moving the layout in design mode. Also an
important point to note is that the right and bottom scroll widths and
heights is not included in calculating the size of the window. To get the
actual size of the window, size the window right and bottom to the last
printable line on the layout.
Note Visual Arts presets the window flags and window IDCMP according to
the window type you’ve selected and the type of objects your have created in

the layout.

AppWindow Option WindAttr (IFF)

To make the window an AppWindow, select the AppWindow option. This will enable the window to receive any icons dragged into the window, and at the same time retain its original form. To edit the AppWindow handler go to the Edit menu and select 'AppWindow Handler', this will allow you to edit the script for this AppWindow handler.

Multi-Processing Windows WindAttr (IFF)

One of the powerful feature in Visual Arts is the ability to generate codes for multi-processing windows which are independent of other windows when opened. You are no longer stuck performing operations on one window, now with Visual Arts you can enter the world of multi-processing windows. To set up multi-processing windows, select the Multi-Process check box. This will enable your window to do multi-process, is that simple! When you generate the codes, the program will be set up for multi-processing for that window.

In order to make other windows multi-process besides the main program, you will need to make every window a multi-process by selecting the 'Multi-Process' option in the Attr setup. Once this is done, you will also need to give it the Main program project name. This is done when generating the actual codes. The entry name can be any sub-program name but the main project name must be the same as the entry name eg: void main(void).

Code Pref

You must select the ALL option for the Main Program which it creates all the neccessary routines for doing multi-processing. For sub-programs do not select the Screen Routines option because its created in the main. You will also need to change the Entry Name for sub programs because you can only have one main(...) function. Entry Name can be any name. You will also need to enter the Main Project name which can found in the main project in the Window setup's WindAttr (Project Name). WindAttr (IFF)

In order for the main program to link to your sub programs, you will need to edit the your main program and add the following line at the beginning of the main program, this is the prototype function name.

extern void entry_name(arg1, arg2 ... argn)

where

entry_name is the name of the sub program entry point.

arg1..argn is the arguments

for example:

extern void SubPrg(void);

extern int OpenDoc(char *filename);
now add the second line just before the main program opens its window in
the main program.
for example:
OpenDoc("mydoc.data"); //open the doc
if (!(OpenMain("main window)) //open main program window
....

1.20 creating grid

Layout Editor : Set Grid On/Off Tool Bar (IFF)
Grids can be turned on or off anytime in the design mode. This gives
the designer precession when placing objects together. You can set the
X and Y grid size in the Project menu under Preference menu item.

1.21 Object Justifications

Object Justifications Tool Bar (IFF)
Selected objects can be justified in any of the four orientations. To
justify objects, select the objects to be justified and select the justification
button on the tool palette. The first selected object is the anchor object.
All objects will be justified according to that object. Currently line
objects can not be justified, selected line objects will not do anything if
justified with grouped objects.

1.22 Layout Editor Short Cuts

In order to simplified getting around and manipulating objects in the
layout editor, there are a number of short cuts which will make your editing
much more easier.
Action Descriptions
Alt - Double click on an object -hold the Alt key down and double
click the left mouse to go directly
to the script editor.
Control - Double Click -hold the control key while double click
will open the general requester.
MOVING AN OBJECT
Up Arrow -move the selected object one pixel up.
Down Arrow -move the selected object one pixel down.

Left Arrow -move the selected object one pixel left.
Right Arrow -move the selected object one pixel right.
Alt Up Arrow -move the selected object 5 pixels up.
Alt Down Arrow -move the selected object 5 pixels down.
Alt Left Arrow -move the selected object 5 pixels left.
Alt Right Arrow -move the selected object 5 pixels right.
SIZE AN OBJECT
Alt Shift Up Arrow -size the selected object 1 pixel less.
Alt Shift Down Arrow -size the selected object 1 pixel larger.
Alt Shift Left Arrow -size the selected object 1 pixel more
in width.
Alt Shift Right Arrow -size the selected object 1 pixel less
in width.
BORDER AN OBJECT
Alt 1 -draw a STANDARD border around the rectangle
object such rectangle and GadTools.
Alt 2 -draw a BEVEL border around the rectangle
object.
Alt 3 -draw a RECESSED border around the rectangle
object.
DEL -delete all the selected objects.

1.23 Screen Setup

Screen Setup
To setup your own custom or public screen select Screen from the menu name 'Design'. In screen setup you can choose the Workbench or Custom screen, the screen type and the font for the screen. If you opened an screen with depth of 2 or greater Visual Arts does not know of this, and you will need to select the Custom Screen otherwise Visual Arts will assummed its a Workbench screen.
If you have selected Custom as the screen. You can make your screen into a Public screen so that other applications can open their window's on your screen. To make your screen a Public screen, enter the name for your screen. eg: VArts is Visual Arts public screen name. When your application is run other applications can use the following to open their window on your screen by defining a global variable:
UBYTE *PubScrName = "VArts";
and somewhere in the codes

if (!(Scr = LockPubScreen(PubScrName)))
return(1L);
or
if (!(Scr = LockPubScreen("VArts")))
return(1L);
this will lock the screen return the a pointer to the Scr variable.

1.24 Visual Arts Preference Setting

Preference (IFF)
Project Paths

The Preference menu option lets you set the path name for the scripts and the editor to be used for editing scripts etc. You must set these setttings if you're going to use the scripting feature. Path names must be a full path name eg: dh1:VisualArts/Objects. It is recommended that you use directories and not volumes.
Grids & Save History

You can also set the X and Y grid size for the layout grids. The interval saving feature is default to save every 15 minutes. If the history is selected, Visual Arts will make backup copies of the project. The way the history works is, suppose your project name is Inventory. Visual Arts will save the next file to be Inventory1, Inventory2 and so on. But the next time you edit Inventory3 for example Visual Arts will not save the next file to be Inventory4 instead Inventory41. Visual Arts always starts with one and keeps incrementing.
Public Screen

Visual Arts now open a public screen name "VArts". By checking the 'Editor on VA' checkbox, when you edit a script the editor with open on Visual Arts screen rather than flipping screens. Note that you will need to configure your editor to tell it that the screen to open on is "VArts" otherwise the editor will not open properly.
WorkBench 3.x

With Version 2.1 comes WB3.x features. This is the inclusion of new features only found in WB3.x such as new GadTool tags, new boxes and the new look. By checking this checkbox, you will be able to use WB3.x

features. Visual Arts still emulates the WB3.x featuers. Use the 'Test' function to see the real GUI. If you are running WB2.x and have WB3.x features turned on, you will not be able to see the WB3.x features in Test mode but you can still generate WB3.x codes.
Note: if you are generating WB3.x codes, 'WB3.x' checkbox must be selected otherwise WB3.x codes will not be generated correctly.
Visual Arts searchs in the ENVARC: path for VisualArts.prefs. If it finds the .prefs file then it uses it as the prefernece file. Any saving will be saved in the paths described in the preference setting.

1.25 Object Master

There are times when you want to save an object for use in other projects and that's where Object Master comes in. Any GadTools can be saved as an Visual Arts object or as an Visual Arts Script. The difference between the two is that a Script is codes and can not be loaded into the layout editor but an Object can be loaded into the layout editor.
To save the object select the object to be saved in the layout editor and select one of the menu item option from the Object menu choice. A file requester will be displayed for you to enter a name for the object to be saved. Objects should be saved in one central location since objects can be shared between different projects. Note only the first selected object will be saved, Object Master does not support multiple save objects.
To load an object or script into the current project, select 'Object Master' from the Object menu. Object Master (IFF)
To edit or add a new script select the 'Add/Edit Script' from the Object menu.
When you have a piece of code and wish to share it between different projects, Visual Arts will convert the script into a Visual Arts script object for sharing. Select 'Script -> Object' to convert the script.
When you add a script object to your project, you can view all the scripts by selecting the 'Object List' in the Design menu. This will list all the scripts you have included for this projects. Now you can call these functions in your other functions. Object List(IFF)
When saving as an object, Object Master also saves the script for that object if it exists.

1.26 Scripting

The Scripting (custom code) in Visual Arts lets you insert custom codes into any active object such as GadTools and menu items. Script (IFF) To enter a script for an object, double click on the object to open the general requester, or select 'Edit' on GadTools Type and select the 'Script' button; key short cut is Alt - double click on the object will bring you directly into the script editor. Short Cuts If the script has not been attached to the object, Visual Arts will create a stub function:

for all GadTools - Obj meaning GadTools object

```
int mybuttonObj(struct VAobject VAObject)
{
return(1);
}
```

for Menu items - MenuObj meaning Menu object

```
int QuitMenuObj(struct VAobject VAObject)
{
return(1);
}
```

VAobject structure is defined as:

```
struct VAobject
{
struct Window *va_Window; //current window
struct Gadget *va_Gadget; //gadget that the user have selected
struct IntuiMessage *va_IntuiMsg; //IntuiMessage
APTR va_UserData; //user data for functions etc..
ULONG va_Flags; //user flags READ/WRITE
}
```

Visual Arts will fill VAObject structure for GadTools objects as follows:

VAObject.va_Window = project window

VAObject.va_Gadget = current gadget selected

VAObject.va_IntuiMsg = msg - Intuition message

VAObject.va_UserData = 0 (user definable)

VAObject.va_Flags = 0 (user definable)

Visual Arts will fill VAObject structure for Menu objects as follows:

VAObject.va_Window = project window

VAObject.va_Gadget = NULL

VAObject.va_IntuiMsg = msg - Intuition message

VAObject.va_UserData = 0 (user definable)
VAObject.va_Flags = 0 (user definable)
for menu item functions DO NOT read or write the VAObject.va_Gadget
field because it contains NULL values.
example function that use VAobject structure

```
int LoadObj(struct VAobject VAObject) //button object
{
char filename[80];
int rc;
//open file requester and get a file
rc = GetFontName(FALSE, VAObject.va_Window, "Load File", "ram:",
"", (char *)&filename);
if (rc) //got a filename so print it using a TEXT GadTool
{
SetGadgetAttrs(TestGadgets[ID_filename], VAObject.va_Window,
NULL, GTTX_Text, filename, TAG_END);
LoadFile(filename);
//modify VAObject UserData to include a function for displaying
//the text file
VAObject.va_UserData=(APTR)DisplayTextFile;
DisplayTxtObj(VAObject); //call other objects
}
return(1);
}
```

When an Intuition event ocurrs such as clicking on any GadTools or
selecting a menu item, Visual Arts will fill this structure and pass it to
the script for this object. In the function, you can programmatically
handle the event. In turn, you could pass VAObject structure to another
object script and have it execute its function.
VAObject structure can be READ or WRITE by your program with caution.
All scripts must contain atleast one function, and also must return a
int value. You can start to insert the codes inside the function. A script
may contain more than one function, declare variables or whatever a program
maybe contain, think of scripts as an external program.
If your choice of editor supports opening of multiple files, you should
see the script on the top (SAS/C SE) and on the bottom it shows the actual
codes of the main program. This file is for references only do not edit,
it’s only for you to reference variables. The second part of the file
contains the header include files, this is where all the stub functions for

GadTools and menu items go. As you will see, all the variables are declared extern so that in your script you don't need to redeclare the variables.
Visual Arts will generate three files when you output to c source codes, the first as mentioned is the main program, and the second one is the include header files containing all the stub functions and the third file contains all the scripts include for the project.
There are no limitations in scripts, because scripts are functions or extern program when Visual Arts merge the scripts with the main program.
When you are done editing the script, save the script and DO NOT rename the script. You will also need to quit the reference file too since you are editing two files. The reference file will not be saved. Use the editor's option to save and quit.
IMPORTANT: Visual Arts uses the objects name as a reference to the actual script stored on the defined path, if the object name is modified, Visual Arts will rename the script to the new name but the script is left unmodified.

IDCMP Script

As of V2.05 most of the IDCMP flags are now functions and can be edited by selecting 'IDCMP Script' in the 'Edit' menu.

Main Script

The 'main() script' menu item option under menu 'Design' lets you add any global variables to your project. This is useful if you are using Visual Arts for complete development environment. Variables declared in this file can be any valid syntax such as declaring arrays, integers, etc..
This file will be appended to the main program, you can reference this file anytime by editing the script.

Script Initialization

Another feature of Visual Arts is the ability to initialize your declared variables in the Main script. This is very useful if you need to assign a value to a variable but can't be done during declaring but only at runtime. This script works in conjuction with the Main script. For example, if you have defined the following in the main script.

main script:

```
char filename[40];
```

init script:

```
strcpy(filename, "my.datafile");
```

1.27 Generation C Source Codes

Design Mode : C Source Code Code Pref

AppWindow Example Program

**Note: to generate Context Sensitive application, the 'Sensitive On' menu item must be checked in 'Context' menu.

If you are generating WB3.x codes you will have to select the 'WB3.x' checkbox in the Preference menu item under Project.

Visual Arts currently only supports for C source codes. Future version may include other languages. Generated source codes can be compiled without adding a single line of code, of course you wouldn't be able to doing anything except for playing with GadTools.

Visual Arts lets the programmer choose what part of the codes to generate. There are times when you only need to generate part of the codes such as modifying some gadgets and you don't need to generate all of the codes just because you have modified the gadgets. You can then replace the older gadget definitions with the new codes. Code Pref (IFF)

The preference settings are:

1. Screen. This will generate codes opening and closing public or custom screens.

2. Function. This will generate function codes that's internal to Visual Arts such as drawing primitive shapes, list routines etc. You probably would only generate this once and reuse the codes.

3. Window. This will generate codes for opening and closing window. It will also generate all window varaibes such as its co-ordinates.

4. Handler. This will generate codes for the IDCMP handler to events.

5. Intuition. This will generate codes for all GadTools, primitive drawings, texts, menus, fonts and lists.

6. All. This will generate a complete appllication.

7. Console - this will add a console window to the application. Select the 'Edit' button to edit attributes. Use the 'Edit' option to edit the size of the console window. The attributes of the console window can not be changed.

-If you want to add your own custom codes to the Console handler, you can select the 'Console Handler' in the 'Edit' menu.

8. Serial - this will generate codes for accessing the serial device. If you have checked both 'Console' and 'Serial' checkbox in the C preference, Visual Arts will generate codes that will intergrate the Console with the Serial. Typing commands in the Console will be sent

to the Serial device. Use the 'Edit' option to edit the serial
parameters. Note: if your serial device can not handle higher baud
rates than do not set the baud rate to the max.
-If you want to add your own custom codes to the Console handler, you
can select the 'Console Handler' in the 'Edit' menu.
*** note: the Console and Serial are global in Visual Arts, meaning that
if you selected these items, other opened projects will have the console
and serial checked in the 'C Preference' setting and codes will be
generated.
Another important note if you are using both the Console and the Serial.
If you selected to add a Console and later went back and add the Serial
and you have edited the Console script and then generated the codes for
your project, the Console and the Serial will not be in sync. To avoid
this, in the beginning select both the Console and the Serial or do not
use the edit options 'Console Handler' and 'Serial Handler'.
There is also an option to name the entry function. Its default is
'int main(int argv, char *argc[])'. By changing the entry function name,
you can create a complete sub program which the main can call. The entry
name can not contain any spaces and it must be valid in syntax.
For example:
My function --> incorrect
void My_function(void)
or
int Myfunction(short x, short y) --> correct
the function can also return values and accept arguments such as above.
Note the entry function name does not contain a semi-colon.
Visual Arts generates the following files.
Calculator.c -the main file eg: GUI interface
Calculator_func.c -function file eg: handlers, drawing
Calculator_scripts.c -any included scripts (may not contain any codes)
Calculator_images.c -images and colormaps (may not contain any codes)
Calculator_makefile -the makefile
SCOPTIONS -SAS/C linker options
The main file contains all the intuition codes such as gadget, menu, text,
and other definitions, window and screen opening and closing and the event
handler. The header file contains all the functions that supports the main
program such drawing primitive shapes, text, list creation gadget and menu
functions when the user selects a gadget or a menu item.
Things you should know

1. Each Gadget Tools contains a function. Each time the user clicks on a
gadget tool it calls the function name. Do your coding in the function.
The function name usually starts with the object name that you defined
in the layout editor. eg: mybuttonObj
The menu and gadget tools functions are in the header file. DO NOT
have objects with the same name, this is thru out your projects.
2. Menus also has the same characteristics as gadget tools in 1. Each time
a menu item or sub item is selected, the function will get executed.
3. Multiple font names with the same size but different styles are
supported in this release.
Built-in Function (these functions are in VisualArts.lib link library)

Quick Help

1. DrawBox - this function draws a box in the specified window with the
specified dimension, color and pattern type.
void DrawBox(struct Window *Wind, int Left, int Top, int Wid,
int Hi, UBYTE APen, short Pattern);
APen - any valid pen color for this screen
Pattern - line pattern number support by Visual Arts (0-5)
2. DrawFBox - this function draws a filled box in the specified window with
the specified dimension, color, pattern type, outline and
filled.
void DrawFBox(struct Window *Wind, int Left, int Top, int Wid, int Hi,
UBYTE APen, short Pattern, UBYTE Outline, short Fill);
APen - any valid pen color for this screen
Pattern - line pattern number support by Visual Arts (0-5)
Outline - any valid pen color for this screen
Fill - filled pattern number support by Visual Arts (0-47)
3. SetRPortFill - This function set the rast port to a specified fill type.
void SetRPortFill(struct Window *Wind, short Type);
Type - filled pattern number support by Visual Arts (0-47)
4. ButtonSelected - This function emulates the user as if the button or
cylce button has been selected. This only hilights the gadget but it
won't cycle to the next item if the gadget selected was a Cycle.
void ButtonSelected(struct Window *wind, struct Gadget *gad);
wind - current open window
gad - gadget to select Button or Cycle gadgets
5. GetNewList - This function allocated memory for a new list and returns
it address. Returns NULL if can not allocate.
struct List *GetNewList(void)

Example: mylist = GetNewList();

6. AddNewNode - This function adds a new name node to specified list.
Returns 1 if no memory and 0 for name added.

int AddNewNode(struct List *list, char name[255]);

Example: rc = AddNewNode(mylist, "Amiga Computers");

7. DeleteNode - This function deletes a name node from a specified list.
Returns 1 if name node not found and 0 for deleted.

int DeleteNode(struct List *list, char name[255])

Example: rc = DeleteNode(mylist, "Ibm");

8. FindNodeName - This function search a list for a specific name. Returns
NULL if not found.

struct NameNode *FindNodeName(struct List *list, char name[255]);

Example:

```
struct NameNode *mynode;
mynode = FindNodeName(mylist, "Amiga");
if (!mynode)
printf("can't find name\n");
```

9. FreeList - This function frees a all the name nodes created using
AddNewNode for a specified list name.

void FreeList(struct List *list);

Example: FreeList(mylist);

IMPORTANT NOTE:

Since Visual Arts lets you output part of the project, it is important to know that each project shares the same built-in routines as described above. When generating only parts of the project, there maybe duplicate routines generated such in the project that generates ALL option. When linking your objects you may encounter duplicate functions. To correct this problem, change or delete the duplicate codes in all your projects except your main project. eg: suppose that your main project generates ALL and your other projects only needs to generate the Window and Intuition parts. And suppose it does some primitive drawings such as a Circle. Visual Arts may generate the drawing functions that your main program already have.

1.28 General Attribute Requester

General Attribute Requester General Requester (IFF)

The Genenal Object Attribute Requester lets you customize each object such as colors, line width, line color etc. Each object's attribute is different from others. For example, object Rectangle is different from

object Circle in that the object Rectangle can be a Bevel or Recessed box.
The Attribute Requester is divided into four parts. The top part deals
with the objects color and pattern, the middle deals with the positioning
of the object on the layout, the bottom deals with only Rectangle objects,
a Rectangle object can be any of the three types. The right deals with
GadTools only.

1.29 Future Ehancements

Future Ehancements
1. Structure builder for designing your data models. (Designing)
2. AmigaGuide support.
3. Browser support (Help Files)
4. Code generator for other languages (eg. C++)
5. BOOPSI

1.30 Demos

Demos
There are some demos program included in the archive. The demos are no
mean complete, thery're just to show some of features in Visual Arts.
In the source drawer of demos drawer you will find more examples and demos
that includes source codes. Almost all of the demos are compiled without
any additional codes added.

1.31 Bugs! Bugs! and more Bugs! This really Bugs Me!

Bugs! Bugs! and more Bugs! This really Bugs Me!
An application is not complete without any bugs right? :-) There are
a few known bugs if you have Enforcer (Great Tool) installed.
1. Menu Manager - When deleting a Menu Title, Visual Arts doesn't seem to
free up or try to free other allocated memory. This only happens once
a while and it doesn't GURU or crash your system. I'm trying very
hard to find this bug but this is a very smart bug indeed.
2. The very first thing you create is a text object in the design mode,
sometimes it doesn't get stored and thus when you try to manipulate it
the object wouldn't get selected. Can't really explain, it shouldn't
happen.
3. This bug is related to the bug number one. When quitting Visual Arts,

the allocated memory of the Menu Manager some how got corrupted and
when freeing it, it gives you hits in Enforcer.
4. When Closing a window or quitting and you selected CANCEL, it sometimes
won't quit! Go figure. The work around when you can't quit is to
resize or drag a window and then it will quit. What stupid coding! :-(
comment: This should be fixed in V0.40.
5. There seems to be a problem when making lo-res screens larger than
the default 320x200 or interlaced with auto-scrolling. If you set
these settings other than the defaults it may crash!!! Looking into
the problem. Version: V0.40
6. When creating a image object, the actual size representation on the layout
doesn't seem to draw correctly. You will need to double click on the
object and select OK again to actually update itself. Version: 1.1
As of V1.1 bug report are no longer included in the documentation.
A great deal of time have spent on killing these nasty bugs and there is
no guarantee that more bugs won't resurface. If we programmers can write
error-free programs, we would be out of a job. So please keep that in
mind and support other authors who make THE Amiga as their CHOICE.

1.32 History

Version Released Notes

V 0.10 Feb 2/94 -first initial release (very buggy indeed)
V 0.20 Feb 14/94 -alot of new features added and bug fixes
V 0.30 Apr 01/94 -No April 1 Joke Here :-) lots of fixes
-Added ASL font
-Fixed AppWindow event handler
V 0.40 Apr 03/94 -Added scripts for all active objects and
menu items, and sub items.
-fixed the duplicate objects for text
objects.
-added key short cuts for manipulating
objects.
-added object bordering for GadTools
V 0.41 Apr 17/94 -reworked main preference
-minor fixes on code generation
-added Object Master
May 1/94 -reorganized the doc

V 1.0 May 9, 1994 -offical general release
V 1.01 May 15/94 -modified so that you can change between
text and numeric display GadTools
-modified so that the GadTools requster
shows up instead the General requster
V 1.02 May 20/94 -Added PopUpMenuClass as a standard
V 1.03 May 25/94 -Added AREXX
-made most the functions into a library
V 1.04 May 27/94 -Added an include file.
V 1.05 June 1/94 -pre 1.1 offical release
V 1.06 June 10/94 -added custom button images
June 15/94 -added images
-fixed the save error during startup
V 1.07 June 23/94 -added get color map from any IFF pic file.
-added color palette
-reworked and add more patterns
V 1.08 July 1/94 -removed recttype when creating a non
rectangle object.
-added automatic object ID
V 1.1 July 5/94 -Offical release
V 1.11 July 10/94 -released the CPU when not busy
V 1.12 July 12/94 -added String gadget for ListView
V 1.15 July 19/94 -fixed the Color Map and Color Palette
V 1.2x Aug 3/94 -Beta releases.
V 2.0 Aug 25/94 -Offical release
V 2.1 Jan 1/95 -Offical release
V 2.2 Apr 2/95 -Officea realease

1.33 Registered Users

If you are registered user and want an upgrade please send me mail with
$5.00US for shipping and handling and I will send you a keyfile. If you
have not received an upgrade already, I will send you the keyfile free of
charge via email only. If want it thru regular postal mail, you'll need
to include $5.00 for shipping and handling costs.
internet: danwong@cadvision.com --> still valid
danwong@oanet.com --> currently I'm here

1.34 User Registeration

Visual Arts is Shareware meaning that you have the rights to use Visual
Arts in a limited time of two weeks. After this date, you must register
Visual Arts if you continue to use it. Registeration fees are as followings:
1. $25.00 US per copy
2. $35.00 US per copy Major upgrades and beta versions
outside of North American please add additional $5.00 US per copy
1. This option entitles you to 1 free upgrade when a major version
becomes available.
2. This option entitles you to 1 free upgrade and beta versions. It
also allows the registrator use it to develop commerical software and
special features.
Additional upgrades is $5.00 US. This is mostly for shipping and handling
costs. You can include the upgrade costs or you can send for upgrades
when I announce the release.
When you become a registered user, you will receive a keyfile that
removes crippled codes.
If ordering by cheque or money order please make payable to: Danny Y. Wong
and mail registration form to: (please state that this is for Visual Arts)
RE: Visual Arts
Danny Y. Wong
131 64 Ave NW
Calgary, Alberta
T2K 0L9 CANADA
Thank you for supporting shareware authors.
Bug Fixes, Comments and Contact:
Home 403-274-9348 (6:00pm to 11:59pm MST)
Internet: danwong@cadvision.com
danwong@oanet.com

1.35 Disclaimer

Disclaimer: The author makes no warranties, either expressed or implied.
This program is provided on an "as is" basis and the author will not be
liable for any damages caused or alleged to be caused directly by using
this program. Use at your own risk.

1.36 Copyright Notice

1.37 Arexx Setup

Arexx Setup(IFF)
Arexx is now part of Visual Arts. Any program created in Visual Arts can
now have its own Arexx port for other programs to access. When setting up
Arexx you first must enter the Arexx port name. This is the most important
step, if Arexx port name is not defined Arexx will not be included in
the source output. Once you have entered the port name, the rest is very
easy. To add a Arexx command simply enter the Arexx command in the entry
field and press return. Note that Arexx commands are case senesitve. To
rename the command, simply click on the name to be changed and type over it
and press return when done. To delete a command, select the name and click
on the delete button. Select OK when done.
As with all active objects, Visual Arts lets you attach or insert your
own custom codes to each Arexx command. When other applications sends a
Arexx command to your application the scripts for this command is automatically
get executed.

1.38 Custom Images

Visual Arts allow any IFF brush to be loaded although the layout editor
doesn't show the actual image, it only draws the size of the image. Images
can not be sized in the layout editor. Simply put, you can't change any
its attributes except images. One important note that if you load a
different depth size brush into the current depth screen the image would
have different colors in the compiled application.
Custom Image: Button
Custom Button (IFF)

Visual Arts allow you to use the regualar standard Intuition boolean gadget. Boolean gadgets can have its own custom primary and secondary image or one or no image at all. But Visual Arts won't draw the border if the boolean gadget has no images. Visual Arts only recongize IFF brushes they can be any depth as long as your project can handle the brush depth. Multiple brushes can be used and Visual Arts only generate one instance of the same brush when outputing to source. If the brushes to be loaded are in a different directory but with the same name, Visual Arts treats it as the same.
Important: Due to a bug, the initial display of the brush doesn't show the actual size of the image, you will need to double click on the custom button image and select OK again to refresh the object.

1.39 Setting a Color Map

Color Map(IFF)
A totally different color map can be selected from any IFF picture file as long as it has a color map and use it in Visual Arts. If the color map has more or less than the current color map colors, Visual Arts only loads the maximum colors allow for the current screen mode. Note that Visual Arts picks its own random color palette if the project screen has more than 4 colors.

1.40 Why Another GUI Designer?

Why another GUI designer since there are so many already on the market? And how good is Visual Arts?
Simply put: There are NO PD/Shareware GUI designer in the Amiga market that is simple to use and yet powerful enough for professional use. We've seem many GUI designer lack simplicity in term of use and not enough features to build professional looking interfaces.
Visual Arts was designed to be simple and yet powerful. It was bulid from the start to be user friendly and object oriented. There are features in Visual Arts which aren't found in any GUI designer.
I'm a lazy user and I'm a creative programmer. I want to do as minimal work as possible to get the task done and don't care how its done as long as the task is satisfying with quality.
When is the last time you've evaulated a GUI designer? How's the user interface? Was it easy to use? Powerful features? Keep these questions

in mind when you evaulate Visual Arts and compare.
If you don't like Visual Arts then write a better one your yourself!
if (VisualArts == WAYCOOL)
Cool! Let's register!
else
{
'rx WB Trash 'VisualArts"
'rx WB EmptyTrash'
}

1.41 Visual Arts Installation

Visual Arts requires that you install the following libraries in your
lib: directory and the three include files in the include: directory.
These are not runtime libraries, they are linkable libs and don't required
distribution when you distribute your applications.
Double click on the VAInstall icon in the VAInstall directory or you can
copy to the lib: directory: copy #?.lib lib:
1. VisualArts.lib
2. PopUpMenuClass.lib
copy to the include: directory
1. VisualArts.h
2. PopUpMenuClass.h
copy to the include:clib directory
1. VisualArts_protos.h
In your SASCOPT add the two libraries to the library listview.
Note: you must link with the two libraries. You are only required to
link with the PopUpMenuClass if you are using the PopupMenuClass.
Example of Slink:
1> slink lib:c.o,myprg.o to MyPrg lib lib:VisualArts.lib lib:PopUpMenuClass.lib
lib:sc.lib lib:amiga.lib
You can also you the SMake to compile and link your program:
1>smake -f example_makefile
The two libraries should be searched first before any other libraries as
above.
TextField Gadget BOOPSI

Visual Arts v2.5 supports the Textfield Gadget. The installation will copy
all the textfield.gadget files to your development environment. If you get
compilation errors for textfile gadgets then I might not have included all
the neccessary files. You can get textfield gadget on Aminet site.

1.42 What's New In Version 2.11

New for V2.2 Built-In Functions

---

*Separated the VisualArts.h into one header file one proto file.

*Defaulf Font - fixed the problem if WB defualt font is other than topaz 8.

*Font, File and Screen mode ASL reqester

-You can now access the standard Font, File and screen mode

requesters.

*Clip Board - you can now read and write to the clipboard.

* Serial -Access the serial device. If used with the Console feature

you can create a simple terminal application without writting

a single line of code!

How?

Go to the 'Preference' option in the 'Code' menu and check

both 'Console' and 'Serial', then generate the code. You the

console window to input commands.

* Speech

-It is now possible to add speech to your applications. See

'Speech' for more detail.

* Console Window

-Applications can now add a console window.

* VA Preference

-Added checkbox 'Editor on VA'. When this is checked the editor

you selected will open on VA's screen. Of course you'll have to

specify that your editor is to open on screen name 'VArts'.

-Added WB3.x. When this is checked you will be able to use WB3.x

features such as scaled checkbox and MX and other GadTool features.

* Screen Mode

-When you click on 'Load' in the screen mode setup when VA is

started it will default to the right path specified in the .prefs

file.

* Generate C codes

-Both WB2.x and WB3.x is now functional. Note that when generating

codes for WB3.x you will need to check 'WB3.x' checkbox otherwise

some of the codes will NOT be generated.

-A makefile and SCOPTIONS files are generated when generating the

source codes.

-Most of the IDCMPs are now functions and its moved to the function

file.
-Generated files are renamed in V2.05 as follow. (there are no
.h files)
Main File --> unnamed1.c
Function File --> unnamed1_func.c
Script File --> unnamed1_scripts.c
Image File --> unnamed1_images.c
Make File --> unnamed1_Makefile
SCOPTIONS --> SCOPTIONS
There reason to get rid of the .h file is not CPR doesn't allow
.h files to be included when debugging.
* Context Sensitive Layout
-It has been enhanced for WB3.x. A TRUE WYSIWYG when the window
is sized, codes in the object scripts are eliminated but it is
still in WB2.x for compitability.
* Saving
-When saving a project file a project icon is generated. There is
a bug somewhere in the codes. DO NOT double click on the icon
project if run from RAM. Workbench will not work when you exist
from VA. However, it does work if you started other than RAM.
* GadTool Fonts
-When selecting a font for an Object previous version did not
default to the selected font attributes such as font name, size etc.
This has been fixed and will display the font attributes for the
selected font when you click on 'Font' gadget.
* BevelBox for WB3.1
-Added Ridge and IconDropBox boxes for WB3.1. You can still selected
if you are using WB2.1 have to delete the last two items for WB2.1
* Misc Enhancements
-You can now edit IDCMP scripts. Select 'IDCMP scripts' in the
edit menu. Only IDCMP flags selected in the Window attribute's
window are listed in the listview.
-Screen Mode selection and Label Listview selection now has double
click detection. Just double click on the item.
-Add WB3.1 GadTools features will be ghosted if WB2.x is selected.
-Added memory availability and VA screen name in About menu item.
New for V2.01
---
* Extended String Gadget

-GadTool and standard intution string/integer gadget is now extended
to support the extentsion structure.
-This feature allows the gadget to have its own active and inactive
foregound and background color.
-When a new project is created, Visual Arts uses the standard
WorkBench colors. You can modify the extended structure in the
'Edit' menu under 'Extended Gadgets' menu item.
-The InitialMode flags defines how the editing will be.
-All string/integer gadgets uses this structure and thus the
behavior of all string/integer gadgets will be the same.
* WorkBench 1.x String and Integer Gadget
-WB 1.x string and integer gadget is now supported.
-When creating a String/Integer GadTool, click on the 'WB 1.x Style'
check box to make it a WB 1.x gadget.
-WB 1.x gadgets has no borders. If you want a border you'll have to
draw a rectangle object around it. The easiest is to select the
gadget, alt-1 or alt-2 or alt-3.
-The maximum string length of string is defined to be 100 chars and
the integer is defaulted to be 10 chars.
-The WB 1.x string/integer gadget is represented by a standard
rectangle object with a back slash '/'. Note that the hilight
button object is represented by '\'.
* Test GUI
-It is now possible to test your current interface. Select
'Test GUI' item in 'Design' menu.
-The test function is WYSIWYG. Window Flags and IDCMP flags are
from the current project, the CLOSEGADGET box is added so that
you can exit the test function. ESC key will also exit the test
function.
-Currently the Test GUI DOES NOT support the following:
- hilight or custom buttons.
- string gadget (emulated by String/Integer GadTools)
These two are standard boolean and string/integer gadgets.
-Popupmenu is not support.
-Images is supported.
-If the window type has scrollable arrows the Test function will not
show it, although the BORDERRIGHT and BORDERBOTTOM is showing.
-The output of the Test function is the final look when complied.
-To exit the Test function click the close box gadget or press the

ESC key.

Enhancements

-Add hilight, custom buttons and string/integer gadgets.

-Add scrollable arrows for scrollable windows.

-Add popupmenu gadget.

Object Alignment

-Objects can now be aligned on width and height. This function behave the same as aligning objects left or right. The first selected object in the list will be the anchor.

-Again Line and text objects CAN NOT be aligned on width or height.

-Alignment functions are located in the 'Object' menu.

Enhacements and Bug Fixes in 2.0

---

* Visual Arts Preference setup

-The 'Use' option is now added.

-History check box is now fixed.

* Window Setup

-Added Zoom gadget fields for window.

-Added the SuperBitMap check box. It you want the window to be a scrollable super bitmap you'll need to check the 'SuperBitMap' checkbox. The default for window with scrollable arrows is no super bitmap. (only for window types (13-16)

Future Enhacement

-Since the Zoom option is added to the window attribute, every window now has zoom gadget no matter if you want one or not. Future will allow you to select Zoom on or off.

* Border an Object

-When adding a border to an object using the keyboard short cut, (alt-1, alt-2, atl-3) before it creates the border but the object ID doesn't change for the next border.

* Hilight/Custom Button

-When creating a hilight button before it didn't read the widht and height correctly thus creating a one by one pixel object, although it works for the custom button.

* Context Sensitive Layout

-Before when adding/editing a script the codes for CSL was not added. This only works if you are editing the object for the first time, if you are editing an existing script the codes will not be generated thus the context sensitive layout will NOT work!

Future Enhacements

-Remember listview selected item.

-Support primitive objects and custom buttons and images.

* Source Generation

-Alot has been changed to support non scrollable bitmap windows.

-Extended String Gadgets

-Minor changes

Pre-Version 2.0

Version 2.0 fixes a major bug that hogs the CPU and wouldn't let go until hell freezes. Other fixes and ehancements are:

1. Freed up the CPU.
2. Creating a GadTool and selecting OK will draw an visible GadTool, its now fixed.
3. Problem with the Color Map and Color Palette if opening a different project with higher depth.
4. Added 'Save' menu option. This will automatically save into the path specified in the Preference setup without asking for a file name.
5. Add a String gadget option for ListView gadgets. Both ListView and the String gadget MUST be the SAME width.
6. Automatically create Button gadget.
7. Context Sensitive layout. Any standard GadTool can be used in context sensitive layout.
8. Test function for testing User Interface.
9. Extend string gadget support.
10. Standard string/integer gadgets.
11. Support WB3.x gadtools.
12. Visual Arts now open as a PUBLIC screen. Name is "VArts".
13. Many enforcer hits removed.

1.43 Context Sensitive

Context Sensitive (IFF)

Context Sensitive layout applications can dynamically resize the window's content when the window size changes. Visual Arts supports the standard Intuition GadTools for use in context sensitive. Any GadTool can be made into context sensitive.

Visual Arts does not support text, graphic objects and popup in this release. It will be added in the future releases. If you are designing

context sensistive layouts please DO NOT use text, graphic objects and
the popup.
In order to set up your project to be context sensitive you will need to
do the following:
1. Select 'Sensitive On' menu item from the 'Context' menu.
2. Create an GadTool object.
3. Select the 'Context' button.
4. Select the context type for the GadTool. The valid types are:
MoveX - object can be moved to left or right.
MoveY - object can be moved up or down.
ExpandX - object can be expanded or shrinked.
ExpandY - object can be made taller or shorter.
Static - object can not be moved or expanded.
IMPORTANT:
If you are using scripts in your current project and have never set the
context sensitive option, and wants to use context sensitive then you will
have problems. When you edit the script the context sensitive codes are
not generated if there is a script for the object already exist. The codes
will only be generated if editing the script for the first time.
If you want to add context sensitve layout to your existing project, you
will need to assign the script path in the 'Preference' setting to a path
that doesn't contain any scripts. Once you have done it, select Context
Sensitive Layout and geneate the codes. Once you have generated the codes
for context sensitive, using a editor copy and append the context sensitive
codes into the beginning of your current project. You'll need to do this
for every object that has a script attached.
An good example of context sensitive is the standard ASL requester. To
emulate the ASL requester in Visual Arts, create the following GadTools and
list name.
1. Select 'Sensitive On' menu item from the 'Context' menu.
2. Create a list using the List Manager. Any list will do, make sure that
you have enough items.
3. Create a ListView object on the layout, select the label (list) you've
created in step number 2 by clicking on the 'Label' button in the requester.
4. Select 'Context' button.
5. Check 'ExpandX' and 'ExpandY'. This will allow the listview to grow in
width and height. Select OK.
6. Create a new String object just under the ListView object. Make sure
that the string gadget is the same size of the list view gadget and that

the x co-ordinate is the same too. Use Control-Double click on the object
to verify their settings.
7. Select 'Context' button.
8. Check 'MoveY' and 'ExpandX'. This will move the string gadget up or down
and grow/shrink left or right.
9. Double click on the ListView object, select 'Gadget' button and select
the string gadget you just created. This will allow the string gadget to
display the item the user selects from the listview. Make sure that the
width of the listview is that same as the string gadget otherwise the
program will not run if compiled. To check this hold the 'Control' key
and double click on the listview object and note the position values. Now
doublel click on the string gadget and compare the position values with the
listview value and make sure that it is the same.
10. Select the window type with a size gadget and generate and compile.

1.44 Requester and ASL Functions

With version 2.2 you can now accces the standard requster and ASL
libraries. See included examples on how to call these functions.
1. Standard requester
2. Font requester
3. File requester
4. Screen Mode requester
Built-In Functions

1.45 Speech

With version 2.1 you can now add speech to your applications. You will
not find the speech option in the layout editor, it is part of the linkable
library.
It is straight forward to intergate the speech. There are only three
functions in order to use speech.
1. InitSpeech() - initialize the speech module
2. Speak(char text[], short volume, short rate, short sex)
text - any text string upto 256 characters.
volume - volume of the speech 0 - 64
rate - the rate of the speech 40 - 400
sex - voice 0 - male 1 - female
-the Speak() function does the actual speaking of your text.

eg: Speak("Hi, I am Amiga", 32, 200, 1);

3. DeInitSpeech() - deinitialize the speech module.

In your application do the following:

1. Add InitSpeech() function in the main() function of your application.

You must add this before calling the Speak() function.

2. Use the Speak() function to speak your text.

3. Add DeInitSpeech() function to the end of the main() function. You must

call DeInitSpeech() to free up the allocated memory.

Example:

```
void main(void)
{
OpenWindow...
InitSpeech(); --> 1
Do some processing...
Speak(myString, 64, 200, 1); --> 2
Do some more...
Cleanup...
DeInitSpeech(); --> 3
}
```

1.46 Adding a Console Window

To add a console window to your application it is very simple. Select the ’Code’ menu option and select the ’Preference’ menu item. Next check the ’Console’ checkbox and select the ’Edit’ button. This will let you enter the Console window’s title and the size of the window. If the ’Console’ checkbox is not checked, the codes for the console window will not be generated. More Notes

1.47 Console and Serial Notes

Note: with the combinations of the console and serial handler, you can create a simple terminal program without writing a single line of code! How?

1. create a new project.
2. open the code preference window. (’Code’ menu item)
3. check both console and serial check boxes and enter their attributes.
4. generate and compile your project.
5. onced compiled run the programm and try do the following in the

console window.

a. atl1

b. at

c. atdt123-4567 (replace the number 123-4567 with a valid bbs number)

Note that there are two windows open. One is the main window and one is the console window. You could ignore the console when you generate the codes. But then you'll have to programatically access the serial. This is only if you don't want to add a console window. Refer to the Built-In function topic on how to call the built-in functions. Built-In Functions

1.48 Adding a Serial Handler

To add a serial handler to your application it is very simple. Select the 'Code' menu option and select the 'Preference' menu item. Next check the 'Serial' checkbox and select the 'Edit' button. This will let you enter the serial attributes such as baud rate, parity etc. If the 'Serial' checkbox is not checked, the codes for the serial handler will not be generated. More Notes

1.49 AppWindwo Example Codes

```
/* ********************************************************************** */
/* C code generated by: */
/* Visual Arts Version 2.1 */
/* Copyright 1994-95 Danny Y. Wong All rights reserved */
/* Calgary, Alberta (CANADA) */
/* Partial of the code is copyright by Jaba Development */
/* ********************************************************************** */
#include "VisualArts.h"
//gadget ID's defined by Visual Arts
#define ID_clear 0
#define ID_quit 1
#define ID_icons 2
//number of gadgets in this project
#define Project0NumGads 3
#include "AppWindow_func.c"
//function prototypes for the three gadgets
int clearObj(struct VAobject VAObject);
int quitObj(struct VAobject VAObject);
```

```
int iconsObj(struct VAobject VAObject);
//program prototypes
int GetPubScreen(void);
void ClosePubScreen(void);
int OpenProject0Window(char windtitle[80]);
void CloseProject0Window(void);
int Project0Handler(void);
int Project0MainHandler(void);
int Project0AppWindHandler(struct MsgPort *appwindport);
void DrawProject0Objs(void);
int main(int argc, char *argv[]);
UBYTE *PubScrName = NULL;
struct DrawInfo *ScrDrawInfo = NULL;
APTR VisualInfo = NULL;
struct Screen *Scr = NULL;
struct Window *Project0Wnd = NULL;
struct AppWindow *Project0AppWind = NULL;
struct MsgPort *Project0AppWindPort = NULL;
struct Gadget *Project0GList = NULL;
struct Gadget *Project0Gadgets[Project0NumGads];
struct IntuiMessage Project0Msg;
UWORD Project0Left = 158;
UWORD Project0Top = 48;
UWORD Project0Width = 224;
UWORD Project0Height = 103;
//program fonts
struct TextAttr topaz8 = { (STRPTR)"topaz.font", 8, 0x00, 0x01 };
struct TextAttr topaz800 = { (STRPTR)"topaz.font", 8, 0x00, 0x00 };
//using Visual Arts internal list, create a NULL list with no
//items in it because you gonna dynamically add them
UBYTE *IconsLabels[] = {
NULL
};
//the three gadget types
WORD Project0GadTypes[] = {
BUTTON_KIND,
BUTTON_KIND,
LISTVIEW_KIND,
};
```

```
//the three gadget structure definitions
struct NewGadget Project0NGads[] = {
6, 73, 56, 12, (UBYTE *)"_Clear",&topaz800, ID_clear, PLACETEXT_IN, NULL, (APTR)clearObj,
153, 73, 56, 12, (UBYTE *)"_Quit",&topaz800, ID_quit, PLACETEXT_IN, NULL, (APTR)quitObj,
6, 2, 203, 68, (UBYTE *)"", &topaz800, ID_icons, PLACETEXT_ABOVE, NULL, (APTR)iconsObj,
};
//tags for the three gadgets
ULONG Project0NTags[] = {
(GT_Underscore), '_', TAG_DONE,
(GT_Underscore), '_', TAG_DONE,
(GTLV_Labels), NULL, (GTLV_Top), 0, (GTLV_ReadOnly), TRUE, (GTLV_ScrollWidth), 16, (LAYOUTA_Spacing), 0, TAG_DONE,
};
//get the public screen info
int GetPubScreen(void)
{
if (!(Scr = LockPubScreen(PubScrName)))
return(1L);
if (!(VisualInfo = GetVisualInfo(Scr, TAG_DONE)))
return(2L);
if (!(ScrDrawInfo = GetScreenDrawInfo(Scr)))
return(3L);
return(0L);
}
// close all the resources
void ClosePubScreen(void)
{
if (VisualInfo)
FreeVisualInfo(VisualInfo);
if (Scr)
UnlockPubScreen(NULL, Scr);
if (ScrDrawInfo)
FreeScreenDrawInfo(Scr, ScrDrawInfo);
}
int OpenProject0Window(char windtitle[80])
{
struct NewGadget NewGad;
struct Gadget *Gad;
register UWORD i, j;
UWORD offsetx = Scr->WBorLeft;
```

```
UWORD offsety = Scr->WBorTop + Scr->RastPort.TxHeight + 1;
//create the context first, if NULL then something is really wrong
if (!(Gad = CreateContext(&Project0GList)))
return(1L);
//for the three gadgets create it them
for (i=0, j=0; i < Project0NumGads; i++)
{
CopyMem((char *)&Project0NGads[i], (char *)&NewGad, (long)sizeof(struct NewGadget));
NewGad.ng_VisualInfo = VisualInfo;
NewGad.ng_LeftEdge += offsetx;
NewGad.ng_TopEdge += offsety;
Project0Gadgets[i] = Gad = CreateGadgetA((ULONG)Project0GadTypes[i], Gad, &NewGad,
(struct TagItem *)&Project0NTags[j]);
while (Project0NTags[j])
j +=2;
j++;
if (!Gad)
return(2L);
}
//now open the window
if (!(Project0Wnd = OpenWindowTags(NULL,
WA_Left, Project0Left,
WA_Top, Project0Top,
WA_Width, Project0Width,
WA_Height, Project0Height - Scr->WBorTop,
WA_NewLookMenus, TRUE,
WA_IDCMP, IDCMP_CLOSEWINDOW | IDCMP_MOUSEBUTTONS | IDCMP_MOUSEMOVE | IDCMP_GADGETUP |
IDCMP_GADGETDOWN | IDCMP_VANILLAKEY | IDCMP_INTUITICKS ,
WA_Flags, WFLG_CLOSEGADGET | WFLG_SMART_REFRESH |
WFLG_ACTIVATE | WFLG_DRAGBAR | WFLG_DEPTHGADGET,
WA_Gadgets, Project0GList,
WA_Title, windtitle,
WA_ScreenTitle, "Visual Arts V2.0 Copyright 1994 Danny Y. Wong All Rights Reserved.",
WA_PubScreen, Scr,
WA_MinWidth, 160,
WA_MinHeight, 50,
WA_MaxWidth, 640,
WA_MaxHeight, 200,
TAG_DONE)))
```

```
return(3L);
//create the list gonna hold the icon names
CreateProject0Lists();
//create the AppWindow port and inits here
if (Project0AppWindPort = CreateMsgPort())
{
Project0AppWind = AddAppWindow(1, 0, Project0Wnd, Project0AppWindPort, NULL);
if (Project0AppWind == NULL)
{
CloseProject0Window();
return(-1L);
}
}
else
return(-2L);
//we need to refresh the GadTools we've just create. very important
GT_RefreshWindow(Project0Wnd, NULL);
//refresh other gadgets if any
RefreshGadgets(Project0Gadgets[0], Project0Wnd, NULL);
//now attach the list to the list view gadget, we need to do this here
//because the list is not created before the listview gadget
GT_SetGadgetAttrs(Project0Gadgets[2], Project0Wnd, NULL,
GTLV_Labels, Project0Lists[0], TAG_END);
return(0L); //return 0 means OK
}
//close everything cuz we are done
void CloseProject0Window(void)
{
struct AppMessage *appwindmsg;
if (Project0Wnd)
CloseWindow(Project0Wnd);
if (Project0GList)
FreeGadgets(Project0GList);
//remove the appwindow port and any msgs
if (Project0AppWind)
{
RemoveAppWindow(Project0AppWind);
while (appwindmsg = (struct AppMessage *)GetMsg(Project0AppWindPort))
ReplyMsg((struct Message *)appwindmsg);
```

```
DeleteMsgPort(Project0AppWindPort);
}
}
//window msg port handler, for all the msgs
//when running = 1 then exit handler
int Project0Handler(void)
{
struct IntuiMessage *msg;
struct VAobject VAObject; //Visual Arts object
int running = 1;
int (*func)(struct VAobject VAObject);
ULONG class;
UWORD code;
while (msg=GT_GetIMsg(Project0Wnd->UserPort))
{
CopyMem((char *)msg, (char *)&Project0Msg, (long)sizeof(struct IntuiMessage));
class = msg->Class;
code = msg->Code;
GT_ReplyIMsg(msg); //reply to the msg
switch(class) //which msg
{
case IDCMP_MOUSEBUTTONS :
break;
case IDCMP_MOUSEMOVE :
break;
case IDCMP_INTUITICKS :
break;
case IDCMP_CLOSEWINDOW: //done
return(0);
break;
//use has selected a gadget so fill the VAObject and pass it on to
//the function.
case IDCMP_GADGETUP:
VAObject.va_Window = (struct Window *)Project0Wnd;
VAObject.va_Gadget = (struct Gadget *)msg->IAddress;
VAObject.va_IntuiMsg = (struct IntuiMessage *)msg;
VAObject.va_Flags = 0;
VAObject.va_UserData = 0;
func = (void *)((struct Gadget *)Project0Msg.IAddress)->UserData;
```

```
if (func != NULL)
running = func(VAObject);
break;
//short key cuts
//Visual Arts does not generate the function call for you, you'll
//have to add them your self.
case IDCMP_VANILLAKEY:
switch(code)
{
case 'C':
case 'c': //not generated by Visual Arts
ButtonSelected(Project0Wnd, Project0Gadgets[0]);
running = clearObj(VAObject); //not generated by VA
break;
case 'Q':
case 'q': //not generated by Visual Arts
ButtonSelected(Project0Wnd, Project0Gadgets[1]);
running = quitObj(VAObject);
break;
}
break;
}
}
return(running);
}
//appwindow handler
int Project0AppWindHandler(struct MsgPort *appwindport)
{
struct WBArg *wbargptr;
struct AppMessage *appwindmsg;
int i;
int rc;
while (appwindmsg = (struct AppMessage *)GetMsg(appwindport))
{
wbargptr = appwindmsg->am_ArgList;
/* examine the icons dragged into the window here. */
/* wbargptr->wa_Name ---> name of the icon */
/* wbargptr->wa_Lock ---> directory this icon is in. */
//have to detach the list from the listview gadget first.
```

```
//very important
GT_SetGadgetAttrs(Project0Gadgets[ID_icons], Project0Wnd, NULL,
GTLV_Labels, ~0, TAG_END);
for (i=0; i < appwindmsg->am_NumArgs; i++)
{
//add the new icon name to the list
rc = AddNewNode(Project0Lists[0], wbargptr->wa_Name);
wbargptr++;
}
//now reattach the list to the listview with the update items
GT_SetGadgetAttrs(Project0Gadgets[ID_icons], Project0Wnd, NULL,
GTLV_Labels, Project0Lists[0], TAG_END);
ReplyMsg((struct Message *)appwindmsg);
}
return(0);
}
//this is the main handler for all msgs
int Project0MainHandler(void)
{
int running = 1;
ULONG windsig, signals;
ULONG appwindsig;
while (running == 1)
{
appwindsig = 1L << Project0AppWindPort->mp_SigBit;
windsig = 1L << Project0Wnd->UserPort->mp_SigBit;
signals = Wait( windsig | appwindsig );
if (signals & windsig) //regular intuition msgs
{
running = Project0Handler();
}
else //appwindow msgs
if (signals & appwindsig)
Project0AppWindHandler(Project0AppWindPort);
}
return(running);
}
int main(int argc, char *argv[])
{
```

```
int rc;
if (!(GetPubScreen()))
{
for (rc=0; rc < 1; rc++) //create a list
Project0Lists[rc]=GetNewList();
if (!(OpenProject0Window("Visual Arts -- AppWindow Demo")))
{
rc = Project0MainHandler();
CloseProject0Window();
}
for (rc=0; rc < 1; rc++) //free lists
FreeList(Project0Lists[rc]);
ClosePubScreen();
}
}
```

1.50 AppWindow Example Codes -- Function File

```
/* C code generated by: */
/* Visual Arts Version 2.1 */
/* Copyright 1994-95 Danny Y. Wong All rights reserved */
/* Calgary, Alberta (CANADA) */
extern struct Window *Project0Wnd;
extern struct Gadget *Project0Gadgets[Project0NumGads];
extern void CreateProject0Lists(void);
extern UBYTE *IconsLabels[];
struct List *Project0Lists[1]; //list
//add the items to the lists it there are any
void CreateProject0Lists(void)
{
short i;
i=0;
while (IconsLabels[i])
AddNewNode(Project0Lists[0], IconsLabels[i++]);
}
/* gadget functions */
int clearObj(struct VAobject VAObject)
{
GT_SetGadgetAttrs(Project0Gadgets[ID_icons], Project0Wnd, NULL,
```

```
GTLV_Labels, ~0, TAG_END);
FreeList(Project0Lists[0]);
Project0Lists[0]=GetNewList();
GT_SetGadgetAttrs(Project0Gadgets[ID_icons], Project0Wnd, NULL,
GTLV_Labels, Project0Lists[0], TAG_END);
return(1L);
}
/* Button Clear */
int quitObj(struct VAobject VAObject)
{
return(-1L);
}
/* Button Quit */
int iconsObj(struct VAobject VAObject)
{
return(1L);
}
/* ListView */
```

1.51 Visual Arts Built-In Functions -- Quick Help

The following are Visual Arts built-in functions. Look in VisualArts.h
file of further prototypes.

SPEECH FUNCTIONS

NAME InitSpeech -- Initializes the speech module

SYNOPSIS status = InitSpeech(void)

int InitSpeech(void);

FUNCTION Initializes the speech module and allocate resources

INPUTS NONE

RESULTS

NAME DeInitSpeech -- DeInitializes the speech module

SYNOPSIS DeInitSpeech(void)

void DeInitSpeech(void);

FUNCTION DeInitializes the speech module and deallocate resources

INPUTS NONE

NAME Speak -- speak the text in english

SYNOPSIS status = Speak(sentence, volume, rate, sex)

int Speak(char sentence[], short, short, short);

FUNCTION Speak the text using the passed parameters

INPUTS sentence = array of char upto 255 characters

volume = the volume (0-64)

rate = the rate of the speech (40-400)

sex = (0 = male, 1 = female)

RESULTS status

INTUITION FUNCTIONS

NAME ButtonSelected -- emulates the selected button

SYNOPSIS ButtonSelected(wind, gad)

void ButtonSelected(struct Window *, struct Gadget *);

FUNCTION Emulates the BUTTON_KIND Gadtools when the user selects the

keyboard equivlent.

INPUTS wind = pointer to the current window

gad = point to the Button gadget

RESULTS NONE

NAME SetRPortFill - set the fill pattern type of the current rastport

SYNOPSIS SetRPortFill(wind, type)

void SetRPortFill(struct Window, short);

FUNCTION Sets fill pattern type of the current window’s rastport.

INPUTS wind = pointer to the current window

type = fill pattern type (0 - 47)

RESULTS NONE

NAME DrawBox - draws a rectanlar box

SYNOPSIS DrawBox(wind, left, top, width, height, pen, pattern)

void DrawBox(struct Window *, int, int, int, int, UBYTE, short);

FUNCTION Draws a rectanglar box with a pen color and pattern type.

INPUTS wind = pointer to the current window

left = leftedge of the window

top = topedge of the window

width = width of the box

height = height of the box

pen = color of the box

pattern = pattern type (0 - 5)

RESULTS NONE

NAME DrawLine - draws a line

SYNOPSIS DrawLine(wind, left, top, width, height, pen, pattern)

void DrawLine(struct Window *, int, int, int, int, UBYTE, short);

FUNCTION Draws a line with a pen color and pattern type.

INPUTS wind = pointer to the current window

left = leftedge of the window

top = topedge of the window

width = width of the box

height = height of the box

pen = color of the box

pattern = pattern type (0 - 5)

RESULTS NONE

NAME DrawNCircle - draws a color circle

SYNOPSIS DrawNCircle(wind, left, top, right, bottom, pen)

void DrawNCircle(struct Window *, int, int, int, int, UBYTE);

FUNCTION Draws a circle with a pen color

INPUTS wind = pointer to the current window

left = leftedge of the window

top = topedge of the window

right = rightedge of the window

bottom = bottomedge of the window

pen = color of the box

RESULTS NONE

NAME DrawFBox - draws a rectanlar filled color box

SYNOPSIS DrawFBox(wind, left, top, width, height, pen, pattern, outline, fill)

void DrawFBox(struct Window *, int, int, int, int, UBYTE, short, UBYTE, short);

FUNCTION Draws a rectanglar filled box with a pen color and pattern type.

INPUTS wind = pointer to the current window

left = leftedge of the window

top = topedge of the window

width = width of the box

height = height of the box

pen = color of the box

pattern = pattern type (0 - 5)

outline = outline color

fill = (0 - 47)

RESULTS NONE

NAME DrawFCircle - draws a filled color circle

SYNOPSIS DrawFCircle(wind, left, top, right, bottom, pen, pattern, outline, fill)

void DrawFCircle(struct Window *, int, int, int, int, UBYTE, short, UBYTE, short);

FUNCTION Draws a filled circle with a pen color

INPUTS wind = pointer to the current window

left = leftedge of the window

top = topedge of the window

right = rightedge of the window

bottom = bottomedge of the window

pen = color of the box

pattern = pattern type (0 - 5)

outline = outline color

fill = (0 - 47)

RESULTS NONE

EXEC LIST FUNCTIONS

NAME GetNewList - allocates memory for a new list

SYNOPSIS GetNewList(void)

list = GetNewList(void);

Struct List *list;

FUNCTION allocates memory for a new list

RESULTS list - a pointer to a List structure or NULL if no memory

---

NAME FreeList - deallocates memory for a new list

SYNOPSIS FreeList(list)

void FreeList(struct List *);

struct List *list;

FUNCTION deallocates memory for a new list and all list items

RESULTS list - a pointer to a List structure

NAME AddNewNode - add a new node to an existing list

SYNOPSIS AddNewNode(list, name)

status = AddNewNode(struct List *, char name[]);

int status;

struct List *list;

char name[255];

FUNCTION Add an node to an existing list

RESULTS status = 0 success

NAME DeleteNewNode - delete a node to an existing list

SYNOPSIS DeleteNode(list, name)

status = DeleteNode(struct List *, char name[]);

int status;

struct List *list;

char name[255];

FUNCTION Delete an node to an existing list

RESULTS status = 0 - success

1 - error

NAME FindNodeName - search the list for a name

SYNOPSIS FindNodeName(list, name)

node = FindNodeName(struct List *, char name[]);

struct NameNode *node;

struct List *list;

char name[255];

FUNCTION Searchs a list for the a specific name

RESULTS node - pointer to the current node if found otherwise node

will be NULL

NAME FindNodeNo - search the list on index

SYNOPSIS FindNodeNo(list, index)

node = FindNodeNo(struct List *, UWORD);

struct NoNode *node;

UWORD index;

FUNCTION Searchs a list based on the index value

RESULTS node - pointer to the current node if found otherwise node

will be NULL

List Example

This function basically walk down the list and print the content of each

node.

/* structure taken from VisualArts.h file*/

struct NameNode

{

struct Node nn_Node; /* linked list node to previous or next node */

UBYTE nn_Data[255]; /* name of the node, this is the same as */

/* nn_Node.ln_Name */

};

Calling Example: PrintListName(Project0Lists[0]);

void PrintListName(struct List *list)

{

struct NameNode *worknode, *nextnode;

short i;

if (list == NULL) // if NULL then return

return(NULL);

// position at the head of the list

worknode=(struct NameNode *)(list->lh_Head);

while (nextnode=(struct NameNode *)(worknode->nn_Node.ln_Succ))

{

printf("list item name %s\n", worknode->nn_Data);

worknode=nextnode; // move the next node

}

}

CONSOLE WINDOW FUNCTIONS

NAME ConPutChar - write a single character to the console

SYNOPSIS ConPutChar(request, character)

status = ConPutChar(struct IOStdReq *, char);

int status;

struct IOStdReq *request;

char character;

FUNCTION Write a character to the console

RESULTS status = none zero for success

NAME ConPutStr - write a string to the console

SYNOPSIS ConPutStr(request, string)

status = ConPutStr(struct IOStdReq *, char *);

int status;

struct IOStdReq *request;

char *string;

FUNCTION Write a string to the console

RESULTS status = 0 for success

SERIAL COMMUNICATION FUNCTIONS

NAME SerPutChar - write a single character to the serial device

SYNOPSIS SerPutChar(request, character)

status = SerPutChar(struct IOExtSer *, char);

int status;

struct IOExtSer *request;

char character;

FUNCTION Write a character to the serial device

RESULTS status = none zero for success

NAME SerPutStr - write a string to the serial device

SYNOPSIS SerPutStr(request, string)

status = SerPutStr(struct IOExtSer *, char *);

int status;

struct IOExtSer *request;

char *string;

FUNCTION Write a string to the serial device

RESULTS status = 0 for success

CLIP BOARD FUNCTIONS

NAME CBReadLine - read a single line from the clip board

SYNOPSIS CBReadLine(string)

status = CBReadLine(string *);

BOOL status;

char *string;

FUNCTION read a line from the clip board

RESULTS status = TRUE no error

= FALSE error

NAME CBWriteLine - write a single line to the clip board

SYNOPSIS CBWriteLine(string)

status = CBWriteLine(string *);

BOOL status;

char *string;

FUNCTION write a line to the clip board

RESULTS status = TRUE no error

= FALSE error

REQUESTER

NAME VARequester - display the standard requester

SYNOPSIS VARequester(window, title, format, choices)

status = VARequester(struct Window *, char *, char *, char *);

LONG status;

struct Window *window;

char *title;

char *format;

char *choices;

FUNCTION displays the standard requester

RESULTS status = positive or negative response

FONT, FILE and SCREEN MODE ASL LIBRARY

NAME ASLGetFileName - get a file name using the ASL File requester

SYNOPSIS ASLGetFileName(filename, window, top, left, height, width,

title, path, flags)

status = ASLGetFileName(char *, struct Window *, int, int, int,

int, char *, char *, ULONG);

BOOL status;

char *filename;

struct Window *window;

int top;

int left;

int height;

int width;

char *title;

char *path;

ULONG flags;

Possible flags for the File Reqeuster. You can or the flags.

#define ASLFR_DOSAVEMODE 1

#define ASLFR_DOMULTISELECT 2

#define ASLFR_DOPATTERNS 4

#define ASLFR_DRAWERSONLY 8

#define ASLFR_REJECTICONS 16

#define ASLFR_FILTERDRAWERS 32

#define ASLFR_SLEEPWINDOW 64

FUNCTION open the ASL file requester and return a file with full path

RESULTS status = TRUE no error

= FALSE error

NAME ASLGetFontName - get a font struct using the ASL Font requester

SYNOPSIS ASLGetFontName(textAttr, window, top, left, height, width, flags)

status = ASLGetFontName(struct TextAttr *, struct Window *,

int, int, int, int, ULONG);

BOOL status;

struct TextAttr *textAttr;

struct Window *window;

int top;

int left;

int height;

int width;

ULONG flags;

Possible flags for the Font Reqeuster. You can or the flags.

#define ASLFO_SLEEPWINDOW 1

#define ASLFO_DOFRONTPEN 2

#define ASLFO_DOBACKPEN 4

#define ASLFO_DOSTYLE 8

#define ASLFO_DODRAWMODE 16

#define ASLFO_FOXEDWIDTHONLY 32

FUNCTION open the ASL font requester and return a TextAttr structure

RESULTS status = TRUE no error

= FALSE error

NAME ASLGetScrMode - get a screen mode structure using the

screen mode requester.

SYNOPSIS ASLGetScrMode(scrmodereq, window, top, left, height, width, flags)

status = ASLGetScrMode(struct ScreenModeRequester *,

struct Window *,

int, int, int, int, ULONG);

BOOL status;

struct ScreenModeRequester *scrmodereq;

struct Window *window;

int top;

int left;

int height;

int width;

ULONG flags;

Possible flags for the Screen Mode Reqeuster. You can or the flags.

#define ASLSM_INITIALAUTOSCROLL 1

#define ASLSM_INITIALINFOOPENED 2

#define ASLSM_DOWIDTH 4

#define ASLSM_DOHEIGHT 8

#define ASLSM_DODEPTH 16

#define ASLSM_DOOVERSCANTYPE 32

#define ASLSM_DOAUTOSCROLL 64

#define ASLSM_SLEEPWINDOW 128

FUNCTION open the ScreenMode requester and return a ScreenModeRequester
structure.

RESULTS status = TRUE no error

= FALSE error

1.52 Clip Board

Visual Arts now supports the clip board feature. You can copy and write
to the clip board using the following two built-in functions.

BOOL CBReadLine(char *);

The CBReadLine() function will read a line from the clip board and return
it a pointer to the string.

```
char mytext[80];
```

eg: ok = CBReadLine((char *)mytext);

BOOL CBWriteLine(char *);

The CBWriteLine() function will write a line to the clip board.

```
char mytext[80];
strcpy(mytext, "Testing the Clip Board");
```

eg: ok = CBWriteLine((char *)mytext);

Visual Arts does not support reading and writing of the entire clipboard.
You can easily coded the routines to read/write the entire clipboard.
To read the entire clip board use this function which is taken from the
Rom Kernal Manual and its not included in Visual Arts.

```
void ReadClip(void)
{
struct IOClipReq *ior;
struct cbbuf *buf;
/* Open clipboard.device unit 0 */
if (ior=CBOpen(0L))
{
/* Look for FTXT in clipboard */
if (CBQueryFTXT(ior))
{
```

```
/* Obtain a copy of the contents of each CHRS chunk */
/* The while loop will read the entire clip board */
while (buf=CBReadCHRS(ior))
{
/* Process data */
printf("%s\n",buf->mem);
/* Free buffer allocated by CBReadCHRS() */
CBFreeBuf(buf);
}
/* The next call is not really needed if you are sure */
/* you read to the end of the clip. */
CBReadDone(ior);
}
else
puts("No FTXT in clipboard");
CBClose(ior);
}
else
puts("Error opening clipboard unit 0");
return(0L);
}
```

To write multiple lines to the clip board use the template CBWriteLine function and customize it.

```
void WriteClip(char *string)
{
struct IOClipReq *ior;
if (string == NULL)
{
puts("No string argument given");
return(0L);
}
/* Open clipboard.device unit 0 */
if (ior = CBOpen(0L))
{
/* use multiple CBWriteFTXT() to write multiple lines */
if (!(CBWriteFTXT(ior,string)))
printf("Error writing to clipboard: io_Error = %ld\n",ior->io_Error);
CBClose(ior);
}
```

```
else
puts("Error opening clipboard.device");
return(0);
}
```

Example to write a file to the clip board.

```
void WriteFont2Clip(char *file)
{
FILE *io;
struct IOClipReq *ior;
char string[80];
if (!(io = fopen(file, "r")))
{
printf("can’t open file %s for read\n", file);
return(0);
}
/* Open clipboard.device unit 0 */
if (ior = CBOpen(0L))
{
/* use multiple CBWriteFTXT() to write multiple lines */
while (!(feof(io))
{
fgets(string, 80L, io); /* read a line from the file */
if (!(CBWriteFTXT(ior,string))) /* write to clip board */
printf("Error writing to clipboard: io_Error = %ld\n",ior->io_Error);
}
CBClose(ior);
}
else
puts("Error opening clipboard.device");
fclose(io);
return(0);
}
```