NAME

Vlib − library of vision-related software

SYNOPSIS

cc ... **-I/u/cs449/include**

#include <vista/Vlib.h>

ld ... **-lvista -lm -lgcc** ...

DESCRIPTION

Introduction

The Vista library is a collection of C-callable routines for performing operations that are common in computational vision applications.

A tutorial introduction to the library can be found in the document ''Introduction to Vista Programming'', which is available on-line in the directory **/u/cs449/src/doc/intro**.

The library defines representations for both file-resident and memory-resident objects such as images, and it includes routines for operating on those objects. The library is designed to be easily extended to support additional operations and new types of objects.

A machine-independent format is used for storing data objects in files. This format is described by the **Vfile**(5Vi) manual page.

Usage

As a Vista library user you will need to direct the C compiler to header files located in the **/u/cs449/include** directory (usually by specifying **-I/u/cs449/include**), and you will need to include **<vista/Vlib.h>** in your C source code files. Various section 3Vi and 5Vi manual pages indicate any other header files that may be needed. If you are working with images, for example, you must also include **<vista/VImage.h>**, as indicated by **VImage**(3Vi).

Three versions of the library are available. They are functionally equivalent but they differ in the manner in which they were compiled. To link with an optimized version of the library, specify the option **-lvista** to **cc**(1) or **ld**(1). To link with a debugging version that includes symbol information for **dbx**(1), specify the option **-lvista_d**. The two library archives are found in the **/u/cs449/lib** directory.

The Vista library should be followed in the linking sequence by the math library, which is specified by **-lm**. Depending on what routines you use you may need to include additional libraries; a routine's manual page will say so if this is necessary. Also, if you're using something other than **gcc**(1) (the GNU C compiler) to perform the linking, you will need to include a **gcc**-related library by specifying **-lgcc**. (This is necessary because the Vista library was compiled using **gcc**; when **gcc** links, it includes **-lgcc** automatically.)

Conventions

Symbols defined by the library follow a naming convention that helps to ensure that they are unique and easily recognized as belonging to Vista. All type names, constant names, macro names, and externally-visible function and data structure names begin with a V (capital v). When such names are formed from multiple words they are constructed by capitalizing the first letter of each word. A typical function name, for example, is **VCreateImage**.

Names of structure members and names of function parameters mentioned in the documentation use entirely lower case letters. When such names are formed from multiple words the words are separated by underline characters. A typical structure member is *row_index*.

For those using ANSI C, function prototypes are included in Vista header files. The prototypes are included automatically if either of the preprocessor symbols **__STDC__** or **FUNCPROTO** is defined. ANSI C compilers such as **gcc**(1) normally predefine **__STDC__**, causing the prototypes to be included.

Error Handling

Vista library functions that encounter errors report them using the routines **VError**(3Vi) and **VWarning**(3Vi). The default behavior for these routines is to print an error message to the standard error stream, and, if the error is fatal, to halt the program. The programmer can, however, alter this behavior using **VSetErrorHandler**(3Vi) and **VSetWarningHandler**(3Vi).

Typically, a Vista library function will return a non-zero result (such as **TRUE** or an address other than **NULL) if successful. If it encounters a recoverable error, it will issue a warning message using VWarning(3Vi) and return a zero result. If it encounters a fatal error, it will call VError(3Vi) to issue a message and halt the program.**

An error message produced by the Vista library identifies the routine that encountered the error. The section 3Vi manual page for a routine will usually list the routine's error messages and provide further information about them.

Numeric Data Types

The following types are used for representing numeric data such as pixel intensities, distances, and other measurements. They may be defined in different ways for different machine architectures, but each is guaranteed to have a certain minimum precision. There definitions are in **<vista/Vlib.h>**.

VBit represents an unsigned integer in the range [0,1].

VUByte represents an unsigned integer in the range [0,255].

VSByte represents a two's-complement integer in the range [-128,127].

VShort represents a two's-complement integer in at least 16 bits.

VLong represents a two's-complement integer in at least 32 bits.

VFloat represents a floating-point number in at least 32 bits.

VDouble represents a floating-point number in at least 64 bits.

Each is at least as large as the minimum addressable unit on the present machine (usually a byte), and also large enough to represent its full range of values (e.g., at least 16 bits in the case of **VShort**). It may be larger. For example, **VBit** occupies a byte on most machines, and **VShort** might occupy 32 bits on some machines.

Note for users of ''Kernighan and Ritchie'' (K&R) C: In ANSI C, **VSByte** is declared as **signed char** and an operation involving a **VSByte** value preserves its sign. However, in the older K&R C dialect **VSByte** is declared as simply **char**, which may be either signed or unsigned depending on the machine. If you are using the library with a K&R compiler that thinks **char** is unsigned, take care to preserve the sign of a **VSByte** value when promoting it to an **int** or to some other signed type.

Other Generic Types

The following types are used for various purposes. They are defined in **<vista/Vlib.h>**.

VAttrList represents an *attribute list*, which is described by the **Vattribute**(3Vi) manual page.

VBoolean represents either **TRUE** or **FALSE**

VList represents a list of pointers to opaque objects. It is described by the **VList**(3Vi) manual page.

VBundle represents an object as a type name, an attribute list value, and a block of binary data. It is used to contain objects of unfamiliar type found in data files. **VBundle** is described by the **Vtype**(3Vi) manual page.

VPointer represents a pointer to an unspecified type. In ANSI C it is defined as **void ∗**.

VString represents a null-terminated string. It is defined as **char ∗**.

VStringConst represents a null-terminated string whose contents are guaranteed not to be changed. In ANSI C it is defined as **const char ∗**.

Standard Object Types

The Vista library includes built-in support for some complex types of objects, such as images and edge sets. For each of these standard object types the library provides:

- a format for representing objects in a data file

- a format for representing objects in memory
- routines for transferring objects between data files and memory
- routines and macros for accessing objects in memory
- routines for manipulating objects in memory

There is a section 5Vi manual page for each standard object type. Currently, the standard object types are:

camera Camera model. See **VCamera**(3Vi).

edges Set of edges. See **VEdges**(3Vi).

image Multi-band two-dimensional array of pixels. See **VImage**(3Vi).

It is possible for the programmer using the Vista library to extend this list, adding support for his or her own, custom object types. The **Vtype**(3Vi) manual page explains how.

Identifying Types

One sometimes needs to identify, to a routine, what representation a data value has. The **<vista/Vlib.h>** header file defines the following constants for this purpose:

```
typedef enum {
        VUnknownRepn,
        VBitRepn, VUByteRepn, VSByteRepn, VShortRepn, VLongRepn,
        VFloatRepn, VDoubleRepn,
        VAttrListRepn, VBooleanRepn, VBundleRepn, VListRepn, VPointerRepn,
        VStringRepn,
        VEdgesRepn, VImageRepn
} VRepnKind;
```

The **VRepn**(3Vi) manual page describes some macros for obtaining information about data types.

Constants

The following constants are defined in **<vista/Vlib.h>**, and used as arguments or return values by some library routines:

FALSE is defined to be 0.

TRUE is defined to be 1.

NULL is defined to be 0.

Other Facilities

The Vista library also includes support for

- parsing command line options; see **Voption**(3Vi).
- obtaining parameter values from files; see **Vpd**(5Vi).
- generating PostScript documents; see **Vpostscript**(3Vi).
- writing X Windows applications; see **VX**(3Vi).

FILES

/u/cs449/include/vista
the directory containing Vista library header files

/u/cs449/lib
the directory containing Vista library archives

SEE ALSO

Vista(7Vi), **Vattribute**(3Vi), **Vfile**(5Vi), **Vtype**(3Vi),
VCamera(3Vi), **VEdges**(3Vi), **VImage**(3Vi), **VList**(3Vi),
VError(3Vi), **VWarning**(3Vi), **VRepn**(3Vi), **VConst**(3Vi),
Voption(3Vi), **Vpd**(5Vi), **Vpostscript**(3Vi), **VX**(3Vi)

Introduction to Vista Programming
Vista Contributor’s Guide

AUTHOR

Art Pope <pope@cs.ubc.ca>