Techniques for organizing interactive graphics programs

Rick Ottolini

ABSTRACT

Programs are easier to understand and maintain if they are built out of data modules. A data module is the set of subroutines that share the same variable(s). An important component of the module is the *header* that describes the contents of the module. Module coding techniques are illustrated with examples from the C programming language.

DATA MODULES

A *data module* is the set of subroutines that share the same variable(s). More abstractly, a data module defines a data object and operations upon it. For example, a seismic data module has variables for the data files, data dimensions, and data buffer. Subroutines include parameter fetching, memory allocation, data loading, etc.

Data modules derive their advantages from being relatively self-contained conceptual and syntactic units. Larger code units increase complexity, while smaller code units decrease coherence.

Modularization strategy for interactive seismic graphics

This section illustrates how to partition a program into modules. An interactive seismic graphics program consists of accessing a seismic database, performing data transformations, and displaying the result. First, identify the major data concepts and operations on these data. Then assign a module to each and write the module. Interactive seismic graphics data concepts include accessing a seismic database (e.g. SEGY, seplib, internal program data representation), data transformation (database floating point to display integers, windowing, normal moveout), and graphics objects (image buffers, color tables, menus). Figure 1 lists modules in a seismic cube dissection program.

Figure 1: Modules for a seismic cube dissection program.

NAME	FUNCTION	SAMPLE SUBROUTINES
Axis	dataset & image axes	AxisGetScale, AxisDraw
Brick	seismic cube sub-bricks	BrickResize, BrickDraw
Color	color schemes	ColorLoadGray, ColorLoadFlag
Cursor	cursor shape	CursorShrinkShow, CursorExpandShow
DataCube	seismic data cube	DataCubeCreate, DataCubeLoad
Font	graphics fonts	FontInit
Image	display image	ImageCreate, ImageSave
MenuBar	command menus	MenuBarInit
Mouse	mouse events	MouseInfo
Window	display window	WindowInit, WindowQuit

MODULE HEADERS

A data module consists of a *header* that describes the variables and subroutines in the module and a *code body* that implements the subroutines. The header documents the module for programmers and tells the compiler how to link modules together. It defines data declaration templates, global variables, and global subroutines. For example, the header for a color table module in the C language is:

```
/*
 color_table module header
 */
/* templates for data declarations */
#define CT_SIZE 256
typedef unsigned charColorComponent;
typedef char        Name;
typedef struct {
            ColorComponentred;
            ColorComponentgreen;
            ColorComponentblue;
            } Color;
typedef struct {
            Name   name[NAMESIZE];
            Color  color[CT_SIZE];
            } *ColorTable[1]
/* global variables */
extern ColorTable   grayscale;
extern ColorTable   flag;
extern ColorTable   rainbow;
/* global subroutines */
extern ColorTable   ColorTableCreate (/* name,color_neg,color_zero,color_neg */);
extern void ColorTableLoad (/* color_table,window */);
```

Data declaration templates

A *data declaration template* defines the syntax of a data declaration and gives it a custom name[1]. The syntax can be a single variable (e.g. 'ColorComponent'), or an aggregate[2] of variables (e.g. 'Color'). Templates can derived from other templates by nesting (e.g. 'ColorComponent' in 'Color') or by modification[3].

A template gives meaningful and distinguishing names to variable declarations. 'ColorComponent' is more meaningful than 'unsigned char'. 'ColorComponent' can be distinguished from image buffers that are also 'unsigned char' in C. 'ColorTable' is a more meaningful than the aggregate of variables it represents.

A template hides the details of a variable declaration and makes it easier to change the details. All of the 'ColorComponent' variables could be changed to floating point color components by just changing the 'ColorComponent' definition and nothing else in the module. A template may hide a large number of internal variables, a frequent occurrence in graphics code, allowing compact global variables and subroutine arguments.

Separate module header files

Keeping module headers in separate files to be prepended to code bodies eliminates code redundancy. This means smaller and consistent code.

Commented header files are a convenient source of program documentation. The collection of header files describes the variables and subroutines used in the program or available in a library.

Naming and spelling conventions.

It is good practice to preface every name introduced in a module, whether a constant, template, variable, or subroutine, with the name (or abbreviated name) of that module. Then the location of the name's definition can easily be found.

Constant names are entirely capitalized. The first letter of template and subroutine names are capitalized while variables are not capitalized. There is no fixed convention for multiple word names. I prefer capitalizing the first letter of new words. Others use underscores, or run words together.

[1] In C, template naming is done with *#define* and *typedef* statements. In FORTRAN, custom naming is done with macros.
[2] In C, variable aggregating is done with arrays, *struct*, *union* and *enum* statements. In FORTRAN, variable aggregating is done with arrays, *common* and *record* statements.
[3] Not possible in C or FORTRAN.

MODULE CODE BODY

The code body implements the subroutines described in the module header. Continuing with the color table example

```
/*
 color_table module body
 */
/* prepended header files, including self */
#include    <stdio.h>
#include    "data_cube.h"
#include    "color_table.h"
/* global variable declarations */
ColorTable grayscale;
ColorTable rainbow;
ColorTable flag;
/* global subroutine definitions */
ColorTable ColorTableCreate (color_neg,color_zero,color_pos)
           { ... }
void        ColorTableLoad (color_table,window)
           { ... }
```

Descriptions of variables and subroutines used from other modules, via the header files are prepended[4] to the code body. Due to a quirk in C, the global variables must be redeclared in the module body, except without the 'extern' statements. Otherwise, the code body is straight forward.

MORE ABOUT DATA MODULES

Library interface modules

Some data abstractions are more or less defined in external code libraries. It is good practice to write interface modules for external code. In the example of Figure 1, 'Cursor', 'Font', 'MenuBar' and 'Window' are interface modules. A interface module might amplify the functionality of external code (e.g. custom cursors in 'Cursor'). It isolates external code that might change in the future. It also standardizes the interface to external code.

Joint subroutines

Sometimes a subroutine uses variables from more than one module. A common example is a format conversion subroutine. If these subroutines use external code, then they should go into the appropriate interface module in order to isolate external code. Otherwise, the subroutine should go into the module least likely to be used in future

[4] The *#include* statement prepends files in C and FORTRAN.

programs in order to reduce code in future programs. If there are a large number of subroutines sharing the same variables, then the original modular design should be re-evaluated to see if a new joint-data module should replace or supplement the original.

Multiple and dynamic data declarations

Data modules can be designed such that an arbitrary number of variables of the same kind can be created and discarded during the course of a program. For example, a seismic data viewing program might want to simultaneously display any number of datasets. To implement this, special *create* and *destroy* subroutines are defined. A creation subroutine allocates space, initializes variables, and returns a *reference variable*. The reference variable is a memory pointer or table index used to identify a piece of data. Take a file input/output module for example. Open and close subroutines create and destroy the input/output channels. The reference variable is the channel number.

Module dictionary

A *dictionary* is a condensation of the module header file descriptions into a table and is a useful programming aid. Figure 1 is a partial example. Table entries include the name, subroutines, and dependencies of each module. The templates or subroutines of some modules depend on other modules and would have to be included with that module in future programs.

Special programming languages

There are special languages and pre-processors that help organize programs into data modules. These include Modula, Ada, Smalltalk, ObjectiveC and others. Unfortunately, not one of these is widely enough available to be a replacement for C or FORTRAN. The techniques presented in this paper reap many of the benefits of these languages.

Interface subroutines

An interface routine provides programmers with a concise and uniform access to module contents. Keyword-value arguments in the argument list trigger variable assignments or operations. The programmer who uses the module need not know how the argument is implemented, and the programmer who writes the module is free to change the implementation. Examples of interface routines are FORTRAN file input/output and SunView graphics.

Writing the interface routine requires adding a keyword list to the header file. A keyword list for the color table example is

```
typedef enum {
        CT_NAME,/* set color table name: value is name */
```

```
CT_ENTRY,/* set color table entry: values are index,red,green,blue */
CT_SET,   /* set the color table; reference variable is value */
} CT_keyword;
```

Enumeration variables make good keyword names in C.
The interface subroutine parses and executes each keyword

```
ColorTableSet (color_table,arglist)
ColorTable color_table;
CT_keyword arglist;
        {
        CT_keyword *arg;
        int index, do_load = NO;
        /* process the argument list until a zero value */
        for (arg=&arglist; *arg != 0; arg++) switch(*arg)
                {
        case CT_NAME:
                /* set the name variable */
                strcpy (color_table.name,(Name *)(++arg));
                break;
        case CT_LOAD:
                /* set the load subroutine flag */
                /* delay load until all arguments processed */
                do_load = YES;
                break;
        case CT_ENTRY:
                /* set the four value entry variable */
                index = (int)*(++arg);
                color_table.color[index].red = (ColorComponent)*(++arg);
                color_table.color[index].green = (ColorComponent)*(++arg);
                color_table.color[index].blue = (ColorComponent)*(++arg);
                break;
                }
        /* execute the load color routine */
        if (do_load) ColorTableLoad (color_table);
        }
```

The interface routine interprets the number and type of each argument. (This could be more systematically encoded into a table.) The interface routine also makes sure that arguments are executed in their proper order, even though they may be specified in an arbitrary order.

SYNTHESIS

Program:

- Program modules,
- Interface modules,
- Special files.

Each module:

- Header:
  - Data declaration templates,
  - Global variables,
  - Global subroutines.
- Code body:
  - Include files,
  - Global variable declarations,
  - Subroutines.

Special files:

- Main routine,
- Module dictionary.

STANFORD UNIVERSITY

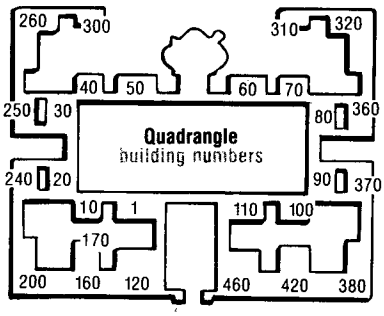
North

Free TIME LIMIT Parking
8 to 5, Monday through Friday

PAY LOT and/or METER Parking
(coins needed except at Medical Center)
8 to 5, Monday through Friday

- **HANDICAP Parking**
For use **only** by persons with handicap parking permits. Enforced at all hours every day. **$50 minimum fine** for parking in these zones without proper permit.

Parking permits are required in 'A' 428 'C' lots from 8 to 5, Monday ugh Friday. At other hours, e lots are free and open to the public.



Information for visitors

MITCHELL BUILDING

TO Foothill Expressway and Los Altos

Junipero Serra Freeway (Interstate 280) via Page Mill Road

SANTA MARIA

JUNIPERO SERRA BOULEVARD

GERONA RD

ESTUDILLO ROAD

MIRADA AVE

SANTA YNEZ ST

HANNA HOUSE

FRENCHMAN'S RD

CABRILLO AVE

LOU HENRY HOOVER HOUSE

EL ESCARPADO WAY

ALVARADO CT

SAN RAFAEL PL

SAN JUAN ST

PORTOLA

CONSTANZO ST

LOOSLEY LANE

CAMPUS DRIVE EAST

TO Golf Course
Center for Advanced Studies in the Behavioral Sciences
National Bureau for Economic Research

TO Menlo Park

Junipero Serra Freeway (Interstate 280 via Alpine Road or Sand Hill Road

MAYFIELD AVE

ESPLANADA WAY

VALPARAISO ST

CORONADO AVE

ALVARADO ROW

SANTA YNEZ LANE

SALVATIERRA ST

PEARCE MITCHELL HOUSES

CAMPUS DRIVE

FLORENCE MOORE HALL

ROW HSO OFF

LOMITA CT

THE KNOLL

LOMITA DR

Lagunita

BOAT HOUSE

GOLF DRIVING RANGE

ELLIOTT PROGRAM CENTER

GOVERNOR'S AVE SOUTH

Governor's Corner

Sterling Quad

GOV AVE EAST

CAMPUS DRIVE WEST

FINE HILL RD

SAN FRANCISCO CT

S.F. TERRACE

BOLIVAR HOUSE

HUSTON HOUSE

COWELL LANE

COWELL STUDENT HEALTH CENTER

THE BRIDGE

OWEN HOUSE

SERRA HOUSE

ROSA HOUSE

ROGERS

NATHAN ABBOTT WAY

INTERNATIONAL CENTER

FACULTY CLUB

HARMONY HOUSE

BOWMAN ALUMNI HOUSE

BLACK HOUSE

ROBLE HALL

ROBLE DR

LAGUNITA COURT

Lane L

BOWDOIN ST

TO Stanford Avenue

DRIVE

CAMPUS

Escondido Village

MIRRIELEES HOUSE

WILBUR HALL

STERN HALL

ESCONDIDO RD

BRANNER HALL

High rise buildings in Escondido Village

Abrams
Barnes
Blackwelder
Hoskins
Hulme
McFarland
Quillen
Mirrielees

SCHOOL OF LAW

(CERAS) CENTER for EDUCATIONAL RESEARCH

BRAUN MUSIC CTR

POST OFFICE

BOOKSTORE

PLACEMENT CENTER

White Plaza

Clubhouse

OLD UNION

Bldg 590

DINKELSPIEL AUD

TRESIDDER UNION

DIJENA ST

OLD FIRE HOUSE

Bldg 570

PRESS BLDG

STORKE BLDG

TERMAN ENGINEERING CENTER

SAN HUBERTO WAY

ROBLE GYM

ROBLE POOL

SANTA TERESA ST

SANTA TERESA LANE

GOVERNOR'S AVE

LOS ARBOLES AVE

SEARSVILLE RD

Tennis Courts

PANAMA ST

DURAND

NOBLE

MITCHELL

Bldg 500

Bldg 520

Bldg 530-40

Bldg 550

Clock

MEYER MEMORIAL LIBRARY

SCHOOL OF EDUCATION

MEMORIAL CHURCH

GREEN LIBRARY

QUADRANGLE

SKILLING

McCULLOUGH

SALVATORI APPLIED ELECTRONICS

HEPL END STATION

CARNEGIE DEPT OF PLANT BIOLOGY

GEORGE FORSYTHE HALL (COMPUTER FACILITY)

SPRUCE

CYPRESS

VIA ORTEGA

Jordan Quad

POLYA

REDWOOD

PINE

CEDAR

TOYON HALL

BROTHERS HALLS

BROTHERS WAY

MANZANITA PARK (Trailers)

ENCINA COMMONS

HOOVER MEMORIAL BLDG

CUMMINGS ART BLDG

ART GALLERY

EATING CLUBS

ARGUELLO WAY

ENCINA HALL

LOU HENRY HOOVER BLDG

HOOVER TOWER

PHYSICS LECTURE HALL

VARIAN

SEQUOIA HALL

CALIFORNIA

VIA PUEBLO

SEQUOIA LANE

Hansen Labs

GINZTON LAB

APPLIED PHYSICS

CENTER FOR INTEGRATED SYSTEMS

RESEARCH LIBRARIES GROUP

PANAMA STREET

VENTURA HALL

JORDAN WAY

SERRA ST

TO El Camino Real and Palo Alto

Escondido Village

PUBLIC SAFETY

FIRE STATION

TELECOM-MUNICATIONS UTILITIES

OPERATIONS & MAINTENANCE

CREDIT UNION

RECYCLING CENTER

ATHLETIC GROUNDS DEPT

DAY CARE CTR

FOOD SERVICES

PAMPAS LANE

CENTRAL STORES

MAPLES PAVILION

Football Practice Field

Varsity Tennis Courts

TENNIS CLUBHSE

San McDonald Hall

Angell Field

TRACK HOUSE

Harry Maloney Field

DE GUERRE POOLS AND COURTS

Sunken Diamond

EUCALYPTUS ROAD

CAMPUS DR

JOSEPHINE SIDING

Tennis Courts

OLD PAVILION

CALVEZ ST

MEMORIAL HALL

GRADUATE SCHOOL OF BUSINESS

The Oval

MEMORIAL WAY

DEPT OF ATHLETICS

ENCINA GYM

BAND SHAN

LAURENCE FROST AMPHITHEATER

GALVEZ HOUSE

HERRIN HALL AND LABS

Stauffer Bldgs

CHEM ENGRG

PHYS CHEM

ORG CHEM

ORGANIC CHEMISTRY

MUDD CHEM BLDG

OLD CHEM BLDG

u.c.

ROTH WAY

PLUM TREE LANE

LASUEN ST

MUSEUM WAY

PALM DRIVE

CAMPUS DRIVE

MUSEUM

ANATOMY

HEALTH RES & POLICY BLDG

LOMITA DR

RESEARCH ANIMAL FACILITY

FAIRCHILD CENTER

MAYER CANCER RESEARCH LAB

Stanford Medical Center

PASTEUR DR

TO Sand Hill Rd

EMERGENCY

FALK CVR BLDG

HOSPITAL MODERNIZATION PROJECT (u.c.)

ADMIN ANNEX

BLAKE WILBUR DRIVE

HELIPORT

WELCH RD

MAUSOLEUM

700 STANFORD BARN

730

750

770

780

800

851

900

QUARRY RD

Stanford Shopping Center

ARBORETUM RD

PALO RD

HOOVER PAVILION (Hospital)

CALVEZ ST

TO El Camino Real and Palo Alto Train Station

EMBARCADERO ROAD

EL CAMINO REAL (CALIF. 82)

TO Bayshore Freeway (US 101)

AUDITORIUMS and AMPHITHEATERS
numbered on map

1 — **Annenberg** — Cummings Art Building
2 — **Baldo** — Braun Music Center
3 — **Bishop** — Graduate School of Business
4 — **Braun Aud** — Mudd Chemistry Building
5 — **Braun Rehearsal** — Braun Music Center
6 — **Campbell** — Braun Music Center
7 — **Cubberley** — School of Education
8 — **Dinkelspiel**
9 — **Elliott**
10 — **Fairchild** — Fairchild Center
11 — **Frost**
12 — **Jordan** — Bldg 360, Quad
13 — **Kresge**
14 — **Little Theater** — Memorial Hall
15 — **Maples**
16 — **Memorial** — Memorial Hall
17 — **Nitery, The** — Bldg 590
18 — **Physics Lecture Hall**
19 — **Schultz** — Health Research & Policy Bldg
20 — **Skilling**
21 — **Stadium**
22 — **Stauffer** — Hoover Memorial Bldg
23 — **Sunken Diamond**
24 — **Terman** — Terman Engineering Center

TICKET OFFICES

25 — **Tresidder Memorial Union**
26 — **Department of Athletics**

The Stanford MARGUERITE shuttle service circles the inner campus every ten minutes on a two way loop. The service operates from 7:20 am to 6:10 pm, Monday through Friday. The Marguerite extends to the Palo Alto Caltrain depot during the morning and afternoon commute hours, and to the Stanford Shopping Center from 11:30 am to 1 pm. For more detailed information, call 497-4375.

The Santa Clara Transit District serves the Stanford campus. For specific information, call (415) 965-3100.

u.c. — under construction

Revised JUNE 1985
Prepared by Stanford News and Publications Service
STORES Stock No. 954401