Begin-BIG

An Approach to the Introductory Computing Course

John Motil

Computer Science Department
California State University, Northridge
9 1 3 3 0

Abstract

Begin-BIG is a way of introducing students to computing with a "big" application which the students first use and later modify. The application, which is big for the student, is actually a small version of a spreadsheet, database, equation solver, etc. One example discussed in this paper is SpreadPage, a spreadsheet of one page (26 columns and 50 rows) which involves big objects (rows and columns) and big actions (procedures acting on the rows and columns). This approach will also "End-Big" with students finally modifying or maintaining the application. In mid-semester the "Big" mind-set often helps the students appreciate many software engineering concepts.

Introduction

Programming can be introduced to students in many ways [1], [2], [4]; some ways may be better than others. This paper proposes a way which is consistent with recent developments in programming principles, software engineering, and pedagogy. It is called "Begin-BIG" for it begins with a big top-down view of applications which involve both bigger objects and bigger actions.

Begin-BIG can be viewed as the "systems" approach applied to computing. It is "Programming in the Large" at an early stage. It can also be viewed as a variant of Top-Down (where students especially have problems finding out where the "top" is).

History

Historically, introductions have evolved from low levels to high levels; we have come far in a short time, but perhaps not far enough. In the distant past the introduction was done mainly from a very low level; "in the beginning was the machine". The details of the machine and the structure of the assembly language were at such a low level as to obstruct the structure of the algorithms.

Tony Hoare [3], after having created Quick-Sort, found it extremely difficult to describe it to others at a low machine level. Written in a higher level language, it became much easier to describe, analyze and modify.

At present, programming is usually introduced at a mid level using a language such as Pascal. Unfortunately the level often drops quickly for various reasons. One reason is that instructors often feel pressure from students to get onto the computer early, and so they begin programming prematurely. Students begin to write before they have ever read or used any programs. As a result, what they write is very trivial (details of syntax and bottom up preference) and a waste of time. Even worse, it may instill bad habits (such as using only global variables and parameterless procedures) which must later be unlearned.

This paper describes an approach which can be called "Begin-BIG" as opposed to the previous "begin at the bottom". It begins at the top and, most importantly, remains at that level until the most significant concepts are instilled. It still allows students to get to the computer early, but not to program at a low level, or intermediate level, but at a very high level. Students learn early to use application modules involving large data structures; then later they actually create such structures and modify them. So this method begins big and ends big with smaller details sandwiched between.

Begin-BIG can be done in many ways. A survey of a large field is one way, but it often lacks substance, and students lose motivation quickly. Selecting a computer application is another way which will be described here. Students can easily relate to the use of an application seeing the big-picture without getting bogged down in details too early. It encourages a bird's-eye view rather than a worm's-eye view. The details will come later.

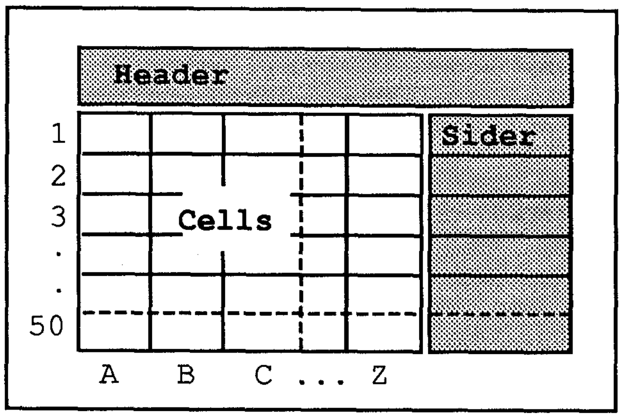
SpreadPage: One "Big" Application

Many application programs having large structures of data and action are possible; only a few will be described here. The most successful application structure used so far was a simple spreadsheet application called "SpreadPage" to put it into proper perspective.

SpreadPage is basically a "four-function" spreadsheet which illustrates many principles and practices without overwhelming students. It does not assume previous knowledge of spreadsheets and does not take long to understand and use. The entire application is described on just a few pages, and is implemented as a Library in Modula-2 so all the details are hidden. Students can use this application package easily on any platform including Apple Macintosh, IBM PCs or clones, Vax, Atari, Amiga, etc. The application is entirely portable and runs the same on all these platforms!

Structure of SpreadPage

SpreadPage has the following form, consisting of a Header, a grid of cells, and a "Sider".



Cells of this spreadsheet hold integers only. Text is allowed only in the one-line Header, and a Sider following the last column. These restrictions can be relaxed later.

Inventory of chair parts with columns representing their price and quantity is shown on the given SpreadPage. A third column shows the Total cost of each part (by multiplying price and quantity). The sum of this Total column yields the grand TOTAL cost of $10,000.

	Price	Quant	Total	Part
1	10	20	200	Backs
2	30	40	1200	Legs
3	50	60	3000	Rungs
4	70	80	5600	Seats
5		200	10000	TOTAL
	A	B	C	

Objects of the small spreadsheet (SpreadPage) are reasonably "big" objects which are rows and columns. The spreadsheet is limited to one page of 26 columns (labelled by the letters of the alphabet) and 50 rows (to fit one page). The values in each cell are initially integers; later they can be extended to real values.

Actions on SpreadPage

Actions on the rows and columns are procedures which are imported from the Library SpreadLib. The high-level actions include procedure calls such as the following:

`SumAColumn(C1,R1,R2,C2,R3)` sums the values in column C1, from rows R1 to R2, and puts the resulting value into the cell in Column C2 Row R3. Usually columns C1 and C2 are the same.

`AddColumns(C1,C2,R1,R2,C3)` adds the two columns C1 and C2 from rows R1 to R2, and puts the resulting sums into column C3.

`SumOfARow(R1,C1,C2,C3,R2)` sums the cells of row R1 from column C1 to column C2 and puts the results in the cell of Column C3 and Row R2.

Other actions such as MaxAColumn, MinOfARow, AddTwoRows and MeanColumn are similar. Also included are Initialize (to set the column width), LoadSpread (to load from a file), ShowSpread (to display it on the screen), and SaveSpread (to save it to a file).

Objects of SpreadPage

Data of the spreadsheet consists of a two-dimensional image of integers in the form of a file which can be created by any editor. An example of an inventory SpreadPage follows:

```
2 Columns
5 Rows
Price Quant Total Part
   10    20       Backs
   30    40       Legs
   50    60       Rungs
   70    80       Seats
    0     0       TOTAL
```

The first two lines of text indicate the size; first is the number of columns followed by the number of rows. The third line is a header, a single string of descriptive text. This is followed by an array of integers of the above specified size, each row followed by a "Sider" of descriptive text.

Programming a SpreadPage consists simply of creating a Module which imports the required procedures from SpreadLib and calls them as follows:

```
MODULE SpreadProg;
(* Chair Inventory System *)

FROM SpreadLib IMPORT Colname,
   Initialize, ShowSpread,
   LoadSpread, SaveSpread,
   MulColumns, SumAColumn;

BEGIN

   Initialize;  LoadSpread;
   SumAColumn( B,1,4,B,5 );
   ShowSpread( A,C,  1,5 );
   MulColumns( A,B,1,4,C );
   ShowSpread( A,C,  1,5 );
   SumAColumn( C,1,4,C,5 );
   ShowSpread( A,C,  1,5 );
   SaveSpread( A,C,  1,5 );

END SpreadProg.
```

Running of this application consists of executing the Modula program, which in this case is:

```
Enter Column Width 6
Enter source file: Chair.in

Price Quant Total Part
   10    20     0 Backs
   30    40     0 Legs
   50    60     0 Rungs
   70    80     0 Seats
    0   200     0 TOTAL

Press any key to continue
```

Continuing in this way the program provides a trace, stopping at each ShowSpread to display the page and ask "Press any key to continue". The SaveSpread action saves the spreadsheet to a file, which can be "beautified" in any text editor or word processor.

Effects and Outcomes

Students quickly learn to think big in terms of big objects: rows and columns. They also learn to think big in terms of procedures; they only use them now, but later they create such procedures. They think "Big" in terms of both objects and actions; they become object-oriented as well as control or action oriented.

Students can easily create their own personal spreadsheet applications which can be useful, including:

keeping track of their income/expenses
balancing of checkbooks
analyzing gas mileage
monitoring their performance in sports
and others in the following appendix.

Also, I use this SpreadPage instead of a "professional" spreadsheet for keeping class grades. In mid-semester I give each student a copy of his/her particular "row" of the spreadsheet. This serves as a check on whether I have entered the proper percentages. Students often suggest interesting actions on these grades (drop the lowest one, weigh the exams, etc). They are motivated to suggest many extensions of this application.

At the same time that students are using this spreadPage application they can be learning how to use an operating system, and how to use an editor to create modify and store files.

Using an application provides students the viewpoint of a user and they can be learning about how an application communicates with a user; how user friendly it is. They could be getting ideas of how to improve it later.

Programming is another thing that students are learning at this time, but it is programming at a very high level. The program consists of simply importing the required resources (both actions and objects) and then specifying various procedures to act on the objects. Control structures such as Loops and Choices are not necessary at this stage (but are possible). The syntax of all these high-level programs is extremely simple, consisting of a Header statement, an Import statement, and a block or series of procedure calls all separated by semi-colons.

Maintenance of the existing Implementation Module could be a significant learning project. SpreadPage could be modified in many ways as suggested by "real" spreadsheets, including:

changing the type to Real values
allowing the column widths to differ
placing a Sider to the left of the grid
comparing values and indicating +,0,-
sorting or ranking of values
deciding, with an IF construct
plotting (bar plot, pie plot, etc)

SpreadPage differs from "real" spreadsheets in some rather significant ways. Firstly, it is compiled rather than interpreted. This could provide considerable power and also speed. Secondly, it is strongly typed; all values are of the single same type (Integer, Real, etc). Other differences are minor.

Readability of SpreadPage programs can be improved easily by using meaningful names (such as Price and Total) for columns instead of the letters A, B etc. This is done by declaring constant names as shown in the following SpreadPage program which describes a trip with mileages, costs and gallons of fuel used.

```
MODULE SpreadPage;
(* Mileage and cost for Canada trip *)

FROM SpreadLib IMPORT ColName,
   Initialize, LoadSpread, ShowSpread,
   SaveSpread, SumAColumn, SubColumns,
   DivColumns, MeanColumn, MaxAColumn;

CONST (* A Renaming of the Columns *)
   Last = A;     (* The last refill *)
   Next = B;     (* The next refill *)
   Galn = C;     (* Gallons used up *)
   Cost = D;     (* Cost of fuel $$ *)
   Mile = E;     (* Miles travelled *)
   MPG  = F;     (* Mile per gallon *)
   CPG  = G;     (* Cost per gallon *)

BEGIN
   Initialize; LoadSpread;
   SubColumns( Next, Last, 1,23, Mile);
   DivColumns( Mile, Galn, 1,23, MPG );
   DivColumns( Cost, Galn, 1,23, CPG );

   SumAColumn( Galn, 1,23, Galn, 24   );
   SumAColumn( Cost, 1,23, Cost, 24   );
   SumAColumn( Mile, 1,23, Mile, 24   );

   MeanColumn( MPG , 1,23, MPG , 25   );
   MaxAColumn( CPG , 1,23, CPG , 26   );
   ShowSpread( Last,  CPG, 1   , 26   );
   SaveSpread( Last,  CPG, 1   , 26   );
END SpreadPage.
```

A run of this program is shown in the following appendix. There the output has been "beautified" in an editor. Notice that the Cost-Per-Gallon, or CPG, is almost twice as high in Canada as it was in the USA. Notice also that there is a simple way to check some of the results. For example, the total miles travelled is the sum of all 23 miles in the fifth column, and is also the difference between the first and last mileages.

Between the beginning and the end of a course this big application is also useful to discuss and illustrate software engineering principles. Students may not appreciate procedures, modules, parameter passing, coupling, cohesion, information hiding or avoidance of global variables when programming in the small but do begin to appreciate these in the context of software engineering of such large systems.

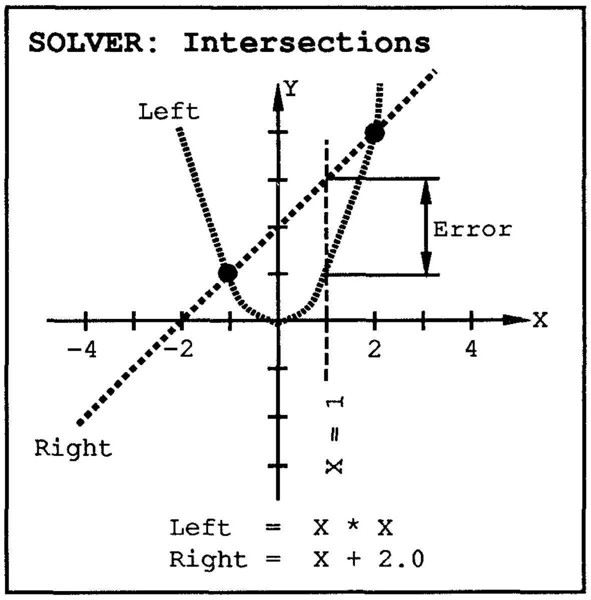
Other "Big" Applications

Retriever, a tiny data base, is yet another application which can be used to Begin-BIG; it is somewhat simpler than SpreadPage but not as motivating.

Retriever is a simple linear search program which retrieves any data from a file that students have created. For example, students could create a file of their friends' phone numbers and then ask for all friends in a given area code, or zip code, or having the name "Bill", etc. Other small "data bases" of personal use to them include their lists of music, books, movies, and other "collectibles".

Tiny data bases of even a page in size are quite useful to students. At first just entering such data is a useful exercise in using the operating system, using the editor and even typing. Later such a Retriever application can be modified to create labels, to make reports and to do logical searches (involving AND, OR, and NOT operations).

Solver is another "big" application program which is more successful with students who are oriented to science and engineering. Solver aids in the solution of equations. The method involves use of random numbers. Basically, two equations are given in one unknown X as shown in the following diagram.



The user specifies a range of values and the program simply tries many random values within the range and evaluates the two equations storing the minimum distance between the two equations.

For example, a random value of X = 1 provides values of Left = 3 and Right = 1 for a difference or error of 2 as shown. The range may be shortened and this process of trying many values and finding the minimum difference (or error) could be continued to obtain any degree of accuracy.

Solver can also be used to find the roots of an equation by having one equation set to 0. Solver can similarly be used to find extreme values of an equation.

Applications other than SpreadPage or Solver can also be used to Begin-BIG. They include: Calculator, Typer, CalendarPlanner, Grapher, Plotter, Driller and Flasher. These are described in a forthcoming book to be published by W.C. Brown [4].

Conclusion

"Begin-BIG" helps students to see the big picture and to begin at the top. This approach can also "End-Big" when this application is programmed or modified at the end of the course. Last, but not least, the application can often be referred to during the course to illustrate many software engineering concepts.

- - - - -

Disks containing this Spreadsheet library are available from the author in both IBM and Macintosh versions.

Bibliography

[1] Baxter, N. and Hastings, D. et al, "Introduction to Computer Science: An Interactive Approach Using ISETL", SIGCSE 21st Technical Symposium on Computer Science Education, SIGCSE Bulletin, Vol 22, No. 1 (February 1990).

[2] Denning, P.E., D.E. Comer, D. Gries, M.C. Mulder, A. Tucker, A.J. Turner, and P.R. Young, "Computing as a Discipline", Communications of the ACM, #2, 1 (January 1989), 9-23.

[3] Hoare, C.A.R. "The Emperor's Old Clothes", Comm. ACM 24(2), 75-83 (February 1981).

[4] Motil, John, "Programming Principles and Practice with Modula-2", W.C. Brown, 1990.

Appendix: SpreadPage of TRIP (Dashed lines separate the initial input data)

```
TRIP MILEAGE and COST        ¦        (1989 Costs)
                             ¦
Start  Finish   Galn  Cents ¦Miles   MPG    CPG     Place of Refill
 26430 26731     10   1182 ¦ 301     30    118     Northridge     CA
 26731 26981     11   1300 ¦ 250     22    118     CasaDeFruta    CA
 26981 27190      9   1200 ¦ 209     23    133     Willows        CA
 27190 27415      9   1050 ¦ 225     25    116     Ashland        OR
 27415 27632     12   1500 ¦ 217     18    125     Arlington      OR
 27632 27881     12   1435 ¦ 249     20    119     Cheney         WA
                            ¦
 27881 28140     10   1710 ¦ 259     25    171     Fernie         BC
 28140 28298      7   1150 ¦ 158     22    164     Blairmore      AB
 28298 28575     11   1725 ¦ 277     25    156     Lethbridge     AB
 28575 28864     12   1935 ¦ 289     24    161     Blairmore      AB
 28864 28976      4    650 ¦ 112     28    162     Hillcrest      AB
 28976 29219     12   2150 ¦ 243     20    179     Banff          AB
 29219 29383     10   1665 ¦ 164     16    166     Bellevue       AB
 29383 29679     12   1890 ¦ 296     24    157     Blairmore      AB
 29679 29937     10   1620 ¦ 258     25    162     Creston        BC
 29937 30170     10   1530 ¦ 233     23    153     Sardis         BC
                            ¦
 30170 30356      8    900 ¦ 186     23    112     Bellingham     WA
 30356 30665     13   1400 ¦ 309     23    107     Eugene         OR
 30665 30875     10   1200 ¦ 210     21    120     GrantsPass     OR
 30875 31117     12   1400 ¦ 242     20    116     Orland         CA
 31117 31338      9   1000 ¦ 221     24    111     SanFrancisco   CA
 31338 31556     10   1338 ¦ 218     21    133     KettlemanCity  CA
 31556 31755      9    988 ¦ 199     22    109     Northridge     CA
----------------------------+-------------------------------------------
     0      0    232  31918 ¦5325      0      0     TOTALS
     0      0      0      0 ¦ 231     22    137     MEANS
     0      0      0      0 ¦   0     30    107     EXTREMES
```