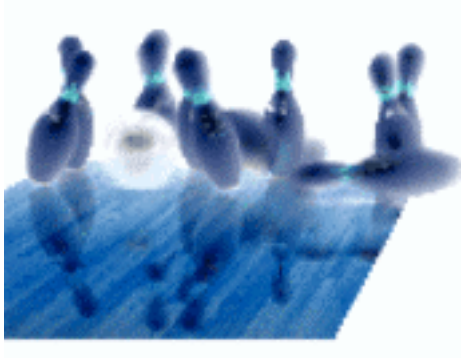
CS 2574 Spring 99
Project Specifications
Bowling Analysis Manager



Bowling Analysis Manager (**BAM**) is an interactive software tool that would be used by Bowlers for the management of their bowling game database. The program is designed to allow stored games to be browsed, created, deleted, modified and viewed hierarchically. BAM is menu-driven single-mode system. In this first graphical interface version only browsing, (viewing and navigation), and limited searching operations to the BAM data file will be provided.

The user will traverse the stored bowling game data linearly and by searching for a bowling alley, event or date of a game. The discussion that follows describes the BAM interface and operations in detail. Deviations from these specifications may *not* result in grade penalties. However, developers would be well advised to receive prior approval before altering any of these requirements.

Discussion

BAM will begin execution with the display of the BAM interface screen as shown in figure 1 on the following page. The BAM data fields will be blank initially. Alternatively, a startup/splash/about screen window giving the usual information about the system and developer, along with the current product version number could be displayed . The startup screen window may be presented in a clever manner in order to capture a viewer's attention. After the user closes the startup screen window, the BAM menu commands will be available to the user.

At startup, command line parameters are accessed to open the optional *bamfile.bam* command line file. The *bamfile* file name, if present, follows the BAM program name on the operating system command line: "*bam bamfile*". A file existence check for any file opened is required and an error message displayed if the file is not present in the current directory or specified file name path. The user may elect to start BAM without any command line parameters. In this case the user must first open an existing *bamfile.bam* file before most BAM menu commands become available. (For an complete explanation of the *bamfile.bam* file, bac file and bal file formats see: http://ei.cs.vt.edu/~cs2574). The remaining discussion will be made with regards to the BAM interface screen and how the system execution affects its contents.

The BAM screen will be divided up into several areas: the menu area, display area, I/O area and the status area. The contents of these areas may change during execution. Only two of the areas, (menu line and display area), are required. The menu area will hold the names of the pull-down containing all BAM system commands. The display area will be used to show a single bowling game information. Note – not all of the BAM game information will be made available during this initial development effort. The indexes of other sections stored in the currently displayed game record will be used for the display of related game information. E.g., the Adex of a game record will be used to locate the alley record with the same alley index in the alley list and display the name of the bowling alley where the game was rolled. Likewise the current game record's conditions index is used to display the related Block field for the oil pattern. (The Block field character codes are: **C**rown, **B**lock, **R**everse, **T**ight, **L**oose, **U**nknown.) The ball and lift display area text are determined from the first game frame record. The first fame record Bdex and Rdex values are used to locate the corresponding ball and release section records. The Model and Lift fields from each of these sections are then displayed. (The Lift field character codes are: **N**one, **T**ouch, **L**ight, **M**edium, **H**eavy, ma**X**.) The Prev and Next buttons in the display area will function analogously to the Prev and Next options of the Games menu described later.

The Input/Output (I/O) area, (optional area near the bottom of the screen), is employed to give error/warning or prompt messages and to accept keyboard input. Alternatively, the I/O area may be replaced by dialog boxes for user inputs and alert messages. The status line, if implemented, will contain 2 fields: 1. the name of the current open BAM bowling data file, left justified; and 2. the number of the currently displayed game record divided by the total number of games in the BAM file, (e.g., Game: 3/12).

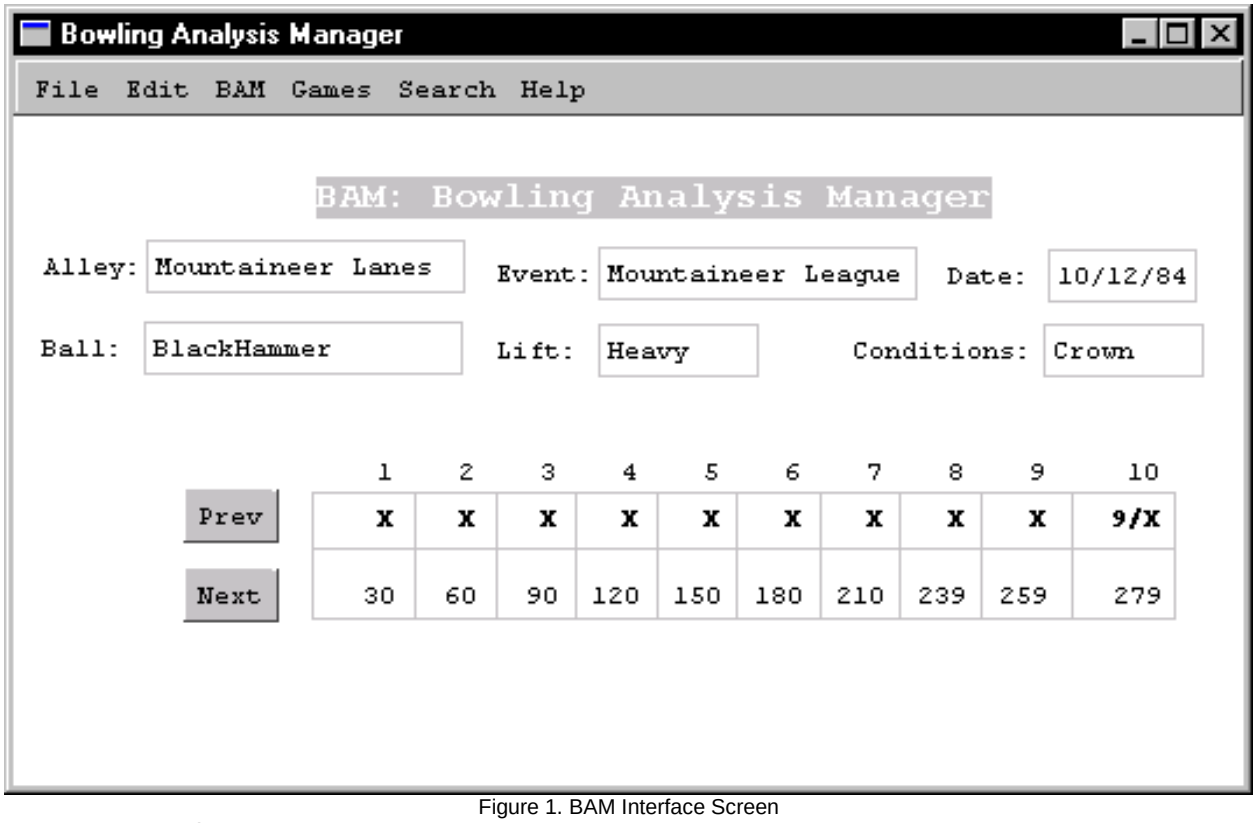
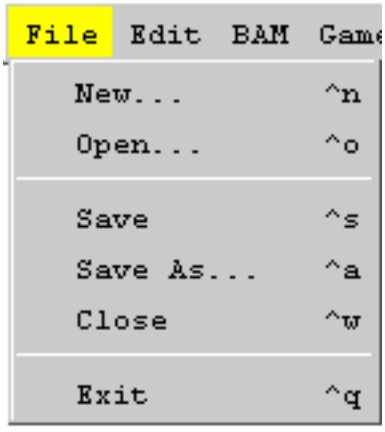


Figure 1. BAM Interface Screen

Menu Commands

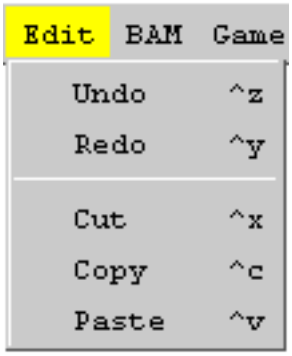
This section explains the functionality of the pull down menus and how they will operate in BAM. Following this general menu operation explanation, the rest of this section will discuss each pull down menu and each of the commands that they contain. Not all menus and menu options are functional all the time. Whether a particular option is functional is dependent upon the current contents of the display area. For example, initially after startup with no open file the only menu commands that would be available would be the be the New…, Open…, Exit options of the File menu and the implemented options on the Help menu. Some menu options will have an associated accelerator keyboard shortcut associated with it for fast selection without using the mouse. The accelerator keys are denoted as "^k" in the menus and are accessed with the control key. Note that in this initial version of the system some of the menus, (Edit and BAM), are not functional, (or selectable). They will be included as placeholders for future development. In addition, some options are also unimplemented in this initial development effort and also included for future development. These disabled options are given in the following menu section explanations.

File



Two options on the file menu, New… and Save, are not implemented in this initial version of the BAM system. The Open… option will prompt the user for the name of a *bamfile.bam* file. The *bamfile* is checked for existence and opened and processed. This involves building double-linked lists of all of the section record data, ordered upon the section indexes. This list is identical to the list built in the Bowling Analysis List, (BAL), program. (For specifications of the BAL program see: http://ei.cs.vt.edu/~cs2574/.) As the *bamfile.bam* file is processed, a report file, (*bamfile.bac*), containing any encountered errors is built. If the file does contains errors, a warning message should be given instructing the user to view the report file externally. The Open command should prompt the user first to see if they wish to close any current open BAM file. (If the user decides not to close the current BAM file then the file menu is exited with no action taking place). If the user does wish to close a current BAM file then the close command is automatically executed and the user is prompted for the name of an existing BAM file to open. The Save As… option prompts the user for the

name for a new BAM file, which is then created and written to the disk. The created BAM file will contain the current contents of the internal BAM list structure in the BAM file format, becoming the current open BAM file, (causing a status line update). If the user enters a name of an existing file the Save As… option over-writes any other version of a file that might exist in the directory without checking or warning the user. The Close command will simply clear the display area fields and destroy the BAM list structure. The Exit command will prompt the user to confirm that they wish to quit BAM, ( ), destroy the list structure and close any open files and resources, before shutting down and returning to the operating system.



Edit

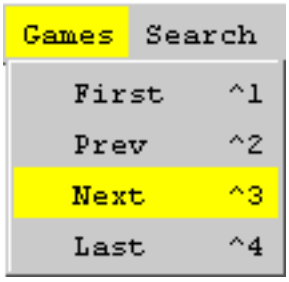
The **E**dit menu is an inoperable menu in this initial development of the BAM system. In later development work it will contain standard clipboard and operation history recovery support.

BAM

The **B**AM menu is also an inactive menu in this initial system version. In later versions, it will contains options for manipulating and managing BAM records.

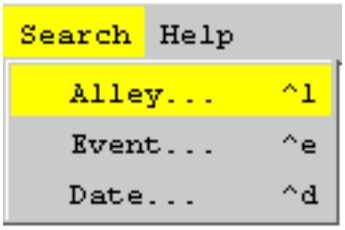
Games

The **G**ames menu allows the user traverse the BAM bowling game data records. The First and Last options will display the game record data that is at the head or the end of the game section list respectively. The Prev and Next options will display the preceding or following BAM bowling game data record that is adjacent to the current displayed game data record. If the user is at the beginning of final record in the list and attempts to move to a nonexistent record a beep should be sounded. All of these options should result in a status line update of the current displayed record number.



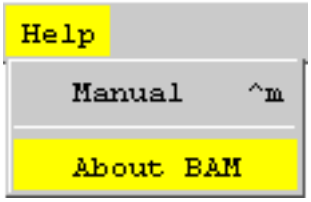
Search

The **S**earch menu contains three options. Each option will allow the user to enter a corresponding string. BAM will then hunt for the string in all of the stored BAM game section and indexed linked records for the string in the corresponding field. If the entered target string has a length of N characters, this will involve a string comparison to match the first N characters of corresponding field game section list entries. If a match is encountered then the game record containing the matched entry or linked to the matched record becomes the current display record, resulting in an update of the display and status area. Non-matched titles should result in a warning message. BAM is only responsible for displaying the first located match. If multiple matches are possible BAM is not responsible for determining that they exist.



Help

The **H**elp menu contains one inactive option, the Manual option and one active option. The about BAM option will display the initial startup/splash/about screen window giving the common information about the system and developer.



GUI Library

The use of a GUI Library is required for this program. Partial credit will be given to implementations that do not use a GUI, but penalties will be incurred. The ability to define windows and menus will greatly aid the task of screen content management. The restoring of the display area's contents after displaying pull-down menus and switching between different display area contents will be made simple. The GUI windowing facilities will greatly enhance the professional look-and-feel of the program.

For students coding a non-GUI interface, the interface specifications described above must still be adhered, with the following modifications. The display area will be lines 1-22, the menu bar on line 23, the status area on line 24 and the I/O area on line 25. When a user selects a menu, the complete screen

should be redrawn with the menu area line, (23), replaced by the menu commands. The user will be allowed to select a menu command by hitting the first letter of the command. A non-text windowing interface will require the creation of a window ADT class and menu ADT class for controlling the screen contents.

List Class

The implementation of a C++ double linked list Class ADT is also a requirement for this program. Partial credit will once again be given to implementations that implement a non-class list ADT, but penalties will be applied.

A specification addendum for this project may be forthcoming shortly if required. The addendum will describe minor enhancements or changes to BAM.

PROJECT GRADING and DOCUMENTATION

Successful completion of this software development and implementation project will result in the following external and internal specification documents. Submission will be made to the auto-acceptor in a zip archive. The project number for the acceptor for this project will be “bam”, typed in lowercase, omitting the quotes.

1. A complete **STRUCTURE CHART** of the final system's design, including all interface specifications stored in a single MS Word document named, “schartp3.doc”. The structure chart must be of sufficient detail to communicate to others WHAT functions are needed to implement this system and how the various parts are interconnected. The structure chart is complete if the algorithms for each module could be developed from it, along with a Data Dictionary, although no data dictionary is required. Use only 8 1/2" X 11" sized pages to draw the chart upon, DO NOT use larger sized pages. Use the symbols described in class for the components and the interfaces. Legibility counts and points will be deducted for sloppily presented work. Careful and CORRECT notation is also required and points will be deducted for a poorly annotated chart. Note — calls to standard C/C++ system routines: cin, cout, etc. and GUI library functions need not be annotated on the chart.

2. An **INTEGRATION PLAN**, that is a plan for HOW and WHEN the various phases of the project will be developed and which modules will be added at each integration point. THREE integration points must be given. The points reflect a semi-equal division of the project work.

a. For each integration point, you must name the TASKS that will be completed by the identified date and clearly define WHAT BEHAVIOR can be expected to be exhibited by your program at that point.

b. The three integration points for the system are: *1. 2/22/99 2. 4/5/99 3. 5/3/99*. Each student will be expected to demonstrate to a GTA the behavior of their project at the second and last integration points.

c. At any integration point or when the program is submitted, any non-functioning commands, actions, etc. of the system are not expected to bomb the system. A message should be displayed informing the user that this operation has not currently been implemented or is still under development. On the final submission, all non functioning aspects of the system MUST be accompanied by a short explanation describing the suspected problem(s).

3. A **USER MANUAL** generated by the word processor of your choice must be written. The manual should enable the novice BAM user to access the features of the system. This document should be at approximately 6 pages in length, excluding the title page, table of contents, index, etc. The user manual must be written from the perspective of a novice BAM user, not from a programming implementation point of view.

QUALITY OF CODE

It is expected that the code will be WELL-DOCUMENTED, appropriately indented and VERY READABLE. Points will be deducted for poorly presented code. Individual functions, excluding their documentation and declarations may be no longer than one page. Each compilation module should contain only related functions. Function headers should resemble the following template:

```
/*******************************************************)
(*FUNCTION NAME                                        *)
(*DESCRIPTION OF FUNCTION                              *)
(*DESCRIPTION OF ALGORITHM : FUNCTION IMPLEMENTATION   *)
(*                                                     *)
(*CALLED BY: (LIST OF FUNCTIONS)                       *)
(*CALLS: (LIST OF FUNCTIONS)                           *)
(*                                                 *)
(*PARAMETERS: NAME AND ROLE IN ALGORITHM OF EACH  *)
(*                                                     *)
(*AUTHOR: name of author                           *)
(*REVISIONS: DATE, REASON                              *)
(*           AUTHOR FOR EACH (if different)            *)
(*VERSION: x.xx                                        *)
(*******************************************************/
```

Each of these documents must be placed in A separate MS Word file included in the submitted archive. Each separate document must be obviously named.

The following scale will be used for grading:

PRODUCT	PERCENTAGE OF PROJECT GRADE	PERCENTAGE OF FINAL GRADE
Structure Charts:	20	5.00
Integration Plan:	10	2.50
Code/Execution:	60	15.00
User's Manual:	10	2.50
Total:	100%	25.00%

Due Date schedule:

There will be NO extensions or late submissions for this project! Any project or portion thereof NOT submitted on time will be rejected!

PRODUCT	DATE
Structure Chart:	05 - 03 - 99
Integration Plan:	05 - 03 - 99
Code/Execution:	ALL work is due by **11:59:59PM** on **05-03-98**.
User Document:	05 - 03 - 99.