A Graphical Interface to Multi-tasking Programming Problems

Nitin Mehra[1]
UNO P.O. Box 1403
New Orleans, LA 70148

and

Ghasem S. Alijani[2]
Graduate Studies Program in CIS
Southern University at New Orleans
6400 Press Drive
New Orleans, LA 70126

Abstract

The foundation of this project lies on the basis of the multi-tasking environment, at the operating system level. An important consideration taken into account is the ability to run platform-independent programs using a common graphical user interface. The application (named 'RunApp') is developed using Microsoft Visual Basic. It is based on the concept of linear programming in a multi-development environment. Currently, this application incorporates the Visual C++ and Visual Basic programming environment by providing editing, compiling, and execution capabilities. RunApp possesses the capability of searching web-oriented resources, which will aid developers to find information about different environments through the Internet. This tool will also help the developer in downloading and updating the latest controls, modules, DLLs, etc. directly into their respective environments such as Visual Basic or Visual C++. This application provides all the editing features of any basic word processor. RunApp shares the personality of a development environment, debugging tool, information access, and application execution module. It is a standalone interface that couples multiple programming environments into one entity with capability of multi-tasking.

Keywords: Multi-tasking, shared memory, error detection, fault tolerance, dynamic link library

1. INTRODUCTION

Microcomputers have become remarkably faster over a short period of time and large networks of microcomputers are cheaper than mainframe computers. Today's computer systems are extremely complex when viewed in all their detail (Brookshear 1997). The increase in the number of computers accompanied by the increase in the number of users has led to the need to have simple and standard user interfaces. The graphical and visual representation of an environment enhances productivity and speeds up development time. Moreover, many administrative tasks are made easier by using graphics instead of plain text to display information (Whitehead and Maran 1997).

An environment is the actual "feel" (available resources, memory variable values, data values, etc.) of the situation that describes a particular process at any given point in time. An environment encompasses tools, database or files, people, hardware, a network, operating systems, standards, and myriad other components. Concurrency is the simultaneous execution of several processes within a system. In a centralized system, interleaving the execution of each process simulates concurrency. The shared-memory architecture provides a very efficient medium for processes to exchange data

[1] nitinmehra@bigfoot.com
[2] dallijani@ix.netcom.com

(Geist, Beguelin, Dongarra, Manchek, and Sunderam 1996).

Most physical user interfaces to electronic systems are tailored to the greatest common denominator of human experience rather than a multiplicity of human experience. A few interfaces, on the other hand, are dynamically tailored to an individual person's needs (Eustice, Lehman, Morales, Munson, Edlund, and Guillen 1999). Most typical development environments work within a uni-tasking mode limiting the users to one programming language.

Intelligent Graphical User Interfaces can simplify a lot of complex processes and reduce the actions that one needs to perform to execute tedious tasks. The objective of this project was to develop a graphical user interface (GUI) that addresses the multi-tasking problems. In many cases, overcoming the hurdles requires the determination of conflicting processes between events in the system. Since perfect implementation and functionality is impossible, several debugging options need to be included.

2. METHODOLOGY

There are many challenges to be resolved with the implementation of an application developer system. The first challenge is the selection of an operating system that allows the user to be ignorant of the multiple resources involved. The application, named RunApp, should be able to handle multi-tasking and intelligent resource management that is supported by Microsoft's Windows operating systems. Some of the other hurdles include implementation of the following.

- Error detection and error handling
- Proper resource allocation
- Assurance of data consistency and
- Providing a fault tolerant system

One of the key characteristics and challenges in the design of this system is the multi-tasking operating system. The RunApp system is the software layer on the operating system that simplifies the task of programming. This system appears as a standard application program, but it runs at the kernel level supporting multiple environments.

Two of the other major concerns in the design of the current system are error detection and error handling. In a multi-environment system, when one or more components (such as memory, client, processes, kernel, etc.) crashes, the failure is fatal. Usually this failure is instantly detected and can be handled appropriately. In single environment system as in the case of current system, error detection and error handling are much more complex.

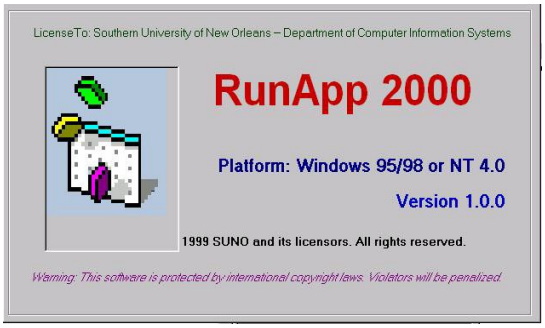

Figure 1. Splash screen

The splash screen as shown in figure 1 is the first screen that introduces the application and provides the user with the version information. This screen stays up until the entire application is loaded in the background. Once RunApp is loaded into memory, the splash screen disappears from the screen and unloads from memory. The next screen that is loaded is the user menu within the application module. As can be seen in figure 2, this is a more of a startup screen that enables the user to pick an option/task that he/she would desire to perform. A convenience advantage of RunApp is that once the user selects a task, it would directly perform that task without having to navigate through all other menus and options. The 'Exit' button on this screen only exits the menu and not the application. If the user exits, he/she would be brought to the main screen as shown in figure 3. This would also directly load Visual Basic or Visual C++ if the user desires that option and the corresponding application have been pre-loaded on the computer. The 'Run a program' option allows the user to directly execute a pre-compiled executable file that has been created by him/her. The 'new' file option opens a new source file in which the user can write code for C, C++, or Visual Basic.

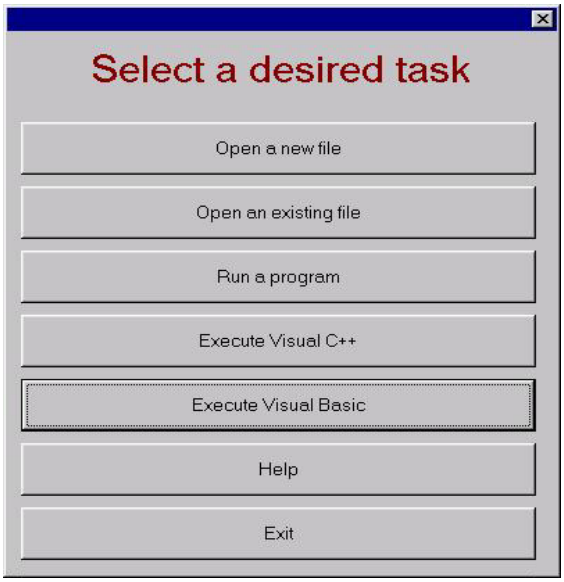

Figure 2. User menu

Figure 3 depicts the main screen of the application. The entire RunApp environment/application is contained within a single process. A process is a program in execution (Silberschatz, Abraham, and Galvin 1998). All processes in the development environment belong to the parent process, which is 'RunApp.' Every sub-process (thread) will show as a standard task in the window's environment (under windows task manager) and will be assigned with a unique task id by the operating system, since it is a new parent process initiated by the RunApp. A process encompasses the current status of the activity, called Process State (Microsoft Corporation 1999a). Multiple processes of RunApp can be initiated at the operating system level (since Windows '95 supports multitasking) and each process will have its own environment, allocated memory and be completed independent of any other task. This is accomplished by the way that the RunApp id is developed under a Single Document Interface (SDI) environment.

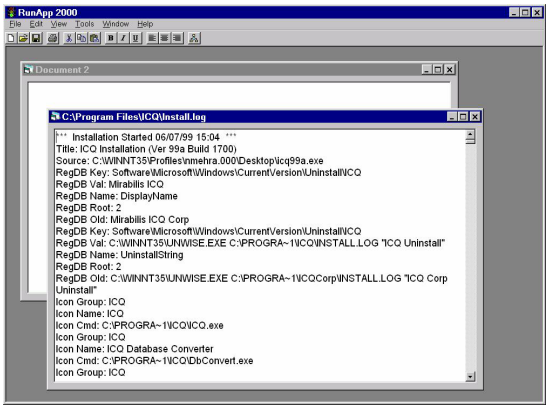

Figure 3. Main screen

The main screen contains the menu bar, toolbar, and status bar that follow all the rules and aspects of a standard window's application. All the rich-text editing features designed within this application are contained in the toolbar along with other file and print functions. All documents opened for editing are displayed within this main window. The option to view and edit them in rich-text format is left to the user but the source files will only be saved in text format. Any new application or development environment is spawned as a new process/task at the operating system level and can continue to execute independent of RunApp, if desired.

RunApp lets the user open any source file (figure 4) from a local or network drive and load it into a rich-text editing environment. These functions can be performed from within RunApp. Multiple files can be edited and compiled simultaneously. The current file versions supported are: C source files, C++ source files, Visual Basic modules, Visual Basic projects and Visual C++ files. Using a file pointer, which points to a structure that contains information about that file (Kernighan and Ritchie 1996), accesses C files. A similar method has been adopted to access the other types of files. Other file formats can be added to this application dynamically or by adding the filters to the Visual Basic code of this module. One can also cut and paste text and data from one file to another in the same manner as it is done in any other word processor. Similar functionality was developed to save information, set the font, view help topics as well as to print source code or program output. All modules have been developed using Microsoft's Windows Application Programmer's Interface (Windows API) and have been modified to suit this application by adding and customizing its features to comply with the file formats for RunApp.

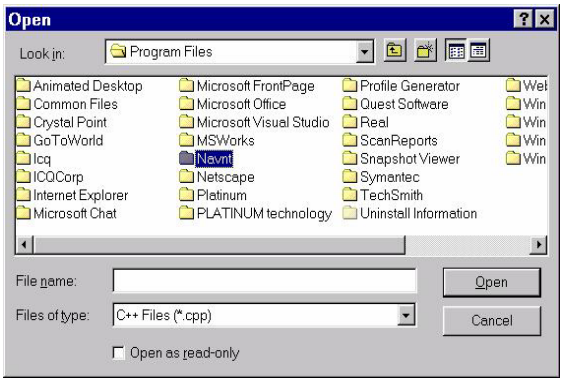

Figure 4. Open file window

The Edit menu provides the user with the functionality to copy, cut, and paste text between source files or to the windows clipboard. Users can also cut and paste text between environments as Visual Basic and Visual C++. This menu also contains the font option, which has been described earlier. All the functions contained in this menu are identical to the ones in any other window's application. In this manner, the interface can be kept simple to the user by avoiding complexity.

The available 'Environments' are shown under the 'Tools' menu (currently – Visual Basic and Visual C++). The menu, when activated, will load the appropriate development environment if and only if it has been pre-loaded on the machine being used. The installed development environments (i.e., Visual Basic or Visual C ++) can be used to compile and create executable files of any source code written via RunApp. All the memory management, context switching, and environment variables are loaded in the background. The user will be provided with the option of directly loading the source file in the selected environment. The executable file can be initiated/run by selecting the 'Run Program' menu. This menu also contains shortcut keys used to initiate the environments directly without having to navigate. The menu provides the user with the list of all the programs that are available and executable. The user-created executable file must have a postfix indicating as an executable file to be run via RunApp. Time-sharing used in a multi-user system is known as multitasking, in reference to the illusion of more than one task being

performed simultaneously (Brookshear 1997). This will execute every program in an independent multitasking environment and multiple programs can be executed simultaneously. Each program will have its own instance, and hence, multiple copies of the same program can be executed at the same time and compared for outputs. The source files can be debugged by going back to the appropriate environment and recompiled. An example has been described later in this paper for running Visual Basic and Visual C++ simultaneously.

RunApp invokes either the Visual Basic or Visual C++ environment when the corresponding selection is made from the Tools menu. Visual Basic will provide the user to start up any new type of application project as desired. This option will only be provided if the user initiates the VB environment without supplying an existing source file, VB form, or module. The same kind of process is followed if the Visual C++ environment is invoked.

Figure 5 depicts an example where RunApp supports a cross environment execution. Here the interface shows how the user can develop and execute two source files in different environments simultaneously. This application allows the user to edit the program files without ever having to actually go into Visual Basic or Visual C++ itself. The right editor has a section of a C++ program source code loaded in it while the one on the left has a Visual Basic program.

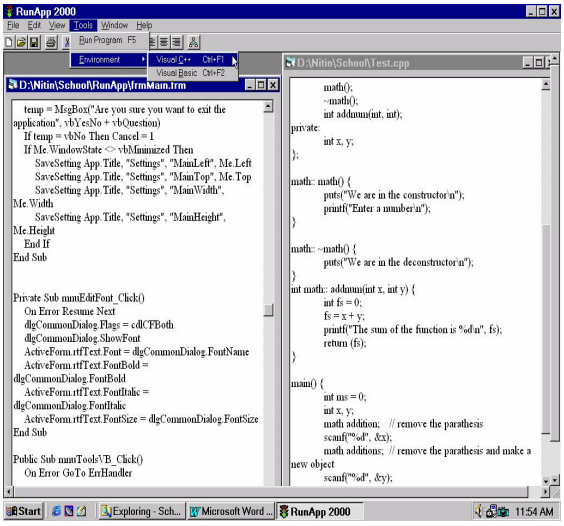

Figure 5. Cross-environment execution

RunApp can also be used to invoke the in-built web browser from the View menu. The browser can be used to refer to development information on the web or use the on-line Microsoft Developer's Network (Microsoft Corporation 1999b, 1999c) help features. One does not have to have a web browser installed on the computer as the Internet Explorer browser control has been incorporated into this application itself. The RunApp Setup program installs all the required dynamic link libraries and reference files. The browser contains only the basic features for surfing.

The help module can be invoked from the help menu bar from the main screen. This loads the help file in the standard windows help module by using the windows application programmer's interface (API) function calls to the operating system. Here the user can use the Index, Search and Find options to locate the help topic that he/she desires. There is also an option that will load the help file in the find mode directly. The help process runs independent of the application as a separate process. It has to be terminated explicitly.

Figure 6 shows the system information tool. The Microsoft System Information is invoked by calling an API function. This tool reads information from the kernel, system repository and windows registry. It is an information tool that supplies the user with information on system resources, tasks, threads, DLL's (dynamic link library), registry settings, active modules, INI settings, etc. This information can be useful for the programmer to monitor memory usage and task/process distribution. This tool runs as an independent process so it does not conflict with the RunApp application environment. In this manner, it can display information about RunApp too (shown). The system information is part of the window operating system. RunApp just links to this tool and helps display the information.

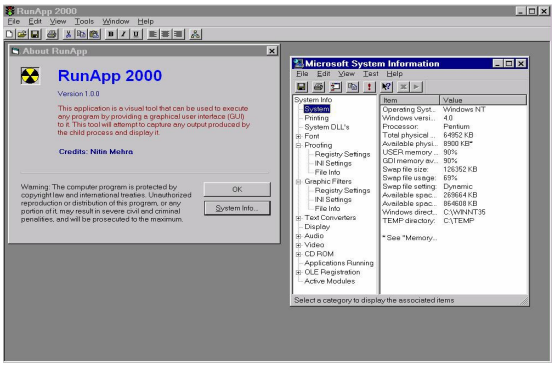

Figure 6. System information

The Setup program as shown in figure 7 is created along with this project to install and deploy the application on a computer. This setup program is created using Visual Basic's deployment kit and will install and the required DLLs, files, programs, data, etc. required to execute RunApp. This setup also contains an Uninstall utility program that can remove the entire application from a computer. All the files installed are logged by the setup program and removed by referring to this log file. The Setup program gives the user the choice on where to install the files and icons.

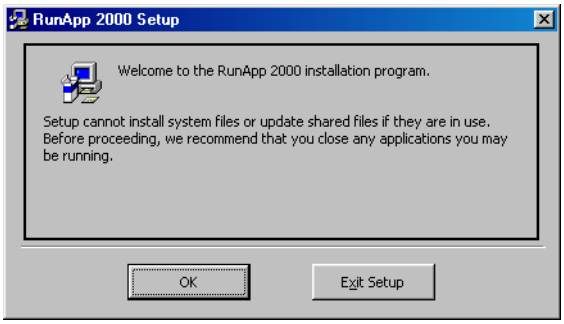

Figure 7. Setup for RunApp 2000

3. CONCLUSIONS

This application was designed and developed using the Windows API functions and sophisticated Visual Basic dynamic objects and Window controls. The main characteristic of this system is the transparency it provides to its user about the existence of multiple processors and environments. This application can be distributed using a setup program developed and provided along with this project. This setup program will install all the necessary components required to run RunApp such as dynamic link libraries, support files, help file, and application executables in a user-desired directory on any IBM compatible personal computer. Moreover, the window registry variables, program group icons, and system initialization files will be added/modified in the computer as well. The application is compiled as a 32-bit executable and requires Windows '95/'98/NT or later operating systems to run. There is an uninstall utility that comes along with this program that will undo and delete all the installed components, files, registry settings, etc. The uninstall program will not remove compiled executables or program source files that have been created by the user. This information will be untouched and retained. The following benefits were identified after the development of the RunApp application.

1) RunApp creates a fault tolerant and secure application development environment.
2) RunApp is flexible and can be molded/customized for specific user requirements.
3) This application can be used as a sophisticated development tool.
4) The system provides the user with high concurrency.

This project has immense amount of scope and opportunity for future enhancements and development. The primary enhancement that can be made to this system is incorporating additional programming environments such as PowerBuilder, Delphi, FoxPro, etc. Moreover, fourth generation languages can be engulfed into this such as SQL, ProC, and others. The graphical user interface can be enhanced to support intelligent editing and file processing. A more concise and detail on-line help system and documentation can be added to RunApp. Advanced options such as personalized user settings, login, and auditing options and version control can be added to make this application more secure. This application can be shared over a network by adding distribution and concurrency methods to enhance speed and share resources.

4. REFERENCES

Brookshear, J. G., 1997, Computer Science – An Overview (5th ed.). Addison-Wesley, USA.

Eustice, Lehman, Morales, Munson, Edlund, and Guillen, 1999, A Universal Information Appliance.

Geist, Beguelin, Dongarra, Manchek, and Sunderam, 1996, PVM: Parallel Virtual Machine. Cambridge: The MIT Press.

Kernighan and Ritchie, 1996, The C Programming Language (2nd ed.). AT&T Bell Labs and Prentice Hall.

Microsoft Corporation, 1999a, Application Programmer's Interface (API). Microsoft Corporation – Windows '98/NT.

Microsoft Corporation, 1999b, Visual Basic Technical Support – MSDN. [Online]. Available at: http://msdn.microsoft.com/vbasic/technical/support.asp

Microsoft Corporation, 1999c, Visual Basic Technical Support – MSDN. [Online]. Available at: http://msdn.microsoft.com/vbasic/technical/support.asp

Silberschatz, A., and P. B. Galvin, 1994, Operating System Concepts. Addison-Wesley, USA.

Whitehead and Maran, 1997, Teach Yourself Networking Visually. IDG Books Worldwide, Inc.