Development In Business Simulation & Experiential Exercises, Volume 18, 1991

AN ARCHITECTURE FOR EXTENSIBLE SIMULATION GAMES

T.M. RAJKUMAR, Miami University

INTRODUCTION

Simulation games have evolved through different stages. The current stage of development is characterized by its use of the microcomputer (Curran and Hornaday, 1989). A difficulty with development of games for microcomputers is that different systems are not compatible with each other. This difficulty gets compounded with advances in technology, since the same game has to be developed for many new platforms such as the Windows, and OS/2 environments in addition to the standard DOS and Macintosh environments. This implies that the game has to be fairly portable. In addition, the user interface of the game has to adapt to each environment. Such adaptation normally implies that the user interface segment of the code has to be rewritten. This rewrite can be avoided by developing an architecture, which is extensible. Extensibility implies that the existing code need not be changed and additions can be made without affecting the existing code. The purpose of this paper is to state some requirements of simulation gaming and develop an extensible architecture for simulation gaming that can meet these requirements. The next section discusses some requirements, which call for extensibility, and section 3 develops an extensible architecture.

REQUIREMENTS

Modularity

Rohn (1986) suggests that 3 levels of modularity be built into games. They are 1) horizontal modularity involving the same level of sophistication in different functional areas; 2) Vertical modularity--where different grades of game complexity exists within a single functional area and 3) diagonal modularity where the game model can be adapted to differing situations such as a small firm, or conglomerate of firms etc.

Hypertext Help

Manuals written for student players tend to be linear (i.e. one chapter after another). In contrast, a students player requires a smorgasbord approach where the student gets a brief overview of all variables involved in making the decision, and is then provided with the ability to jump to related topics easily for a more in depth presentation. Such an on-line browsing tool that is available for microcomputers is hypertext. Hypertext provides the capability to look at related topics in a nonlinear fashion.

Changing Environments

Burkhardt and Watson (1989) specify that the underlying game model must change to accommodate changes in the teal world. This implies that the data structures used in the program must be carefully thought out and the game be extensible.

EXTENSIBLE ARCHITECTURE

Extensibility in simulation games implies that existing code should not change but allow code to be added so that the game is flexible and adaptable to new environments and changing conditions. This can be achieved by dividing the game software into two parts--i) a kernel or situation independent software and 2) situation dependent software.

It is not enough that the kernel just provide software that are situation independent, it should provide basic services that need to be used by the different situation dependent software. The situation dependent software also needs to follow a certain discipline in that they all should provide tot a consistent set of user interface commands, so that an user migrating from one environment to another can easily transfer his knowledge. This consistent common interface allows the developer to change the underlying implementation without the user being aware of such changes.

An example of situation dependent software may be the use of different types of on-line manuals--for example a regular linear help or a nonlinear hypertext based help. The architecture of the extensible system allows for such flexibility because both these help facilities would be add-on software and would use the same common interface provided by the kernel. The kernel would provide the context and retrieve the information from the disk for the help facility, which would then display the information for the user.

CONCLUSION

Object oriented programming languages such as C++, Smalltalk provide facilities to enable the developer to build consistent user interfaces and extensible systems. Such systems are more flexible and adaptable to changes. A brief description of such an architecture has been outlined

REFERENCES

Burkhardt, D.L. and Watson, H. J. (1989) From mainframe to micro. Simulation and Games 20, No. 1: 31-43.

Curran, G.E. and Hornaday, R.W. (1989) A hybrid method of executing a management game simulation. Simulation and Games 20, No. 4: 459-470

Rohn, W.E. (1986) The present state and future trends in management games for management development in Germany. Simulation and Games 17, no 3: 382-392