Developments in Business Simulation & Experiential Learning, Volume 26, 1999

CREATING INTERNET-BASED GAMES USING PERL AND JAVASCRIPT

Sharma Pillutla, Towson University

ABSTRACT

Business Simulation Gaming has been around for quite a few years. The pedagogical and more recently, the assessment value of gaming has been recognized and accepted in many circles. Over the years, as technology has improved, simulation exercises have become more sophisticated and user-friendly. With the advent of the Internet an added dimension of convenience is now available. Using the World Wide Web is a natural extension of making simulation games more powerful and the access more convenient. This paper delineates one such attempt at making the entire game available on the web. We describe the various components of the game and describe how the programming language Perl and JavaScript can provide the functionality for making the game available on the Internet.

INTRODUCTION

Over the last few decades Simulation gaming has become more powerful and more integrated with the curriculum. With the advent of more powerful computers and new user interfaces, simulation games have also kept pace by becoming more sophisticated and more convenient to use. The World Wide Web (WWW) debuted in 1993 and usage of the Internet has exploded ever since. Web-enhanced courses have become the norm in academia. Web-based courses now form the backbone of Distance Learning programs.

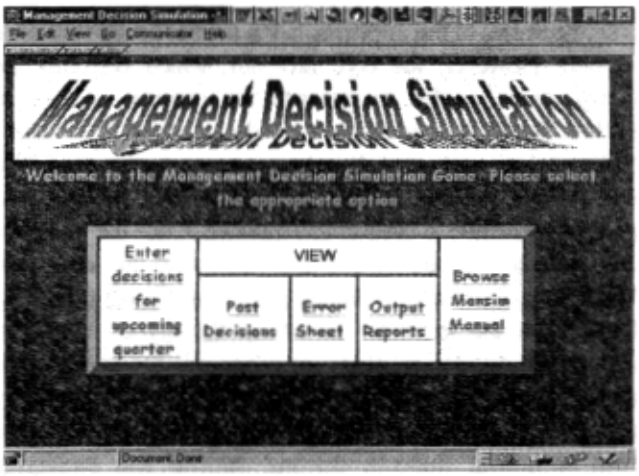
This paper looks at one such attempt to deliver simulation gaming over the Web. The advantages of delivering games over the web are an extremely user-friendly interface and minimal access limitations. Any one with an ISP can now access the game twenty-four hours a day. We discuss the adapting of a simulation game used as a pedagogical tool in the introductory operations management class to the web.

INTERACTIVE **PROCESSING**

In order to have interactive processing, one needs, in addition to a regular HTML web page some back end program which can take user input and process it. This kind of back end programming is done through the Common Gateway Interface (CGI). The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP or Web servers [1]. A plain HTML document that the Web server retrieves is staticA CGI program, on the other hand, is executed in real-time, so that it can output dynamic information. Security concerns need to be addressed before running CGI programs. CGI programs can be written in almost any language such as Pen, JavaScript, Java C, C++ etc. In this project we have used Perl and JavaScript. Perl is a scripting language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information [3]. JavaScript can be thought of as an extension to HTML, which allows authors to incorporate some interactive functionality in their web pages [4].

GAME COMPONENTS

There are two components of the game that have been implemented using Pen and JavaScript - the student interaction system and the game administration system for the instructor.



Student interaction Interface

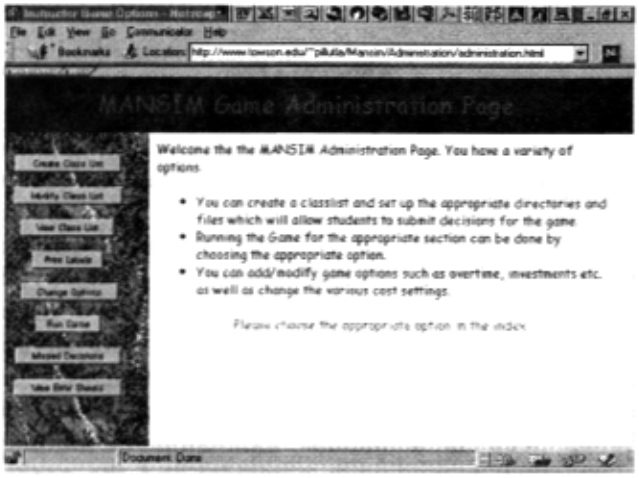
Using this interface the student submits decisions, views the archive of past decisions, views any input errors that they made, browse various outputs such as the income statements, balance sheets and other

information. Implementation is done is Perl and JavaScript. Above is the main page of the student interface. The basic options are entering decisions for the current quarter, viewing past decisions, input error sheets and outputs or browsing the manual

Issues that pertain to entering decisions include authentication of the student's password and other identification information, confirmation of the submission of the decision. All of these are done in Perl. Minimal error checking is done during data entry (numeric Vs text input) by using JavaScript. The instructor can run the game after the deadline for turning in decisions is past. The program has safeguards to ensure that the student doesn't turn in a decision alter the deadline, *i.e.,* after the instructor runs the game for a particular quarter. Students can view all of their past decisions which would help them in determining the future decisions. Here again, it is necessary to authenticate the user by means of a password. Any violation of game rules in the decisions made is considered an input error and an error sheet generated. Password authentication is done for this link also. Students can view their company's performance by viewing their income statements, balance sheets and evaluation of their performance. There is no password authentication for this link. All the instructions for playing the game are provided in the form of an online manual. It is possible for the students to browse the online manual for the game or even print it out if they wish to peruse it at their leisure.

Game Administration by **the Instructor**

Major portions of the implementation for this section are done in Perl. However, the core program, which performs the computations needed to generate the outputs, is done in Fortran. Below is the main page for administration of the game. Since only the instructor should have access to this page. The entire directory is password protected. The *.htaccess* password protection method is used in this instance. Once the instructor enters the correct password, she is allowed to use any of the links in the game administration page. The *Create Class List* links allow the instructor to set up the game. It sets up appropriate directories, generates passwords for students and allows the instructor to enter student names and company assignments. During the course of the game any modifications such as adding students, deleting students and modifying information such as name, password etc. can be done using the *Modify Class List* link. The instructor can also view the list at any time. The View Class List link has an option of either displaying or hiding the passwords so that it does not appear on the screen to inadvertently prevent the passwords from being revealed to an onlooker. The *Print Labels* link allows the instructor to print out the student names, company assignment and password which will be distributed to each student at the outset of the gaming exercise. A



variety of options are present in we game such as allowing overtime, or allowing short term investments etc. The *Change Options* link allows the instructor to modify any or all of the options so that the game would henceforth be run with those new options. The *Create Class List* link creates a file with the default options for each new class. The Run Game link is self-explanatory. The next two links - Missed Decisions and View Error Sheet - are primarily for tracking the student involvement in the game. The *Missed Decisions* link allows the instructor to see whether a student has been turning in the decisions by the announced deadline or whether they have not turned in their decisions. The *View Error Sheet* allows the instructor to view any input errors that the student may have made while yields insight into how well a particular student has understood the basic parameters of the game. This indicates the difference between a student submitting decisions as opposed to a student who is "plugging in numbers".

CONCLUSION

It is seen that implementing a simulation game on the web has its advantages. Some of the major aspects pertaining **to** conducting a simulation gaming exercise have been implemented through the web. Issues such as student authentication, instructor access to the administration, performance logging of missed decisions are all features that stand out in this implementation. Obviously, further enhancements can definitely be made. Running the game Running the game automatically for an industry after all competitor decisions have been obtained is one enhancement. Thus the web offers a new technological medium to offer games that are independent of place and time.

References provided on request