THE DEVELOPMENT OF SITUATIONAL SIMULATION GAME PROGRAMMING

by

Jaturon Jintanawan

A Research Paper

Submitted in Partial Fulfillment of the
Requirements for the
Master of Science Degree in
Management Technology

Approved: 3 Semester Credits

Thesis Advisor

____

Kevin McDonald, M.S., Associated Professor

The Graduate College
University of Wisconsin-Stout
December 2001

The Graduate School
University of Wisconsin-Stout
Menomonie, WI 54751

ABSTRACT

	Jintanawan	Jaturon	
(Writer)	(Last Name)	(First)	(Initial)

The Development of Situational Simulation Game Programming
(Title)

Management of Technology	Kevin McDonald	December 2001	150
(Graduate Major)	(Research Advisor)	(Month/Year)	(No. of Pages)

American Psychological Association (APA) Publication Manual (Fifth Edition)
(Name of Style Manual Used in this Study)

According to Professor Kevin McDonald at UW-STOUT, the students in introductory-level business classes need a tool to encourage them to understand business concepts. In satisfying the need of the students, the computer games are recognized as an possible alternative. However, the computer games available are unsuitable for elementary level courses; the existing games aim at a high-level of understanding of business concepts.

This study will develop a computerized game to fulfill the needs of professors and students in the low-level business courses. The research focuses on adapting the CATscanner model, the class material, to be a main part in the game, using the computer technique. The simulation feature is included as a characteristic of the game. The objectives of the game are to develop a tool in which the user can examine business experiences and the idea of “cross functional area in the organization”. The instructor will use the game to create the series of situations which demonstrate the business lessons in the class.

The development of the simulation game is dependent upon the software engineering technique. Two sections are devoted to this development; one is targeted to an instructor, and the other is for a student. The result of the research includes the software package with installation files, source code, and game documentation.

ACKNOWLEDGEMENTS

This thesis is a part of my graduated education, which I dedicate to Department of Management Technology at University of Wisconsin-Stout. I hope this study will assist the business education.

I would like to begin by thanking my thesis advisor, Professor Kevin McDonald from the Business Department, who inspired me for the thesis topic. I would like to thank Dr. Thomas A. Lacksonen, my program director who suggested me about guidance and technical problems. I would also like to thank Brian at Multicultural Department for proofing my paper.

I would like to thank the people who have developed the simulation games who gave me the direction of doing this thesis. I would like to thank people who developed and implemented the computer technology.

I would specially like to thank to my parents who have been supporting me all the time. I would like to thank all my friends who encourage and assist me to finish the research, specially for Poy and Pooh.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

This is an introduction to the study and the problem under investigation. This chapter is comprised of six sections as follows: (1) overview, (2) need for the study, (3) statement of the problem, (4) research objectives, (5) limitations, and (6) definition of terms.

Overview

Technology has been playing an imperative role in helping education. Some techniques have been used to increase the students' skills, and among these techniques, a simulation game is really popular. "Games and simulations … enable organizations to envision alternative futures within a condensed time frame, and help them get a holistic view of the change journey and its results. The comprehension of the totality of problems and opportunities awaiting them in the future (although at the higher level of abstraction) helps them build cognitive bridges to that future. The result is not only an increased awareness of the future marketplace, but an increased ability to deal with complexity and uncertainty in general"(Ivo and Don, 1999, Games and Simulations and the Big Picture section).

In the business classroom, the simulation game technique has been popularly used to facilitate learning. The game usually simulates real case situations to giving students business experience from those situations. In many cases, a teacher would use the game for a class's material rather than lecture in a traditional way. Their objective is to facilitate learning the complicated subjects. Nevertheless, the simulation games are used in different levels business education. Many of the simulation games are developed to be used in the high level education. In this point, most of the business simulation games in the market are really complex. This means that the players or students are required to know some business knowledge. The direction of the game development focuses on various business functions, from making investments to strategic planning. The results of the modern game may be too difficult for the elementary level business course.

A principle concept of business has been elucidated by the business model at www.catscanner.com and used for business classes at the University of Wisconsin-Stout. “It is fundamental to your understanding of the major business components, and to your understanding of the dynamic and exciting interaction between these components” (Kevin McDonald, Understanding Business Course Outline). The model depicts the various functional areas in an organization as internal environments; finance, information systems, accounting, management, research & development, service, marketing, production materials management, human resources, and strategy. The external business environment consists of customers, competitors in the industry, and the extraneous business conditions: production inputs, economic conditions, technological conditions, political conditions, legal conditions, social/cultural conditions. The external environment cannot be controlled by the company but can be scanned and monitored to anticipate resulting the positive and negative impacts. The concept of “cross functional area” is represented as a relationship between the internal functional areas of a business and a relationship between the organization and the external business environment; events that happen in the external environment will in some way impact the internal environment of the organization, For every action there is a reaction; for every stimulus there is a response and for every cause there is an effect. Proactively scanning the external business environment is recommended in an effort to identify changing business events that can either negatively or positively impact the organization. Anticipating the negative or positive impacts of the changing external business environment consider is a factors to be successful in the business.

The main goal of a business, seeking the maximum profit by providing goods and services, depends on internal and external environments. The actions of unknown and unpredictable situations from inside and outside the company represent the idea of cross functional area. For this principle, it is crucial that the manager practices dealing with potential situation to find their alternatives and either limit the negative impacts or take advantage of the positive consequences.

Using the technique of simulation game is really difficult for this fundamental implication. To develop an appropriate application, a specific business needs to be formed from simple processes yet must demand an adequate performance of student ability. The CATscanner will be appropriately applied to the simulation game representing the fundamental of business level. Using a single situation or case matching to only the best alternative is the best way of the game style, however, critical features have to be enclosed in the application. While the application declares business functions and ordinary rules, it must show the concepts and core working of an enterprise. Active learning, from situation to solution, must aim at the principle concept in the model.

Need for the Study

Professors of business administration have generally been slow to incorporate business gaming simulations in their teaching and research (Wolfe, 1994), but the basic discovery phase of business gaming has not ended as argued by Thavikulwat (1999). Nevertheless, the simulations being used are not that much different from those created in the late 1950s. Although computerized business gaming simulations may enhance business education, as the studies reviewed by Keys and Wolfe (1990) show, the enhancement comes consistently at a high cost in students' time.

Even though business classes at UW-STOUT have referred to the CATscanner model as a important principle of business, it is still a static tool for the students, meaning that the students must learn the model without interacting with the CATscanner model’s graphic. According to Professor Kevin McDonald (personal communication, 2001), the students need some tools for understanding the elementary level business concepts in the class. The CATscanner model will powerfully facilitate the need of the student and instructor in the business class if it can be developed to be a dynamic tool, meaning that student can use it interactively.

Statement of the Problem

Because of the complexity of current business simulation games, there is no application software which fulfills the need of a fundamental online business class in the market. All existed applications confuse the students with many decisions, making it difficult to understand for somebody who does not have knowledge of business. The simulation games in the market may not directly satisfy the need of students in understanding fundamental business idea.

Research Objectives

The objectives for this study are:

1. To create a situational and simulation game and application software, for a fundamental online business class.
2. To create an application to depict the general ideas of operation and perception of a business, including the concept of "cross functional area in an organization," that address the internal and external factors around the business environment.
3. To facilitate hand-on experience of several situations in the business world for the business student.
4. To apply this simulation game to teach students about the case situations in business.
5. To simplify the instruction for people who lack of a business background. On the other hand, they can practice in the common and uniform manner of a business owner.

Limitations

The following are the limitations which have been identified with reference to this study:

1. Since the business simulation game is a restaurant, the knowledge and practical skills that a player gains may not apply to other types of business, especially an organization which needs cooperation and teamwork, in order, to make a decision, for this game depends on only one player.
2. The application of this simulation game is limited to a decision based on short-term goals, since a monthly situation ends and does not continue at the following month. In fact, several business situations or crises in the real world will carry on and be considered as other factors that affect the business operator's future decisions.
3. The ultimate goal of this simulation game is set, which is to gain as high a profit as possible. It may not be applied to some real business situations such as launching a new product.
4. This application can be executed only if a player downloads the program in a computer that has Visual Basic Software.

Definition of Terms

For clarity and understanding, the following definitions of key words have been included:

*Algorithm***.** A prescribed set of well defined, unambiguous rules or processes for the solution of a problem in a finite number of steps; for example, a full statement of arithmetic procedure for evaluating cosine x to stated precision. Contrast with heuristic

Application program. A program that helps the user accomplish a specific task; for example, a word processing program, a spreadsheet program, or an FTP client. Application programs should be distinguished from system programs, which control the computer and run those application programs, and utilities, which are small assistance programs.

Argument. A value that is passed to a program, subroutine, procedure, or function by the calling program; one of the independent variables that determine the output

Database. The collection of all data used and produced by a computer program. In large systems, data base analysis is usually concerned with large quantities of data stored in disk and tape files. Smaller microcomputer systems are more frequently concerned with data base allocations of available memory locations between the program and data storage areas. Also called data bank.

Data Flow Diagram. Distinct pieces of information, usually formatted in a special way. All software is divided into two general categories: *data* and *programs*. Programs are collections of instructions for manipulating data.
Data can exist in a variety of forms -- as numbers or text on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in a person's mind. Strictly speaking, data is the plural of *datum*, a single piece of information. In practice, however, people use *data* as both the singular and plural form of the word.

Front-end applications. The programs that translate source code into object code, are often composed of two parts: a *front end* and a *back end.* The front end is responsible for checking syntax and detecting errors, whereas the back end performs the actual translation into object code.

Graphic user interface. "A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.
Graphical user interfaces, such as Microsoft Windows and the one used by the Apple Macintosh, feature the following basic components:

Pointer. A symbol that appears on the display screen and that you move to select objects and commands. Usually, the pointer appears as a small angled arrow. Text -processing applications, however, use an I-beam pointer that is shaped like a capital I.

Pointing device. A device, such as a mouse or trackball, that enables you to select objects on the display screen.

Icons. Small pictures that represent commands, files, or windows. By moving the pointer to the icon and pressing a mouse button, you can execute a command or convert the icon into a window. You can also move the icons around the display screen as if they were real objects on your desk.

Desktop. The area on the display screen where icons are grouped is often referred to as the desktop because the icons are intended to represent real objects on a real desktop.

Windows. You can divide the screen into different areas. In each window, you can run a different program or display a different file. You can move windows around the display screen, and change their shape and size at will.

Menus. Most graphical user interfaces let you execute commands by selecting a choice from a menu.

The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Center in the 1970s, but it was not until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which until recently were prohibitively expensive.

In addition to their visual components, graphical user interfaces also make it easier to move data from one application to another. A true GUI includes standard formats for representing text and graphics. Because the formats are well-defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph created by a spreadsheet program into a document created by a word processor.

Many DOS programs include some features of GUIs, such as menus, but are not graphics based. Such interfaces are sometimes called graphical character-based user interfaces to distinguish them from true GUIs" (Webopedia, 2001, searching page)

Module. a part of a program. Programs are composed of one or more independently developed modules that are not combined until the program is linked. A single module can contain one or several routines.

Object Oriented. "A popular buzzword that can mean different things depending on how it is being used. Object-oriented programming (OOP) refers to a special type of programming that combines data structures with functions to create re-usable objects (see under object-oriented programming). Object-oriented graphics is the same as vector graphics.

Otherwise, the term object-oriented is generally used to describe a system that deals primarily with different types of objects, and where the actions you can take depend on what type of object you are manipulating. For example an object-oriented draw program" might enable you to draw many types of objects, such as circles, rectangles, triangles, etc. Applying the same action to each of these objects, however, would produce different results. If the action is Make 3D, for instance, the result would be a sphere, box, and pyramid, respectively. (Webopedia, 2001, searching page).

Pseudocode. An outline of a program, written in a form that can easily be converted into real programming statements. For example, the pseudocode for a bubble sort routine might be written: while not at end of list compare adjacent elements if second is greater than first switch them get next two elements if elements were switched repeat for entire list Pseudocode cannot be compiled nor executed, and there are no real formatting or syntax rules. It is simply one step - an important one - in producing the final code. The benefit of pseudocode is that it enables the programmer to concentrate on the algorithms without worrying about all the syntactic details of a particular programming language. In fact, you can write pseudocode without even knowing what programming language you will use for the final implementation.

One-to-many Relationship. A type of data modeling represents the relation between two entities or things. One entities has many relationship to the other. For example, a mother has many children.

Recordset Object. A object technique represents the entire set of records from a base table or the results of an executed command. At any time, the Recordset object refers to only a single record within the set as the current record.

Simulation. “Creating a mathematical model of a real system, to see how the system works and, by changing variables, make predictions about how the system will change.For example, a mathematical simulation of an insect population and its habitat could include such variables as available food supply, natural predators, and rainfall. Changing one of the variables in the simulation would show how the population is affected when that variable changes in the real system. See also real-time simulation”(Hi-tech dictionary, 2001)

SQL command. Structured Query Language (pronounced SQL or Sequel).A language used to create, maintain, and query relational databases. It is an ISO and ANSI standard. SQL uses regular English words for many of its commands, which makes it easy to use. It is often embedded within other programming languages.

Structure chart. Structure Chart is software diagram tool that show hierarchical arrangement of the modules in a structured program. It contains rectangular box and arrow symbol. The rectangular represent one module. The name of a module is written inside the box. An arrow joins two modules that have invocation relation.

Visual Basic. The defending of Visual Basic in Microsoft website (2001) gives the idea of "Visual" part that refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, the developer simply add pre built objects into place on screen. If you've ever used a drawing program such as Paint, you already have most of the skills necessary to create an effective user interface.

The "Basic" part refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.

Data access features allow you to create databases, front-end applications, and scalable server-side components for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.

Internet capabilities make it easy to provide access to documents and applications across the Internet or intranet from within your application, or to create Internet server applications.

Your finished application is a true .exe file that uses a Visual Basic Virtual Machine that you can freely distribute.

CHAPTER 2

REVIEW OF LITERATURE

This chapter provides a theoretical base for the problems and the relevant surrounding issues and concerns associated with the study. Included in this chapter are: (1) history of the simulation game, (2) existing games in the market, (3) complexity of simulation game, and (4) other views of simulation games.

History of the Simulation Game

The first business simulation game has been traced back to use in China around 5,000 year ago (Wolfe and Crookall, 1998). However, the new age of business simulation just started around the nineteen the century. The game, Monopolog, was created to simulate the Air Force supply system for inventory managers for inventory managers (Jackson, 1959). The American Management Association (AMA) later on developed the well-known business game, Top Management Decision Simulation, used in management seminars and administered by the AMA (Meier, Newell, and Pazer, 1969). It was from material used in the business class at the University of Washington in 1957 (Watson, 1981). From this point, the number of business simulation games expanded rapidly. The estimated number, over 100 business games, have been referred to and played by business executives (Kibbee, Craft, and Nanus, 1961), and business schools have used simulation games in their training programs. While Burgess (1991) has reported on business simulation game usage in the Untied Kingdom, McKenna (1991) conducted research on business usage in Australia. Subsequently, Chang (1997) observed the used of simulation games in Hong Kong.

According to Faria (1998), business simulation has been known for 40 years and it has been grown at an exceeding rate. The term “cutting-edge teaching” was introduced by Wolfe (1993).

Existing Games in the Market

The following reviewed simulation games that have been posted and published. It describes the game’s features and details for player.

SimCity

The popular game “SimCity” gives attention to developing and managing a budding city. The player can monitor the details of the city block-by-block and can employ other features to utilize the window’s power to display real-time information (Paul, 1993).

The player starts a game as the mayor of a city. The level of difficulty, which essentially provides you with more start up cash, is chosen. The player is assigned to create and run a simulated city. The game is very detailed in the responsibilities necessary for of overseeing and running an entire city. There is tremendous educational value in having to manage the various aspects of providing for the needs and desires of a community. Players, as the mayor, are responsible for details from trash disposal to land development to addressing the concerns of regulatory agencies and petitioners. There is a lot of micro management that, in real life, would be handled by section heads. For example, put roads too far apart, and Sims (the residents of your city) won’t walk to work. Don't build enough playgrounds, and families will move out, causing your residential areas to run down and taxes to dry up. All the real life problems you hear about in the news can and will happen to your town (Rick, 2000).

Profitania_Deluxe

“Profitania Deluxe,” the third and final trilogy installment from LavaMind is advertised as an educational title or a business simulation. Granted, it educates about supply and demand, and general business concepts by running one’s own manufacturing facility, but it was created as an entertainment title. Imagine a game similar to Monopoly (TM) but much zanier and without the limited of a board. It is designed accumulated up to 6 human players on the same computer. The computer will play any factories not manned, so there is always six manufacturing sites in competition. The game has a level of difficulty that can be adjusted for how "good of a player" the computer is for each competitor. Everyone is in a race to achieve 1 million gems (local currency) and to be recognized as Profitania's Industrialist Extraordinaire.

The game’s story starts with a subterranean civilization powered by magma pools. In a growing economic era, the player is assigned to construct a factory that is surrounded by a diverse community. It can benefit the player’s factory get benefit. The player has the ability to manage before the commodity will change with operates in “real-time.” The game gives the player an idea of business and math as each game turn is equal to one week in game time.

The factory needs materials to work with, and the source for these is goods exchange. The "real-time" aspects of this game are the fluctuations in price, quantity, and quality of the commodities when the players are trying to buy.

The required materials depend on your factory knowledge and your factory’s products. Using the R&D Department is a good strategy for a player. A player can operate what the scientists are working on. The trick is not to research something that several other factories know how to make they probably already have products on the market. Instead, the factory should stay on the culture of innovation; bearing in mind that a more complex a product takes longer to research. In addition, there are levels to product development. The player will not able to research a level 3 product until having researched a couple at level 2. A product’s description is shown in the following detail: what materials are required to produce it, estimated market price, product size to calculate the space it will take up in the warehouse and factory resources involved in producing it. It is safe to say that a factory that does not invest in research will have a difficult time winning the game (Michael Gonsalves, 2000).

The Business Strategy

Snyder (1996) found the following:

“Game the Business Strategy game (BSG) is third version regarding to Wolfe's (1994) review in Simulation & Gaming
The BSG is an international total enterprise game, with separate markets for North America, Europe, and Asia. Student teams compete in four footwear markets: three branded geographic markets of North America (plants in Ohio and Texas), Europe and Asia, and a private-label market in North America. Game administrators have the ability to process yearly decisions for four to sixteen teams of students in a fictitious industry. The third edition of Thompson and Stappenbeck's popular game offers subtle changes for the players and substantial changes for game administrators.
The BSG competes in the arena of total enterprise simulations with other remarkable titles, such as The Business Policy Game (Cotter & Fritsche, 1995); Intopia (Thorelli, Graves, & Lopez, 1995); Strategy (Priesmeyer, 1993); Corporation (Smith & Golden, 1994); CEO (Thavikulwat, 1991); and Micromatic (Scott, Strickland, Hofmeister, & Thompson, 1992), to name a few. The popularity of the BSG is its positioning alongside the most popular strategy text, Thompson and Strickland's Strategic Management: Concepts and Cases (1996).

The third edition of the BSG possesses a wide array of changes and enhancements. One such revision is the depiction of markets. Versions one and two of the BSG modeled one quality level for the four markets available for selling products (three branded areas of North America, Asia, and Europe, and the private-label market in North America). Version three allows teams to differentiate by offering different quality levels in each of the markets. In addition, teams can vary the number of models (e.g., product line breadth) in each of the markets. The intent of the developers was to allow a more varied strategy for game players by using the same variables, but splitting them up by market segment.

The developers enhanced game play by including more powerful "what-ifs." These what-if projections appear at the bottom of the screen and change as data is placed in the various screens of decision inputs (e.g., labor, production, shipping, marketing, finance). Feedback is automatically updated on the bottom of the screen, which helps to avoid the time-consuming process of viewing separate what-if screens. Projections are provided on revenues, net income, earnings per share, return on equity, and cash balance. The decision support system also provides feedback on the number of workers needed for a given level of production, based on productivity levels, and other what-ifs pertinent to the submenu a player happens to be in. Error trapping has been expanded so that the program will not accept any decision entry that falls outside the valid range or is of the wrong type.

Thompson and Stappenbeck took a philosophical stance on the use of decision support systems in total enterprise simulations. Their contention was that users of the simulation should receive any information they could possibly want. However, it is up to the user to assess the information provided and use it in decision making. For this reason, the BSG is replete with screenfuls of pertinent information concerning competitors and what-ifs.

Competitor analysis has been enhanced by including a separate company analysis disk with the program. Teams are provided a submenu that prepares either industry or specific company competitor analysis reports. These reports facilitate both industry and focused analyses of the competition. Teams have the ability to print a chosen year's industry analysis or a year-by-year analysis of the decisions from one of the competitors. The reports provide information from the footwear industry report (FIR), arranged into formats suitable for easy diagnosis of competitor strategies.

The developers have changed the visual feel of the game by making the decision entry screens easier to read, with less clutter. The menu bar at the top of the screen has been redesigned for speedy access to submenus. The levels of menus are easy to follow and intuitive. Perhaps the most valuable change is the graphical user interface (GUI) addition of a mouse. Screens can be passed over, or specific fields for inputs found, by pointing and clicking the mouse. Users may still find themselves using the escape key

and the arrow keys required in previous editions to move around, but once they catch on to the abilities of the mouse, the keyboard becomes secondary.
The crash-proofing ability of the latest version of the BSG is impressive. Administrators were previously limited to rerunning only the latest year in case of industry problems. Version three has the ability to rerun any year. In case team disks become corrupted, the recreate option from the operations menu provides improved flexibility to restore the lost information. An undocumented addition to the new version is a data editor that enables administrators to change individual information stored about the industry or the companies. A warning is presented when entering this submenu (accessed by hitting the F6 key), indicating that changes to the individual files used for running the game may cause irretrievable losses of data. In previous versions of the game, erroneous data could only be corrected by loading specific spreadsheets into Lotus 1-2-3 and then calling the game developer to find out what information was held in the various, number-cluttered fields.
A rarity in running total enterprise simulations is finding help when problems are encountered. Two outlets are available in times of distress while running the BSG: one by contacting the publisher's representative and the second by contacting the developer. Both avenues will provide friendly and concise help. Most game-oriented questions posed by teams can be answered by referring to the 91-page player's manual.
The third edition of the BSG is a welcomed addition to the existing array of total enterprise simulations. Its development follows the trend toward providing GUI for players and administrators alike. By using menus and eye-appealing features, future simulations can further enhance the complexity/ realism dimension without sacrificing understanding. Expect future editions of such games to include hypergraphics and more powerful built-in graphing features."

The Restaurant Game

The restaurant game is developed as a single-period simulation that provides students with the opportunity to plan and implement a strategy in a competitive environment. The players bid for raw materials. This auction results in players' making calculated changes to planned strategies in response to conditions created by the auction. After the players have obtained their necessary raw materials, they create meal menus that are the source of company profits. The integer programming is generated, and the instructor can teach how numerical modeling leads to optimal solutions. The restaurant game provides a rich environment that can be explored in a single period or multiple periods according to the instructor's goals (Brozik and Zapalska, 2000).

The game begins by distributing the instruction sheet that provides students with the information they need to plan an overall strategy. Because the winner of the game needs only, the greatest profit, there are many ways to approach the market. Players have some choices in offering low, medium, high, or a combination of priced meals. These choices will direct strategies of both a narrow and a broad market segment.

The interactions of the players in the auction will establish each menu's relative profitability and randomly drive the strategy for each player. The players need to be aware of the changing environment and to possibly adapt their strategies to meet evolving market conditions.

Competition is important aspect of the game. The mini monopolies situation may be built in with too few players. On the other hand, with too many players, there is insufficient time to auction off enough goods so all players can form complete menus. Following are environments that needs to be concerned with:

1. The optimal number of players.

2. The inventory with the planned menu offerings.

3. Money balance.

4 Competitors’ bidders in the market and the prices offered.

5. Actually doing the bidding.

The teams’ auction allows student to focus on a single event of process in a dynamic environment. Each bid sheet shows to the instructor the items listed in the menus and materials required to make them. The bid runs continuously until the companies run out of money, and the bid sheet will give the number of raw materials available in the market. Once the auction begins, it is possible to achieve this rate without difficulty. The speed at which the auction runs is one of the more important factors in the game. It is the responsibility of the instructor to conduct the auction at a brisk pace. Toward the end of the auction, it is possible that some players will have exhausted their budgets. This could result in a single bidder buying large quantities of goods at extremely low prices. In this situation, the auctioneer can set a minimum bid of $50 or $100 per lot of goods to prevent excessive profiteering.

Once the auction is over, the players are given time to create complete meals with the raw materials they have available. No partial menus are allowed. Because each of the raw materials is used in each menu, it is possible to create many different combinations of menus, each with its own unique level of profitability.

The program is simple enough that the integer programming is not necessary to require that students use such mathematical techniques. Students are usually able to create a set of menus that are close to the optimal solution. After the players have calculated their portion of raw materials and the profit they would make by selling the meals, a winner can be selected. It is also possible to classify if the optimal mix of outputs for each player is found. This can be done fairly easily with a PC computer and a spreadsheet application. Most spreadsheets include a linear programming module with the option for integer outputs. This will help the class to use and practice power of mathematical programming in real-life applications.

The author said, “It may seem surprising, but players often achieve the optimal solution. This is not due to any particular mathematical expertise on the part of the students, however. The players often try to stick to their initial strategy and buy only certain quantities of the raw materials. There is one solution and no waste in this case. The instructor may choose to enrich the allocation experience by awarding each group an additional lot of each of the raw materials, an extra lot of vegetables, an extra lot of meat, and so on. This often will have the effect of changing the optimal solution and requiring students to consider different allocation strategies” (p413). When the class finds the optimal solution, the teacher can demonstrate integer programming. Therefore, the instructor can use the game to set up the following lecture.

After the game is done, a debriefing can review and conclude the games. There are three majors lessons: the auction, strategic and tactical planning, and the allocation process. The teacher may determine if the game will take over one lecture period to complete or not.

For inexperienced students, the game examines the nature of a competitive market environment. The market is characterized by the dynamic interaction of participants, and the students must learn to interpret information that relates to the market. Occasionally, the competitive attitude might develop suboptimal performances due to team's trying to "beat" another by outbidding for a needed commodity. For example, “a team that tries to get all its raw materials early in the auction will often overpay because the market prices have not yet been established. Teams that “sit back and watch” for a few bids have a better idea of the fair price of the raw material and make more reasoned bids and higher profits. This demonstrates that those firms that choose to be industry leaders may not perform as well as firms that choose to enter an established market"(Brozik and Zapalska, 2000, p 417)

Students learn to change their initial strategic plan because of the dynamic nature of the market. If they recognize the opportunity, they can change their strategy to offer high-priced meals and obtain the additional needed raw materials at low prices. This shows a dynamic and adaptive market strategy.

The optimal solution guides students to understand how mathematical techniques can be used in a business environment. An instructor can also illustrate the sensitivity of optimally mixing the inputs, and students can be shown that seemingly minor changes in a mix of inputs can significantly alter the optimal mix of outputs.

Conclusion of games reviewed

These games reviewed give an overview of the simulation game market evaluating both the commercial and educational sectors. Instructors usually prefer to facilitate and use real case situations in their class rather than lecture. Although simulation games are vastly used in the classroom, some researches and articles have pointed out the inherent problems of simulation gaming.

Complexity of Simulation Game

The game's complexity designated the optimum team size for playing. It confirmed that students did not truly understand each other in their group about performing the decisions. Some professors found it was crucial to allow geographical location of team members to be an overriding power when working with part-time MBA students, primarily to facilitate meetings. With e-mail and fax availability, they had found that to be unnecessary (Gentry, 1980; Lucas, 1979; Wolfe and Chacko, 1983). Another point relating to term is cohesiveness versus team conflict. Gentry (1980) concluded that larger teams experienced higher levels of conflict.

Rollier (1992) concluded that the small portion of time advances was optimal in total enterprise simulation over entire duration. Time could be fixed or flexibly scaled, synchronized or unsynchronized among participants, and driven either by the administrator, the participants, the clock, or the level of activity. Flexible scaling allows participants to select the length of successively shackled periods or to independently complete specific decision rules that are effective over any interval. Synchronization affects interrelationship among participants. Gaming simulations with independent decision making by participants are typically unsynchronized. On the other hand, dependent decisions made by participants, are typically synchronized.

The participant with more experience may have a competitive advantage--the reality of business. An administrator-driven simulation inconveniently requires the administrator to gather and run participants' decisions, leaving to the simulating program the simple task of assimilating interdependent decisions. According to the administrator-driven simulation, the programmer must code complex routines that enable the simulating program to manage multiple real-time runs on networked computers, or participants must be restricted to running the simulation on a single designated computer. A clock-driven simulation advances time in concert with the computer's internal clock. An activity-driven simulation captures the advantages of the clock-driven simulation without that design's inherent inadaptability. In theory, an activity-driven simulation advances time when participants are active. It pauses during breaks because they are intervals without activity. An activity-driven design, however, is difficult to program and not easily understood (Thavikulwat, 1996).

Simple functions that plausibly model complex phenomena are convenient. As Wolfe and Jackson (1989) have observed, learning does not increase monotonically with the simulation's complication. If learning increased with usage, and usage increased with convenience, convenience might be the most significant factor. Easy administration depends on the design of the gaming simulation, the adequacy of the computer software, and the availability of suitable hardware.

The complexities can be flexible. By design, the instructor is permitted to set the pace, the complexity, and even the pedagogical focus of the gaming simulation (Thavikulwat, 1988).

From one viewpoint, Stanislaw (1986) concluded that a simulation attempts to duplicate the activities of a real-world system with a computer program. From this viewpoint, reality is the ideal; a simulation that duplicates reality with more reliability is better than one that duplicates reality with less reliability. Thus complex simulations are favored, for they better mirror the complexity of reality (Thavikulwat, 1991).

Because of the complexity’s contribution to learning effectiveness, it has been interested (Butler, Pray, & Strang, 1997). Those complexities in the game often focus on the number of decisions to be made, observed by Brehmer and Allard (1991). Three components make the business game complex: (a) decision types, (b) number of decisions per cycle, and (c) size of the players’ manual interfaces. They can use these to judge a game complexity with quantitative dimension (Wolfe 1978). According to Wolfe’s comment, the decision cycle time is also an important variable for players, but they do not always pay attention to it (Keys and Wolf 1990). However, the time cycle factor tends to make the game more complex. It is considered a process-related factors rather than a structural factor. The business games carefully and closely link two factors, time cycle and number of decisions (Keys and Wolfe 1990). New paradigm of simulations were developed to add a dynamic aspect to the teaching and to provide more involvement of instructors' ideas to present to their classes.

Other Views of Simulation Game

The first view is concerned with simulation games accelerating and changing in various characteristics to try and to cover all business functions (Ivo and Don, 1999). Although the superficial goal of the business simulation is obtain the highest profit, the true goal of learning is to know and understanding the business rules.   (Henshaw and Jackson, 1978). The goals set by the game are hardly considered in large corporations such as Coca-Cola, yet the corporation evaluations are regularly used in making quality managerial decisions (Jusson, 1982).

In the earlier research, the conclusion drawn was that the performance of what could be used to predict the academic success or to appraise managerial potential (Vance and Gray, 1967).

The various environments cause some performance requirements in performing, such as the firm's assets at start of the game. These are not synchronized to situations in the real world. The conclusion in some reports was that the characteristic of simulation was limited by financial measure instead of using human skill. Teach (1990) pinpoints the period over time, which profits are accumulated in the most business simulation, is too short to used profit to provide adequate measurement. The same as association for business and experiential learning meeting (1986) posted the conclusion that it had been clamed the top managers were judged on their ability of bottom-line profits. The utilization and profits, as an evaluation tool, create too many obstacles for realistic design. Also, the short-term tactic may negatively affect short-term profit but it leads the long-term position. The example has been given as the following case: "… the tactic buying market share early in production development and commercialization has proven to have long-run benefits to many actual firms" (Teach, 1990, p13).

CHAPTER 3

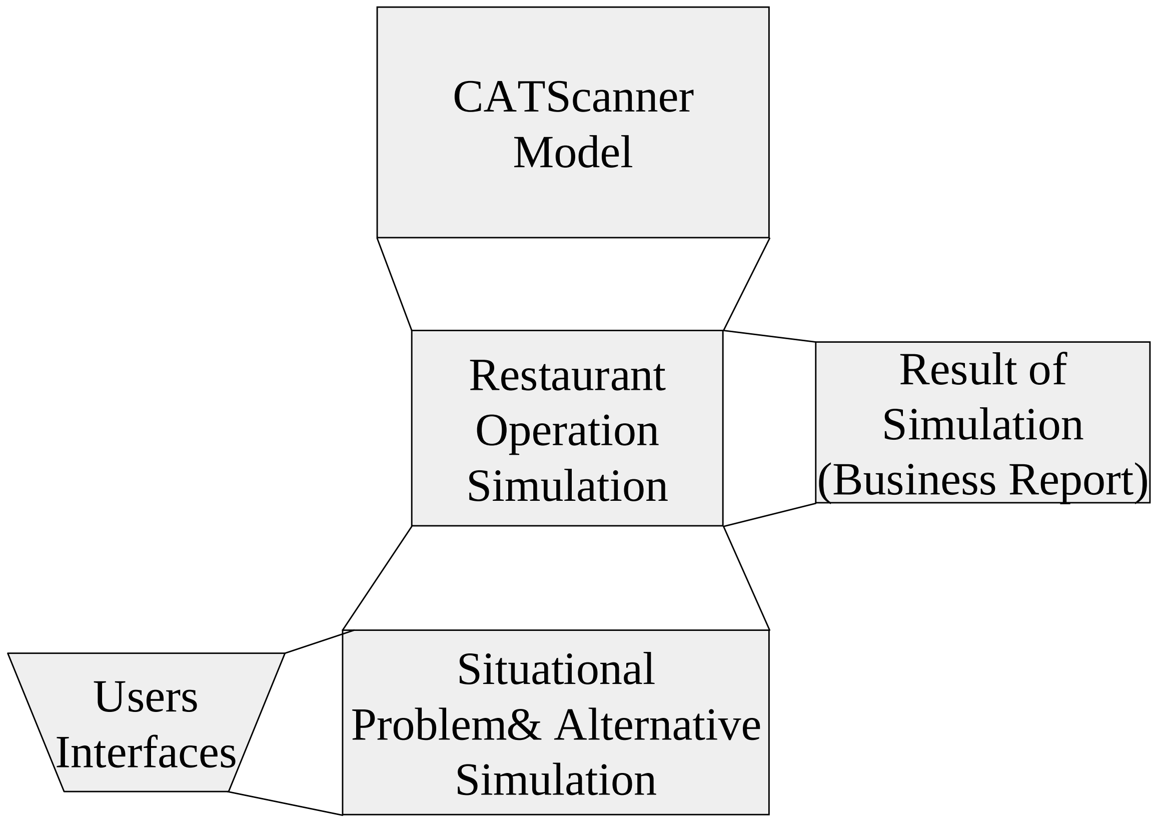
METHODOLOGY

The simulation game represents a tool to help students understand fundamentals business concepts. The game style was adapted in response to the simulation games' problems which had been reviewed in Chapter 2. This chapter investigates methodology used for the game development. The result of the development is a game based on the rudimentary business concepts and the simulation technique. The development process uses software engineering as a tool is explained in the following section. The chapter is divided into three sections. The first section explores the game concept. The second explains how to use the tools to develop the simulation game, given the elements of processes, data structure, and user interfaces. The final section discussed the game's validity.

Game Concept

The first step of the game's development is to identify the game's concept in turn, modeling the logical system. The system's characteristics and architecture are determined in this section. The general idea of a business and the specific type of business operation are considered to add to the game and it effectively showed to the user. This means the CATScanner model represents the general idea, and the pizza restaurant operation represents the specific case study. The following diagram shows the game's elements:

Figure 3.1. Game Concept Diagram


CATScanner Model
Restaurant Operation Simulation
Result of Simulation (Business Report)
Users Interfaces
Situational Problem& Alternative Simulation

The first element in the diagram is the CATscanner model, and it shows a fundamental business concept that is surrounded by internal and external environments, as mentioned in Chapter 1. The environments of the business are related to the factors that impact the profit shown in the report statements. The diagram shows the restaurant simulation, which is a prototype of the CATscanner model and refers to a type of business. The first simulation includes the general business processes, which are automatically run. The characteristics of the game are divided into two parts, students and instructor, and are related to the second simulation, the situational simulation. This idea provides for the instructor to create a scenario or case study, a problem, and a set of solutions. The situational simulation is also linked to the operational simulation. The student and instructor have their own graphic user interfaces to interact with the simulations.

The first simulation brings the idea of a working routine that utilizes the computer timer technique, a set of commands that work in every time period. This first simulation formulates the operational functions, which are manipulated by the business factors. The factor elements use the same platform, which is numerical data, in the second simulation. The situational scenario, therefore, includes mathematical formulas to determine the factors' value. The instructor can create a situation and a set of alternatives in which only one is the best. This is the fundamental idea of the game system.

Development of the Simulation Game

The three sections includs in this part are analysis, design, and software requirement.

Analysis

The game development refers to the software engineering's function, software analysis. This step of development defines what a given application will do based on two types of software engineering methodologies: process-oriented and data-oriented. The process-oriented method is used to analyze the restaurant simulation, but the data-oriented method is used to analyze the situational simulation. The process methodologies take a structured, top-down approach to evaluate the sequential processes and the data flow diagrams with which they are connected. In the data-oriented methodology, the development starts with an analysis of the processes' relationships to determine and underline the data architecture. Some data processing activities or data activities, such as insert and update, are considered, and these activities are grouped into the data flow diagram.

The restaurant simulation is developed according to the business rules in a restaurant operation. The diagram is created according to the process that characterizes a generic business's functional tasks. The data flow diagram is used in this process for graphically representing the application's components that connect processes to other processes, stored data, and identified external entities such as the human interface. The structured composition technique is used to analyze the individual parts, allowing us to identify and isolate small parts of a problem. Because the restaurant operation is designed to be an automatic process, the input element of the process connects to a database interface, not a manual input interface. The database's tables then are identified and related to the diagram; these details were in the appendix section.

The situational simulation is developed when the business factors' structure is determined. The structure element depends on data analysis. It is based on relational database theories, seeking to understand the meaning behind the business factors in the simulation. The data relationship diagrams in this process then are generated. By using standard business concepts, the relationship rules are generated. This development does not accommodate time, has no implied sequence to the processes, and assigns data in analysis without deliberation. The results of the development are attached in the appendix section.

Design

The third step of the development is the design phase of the game, which uses the software engineering technique. The design phase translates "what" the system is supposed to do into “how” the system does it in the particular software and hardware configuration. During this phase, the process generates software components, modules, interfaces, and database’s tables. The details of this phase also verify application control flow, data structure, algorithms, and program components.

The restaurant simulation is designed according to the process-oriented design. The process of design starts with developing a structure chart, a physical database and program specification. The structure chart is the result of transforming the data flow diagram with its functional elements and identifying the clustering of subprocess based on major business functions. The functions are either input, process, or output. The input function mainly retrieves data and prepares it for the process. The process functions transform data values by mathematic functions, according to the business rules. The output functions send the data either to storage, another process or screen. The algorithm then is generated to describe the data structure element. At the moment, the screen input and output are created for this simulation. These are shown in the appendix.

The second part of the game, the situational simulation is designed using the data-oriented concept. This emphasizes on design control, human interfaces, and a reconfirmed database definition. Because the data flow diagram in the second is dependent, the structure to control the process is a matrix process, using the same level, not the sequence process, or the human interface. The human interface is the groups of graphic interfaces, which are designed on the screen and based on developing software. The processes’ algorithm and graphic interfaces are also attached in appendix section.

Software Requirement

After the game system is identified in detail as the analysis and design parts, the application's requirements are determined. Visual Basic version 6 (VB) is selected, which is the developing software tool for creating the game. The features of the VB are as follows: 1) easy to create graphic user interfaces, 2) ability to use SQL command to access database, 3) ability to create object-oriented interface, and 4) computable to almost all computers in the world. The game uses Microsoft Access for storing data element in its tables. The last requirement obviously is a Microsoft Windows operating system. The requirements are fit to the game system and are easy to maintain.

Game Validity

The game's validity is concerned with the standard business concepts that constitute the class's business knowledge. This means the result of the game tests both the process and the output data from the input data. This validity process is shown in the process description, the results of which discussed in Chapter 4

CHAPTER 4

RESULTS AND FINDINGS

This chapter will present the results and discuss the findings. The results were determined in the following order:

1. Outlook of Game

Graphical Interface of Simulation Game

Graphical Interface of Instructor Model

2. Running Simulation Game

2.1 Running Restaurant Simulation

2.1.1 Main Modules

2.1.2 Restaurant Simulation

2.1.3 Restaurant Operation Monthly Report

2.1.4 Business Situation and Alternatives

2.2 Running Situational Simulation

3. Simulation Study Case

3.1 Beginning Simulation

3.2 Situations at the End of the First Month

3.3 Comparing results of the running simulation

Outlook of Game

A pizza restaurant is selected to represent the general business concepts. As the result of the development, the interfaces are divided into two parts, student and instructor.

In the first part of the game, the processes of the restaurant operation are simulated behind user interfaces, meaning that the user will see the results of running processes on the screen. The users learn the business concept by the game process. The game shows various situational problems and the user selects the best alternative decision of solving the problems. In the other part, instructors can create a situational simulation, which is internal or external environmental conditions in the restaurant business, and situation alternatives for a student to make the best decision. The game uses images, text boxes, combo boxes, and list boxes in windows to interact with the users.

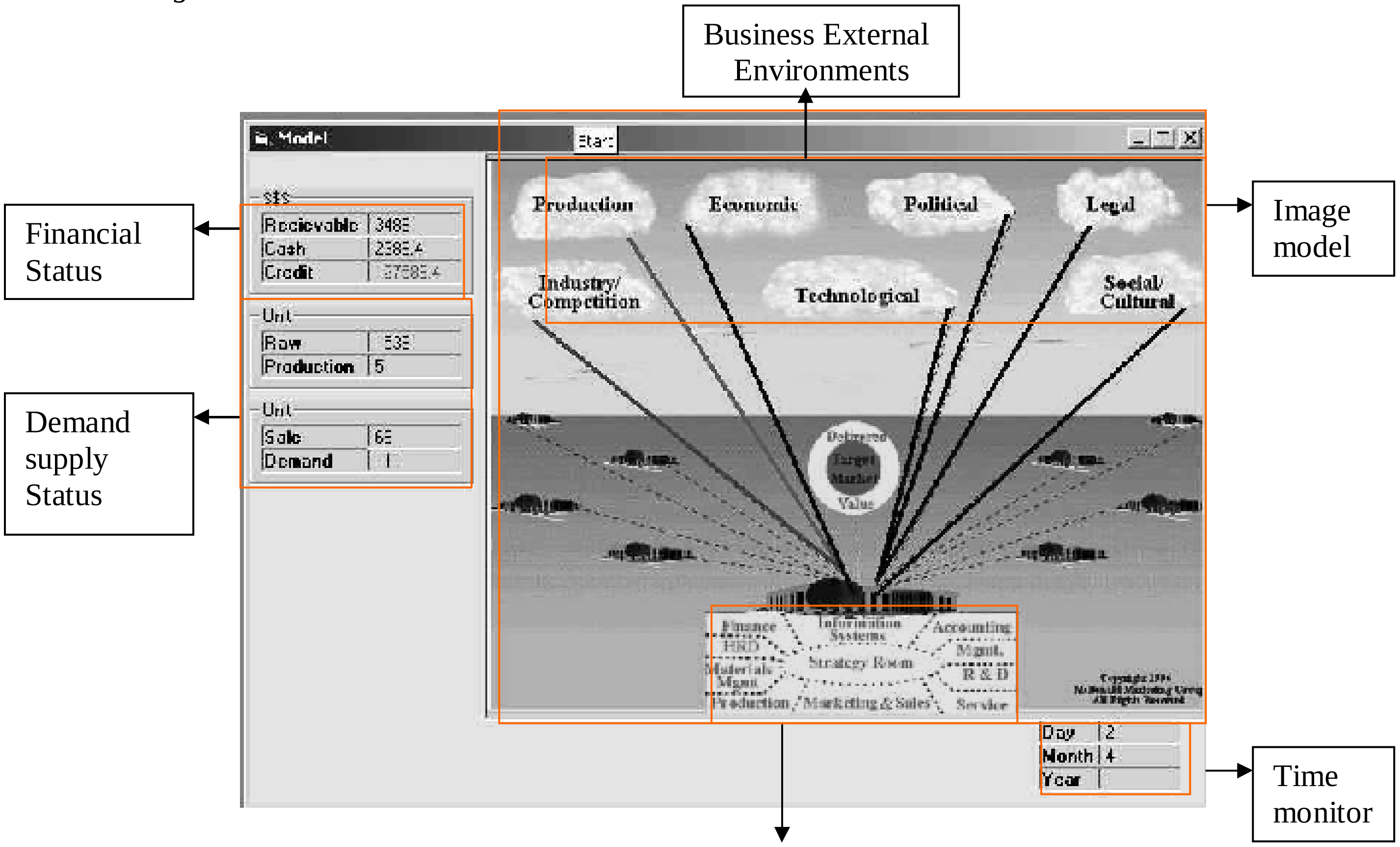
4.1.1 Graphical Interface of Simulation Game

The first part of the game is designed for students in a business class. The main objective in this part is to operate the restaurant simulation and to make the best decision for a given situational problem. The goal of the business simulation is to generate maximum profit, which is measured by monthly earnings. Every alternative brings different consequences for the user's simulation, varying from cutting operation costs to stimulating the restaurant's market share. There are four graphic user interfaces (GUI) in this part. The first two GUIs contain information about the operation, telling the status of the business. The next interface is a report of the operation. The last interface shows a situational problem and alternatives from which a student can select one.

The first graphic user interface is a window that contains running numbers of factors in the operation, timing values to indicate the game time, and the model of the business. The user will consider those numbers in making a decision about the given business situational problem. The CATscanner model on the right side of the form (Figure 4.1) depicts the relationship between the business and the internal and external environments that impact the organization. Internal environments are generic business functions in an organization which consists of production, sales, finance, accounting, human resources, research and development, management, information systems, and strategic planning. The external environments are grouped into seven clouds: production input, industry and competition, economic, technology, political, legal, and social and culture. The relationship is represented by a connection line from the clouds to the organization body.

There are three frames, components of the GUI, on the left side (Figure 4.1). All frames contain text boxes in which data cannot be changed by the user. The first frame shows the amount of cash available, the amount of check received, and the credit balance with suppliers in the text boxes. The second frame attaches supply information. These text boxes show the number of materials and products on hand. The last frame gives the total amount of sales and the customer demand number, which means the amount of pizzas' demanded from the restaurant, in their text boxes. On the bottom right, there are three boxes representing time values, which are day, month, and year. In the internal environment area of the modeling image, the user can click to open the "business's factors" form (second GUI), yet the report interface shows on the screen at the end of every month of game time.

Figure 4.1. CATscanner Screen.



$$$

Recievable	3483
Cash	2383.4
Credit	27583.4

Unit

Raw	533
Production	5

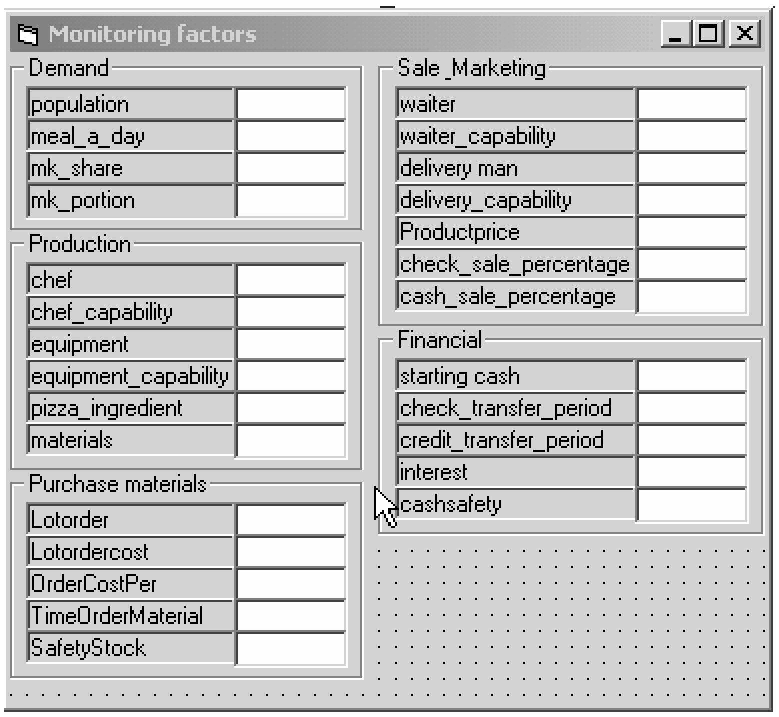
Unit

Sale	63
Demand	1

Day	2
Month	4
Year	

The “business’s factors” form (Figure 4.2) contains the factors’ names and values, so the user can monitor the information concerning the restaurant’s operation. The factors’ names are in a gray-colored box, but the factors’ values are in a white-colored box. The factors are divided into five sections as shown on the Figure 4.2. The screen is shown when the user clicks the mouse on the business’s internal area of CATscanner screen.

Figure 4.2. Monitoring Factor screen.



Monitoring factors

Demand

population	
meal_a_day	
mk_share	
mk_portion	

Production

chef	
chef_capability	
equipment	
equipment_capability	
pizza_ingredient	
materials	

Purchase materials

Lotorder	
Lotordercost	
OrderCostPer	
TimeOrderMaterial	
SafetyStock	

Sale_Marketing

waiter	
waiter_capability	
delivery man	
delivery_capability	
Productprice	
check_sale_percentage	
cash_sale_percentage	

Financial

starting cash	
check_transfer_period	
credit_transfer_period	
interest	
cashsafety	

The data report, as shown in Figure 4.3, consists of two account statements, the Income Statement and the Statement of Owner’s Equity. This report is automatically shown at the end of each month.

The Income Statement carries sale revenue and cost expenses. A section line partitions the revenue and cost expenses. The numbers are the result of revenue (the amount of total sale) and the following expense items:

1. Cost of goods sold is the same as the cost of ordered materials.

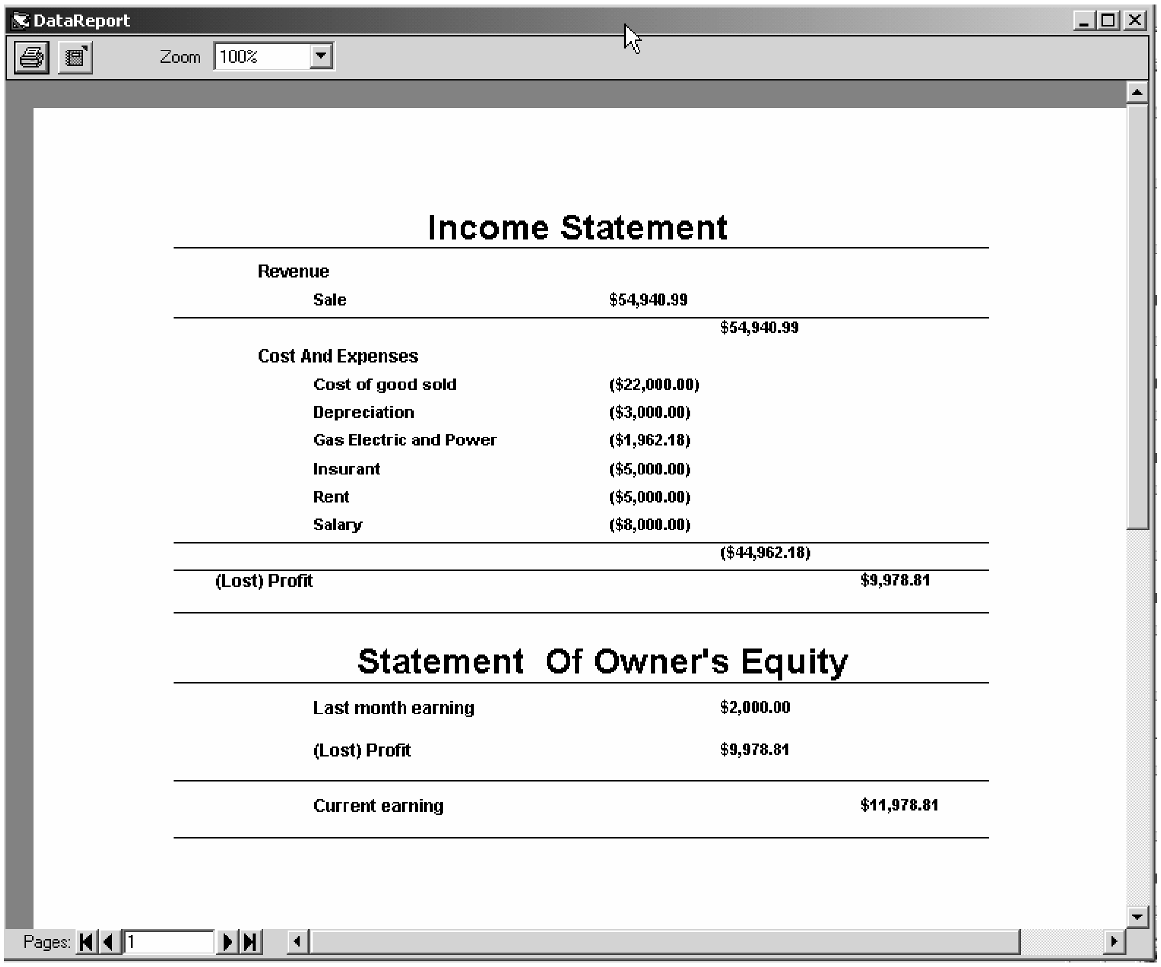
2. Depreciation is a quantified allowance made for a loss in value of equipments and tools.
3. Gas, Electric, and Power are a variable cost that is dependent upon production.
4. Insurance is the cost of properties and equipment being insured.
5. Rent is the cost of the restaurant building.
6. The salary is a cost for all employees.

At the end of each section, a sum of the section's items is shown on the right side, under the line. The loss or profit is calculated and shown on the bottom of this statement.

The Statement of Owner's Equity includes the business's accumulated earning value. There are two lines in this section. The first line is the amount of accumulated earning up to the previous month in the game's time, and the other is the loss or profit of the current month. Then, the new earning's number, which is a summation of all items in this section, is shown in the last line of the statement. The additional information is included in the running simulation section.

The user can print out the result by pressing the print button or save it to a file by using save button, both of which are on the top of the report. The user can also zoom in and out of the report by selecting the zoom percent in the combo box, which is a drop down control containing different choices. The following, Figure 4.3, shows an image of the report. After closing the report, the situation window will appear on the screen.

Figure 4.3. Monthly Report Screen



Income Statement

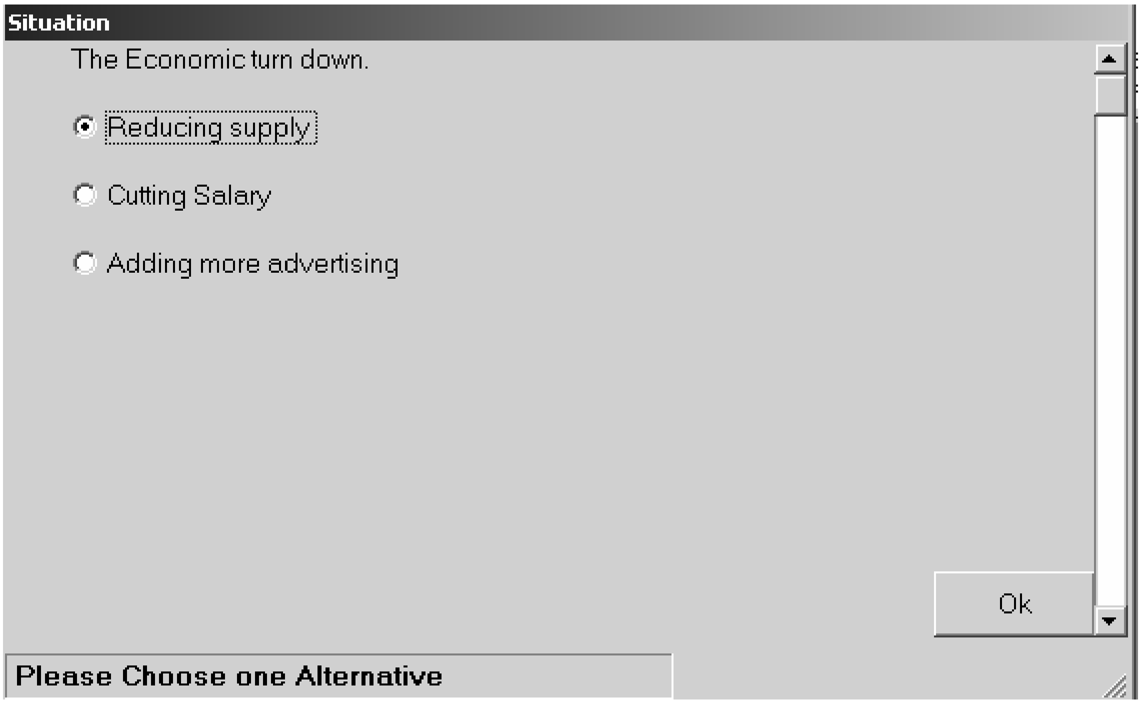
Revenue			
Sale	$54,940.99		
		$54,940.99	
Cost And Expenses			
Cost of good sold	($22,000.00)		
Depreciation	($3,000.00)		
Gas Electric and Power	($1,962.18)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($44,962.18)	
(Lost) Profit			$9,978.81

Statement Of Owner's Equity

Last month earning	$2,000.00	
(Lost) Profit	$9,978.81	
Current earning		$11,978.81

The fourth GUI is a window which contains three components: a situational problem, its solutions, and a confirm button. The user can read text data in the first part. The text described the situation or problem. This is followed by the next component. It provides alternatives to the problem. They are in option boxes, which the user can click on choose a particular answer. The users need the knowledge that they have learned in their business classes to consider each alternative (Figure 4.4). The “OK” button is provided to confirm their answer. The situation and alternative choice will change the factor variables, some of which are shown in Figure 4.2. They will affect the next result, and the best alternative in this round will be explained after closing the following month's report window.

Figure 4.4. Situational Decision screen.



The Economic turn down.

[✓ Reducing supply]

Cutting Salary

Adding more advertising

Please Choose one Alternative

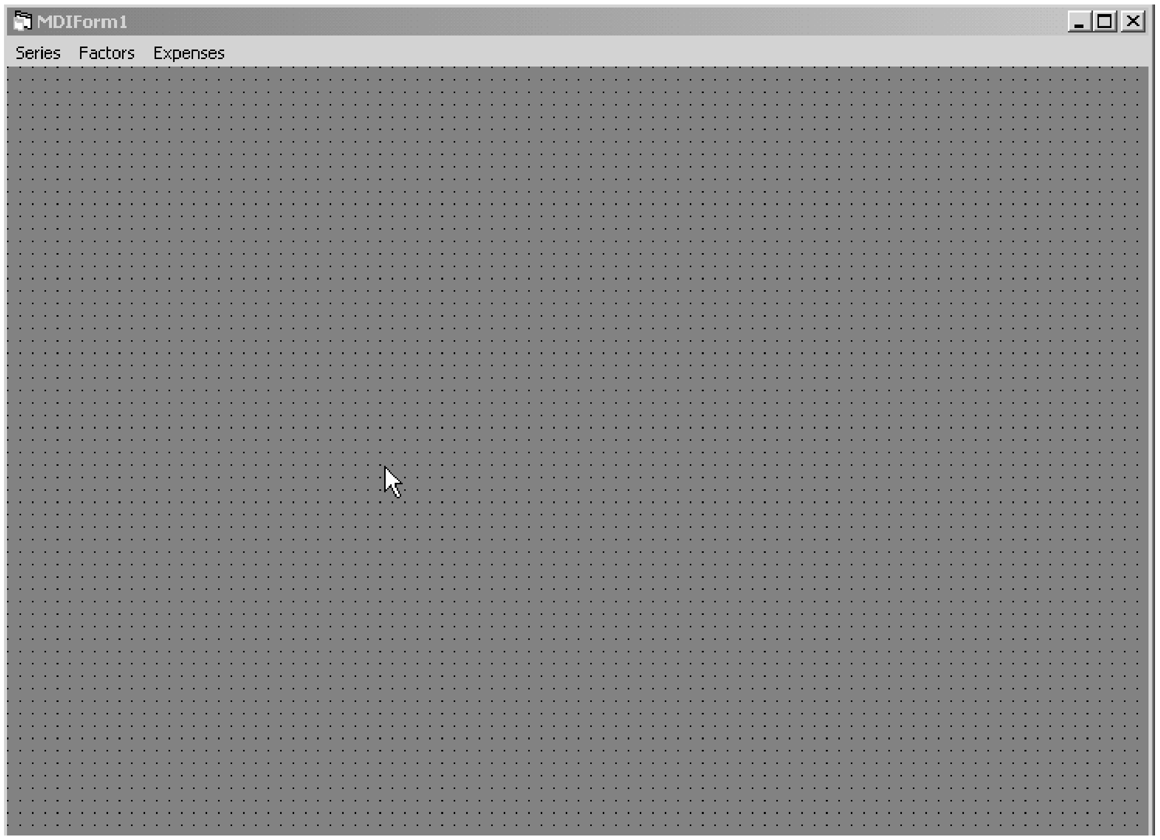
4.1.2 Graphical Interface of Instructor Model

The second part of the game is designed for the instructor of a business class. The main objective in this part is to create a situational simulation and its alternatives which impacted the restaurant simulation, the first part of the game. There were two areas in which the situation can change the model's data: the first area is in the business factors, and the second area is in the business expenses. A situation and its alternatives can have many influences in those areas. The task of an instructor is to come up with a reasonable series of situations and its alternatives. All situational problems and alternatives impact business factors and expenses. When this paper refers to a situation, it means a situational problem of the business. The alternatives obviously are the solutions of the problem. A series refers to a group of situations which comprise a problem situation and its alternative, grouping by the same numbers of month and year.

The task of creating a situational simulation is completed in four GUIs. The first GUI is called the Multiform Document Interface (MDI); it contains other windows it and has menu items that control each of those inside. The second GUI is the main situation control. This window is used to control the situation series and the two object interfaces in that main window, which related to the factor and expense windows.

The first window contains the menu bar object. The users can easily choose the menu item to do what they want to do by controlling the child windows. The child windows are other three windows: “main control,” “situation related factors,” and “situation related to expenses.” They are loaded into the MDI window following the menu item’s action. There are three items to control in each window, and all sub items in the menu are the same. The sub items consist of insert, update, and delete. The action of the main’s sub item acts according to the word in its area. For instance, the sub item “series inserts” new record to a series situation table.

Figure 4.5. Main Menu screen.



The main control window is added to the MDI window when the program started. This window includes four control areas, which are "recordset" control, situational contents control, situation related to factors control, and situation related to expenses control. "Recordset" is referred to as a pointer in the table's record of the database, which is an index of table's column. The "recordset" control has five object interface elements.

1 Button "|<" represents the first record.

2. Button "<" represents previous record.

3. Text box represents the current record as a number out of the total number of records.

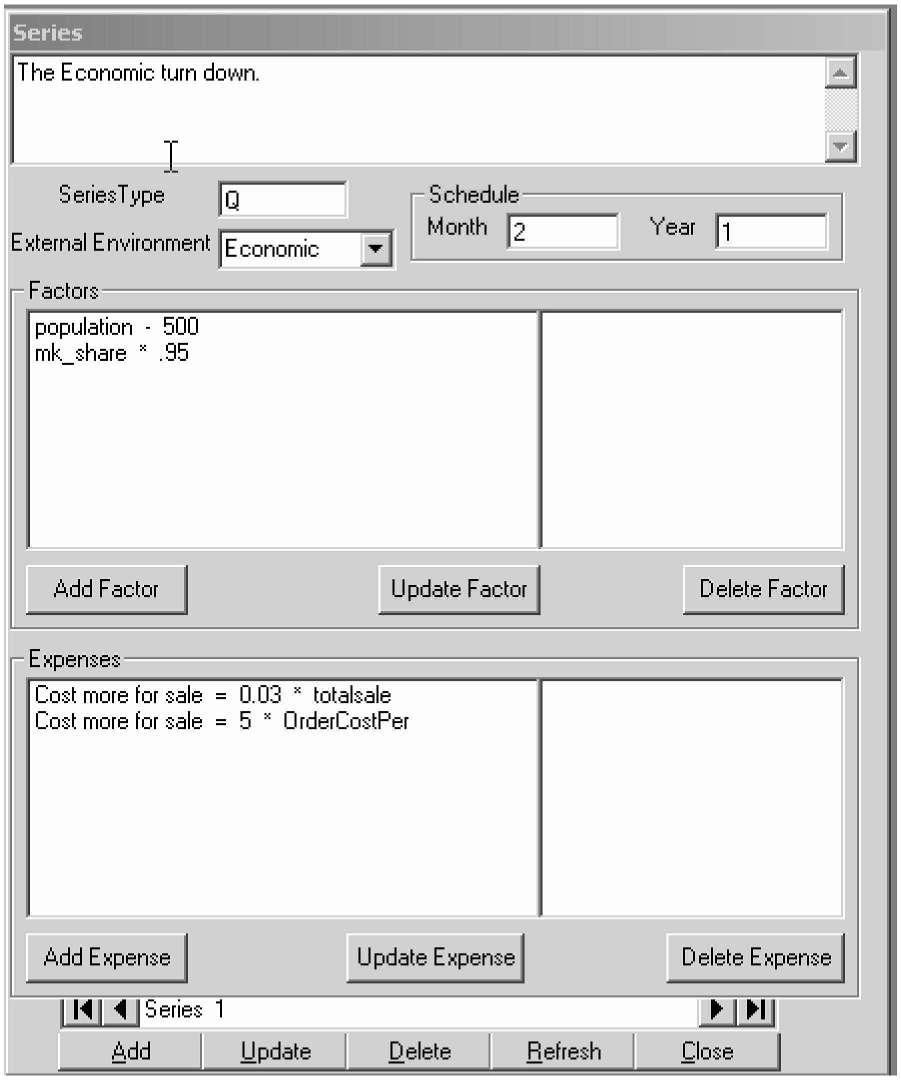
4. Button ">" represents next record.

5. Button ">|" represent the last record

When the user clicks on the buttons, the “recordset” moves a position, following the buttons as represented above. The users can refer to the text box element to know which record they are working.

The situational content area is composed of six objects, five text boxes and a combo box. The first text box represents the description of the situation and is located at the top of the window. The type of situation identifies a series by a character. The character “Q” represents a situational problem while the character “A” represents an alternative. The schedule frame, next to the situation type, contains two text boxes, month and year. These two text boxes are used to get the number of the particular month and year. The numbers are used as the timer to match the situational simulation and the restaurant simulation. For example, if the month and year are the number 2 and 1, then, the schedule is set to February of the first year of the restaurant simulation (Figure 4.1). The situation impacts the restaurant simulation in that month. The last object, combo box, indicates which external environment group fits to the situation. The user can select one environment which is shown in the restaurant simulation model. The last text box is an option box. This is visible when the series situation type is “Q.” The user can insert text to explain the best answer or alternative for this situation.

Figure 4.6 Control Situation screen.



The Economic turn down.

SeriesType: Q

External Environment: Economic

Schedule — Month: 2 Year: 1

Factors

population - 500
mk_share * .95

Expenses

Cost more for sale = 0.03 * totalsale
Cost more for sale = 5 * OrderCostPer

Series 1

The next area is the situation related to factors. This area enables the user to add factors that are impacted by the situation and determine how it will be impacted. As mentioned earlier, a situation can have many impact’s factors. The three buttons in this area will control those factors in the situation by inserting, updating, and deleting. After pressing the buttons, the factor window will operate. The last area in the main situation window deals with “situation related expenses.” This area has the same structure as the “situation related factor” area. There is one list box to monitor expense items and three button boxes to control it.

The following, Figure 4.7, is the situational factor form. The title of this window will tell whether it is in an insertion, update or deletion. The user may opt to set the factor in the first combo box. When the user selects the factor, the textbox down below will show the details, illustrating what the function of the factor is. The second combo contains the operational symbols for a mathematic operation. It is available for the user to select as well. The text box, next to the operational symbols box, is for a number to complete the mathematical calculation for the factor. After the user fills out all data, they may click the ‘ok’ button to select or ‘cancel’ to deny the data. For example, if the factor is population, the operational symbol is a plus, sign and the calculation number is 300, then the new population value is the last population plus 300.

Figure 4.7. Insert Factor screen

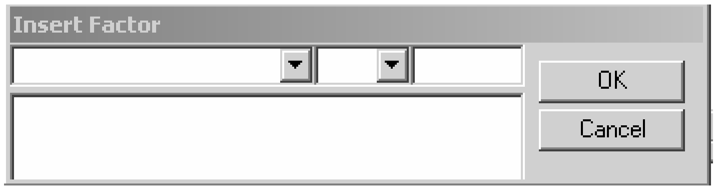
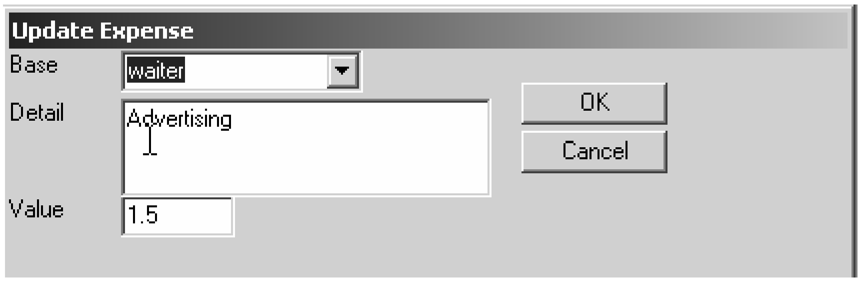


Figure 4.8 shows the situational expense window. The title of this window will tell the status of the window whether it will insert, update, or delete. The user can select the base factor in the first combo box or drop-down menu box. This factor will be multiplied by the value number that is in the last textbox. The textbox ‘detail’ will be used to retrieve the expense text that will be shown in the student report.

Figure 4.8. Update Expense screen.



Base: waiter

Detail: Advertising

Value: 1.5

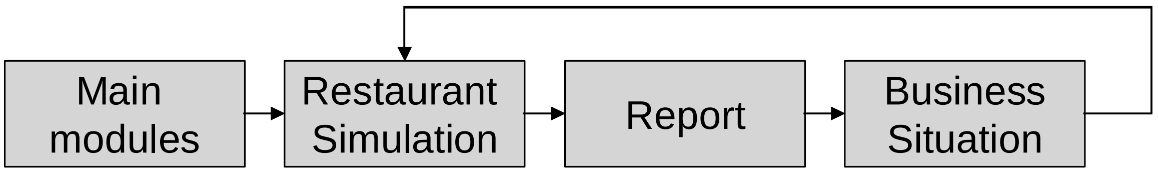
4.2. Running the Simulation Game

This section will sequentially describe all processes, each of which is called a module. It will determine the modules' tasks and their functions, which are controlled by programming statements and programming variables. The modules include some mathematic functions composed of business factors. Some module characteristics are related to graph interface behavior. For example, the click event in a button box will contain codes called the "button_click" module. As mention earlier in chapter 4 section 1, the game's processes are divided into two parts.

4.2.1 Running Restaurant Simulation

The process of the first part is shown in Figure 4.9. The square symbol represents the programming modules grouped by their interface object. The arrow links between modules. This part starts with the main module, followed by the simulation of the restaurant operation, the business report, and the business situation.

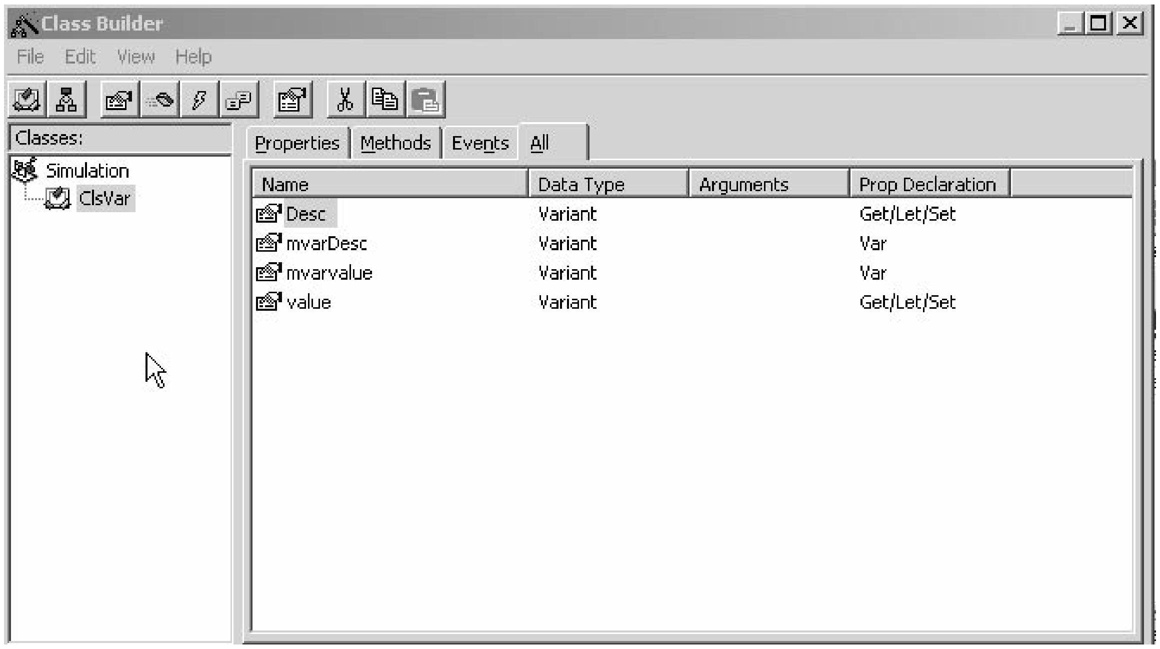
Figure 4.9. The Simulation Game Process.



4.2.1.1 Main Modules

The “main modules” includes the “Getfactors” sub module, the “Savedata” sub module, and the “getmonthlyexpense” sub module. The main module contains two tasks. Before this can be explained, the process of initiating data variable must be described. The business factors used in the program’s modules are declared with the same structure. The object variables constructed by the class object, is shown in the Figure 4.10.

Figure 4.10. Database Class Object.



Name	Data Type	Arguments	Prop Declaration
Desc	Variant		Get/Let/Set
mvarDesc	Variant		Var
mvarvalue	Variant		Var
value	Variant		Get/Let/Set

The class object consists of two properties, “value” and “desc.” The “value” property is a numeric type variable, and the “desc” property is a text type variable. The main module calls on the “Getfactor” sub module to retrieve the factor value from the database in “current table” (appendix, data structure 1). The data object is declared and named as the business factor’s name yet the value and description of the factor will be set according to the object’s properties (definition of term). The advantages of using the object variable are as follows: 1) to work quickly in the memory level, 2) to easily control the object variable using the name of the factors, and 3) to flexibly add more factors by inserting them into the database. The main module then calls the “restaurantsim” window, which is the second task.

These following two sub modules serve as the database interface. The codes inside the module will contain the SQL command to connect to the database. The first module, “saveData,” contains a loop to check for all of the object variables. In the loop, the properties of the object variables will be saved to database table, “CurrentData.”

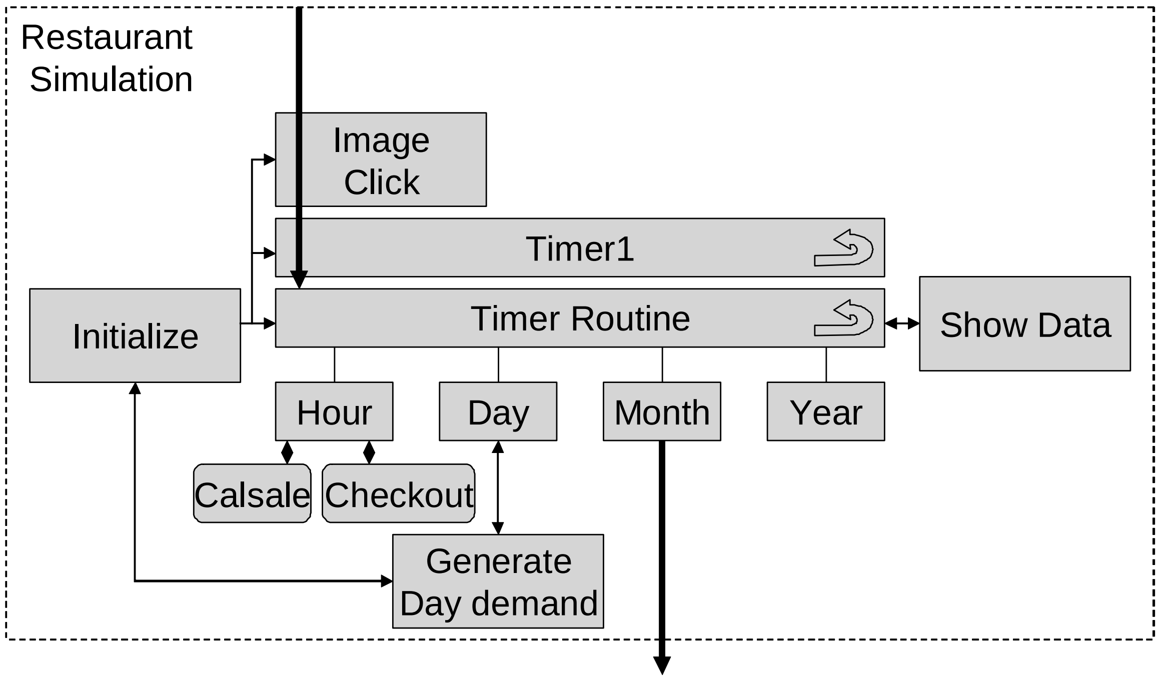
The second sub module, in the first modules group, is used to prepare the business expenses for each month in the game time. The SQL command is used to calculate the total expense from current expenses and situational expenses. The total amount of expense finally is set to the “MonthlyExpense” object variable.

4.2.1.2 Restaurant Simulation

The second group of modules is in the restaurant simulation window. This interface has six modules, two of which are functional modules. The first four modules are related to graphic objects on the window. The last two modules are the user modules, and they are a group of commands used so frequently that they are collected and named to be easily called from anywhere in the game. The functional modules are the user modules that receive parameters for calculation, giving the resulting values back to the variables.

The following, Figure 4.11, is a diagram of the restaurant simulation’s modules. It depicts the modules’ process sequence and is followed by an explanation.

Figure 4.11. Simulation Game: Routinely Process.



The process starts with the initialize module. Then the three modules, which are "timer1_timer," "Imager1_click," and "Timerroutine_timer," are loaded. These are object-event modules that have a special behavior. For timer objects, they work like a schedule, running the timer module in every time period. The "Image1_module" is used when the user clicks on an image in the internal environment area of the business model. The last modules and functional modules support the main four modules as shown in the diagram.

The initialize module first generates a daily customer demand. There are four factors that impact the demand value. The company demand is calculated by multiplying the population, the meals consumed per day, the market portion, and the market share.

The timer routine simulates the restaurant operation tasks in the game time. It uses the "gametime" parameter to check with the timer's clock. When the condition is true, meaning that the two are equal, the process will recount the business time factors. Then, the timer will check condition of the business time factors, which are “business_hours,” “business_day,” “business_month,” and “business_year.”.

Hours: The hourly routine contains production tasks, sales and delivery tasks, and check/cash status tasks. Because tasks in this section are very detailed, they will be represented as a sum.

a) The production routine begins with checking the hourly demand and the amount of product on hand. If the demand exceeds the product on hand, then the materials of operation are checked. The production process will be performed only if there are enough materials. The production is calculated as the amount of pizzas produced in that hour. Four factors are formulated to the following mathematic function:

$$\boxed{(chef \times chef's\_ability) + (machine \times machine' capacity)}$$

b) The sales routine generates the amount of total sales every hour in game time. The number combines the total sales in the restaurant and the total sales by delivery. The first number, total sales in the restaurant, is calculated within the following conditions.

- If the demand number and quantity of production on hand exceed zero, the process continues. The second condition checks that, if the demand exceeds the production number, the process will check the third condition. If this is not the case, the process will jump to the fourth condition. The third condition checks if the product exceeds the service capability, which is the number of waiters multiplied by the waiter’s ability. In the fourth condition, if the hour's demand number exceeds the "calsale" function, the value of sale is set to the "calsale" value, or if not, the value of sale is set to the hour's demand number. After checking the conditions, the demand number and quantity of product on hand are reduced by the value of sales, yet the revenue is calculated and given to the cash and check value. The revenue is the amount resulting from multiplying the units' sold and individual pizza price.

- The total sale by delivery element simulates the pizza delivery system. The routine generates a random number as a time based on the maximum distance of travel for every delivery person. This process is referred to as a shipment of pizza. The routine will generate new random time for that delivery person everyday. In the delivery process, the amount of sales will be calculated much like the selling of pizza in the restaurant process, but instead, it replaces the "calculated_sale" function with the "delivery_capability" factor.

c) The last task in the hourly routine is the checking/cash status. This small code contains a condition statement for when the cash number is less than zero. This will allow a process to borrow money and add that amount to the credit balance and the cash number. The amount is dependent upon minimum cash policy for this company, which is called "cashsafty".

Day

There are four tasks in the daily routine. These four tasks are involved in the daily operation, some of which have trigger factors for checking the days to run the tasks.

a) The first task generates the demand number by calling the "generated_day_demand" module as describe earlier.

b) The second task cashes checks. For the restaurant business, when pizzas are sold, the restaurant may sometimes receive checks. There is a policy in the business for when checks will be changed to cash.

c) The third task is the restaurant's debt payment. The restaurant also has a policy dictating schedule for paying the restaurant's debt. The process will check if the company has cash on hand enough to pay the debt. If there is enough, the cash will be reduced by the amount of debt, and the debt number will be set to zero. If there is not enough money, the company will pay as much cash as is available.

d) The daily routine checks the materials to determine if it is time to re-order. The process will run according to the policy on materials' safety stock and the period for materials' delivery. When the materials are less than the safety stock number, the reordering process will run. This will set the starting trigger to count down the delivery time. The new materials will be added to the inventory when they arrive, and the payment system will be performed. The purchase cost is the cost of materials and the cost of ordering them. The company will pay the cost in cash, and the quantity order in that month is updated and recorded to the "norder" factor.

Month

The monthly routine has only two short tasks that are setting up time parameters, and calling the report object.

Year

The last task in the timer routine is to set-up the time parameters.

When the user clicks the image area, the "factor monitoring" window will be shown. The "factor monitoring" window only has one module that sets some values of the object variable data to text box or object interface. The last task of the timer routine is to set the values of the objects on the "restaurantsim" window. These commands are in the "Datashow" module.

4.2.1.3 Restaurant Operation Monthly Report

The third group of modules will mainly create the business report statements, the Income Statement and Statement of Owner's Equity. The process begins with checking account statements. It retrieves data from tables and separates them into revenue and expense group. The "SaveData" will be called to update the "CurrentData" table as described earlier in this chapter. The "GetMonthlyExpense" sub module then is called to collect expenses' data and calculate the total expense. The total expense is subtracted to the cash value given the current cash value. The closing report module will bring up the business situation window.

4.2.1.4 Business Situation and Alternatives

The last group is composed of six modules, five of which are object-event. The first module, "Form_Load," is called when the window is active. The second, "Form_Unload," is called when window is closed. The third module, “Command1_Click,” is in the confirm button, and the module will be called when the user clicks the button. The forth module, “VScroll1_Change,” is the display controller on the screen, for there are times when the user changes the scroll bar’s position if the detail is larger than the window size. The last object-event module, “VScroll1_Scroll,” is called when the mouse is used to control the scroll bar on the window. The last module, the user module, deals with changing data in the tables. The module will locate the factors in the “CurrentData” table, which match the selected alternative, and change their values.

The “Form_Load” module starts with checking for the answer, the best alternative for last month, and showing it. This process searches the data in the situation table, where the numbers of the month and year are the same as that of the previous month and year in the game time, using the SQL command. The process next selects records where the numbers of month and year are the same as the numbers of the current month and year in the game time. When the process retrieves the records in the table as a loop, it creates a new object to the window. If the situation type is a situational problem, the module is declared to a label box, only showing the text. If the module is an alternative situation, the object type is declared to an option box, with a text property, to describe the alternative. The four properties of the “VScroll” object are the final task. The minimum and maximum lines for the window are set to maximum line properties of the scroll bar object; then the change is set.

After the users read the situational problem and alternatives as a text description, they must select an alternative and click “OK” to confirm their answer. This action relates to the “Command1_Click” module. This module will set the factors in the “currentdata” table, which will impact the restaurant simulation process.

When the window is closed, the “Form_Unload” module is called. This module will cause the timer in the restaurant simulation window to start again.

4.2.2 Running Situational Simulation

Because the situational simulation’s tasks are to generate and to organize a business situation which will impact business factors and expenses, the modules are involved in data access control. This part of the game allows the user to insert, delete, and update data in the main three tables: the situation table, the situation related factors table, and the situation related expenses table. The tasks in the GUIs are mainly object-event modules. These will be divided into five groups in the following the next paragraphs.

The first group of modules will setup the application variables. The tasks of these modules load records from the database’s three tables to recordset variables. In general, the modules create database objects in the computer memory as “db” and then refer to table in the database as “rs.” The recordset object is built-in includes some functions to manage its records.

The user can control the situational record by the recordset object. The object contains graphic control elements. The process will automatically move the record to where the user commands wants it to be and set other objects’ value properties, which will map the data field to the recordset. The movements of a record can be divided to four types: move first, move previous, move next, and move last. After moving the record position, the “Data_Reposition” module is called. This module will reload the factor and expense items which relate to the situation number and will check the situation type. If the situation type is a problem type, the situation’s answer field is visible. If the situational type is an answer type, the situation's answer field is invisible. The situation's answer field will contain a text to explain the answer for the situation problem

The user interfaces of the recordset object will be shown as button boxes. There are four buttons in this object insert, delete, update, and refresh.

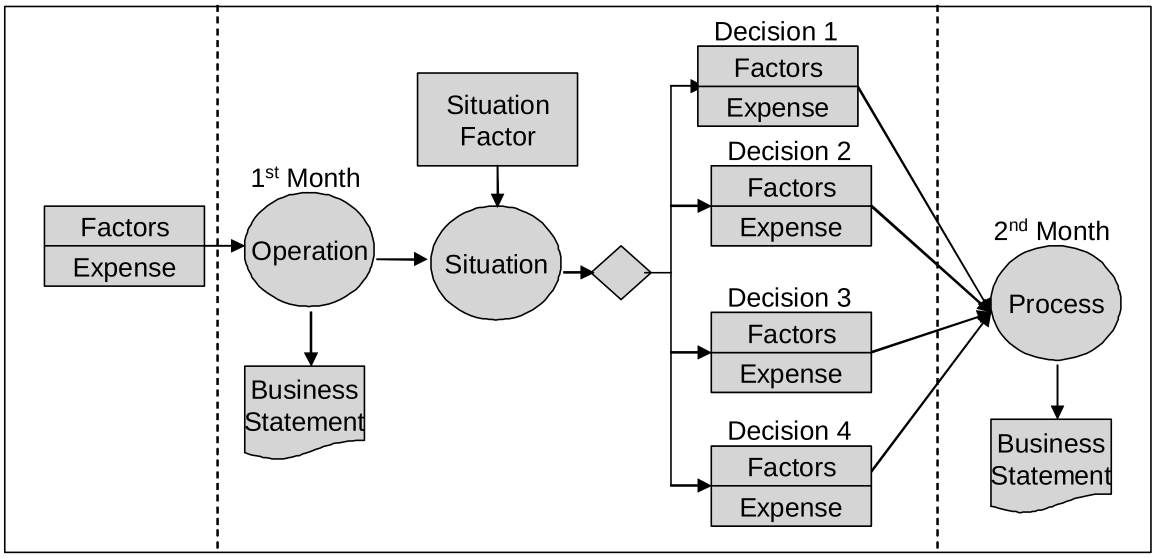
As mentioned earlier, the "situation_factors" table is related to the "situation_factors" element in the situation window in the same manner as the "situation_expense" table is related to the "situation_expenses" element. Each area is composed of three buttons. The "Insert factor" button tells the record in the situation factor table to jump to the last record, and it adds the new record to the table. The "OK" button on the window contains the SQL command to add data from the interface objects to the "situation_factor" table. The "Update_factor" button will set the data, which are selected on the situation window, in the object interface in the "situation_factor" window and show that window. When the user clicks the "OK" button on that window, the process will use the SQL command to delete or add records to the "situation_factor" table. When the user selects "Delete factor," the "situation_factor" window will be shown to confirm the deletion.

The expense's component, in the situation window, uses the same concepts as the factor component containing the same three buttons. The process in the expense modules are the same as the factor modules with the exception of using the "situation_expense" table instead.

4.3. Simulation Study Case

This section will focus on the result of running a simulation. It will investigate the business reports from the first and second month in the simulation game time. The status of the business will be shown in the report and will be based on the situational problem and chosen alternative. The consequences of the situation will impact the factors and expenses of the business. The following paragraphs will show the result of each decision alternative and the details in the tables. The first diagram will show the process of the simulation.

Figure 4.12. Case Study Process.



The diagram shows the factors' table in a rectangular box. It connects to a process in a circular box by an arrow symbol. The alternative case will be expressed by being shown in the factor and expense table's changing data, which will generate the different result in the report. The first and second month processes give the report as a report box that is connected to the circular box. When the first month of the simulation generates a report, the situation decision will be shown. The four decisions in the second month will be considered from which one will be selected. Each selection choice will be related to the resulting impacted factors, which are resulted in the second month report.

4.3.1 Beginning Simulation

Suppose that the restaurant's factors are in the following, Table 4.1, before running the business operation.

Table 4.1. Situation Factor.

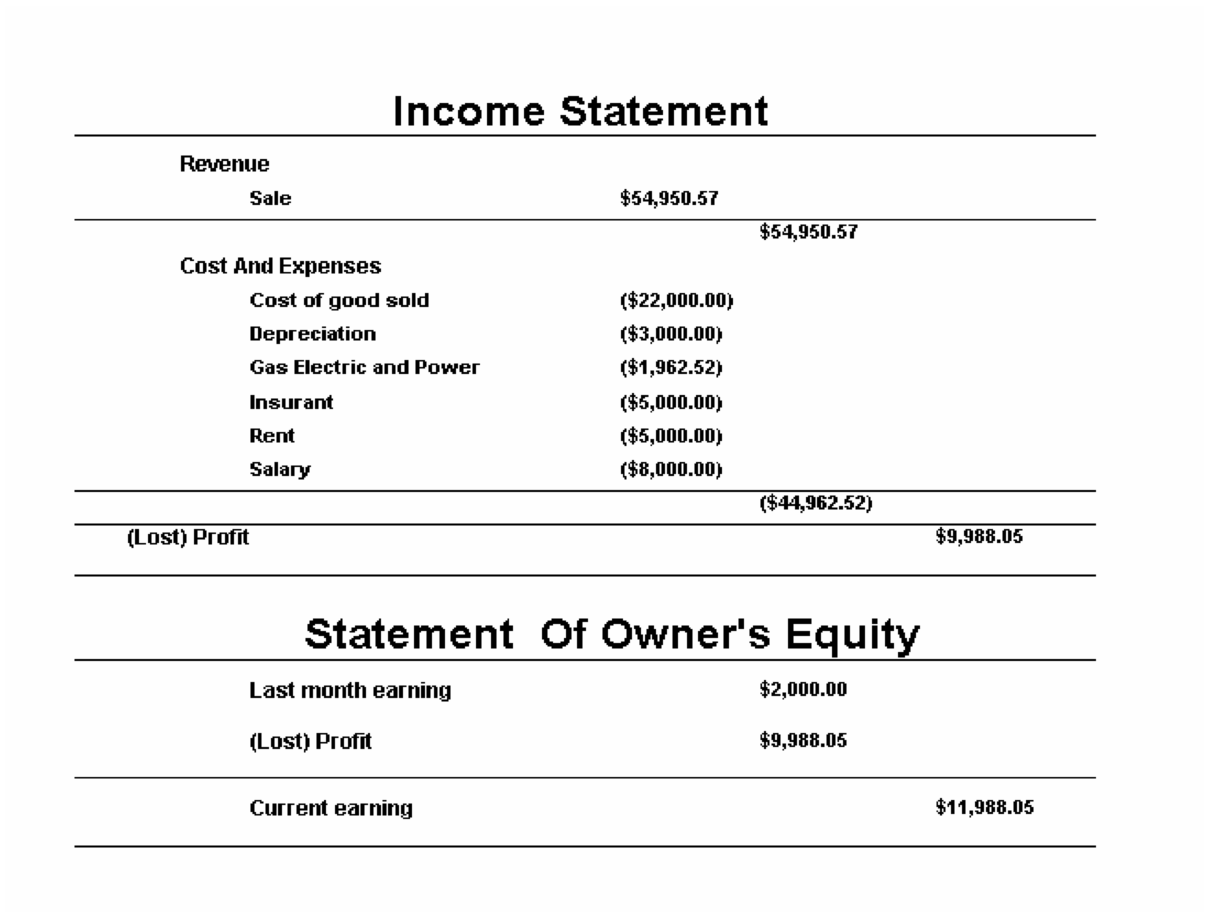
Current Data				
Event	Var_name	Var_value	Type	Desc
	Max_delivery_time	3	ch	Maximum timer to delivery
	Gametime	3	cl	The period of timer
	MK_H_demand	0	cl	Hourly demand for the market
	timeordermaterialwait	0	cl	Time-order tracking
	check_transfer_waiting	0	cl	Check transfer tracking
	credit_transfer_waiting	0	cl	Credit transfer tracking
Production	Chef	2	ch	Number of Chef hiring
Sale	Waiter	1	ch	Employee who work in the restaurant driving sale volume
Sale	delivery_man	2	ch	Employee number to delivery Pizzas
Sale	waiter_capability	10	ch	Capability to serve for an hour
Sale	delivery_capability	10	ch	Capability to send pizza a round
	Business_time	8	ch	The business work time a day, giving hours work a day
Demand	PriceBase	14		Standard price in the market
Production	Equipment	1	ch	Machine and Equipment to produce Pizza
Production	pizza_ingredient	2	ch	Units number of Materials use for a pizza
Production	chef_capability	10	ch	Performance of Chef producing Pizzas, per hr, per person
Sale	Restaurant	1	cl	The number of restaurant
CashRecieve	CheckRecieve	0	cl	Value of total check received
CashRecieve	Credit	0	cl	Total credit value in business
CashRecieve	Cash	2000	cl	Cash on hand

Current Data				
Event	Var_name	Var_value	Type	Desc
Production	Materials	1000	ch	Materials on hand
Product	equipment_capability	40	ch	Capability of equipment producing pizzas, per hr, per machine
OrderSupply	TimeOrderMaterial	2	ch	The period to order material
	check_sale_percentage	0.6	ch	Percentage of sale to check
	cash_sale_percentage	0.4	ch	Percentage of sale to cash
	check_transfer_period	5	ch	The number of a period transferring check to cash
	Cashsafty	2000	ch	Minimum allowable cash on hand
	Creditlimited	100000	cl	Limited of credit value or game over time run
	Overtime	60	cl	Days allowing to have maximum credit or game over
	credit_transfer_period	15	ch	the number of period paying credit
	Interest	0.5	ch	the interest add to credit when pay it late
Sale	Totalsale	0	cl	Total sale of pizzas in a month
Time	Business_month	1	cl	simulation month
Time	Business_year	1	cl	simulation year
Time	Varsale	0	cl	Temporary number accumulate sale number a day
Time	Lotordercost	5000	ch	A materials' cost per order
Time	MonthExpense	22000	cl	Amount of monthly expense
Time	Business_hr	1	cl	Simulation hours
	Mk_demand	0	ch	The demand of pizza for this company, it is generated every simulation day time
	Business_day	0	cl	Simulation day
	Product	10	cl	Pizza on hand
	OrderCostPer	500	ch	Other expenses for order supply
Product	Norder	0	cl	number of order,
	Startcash	2000	ch	the start cash value starting month
Demand	Population	30000	ch	people (demand base)
Demand	meal_a_day	1.5	ch	Average meals that a person consuming a day
Demand	mk_portion	0.01	ch	Pizza market portion among other types restaurant industry
Demand	mk_share	0.3	ch	The company market share of all

Current Data				
Event	Var_name	Var_value	Type	Desc
				competitions
Demand	Flexibility	0	ch	The random boundary, limitation of random
OrderSupply	SaftyStock	1000	ch	The minimum number of materials before sending order
OrderSupply	LotOrder	2000	ch	Units of material order a time
Demand	Productprice	14	ch	Pizza Price

After the first month of running the restaurant simulation, the business factors will be changed and saved into the table. Then, the process will retrieve the data from the table to show in the following report (Table 4.2).

Table 4.2. 1st Month Business Report.



Income Statement

Revenue			
Sale	$54,950.57		
		$54,950.57	
Cost And Expenses			
Cost of good sold	($22,000.00)		
Depreciation	($3,000.00)		
Gas Electric and Power	($1,962.52)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($44,962.52)	
(Lost) Profit			$9,988.05

Statement Of Owner's Equity

Last month earning	$2,000.00	
(Lost) Profit	$9,988.05	
Current earning		$11,988.05

The first month of simulation shows that the company has $9,988.05 profit, excluding the effect of a situational problem. The report uses the business factors in Table 4.1 to run the simulation's process. The Statement of Owner's Equity shows the new total earning, which is the amount of the previously accumulated earning and this month's profit. It shows that the company's current earning is $11,988.05.

4.3.2 Situations at the End of the First Month

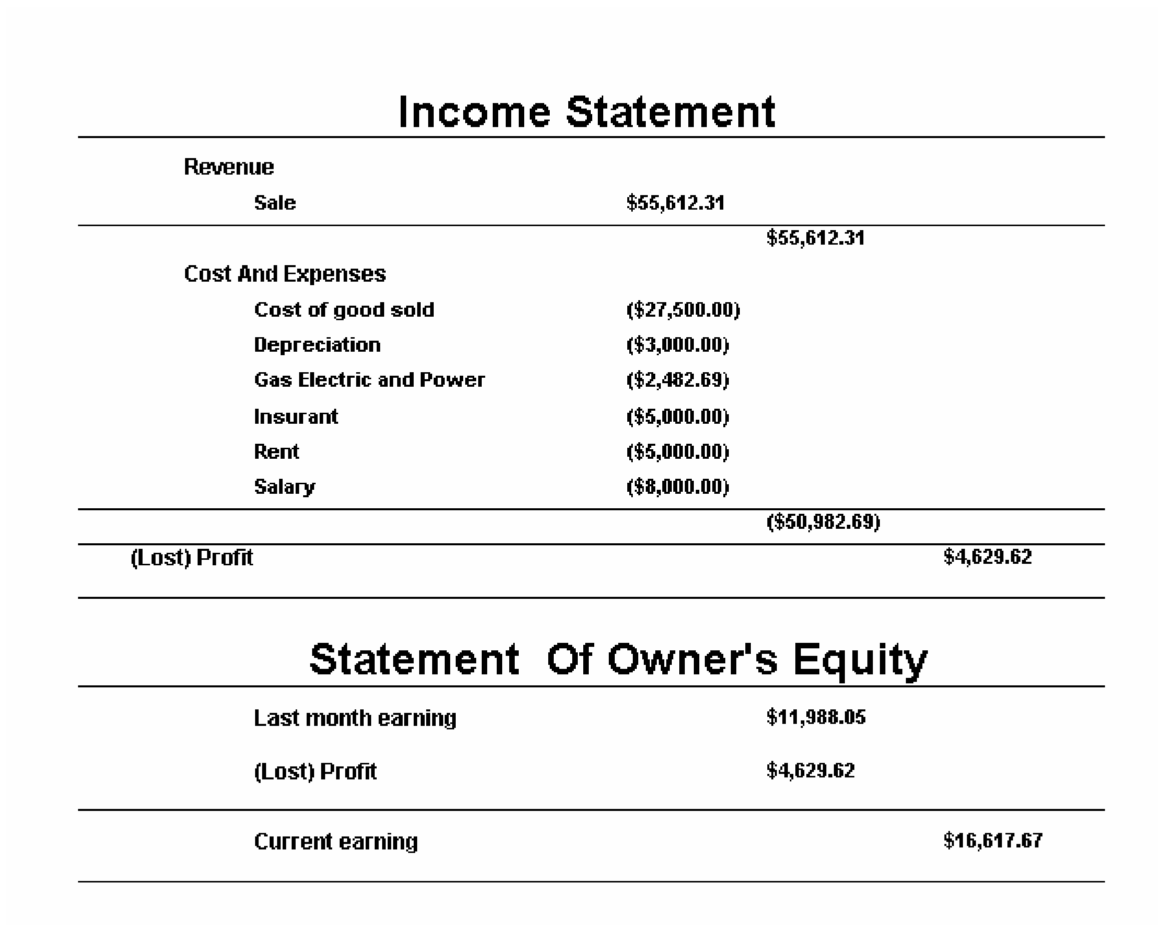
Suppose that the restaurant operation is faced with a new competition, which drops their pizza price by 20%. This situation decreases the original business's market share by 20%. There are the following four alternatives for this situation.

The restaurant could match the competition's price. In this case, the price will be dropped by 20%. The new price may interest people and stimulate the market share, raising it 40% among its competitors.

Table 4.3. Factor Impact Case 1.

Situational Factor	Mathematic Function
Product price	Product price = Product price * 0.8
Market Share	market share = Market share * 1.8

Table 4.4. Report of Alternative1 in 2nd Month.



Income Statement

Revenue			
Sale	$55,612.31		
		$55,612.31	
Cost And Expenses			
Cost of good sold	($27,500.00)		
Depreciation	($3,000.00)		
Gas Electric and Power	($2,482.69)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($50,982.69)	
(Lost) Profit			$4,629.62

Statement Of Owner's Equity

Last month earning	$11,988.05	
(Lost) Profit	$4,629.62	
Current earning		$16,617.67

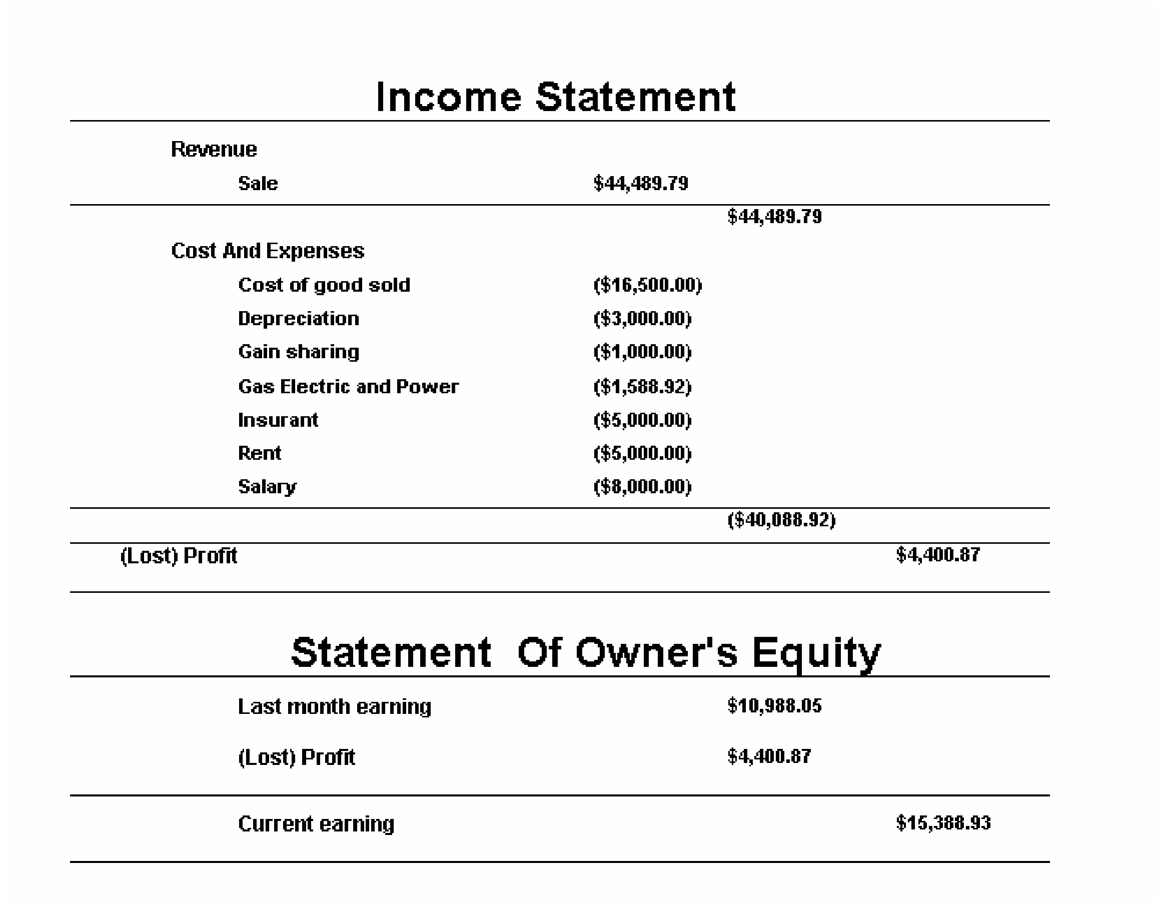
The restaurant realizes that, though the demand factor is important, the production capability does still not meet the current demand, and the price of pizza is more important than the demand number influencing the profit. Therefore, the restaurant should increase productivity and quality of the product. This alternative will impact the delivery capability and production capability. The reward of increasing capability will approximately be returned to employee as 10% of their salary. This amount will be added to the salary expense in the report. The consequence of this alternative is that it will stimulate the market share to increase by 0.66%. The following, Table 4.5, is the details of factors and mathematical functions

Table 4.5. Factor Impact Case 2.

Situational Factor	Mathematic Function
Delivery Capability	delivery_capability=delivery_capability * 1.10
Equipment Capability	equipment_capability=equipment_capability* 1.10
Chef Capability	chef_capability =chef_capability*1.10
Waiter Capability	waiter_capability =waiter_capability *1.10
Market Share	market share = market share + 0.002
Salary Expense	salary expense = salary expense +(deliveryman*200) salary expense = salary expense +(chef*200) salary expense = salary expense +(waiter*200)

After the use chooses this alternative, the report will be set in the following number:

Table 4.6. Report of Alternative 2 in 2nd Month.



Income Statement

Revenue			
Sale	$44,489.79		
		$44,489.79	
Cost And Expenses			
Cost of good sold	($16,500.00)		
Depreciation	($3,000.00)		
Gain sharing	($1,000.00)		
Gas Electric and Power	($1,588.92)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($40,088.92)	
(Lost) Profit			$4,400.87

Statement Of Owner's Equity

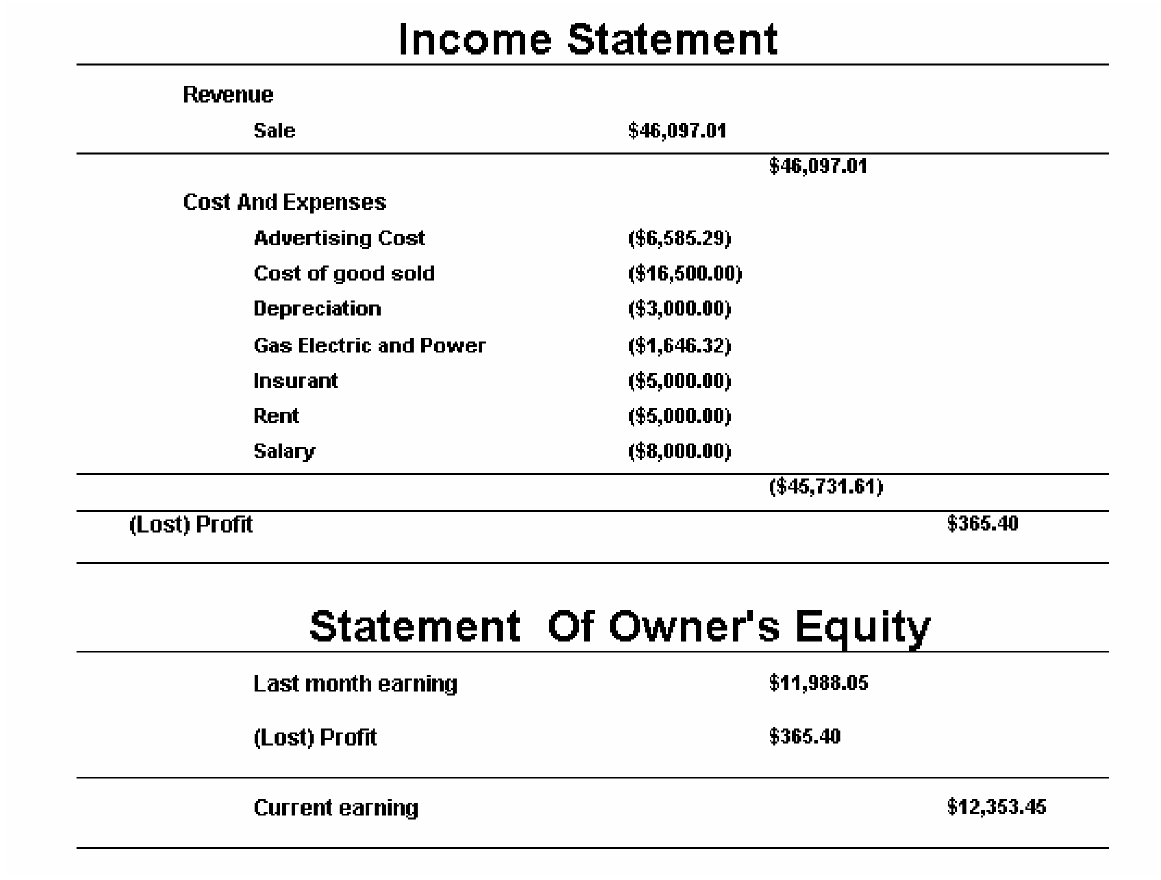
Last month earning	$10,988.05	
(Lost) Profit	$4,400.87	
Current earning		$15,388.93

The restaurant thinks that the price of pizza can greatly impact monthly profit. This alternative chooses paying extra money in the form of advertising to keep the demand. Furthermore, the advertising may impact the demand more than the price factor. This alternative spends an average of $1 per pizza sold for the ads. This stimulation may increase the demand by as much as 5%. The factors impact is in the following, Table 4.7:

Table 4.7. Factor Impact Case 3

Situational Factor	Mathematic Function
Market Share	Market_share=market_share*0.05
Advertising Cost	Advertising Cost = product sale * 1

Table 4.8. Report of Alternative 3 in 2nd Month.



Income Statement

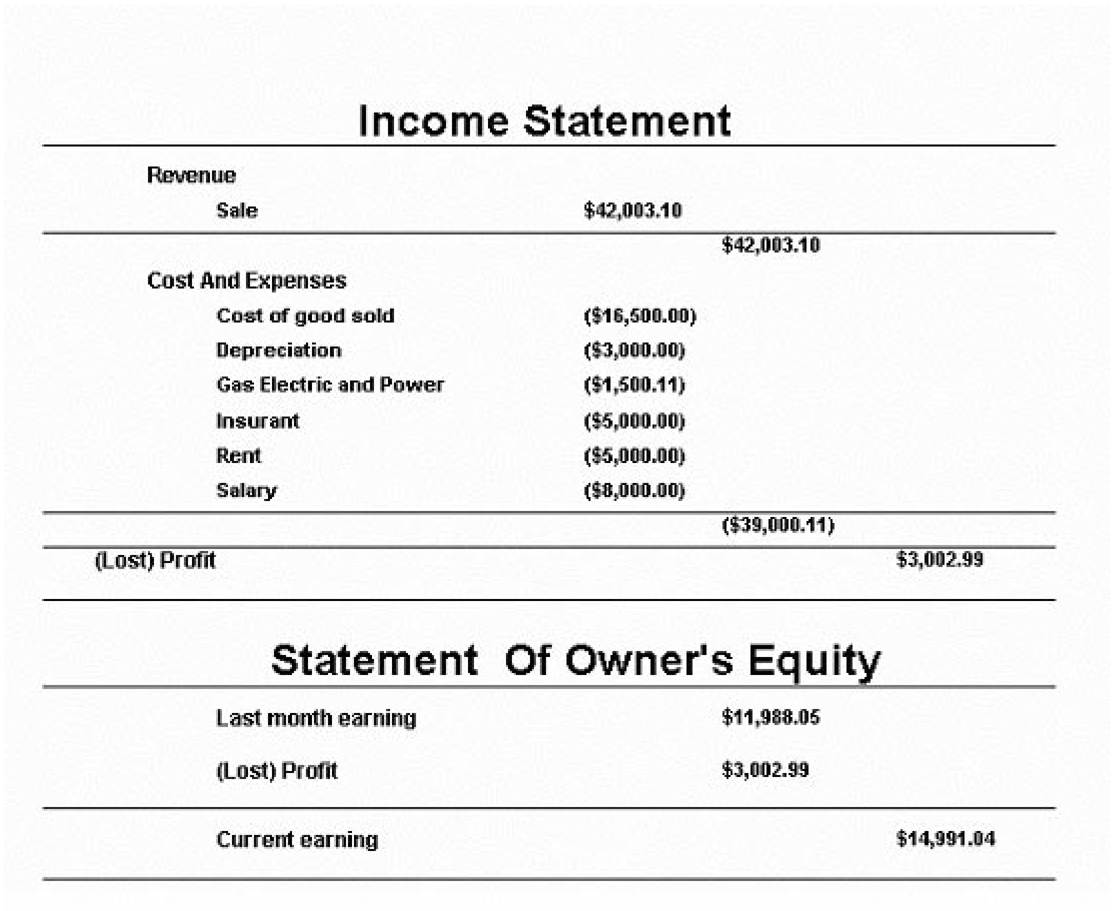
Revenue			
Sale	$46,097.01		
		$46,097.01	
Cost And Expenses			
Advertising Cost	($6,585.29)		
Cost of good sold	($16,500.00)		
Depreciation	($3,000.00)		
Gas Electric and Power	($1,646.32)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($45,731.61)	
(Lost) Profit			$365.40

Statement Of Owner's Equity

Last month earning	$11,988.05	
(Lost) Profit	$365.40	
Current earning		$12,353.45

The last alternative thinks that the restaurant's situation is good enough to not act at in this time. The percentage of demand decreasing is so small that it cannot affect the profit. The production and selling factors are also strong and stable, creating the possibility of obtaining the similar profit as the previous month.

Table 4.9. Report of Alternative 4 in 2nd Month.



Income Statement

Revenue			
Sale	$42,003.10		
		$42,003.10	
Cost And Expenses			
Cost of good sold	($16,500.00)		
Depreciation	($3,000.00)		
Gas Electric and Power	($1,500.11)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($39,000.11)	
(Lost) Profit			$3,002.99

Statement Of Owner's Equity

Last month earning	$11,988.05	
(Lost) Profit	$3,002.99	
Current earning		$14,991.04

With the result of each alternative, the values of the business factors differed greatly after changing. The following spreadsheet shows how the varying alternatives impact each factor in the business, in order to easily compare them.

4.3.3 Comparing results of running simulation

The bold characters emphasize the factors that have been changed by the chosen alternative.

Table 4.10. Comparing Factors' Alternatives.

Comparing Business Factors						
Name	Starting	1st Month	2nd Month Alternative 1	2nd Month Alternative 2	2nd Month Alternative 3	2nd Month Alternative 4
Max_delivery_time	3	3	3	3	3	3
Gametime	3	3	3	3	3	3
MK_H_demand	0	0	0	0	0	0
Timeordermaterialwait	0	0	0	2	2	0

Comparing Business Factors

Name	Starting	1st Month	2nd Month Alternative 1	2nd Month Alternative 2	2nd Month Alternative 3	2nd Month Alternative 4
check_transfer_waiting	0	0	5	5	5	5
credit_transfer_waiting	0	14	11	11	11	11
Chef	2	2	2	2	2	2
Waiter	1	1	1	1	1	1
delivery_man	2	2	2	2	2	2
waiter_capability	10	10	10	11	10	10
delivery_capability	10	10	10	11	10	10
Business_time	8	8	8	8	8	8
PriceBase	14	14	14	14	14	14
Equipment	1	1	1	1	1	1
pizza_ingredient	2	2	2	2	2	2
chef_capability	10	10	10	11	10	10
Restaurant	1	1	1	1	1	1
CheckRecieve	**0**	**0**	**5962.86**	**4573.80**	**4768.24**	**4309.2**
Credit	0	0	0	0	0	0
Cash	**2000**	**11988**	**10654.81**	**11815.12**	**7585.211**	**10681.84**
Materials	1000	1100	1100	720	500	1100
Equipment_capability	40	40	40	44	40	40
TimeOrderMaterial	2	2	2	2	2	2
check_sale_percentage	0.6	0.6	0.6	0.6	0.6	0.6
Cash_sale_percentage	0.4	0.4	0.4	0.4	0.4	0.4
check_transfer_period	5	5	5	5	5	5
Cashsafty	2000	2000	2000	2000	2000	2000
creditlimited	100000	100000	100000	100000	100000	100000
Overtime	60	60	60	60	60	60
credit_transfer_period	15	15	15	15	15	15
Interest	0.5	0.5	0.5	0.5	0.5	0.5
Totalsale	**0**	**3925.04**	**4965.38**	**3177.84**	**3292.64**	**3000.22**
business_month	1	2	3	3	3	3
business_year	1	1	1	1	1	1
Varsale	0	7.89	10	0.77	5.20	1.81
Lotordercost	5000	5000	5000	5000	5000	5000
MonthExpense	**22000**	**22000**	**23482.69**	**22962.52**	**29231.64**	**22500.11**
business_hr	1	0	0	0	0	0
Mk_demand	0	135	194.4	108.9	113.53	102.6
business_day	0	1	1	1	1	1
Product	**10**	**34.96**	**69.5743**	**47.11**	**42.32**	**34.74**
OrderCostPer	500	500	500	500	500	500
Norder	0	4	5	3	3	3
Startcash (Earning)	**2,000**	**11,988**	**16,617.67**	**16,388.92**	**12,353.45**	**14,991.04**
Population	30000	30000	30000	30000	30000	30000

Comparing Business Factors						
Name	Starting	1st Month	2nd Month Alternative 1	2nd Month Alternative 2	2nd Month Alternative 3	2nd Month Alternative 4
meal_a_day	1.5	1.5	1.5	1.5	1.5	1.5
mk_portion	0.01	0.01	0.01	0.01	0.01	0.01
mk_share	**0.3**	**0.3**	**0.432**	**0.242**	**0.258**	**0.228**
Flexibility	0	0	0	0	0	0
SaftyStock	1000	1000	1000	1000	1000	1000
LotOrder	2000	2000	2000	2000	2000	2000
Productprice	14	14	11.2	14	14	14

Table 4.10 shows the result of performing the simulation for the first and second months in the game's time with a situational problem and an alternative case. The first alternative, matching the competitor's price, is the best selection in this case according to the earnings of $16,617.67. After analyzing the process, the important factors, which brought the final result, were mainly in market share and monthly expense factors for it generated 43% of market share. Compared to the second best number which resulted in 25% of market share, these are significantly different. Although the first alternative, the best choice, did not produce extra expenses like the second and third alternatives, it did reduce the pizza price, which changed from $14.00 to $11.20. This caused the profit margin to reduce by as much as $1.80 per pizza. When simulating the total sale for the restaurant in the second month, the first alternative sold 3,925 units, which was the second best number. This increased the revenue to $43,960 in an environment without any extra expenses.

The second alternative, increasing the productivity, came up with the second best solution. It generated $16,388.92 in earning for the restaurant. Because the productivity and quality increased, it brought the market share up to 25%. The idea of gain sharing to increase productivity and quality, that results in increasing the market share, is an excellent idea, but because of the aggressive competitor action of the competitor in this case, the second alternative may not be strong enough to compete in the market.

The third alternative, using advertising strategy, did not fit for this situation. The cost of advertising was so high that it greatly affected expenses, increasing to $29,231. Although the market share in this case was the second best number, it was the worst solution, generating only $12,353 of total earning.

The last alternative, which chose do nothing, did not generate expenses but the market share was impacted and declined to 22.8%. This number was the only factor effected, earning $14,991 for the company.

The simulation change depends on the alternative selected. The business factors realistically relate these decisions.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

This final chapter includes the conclusions and recommendations of this study. In this chapter the objectives will be addressed and compared with the results of the study. Recommendations for instructors and students are made. The chapter will conclude by discussing recommendations for future research.

Conclusions

The restaurant simulation game, the result of the research, contained characteristics which answered the research problem. The game's concept was dependent upon principles of business that the user could learn while playing. The CATscanner model was used as the foundation of and standard for the game. This study achieved the research objectives as follows:

1. It depicted the general ideas of business operation and perception of, including the concept of "cross functional area in an organization" that addressed the internal and external factors of the business environment.
2. Users experienced several business case studies.
3. The game was utilized as a tool to discuss business lessons.
4. The game guided people who lack a business background to understand the principles.

At the simulation game suggestions were reviewed in Chapter 2, the researcher designed the game to include the independent decision feature, allowing the user to run it on a standalone computer. This would reduce the problem regarding group conflict, and the students were able to examine and learn from the game many times before holding a discussion in the class. The game included a timing feature with the sequential processes, making it easy to debug the problems of the game processes. Flexibility was not included in the game so it would give the same result when playing. The illustration of the business cases utilized as mathematical functions and was the significant step to implementing the game.

Recommendations

As the simulation game will be used in a business class, the participants of the game, of course are instructors and students. Recommendations for the two groups are as follows:

Recommendation for instructors:

1. The instructors should carefully read the game instructions to make certain that the game process and the relations of business factors are understood. Only one inputting mistake can generate incorrect report statements, which affect the given concept’s interpretation.
2. The instructors should test all the alternatives and situations in the restaurant simulation to make appropriate recommendations. The recommendations will be inputted in the situational simulation window. This will also be given to students after they run the restaurant simulation and have received the monthly report.

3. The instructors should use Chapter 4 to set the business factors and mathematical functions.
4. After the students have played the simulation game, the case study from the game could be discussed to review a business concepts.

Recommendation for students:

1. They need to read the restaurant simulation process in Chapter 4 to understand the varying operations. This can assist in determining the best answer to the problem.
2. They should discuss the problems with other students to make sure decision regarding the alternative situations.
3. It is necessary for the student to use the financial statements to analyze the case situation
4. The students should use the “factors monitoring” window to help make a decision for the alternative situations.

Recommendations for Future Research

The recommendations for future study are made as follows:

1. Some utility functions can be included such as the “redo the decision” function. Otherwise, the user has to access the database file by using Microsoft Access to perform these functions.
2. The business case studies can be investigated as potential input information for the game. The sets of situations can be researched to find

the relationships between the business factors, which can then be explained by mathematical functions.

3. The game can include some advanced functions which will graphically simulate the restaurant work routine.
4. Situations that affect the company’s long-term benefit should be considered and added to simulation, meaning that the effect of changing in business factors will occur longer than one month.
5. The technique of computer networking should be included in the future simulation game. This enables the users to share data and to see the impacts of other players’ decision making.

REFERENCE

Burgess, T. F. (1991). The use of computerized management and business simulation in The United Kingdom. Simulation & Gaming, 22(2), 174-195.

Brozik, Dallas; Zapalska, Alina (2000). The Restaurant game, Simulation & Game 31(3), 407-417.

Chang, M. T. (1997). The use of business gaming in Hong Kong academic institutions. In J. Butler & N. Leonard (Eds.), Developments in business simulation and experiential learning (pp. 218-220). Statesboro: Georgia Southern University Press.

Gentry. J. W. (1980). Group size and attitudes toward simulation experience. Simulation & Games, 11: 451-460.

Henshaw, R., & Jackson, J (1978). THE EXECUTIVE GAME. Homewood, IL: Richard D. Irwin, Inc

Hi-tech dictionary. Retrieved September 28, 2001, from http://www.computeruser.com/resources/dictionary/dictionary.html

Ivo, W & Don, C (1999) Why do we bother with games and simulation: An organization perspective. Simulation & Game 30(3), 3750-385

Jackson, J. R. (1959). Learning from experience in business decision games. California Management Review, 1(1), 23-29.

Judson, A (1982). The awkward truth about productivity. Harvard Business Review, 60(4), 93-97

Kibbee, . M., Craft, C. J., & Nanus, B. (1961). Management games. New York: Reinhold.

Keys, B., & Wolfe, J. (1990). The role of management games and simulations in education and research. Journal of Management, 16, 307-336.

Lucas, A. C. (1979). Performance in a complex management game. Simulation & Games, 10(1): 61-74.

McKenna, R. J. (1991). Business computerized simulation: The Australian experience. Simulation & Gaming, 22(1), 36-62.

Meier, R. C., Newell, W. T., & Pazer, H. L. (1969). Simulation in business and economics. Englewood Cliffs, NJ: Prentice Hall.

Michael, Gonsalves (2000). Profitania Deluxe game review, http://www.gamesdomain.com/gdreview/zones/reviews/pc/dec98/profit.html

Snyder, LT (1996). BUSINESS Strategy Game The (Game), Simulation & Gaming, 27(4), 520, 524

Stanislaw, H. (1986). Test of computer simulation validity. Simulation & Games, 17(1), 173-191

Rollier, b. (1992). Observation of a corporate facilitator. Simulation & Game, 23(1), 442-456

Rick Casteel (2000). Sim City 3000, Christian Computing Magazine, http://www.gospelcom.net/ccmag/game/SimCity3000.shtml

Teach, Richard D (1990). Profits: The false prophet in business gaming. Simulation & Gaming 21(1), 15-17

Thavikulwat, P. (1988). Emphasizing different modes of learning through a configurable business simulation game. Simulation & Games, 19, 408-414.

Thavikulwat, Precha (1996). Activity-driven time in computerized game simulations. Simulation & Game 27(1), 110-123

Thavikulwat, Precha (1991). BUSINESS –Computer simulation. Simulation & Game 22(1), 350-360

Vince, S., & Gray, C (1967). Use of a performance evaluation model of research in business gaming. Academy of Management Journal, 33(1), 27-37

Watson, H. J. (1981). Computer simulation in business. New York: John Wiley & Sons.

Webopedia Retrieved October 3, 2001, from http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html

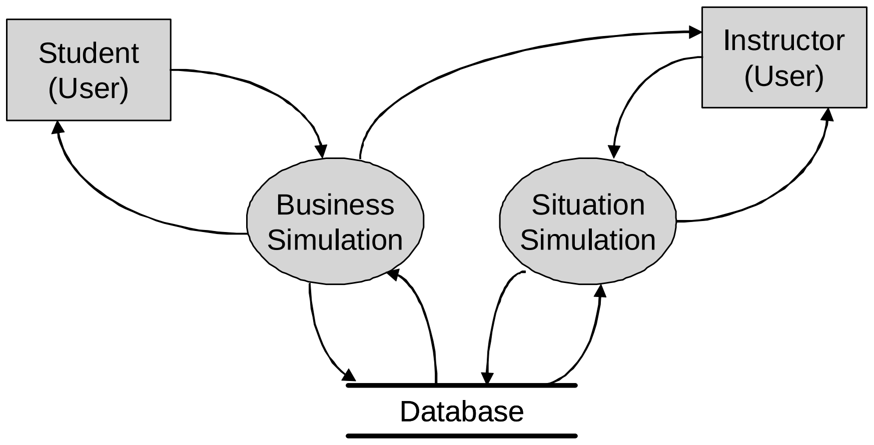
Wolfe, J., & Chacko, T. I. (1983). Team size effects on business game performance and decision making behaviors. Decision Sciences, 14, 121-133.

Wolfe, J., & Crookall, D. (1998). Developing a scientific knowledge of simulation/gaming. Simulation & Gaming, 29(1), 7-19)

APPENDEX

A. Analysis and Design: Section 1: Simulation Game

Figure A.1. Data Flow Diagram



*There are two sections of programming development. The details of two sections are the following:

Section 1: Business Simulation

Structure Chart and Codes

Figure A.2. A Restaurant Process; Business Simulation Structure Chart

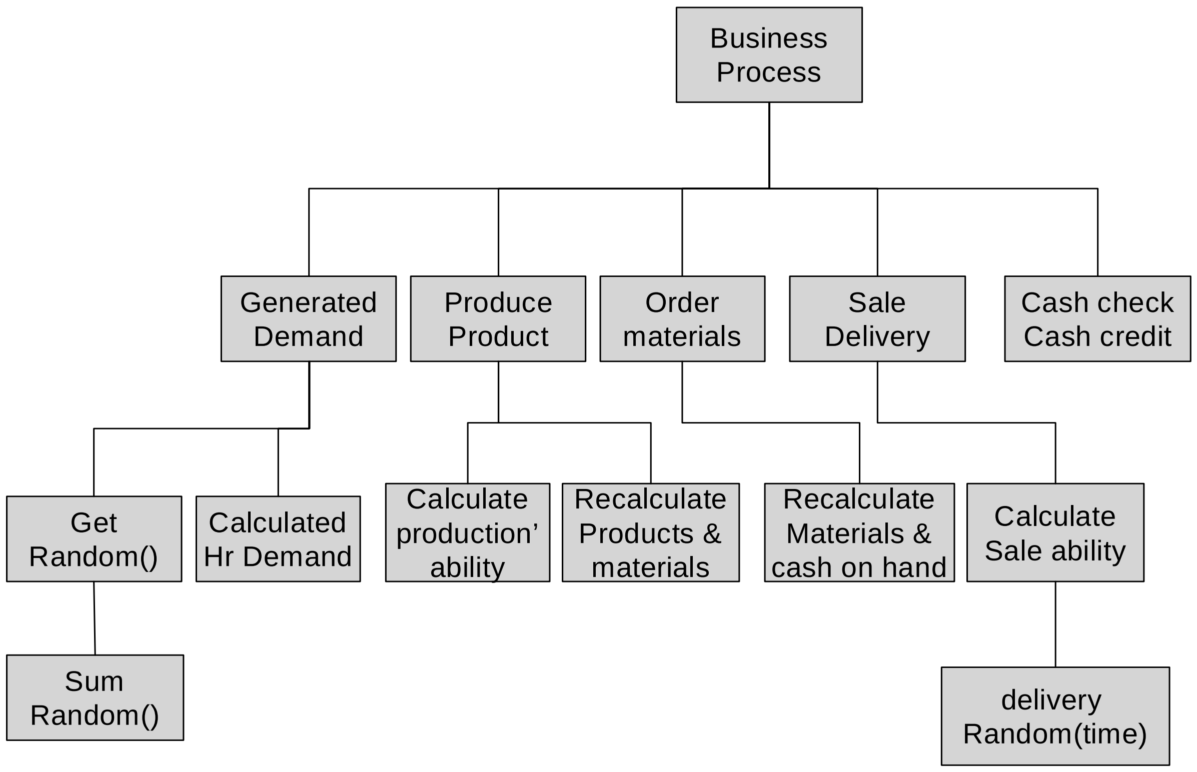


Table A.1 Situation Simulation Modules

Process Item	Codes
Set Data Variable	Set Database variable to Database Object Do Loop Read All records Create object variable named by record data Set object variable properties value to record's value Loop
Generated Demand	$marketdemand = population \times meal\_a\_day \times marketshare \times marketportion$ Clear value of variable "Total Hour demand" For loop of business hour time Set array "demand in hour" = random number Total_hour_demand = $\sum_{n=numberofbu\sin esswork}^{n} Hrdemand(n)$ Next business hour time
Produce Product	Check if hour demand > product on hand Check if $material > ingredientused \times productionability$ Add product on hand Cut materials used
Sale and Delivery	Check if still having demand and product on hand (1) Check if $Demand \geq \Pr oductonhand$ (2)

```
Check if Product > waiters' ability Then (3)
Add waiters' ability to variable varsale
Else (if 3)
Add product on hand to variable varsale
End If (3)
 Else (if 2)
Check if hr demand > waiters' ability (4)
Add waiters' ability  to variable varsale
Else (if 4)
Add hr demand to variable varsale
End If (4)
 End If (2)
Else (if 1)
varsale  = 0
End if (1)
Set update demand
Set update product on hand
Set update total products sold
Set update cash
Set update credit
For number of delivery man loop
Check if delivery finished

Next
For n = 1 To number of delivery people
 If delivery(n) = business hours Then

  If hourly demand > 0 And product on hand > 0 Then
   If  hourly demand >= product on hand Then
     If product on hand > delivery capability Then
      varsale = delivery capability
     Else
      varsale = product on hand
     End If
    Else
     If hourly demand > delivery capability Then
      varsale = delivery capability
     Else
      varsale = hourly demand
     End If
    End If
  Else
  varsale = 0
  End If
  '-------------------------------------------------------------
delivery(n) = checkout(delivery(n))
```

	hourly demand = hourly demand - varsale product on hand = product on hand - varsale total sale = total sale + varsale cash = cash + (varsale * Product price * sale receive cash percentage) check receive = check receive + (varsale * product price * check sale percentage) End If Next
Transfer credit	If credit transfer waiting = credit transfer period Then cash = cash - credit credit = 0 credit transfer waiting = 0 Else credit transfer waiting = credit transfer waiting + 1 End If
Transfer check	If check transfer waiting = check transfer period Then Cash = Cash + Check Receive Check Receive = 0 check transfer waiting = 0 Else check transfer waiting = check transfer waiting + 1 End If

Order Materials	If materials < safety stock hen
	    If time order material <> time order material wait Then
	    Time order material wait = time order material wait + 1
	    Else
	    materials = materials + Lot-order
	    cash = cash - Lot ordercost - Order Cost Per Unit
	    norder = norder + 1
	    time order material wait = 0
	    End If
	End If

Show Data	Lbdayvar.Caption = Str(business day)
	Lbmonthvar.Caption = Str(business month)
	lbyearvar.Caption = Str(business year)
	lbsalevar.Caption = Str(Round(Total sale , 0))
	lbproductvar.Caption = Str(Round(Product , 2))
	LbDemandVar.Caption = Str(Round(MK H demand , 0))
	lbmaterial.Caption = Str(materials)
	Lbcashvar.Caption = Str(Round(cash , 2))

Image Click	txtcash sale percentage.Text = cash sale percentage txtcashsafty Text = cashsafty txtcheck sale percentage.Text = check sale percentage txtcheck transfer period.Text = check transfer period

	txtchef.Text = chef txtchef capability.Text = chef capability txtcredit transfer period.Text = credit transfer period txtdelivery capability.Text = delivery capability txtdelivery man.Text = delivery man txtequipment.Text = equipment txtequipment capability.Text = equipment capability txtinterest.Text = interest txtLot order.Text = Lot order txtLot ordercost.Text = Lot ordercost txtLot orderper.Text = Order Cost Per Unit txtmaterials.Text = materials txtmeal a day.Text = meal a day txtmk portion.Text = mk portion txtmk share.Text = mk share txtpizza ingredient.Text = pizza ingredient txtpopulation.Text = population txtproduct price.Text = product price txtsaftystock.Text = saftystock txtstartcash.Text = startcash txttimeordermaterial.Text = timeordermaterial txtwait capability.Text = waiter capability txtwaiter.Text = waiter

Timer1 Timer	Select Case environment
	Case 1
	  If IndustryLine.BorderColor <> vbRed Then
	   IndustryLine.BorderColor = vbRed
	  Else
	   IndustryLine.BorderColor = vbBlue
	  End If
	Case 2
	  If Productline.BorderColor <> vbRed Then
	   Productline.BorderColor = vbRed
	  Else
	   Productline.BorderColor = vbBlue
	  End If
	Case 3
	  If EconLine.BorderColor <> vbRed Then
	   EconLine.BorderColor = vbRed
	  Else
	   EconLine.BorderColor = vbBlue
	  End If
	Case 4
	  If TechLine.BorderColor <> vbRed Then
	   TechLine.BorderColor = vbRed
	  Else

```
   TechLine.BorderColor = vbBlue
   End If
Case 5
  If PoliticalLine.BorderColor <> vbRed Then
   PoliticalLine.BorderColor = vbRed
  Else
   PoliticalLine.BorderColor = vbBlue
  End If
Case 6
  If LegalLine.BorderColor <> vbRed Then
   LegalLine.BorderColor = vbRed
  Else
   LegalLine.BorderColor = vbBlue
  End If
Case 7
  If SocialculLine.BorderColor <> vbRed Then
   SocialculLine.BorderColor = vbRed
  Else
   SocialculLine.BorderColor = vbBlue
  End If
End Select
```

Figure A.3. Report Process Structure Chart

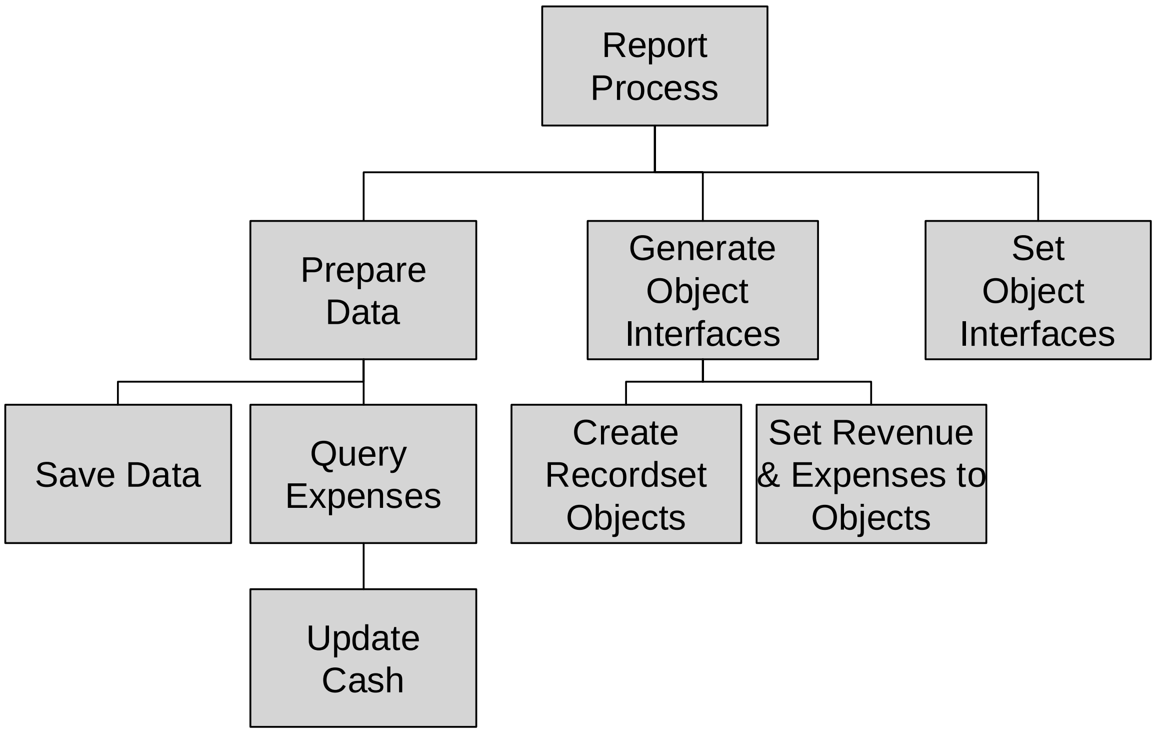


Table A.2. Report Process Modules

Process Item	Code
Initialize Data	Dim rs As Recordset
	Dim child rs As Recordset
	Dim fld As Field
	Dim Exprs, detailvar
	Dim amountvar
	SaveData
	getmonthlyExpense
	cash = cash - MonthExpense
	mainmenu.TimerDay.Interval = 0
	Set rs = New Recordset
	rs.LockType = adLockBatchOptimistic
	rs CREATE TABLE " &
	ADD adVarChar(20) As TopicGroup,
	((CREATE NEW TABLE"
	ADD adVarChar(20) As Topic,
	ADD adVarChar(50) As Name,
	ADD adSingle As Amount)
	RELATE TopicGroup to Topic) As Child
	amountvar = total sale * product price
	rs.AddNew Array("TopicGroup, Array("Revenue
	rs.AddNew Array("TopicGroup, Array("Cost And Expenses
	Set child rs = rs("Child
	sumtotal = sumtotal + amountvar
	child rs.AddNew Array("Topic", "Name", "Amount, Array("Revenue", "Sale", amountvar)
	DBSituation.Recordsets("MonthlyExp.Open
	DBSituation.Recordsets("MonthlyExp.Requery
	Set Exprs = DBSituation.Recordsets("MonthlyExp
	Exprs.MoveFirst
	Do While Not Exprs.EOF
	detailvar = Exprs.Fields("Detail
	amountvar = Exprs.Fields("Total * -1
	sumtotal = sumtotal + amountvar
	child rs.AddNew Array("Topic", "Name", "Amount, Array("Cost And Expenses", detailvar, amountvar)
	Exprs.MoveNext
	Loop
	DBSituation.Recordsets("MonthlyExp.Close
	Sections("Section5.Controls("LmOEq.Caption = Format(startcash , currency)
	startcash = sumtotal + startcash
	norder = 0
	total sale = 0

	SaveData
	Set DataSource = rs
	Sections("Section5.Controls("GT.Caption = Format(sumtotal, currency)
	Sections("Section5.Controls("LPOEq.Caption = Format(sumtotal, currency)
	Sections("Section5.Controls("COEq.Caption = Format(startcash , currency)
	End Sub

Terminate Data	Situation.Show

Figure A.4. Situational Simulation Process

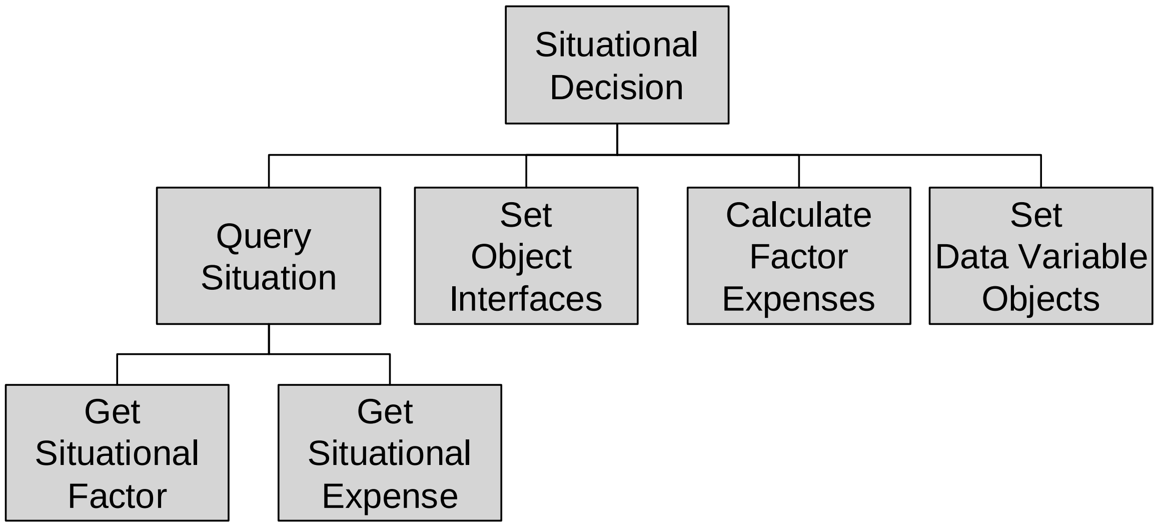


Table A.3. Situational Simulation Modules

Process Item	Codes
Form Load	If business month - 1 > 0 Then sqlstr = "Select * from situation where situation month =" & business month - 1 & "and situation year=" & business year & ";" Set rs = Execute sqlstr base on DBSituation If rs end of file <> True Or rs before of file <> True Then Show rs.Fields("Situation answer End If End If sqlstr = "Select * from situation where situation month =" & business month & "and situation year=" & business year & ";"

```
Set rs = Execute sqlstr base on DBSituation
Do While Not rs end of file
If rs.Fields("Situation type = "Q" Then
Set objControl = Controls.Add("VB.Label", "obj" + CStr(rs.Fields("number
))
    Set ObjControl Visible is true
    objControl.Top = 255 * i
    objControl.Left = 500
    countword = Len(rs.Fields("situationdetail )
    objControl.Width = 5000
    objControl.Height = 255 * (Round(countword / 70, 0) + 1)
    i = i + (Round(countword / 70, 0) + 2)
    objControl.Caption = rs.Fields("situationdetail
  Select Case rs.Fields("situation env
  Case "Industry/Competition"
   environment  = 1
  Case "Production/Input"
   environment  = 2
  Case "Economic"
   environment  = 3
  Case "Technology"
   environment  = 4
  Case "Political"
   environment  = 4
  Case "Legal"
   environment  = 6
  Case "Social/culture"
   environment  = 7
  Case Else
   environment  = 0
 End Select
 End If
If rs.Fields("Situation type = "A" Then
Set objControl = Controls.Add("VB.optionButton", "obj" +
CStr(rs.Fields(“number”))
    ObjControl Visible is true
    objControl.Top = 255 * i
    objControl.Left = 500
    objControl.Caption = rs.Fields(“situationdetail”)
    countword = Len(rs.Fields(“situationdetail”)
    objControl.Width = 8000
    objControl.Height = 255 * (Round(countword / 70, 0) + 1)
    i = i + (Round(countword / 70, 0) + 2)
End If
  rs.MoveNext
Loop
```

	VScroll1.Min = 0
	VScroll1 = 0
	VScroll1.LargeChange = 100 'Arbitary values
	VScroll1.SmallChange = 50
	VScroll1.Max = ((i * 255) - Height) + VScroll1.Height

Form Unload	mainmenu.TimerDay.Interval = temp TimerDay
getchange	Dim sqlstr, rs sqlstr = "Select * from situation factor where situation number=" & n & ";" Set rs = DBSituation.DBSituation.Execute(sqlstr) Do While not rs end of file If rs.Fields(operator = "+" Then rs.Fields("var name) value = rs.Fields("var name) value + Val(rs.Fields(var num) ElseIf rs.Fields("operator = "-" Then rs.Fields(var name) value = rs.Fields("var name) value - Val(rs.Fields("var num) ElseIf rs.Fields(operator = "*" Then rs.Fields("var name) value = rs.Fields("var name) value * Val(rs.Fields(var num) ElseIf rs.Fields(operator = "/" Then rs.Fields(var name) value = rs.Fields("var name) value / Val(rs.Fields(var num) End If rs.MoveNext Loop ' insert base DBSituation.DBSituation.Execute ("INSERT INTO currentexpense (Base, Detail, Name, Amount, Event) SELECT situation expense.Base, situation expense.Detail, situation expense.Name, situation expense.Amount, 'MonthlyExpense' AS Mep From situation expense WHERE (((situation expense.situation number)=” & CStr(n) &); SaveData

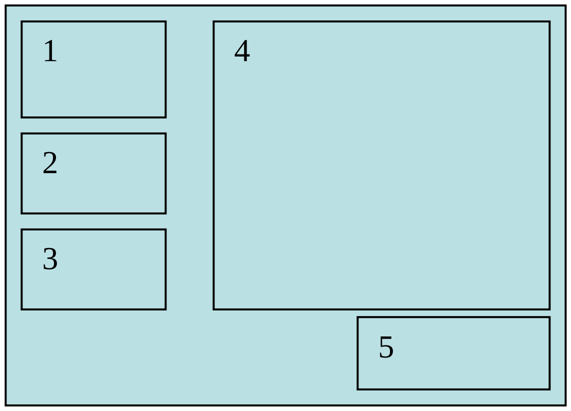
VScroll1 Change	For I = 0 To Count – 1
	  Set objControl = Controls(i)
	   If Left(objControl.Name, 3) = "obj" Then
	    objControl.Top = -VScroll1 + h
	    h = h + objControl.Height + 255
	   End If
	Next

VScroll1 Scroll	Dim i, objControl, h
	h = 0
	For i = 0 To Count - 1
	  Set objControl = Controls(i)
	   If Left(objControl.Name, 3) = "obj" Then
	    objControl.Top = -VScroll1

	`h = h + objControl.Height`
	`End If`
	`Next`

Graphic User Interfaces

Table A.4. Main Window

	1. A finance frame 2. A productivity frame 3. A market-demand frame 4. CATscanner model 5. Timing frame

*The finance frame contains three text boxes, cash receivable, cash, and credit.

*The productivity frame contains raw materials and product on hand textboxes

*The sale-demand frame contains sales and demand textboxes.

Table A.5. Object Script with Properties

```
mainmenu - 1
VERSION 5.00
Begin VB.Form mainmenu
BackColor = &H00E0E0E0&
Caption = "Model"
ClientHeight = 6645
ClientLeft = 1425
ClientTop = 1470
ClientWidth = 9735
LinkTopic = "Form1"
ScaleHeight = 6645
ScaleWidth = 9735
Begin VB.Timer Timer1
Interval = 100
Left = 1920
```

```
Top = 3480
End
Begin VB.Frame Frame3
Caption = "Unit"
Height = 855
Left = 0
TabIndex = 19
Top = 2520
Width = 2295
Begin VB.Label LbDemandText
BorderStyle = 1 'Fixed Single
Caption = "Demand"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 23
Top = 480
Width = 1095
End
Begin VB.Label lbSaletext
BorderStyle = 1 'Fixed Single
Caption = "Sale"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 22
Top = 240
Width = 1095
End
Begin VB.Label lbsalevar
```

```
BorderStyle = 1 'Fixed Single
Height = 255
Left = 1200
TabIndex = 21
Top = 240
Width = 975
End
Begin VB.Label LbDemandVar
BorderStyle = 1 'Fixed Single
mainmenu - 2
Height = 255
Left = 1200
TabIndex = 20
Top = 480
Width = 975
End
End
Begin VB.Frame Frame2
Caption = "Unit"
Height = 855
Left = 0
TabIndex = 14
Top = 1560
Width = 2295
Begin VB.Label lbproductvar
BorderStyle = 1 'Fixed Single
Height = 255
Left = 1200
TabIndex = 18
Top = 480
Width = 975
End
Begin VB.Label lbmaterial
BorderStyle = 1 'Fixed Single
Height = 255
Left = 1200
TabIndex = 17
Top = 240
Width = 975
End
Begin VB.Label lbproducttext
BorderStyle = 1 'Fixed Single
Caption = "Production"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
```

```
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 16
Top = 480
Width = 1095
End
Begin VB.Label lbmaterialtext
BorderStyle = 1 'Fixed Single
Caption = "Raw Material"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 15
Top = 240
Width = 1095
End
End
Begin VB.PictureBox Picture1
Height = 5775
mainmenu - 3
Left = 2400
Picture = (Bitmap)
ScaleHeight = 5715
ScaleWidth = 7275
TabIndex = 6
Top = 0
Width = 7335
Begin VB.Image Image1
Height = 1095
Left = 2160
Top = 4680
Width = 3255
```

```
End
Begin VB.Line SocialculLine
BorderWidth = 3
X1 = 7080
X2 = 3960
Y1 = 1560
Y2 = 4440
End
Begin VB.Line LegalLine
BorderWidth = 3
X1 = 6120
X2 = 3960
Y1 = 720
Y2 = 4320
End
Begin VB.Line PoliticalLine
BorderWidth = 3
X1 = 5280
X2 = 3960
Y1 = 600
Y2 = 4320
End
Begin VB.Line TechLine
BorderStyle = 2 'Dash
BorderWidth = 3
X1 = 4680
X2 = 3960
Y1 = 1560
Y2 = 4320
End
Begin VB.Line EconLine
BorderStyle = 2 'Dash
BorderWidth = 3
X1 = 2040
X2 = 3720
Y1 = 720
Y2 = 4440
End
Begin VB.Line IndustryLine
BorderColor = &H00404040&
BorderWidth = 3
X1 = 480
X2 = 3600
Y1 = 1680
Y2 = 4200
End
```

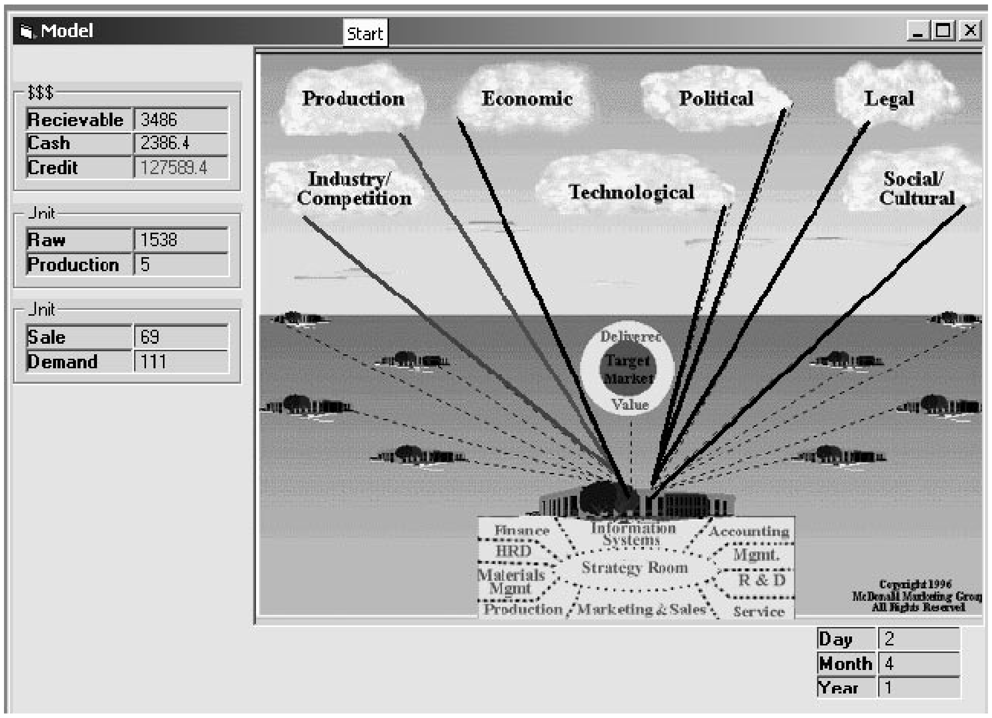
```
Begin VB.Line Productline
BorderStyle = 2 'Dash
BorderWidth = 3
X1 = 1440
X2 = 3600
Y1 = 840
Y2 = 4200
End
End
Begin VB.Timer TimerDay
Interval = 100
mainmenu - 4
Left = 1440
Top = 0
End
Begin VB.Frame Frame1
Caption = "$$$"
Height = 1095
Left = 0
TabIndex = 7
Top = 360
Width = 2295
Begin VB.Label Lbreceivevar
BorderStyle = 1 'Fixed Single
Height = 255
Left = 1200
TabIndex = 13
Top = 240
Width = 975
End
Begin VB.Label Lbpayvar
BorderStyle = 1 'Fixed Single
ForeColor = &H000000FF&
Height = 255
Left = 1200
TabIndex = 12
Top = 720
Width = 975
End
Begin VB.Label Lbcashvar
BorderStyle = 1 'Fixed Single
Height = 255
Left = 1200
TabIndex = 11
Top = 480
Width = 975
```

```
End
Begin VB.Label Lbreceive
BorderStyle = 1 'Fixed Single
Caption = "Recievable"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 10
Top = 240
Width = 1095
End
Begin VB.Label lbpay
BorderStyle = 1 'Fixed Single
Caption = "Credit"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 9
Top = 720
mainmenu - 5
Width = 1095
End
Begin VB.Label lbcash
BorderStyle = 1 'Fixed Single
Caption = "Cash"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
```

```
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 120
TabIndex = 8
Top = 480
Width = 1095
End
End
Begin VB.Label lbdaytext
BorderStyle = 1 'Fixed Single
Caption = "Day"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 8040
TabIndex = 5
Top = 5760
Width = 615
End
Begin VB.Label Lbdayvar
BorderStyle = 1 'Fixed Single
Height = 255
Left = 8640
TabIndex = 4
Top = 5760
Width = 855
End
Begin VB.Label lbyeartext
BorderStyle = 1 'Fixed Single
Caption = "Year"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
```

```
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 8040
TabIndex = 3
Top = 6240
Width = 615
End
Begin VB.Label lbyearvar
BorderStyle = 1 'Fixed Single
Height = 255
Left = 8640
mainmenu - 6
TabIndex = 2
Top = 6240
Width = 855
End
Begin VB.Label lbmonthtext
BorderStyle = 1 'Fixed Single
Caption = "Month"
BeginProperty Font
Name = "MS Sans Serif"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Height = 255
Left = 8040
TabIndex = 1
Top = 6000
Width = 615
End
Begin VB.Label Lbmonthvar
BorderStyle = 1 'Fixed Single
Height = 255
Left = 8640
TabIndex = 0
Top = 6000
Width = 855
End
End
```

Figure A.5. CATscannse Model



$$$

Recievable	3486
Cash	2386.4
Credit	127589.4

Jnit

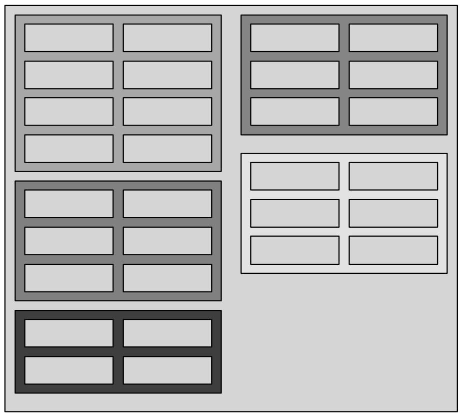
Raw	1538
Production	5

Jnit

Sale	69
Demand	111

Day	2
Month	4
Year	1

Table A.6. Monitoring Factors Window

	1. Green color frame represents demand factors. 2. Gray color frame represents production factor. 3. Blue color frame represents inventory factors. 4. Red color frame represents sale and marketing factors. 5. Yellow color represents financial factors.

* The first frame contains population, meal a day and market share and market portion factor of the restaurant.

* The second frame contains a number of chefs in the restaurant, chefs' ability in cooking per hours, machine to beak pizza, and machine's ability to beak pizzas per hours, Pizza ingredient and, unit of materials in inventory.

* The third frame contains the number of materials ordered in one time, materials' cost in that order, Cost of ordering such as transportation, period of order materials, and quantity of materials for safety stock.

* The forth frame contains number of waiter in the restaurant, waiter's ability to serve pizzas per hours, number of delivery man, delivery man's ability per hours, product price, percentage of customer pay by check and percentage of customer pay by cash.

* The fifth frame contains the number of cash on the first day of that month, the period to cash check, the period to transaction credit, interest and minimum cash on hand policy.

Table A.7. Object Script and properties

```
Form1 - 1
VERSION 5.00
Begin VB.Form Form1
Caption = "Monitoring factors"
ClientHeight = 4875
ClientLeft = 60
ClientTop = 345
ClientWidth = 5655
LinkTopic = "Form1"
ScaleHeight = 4875
ScaleWidth = 5655
StartUpPosition = 3 'Windows Default
Begin VB.Frame Frame4
Caption = "Financial"
Height = 1575
Left = 2760
TabIndex = 8
```

```
Top = 2040
Width = 2895
Begin VB.TextBox txtcashsafty
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 58
Top = 1200
Width = 855
End
Begin VB.TextBox txtinterest
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 57
Top = 960
Width = 855
End
Begin VB.TextBox txtcredit transfer period
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 56
Top = 720
Width = 855
End
Begin VB.TextBox txtcheck transfer period
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 55
Top = 480
Width = 855
End
Begin VB.TextBox txtstartcash
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 54
Top = 240
Width = 855
End
Begin VB.Label Label20
BorderStyle = 1 'Fixed Single
Caption = "starting cash"
Height = 255
```

```
Left = 120
TabIndex = 14
Top = 240
Width = 1815
End
Begin VB.Label Label17
BorderStyle = 1 'Fixed Single
Form1 - 2
Caption = "interest"
Height = 255
Left = 120
TabIndex = 12
Top = 960
Width = 1815
End
Begin VB.Label Label16
BorderStyle = 1 'Fixed Single
Caption = "credit transfer period"
Height = 255
Left = 120
TabIndex = 11
Top = 720
Width = 1815
End
Begin VB.Label Label14
BorderStyle = 1 'Fixed Single
Caption = "cashsafety"
Height = 255
Left = 120
TabIndex = 10
Top = 1200
Width = 1815
End
Begin VB.Label Label15
BorderStyle = 1 'Fixed Single
Caption = "check transfer period"
Height = 255
Left = 120
TabIndex = 9
Top = 480
Width = 1815
End
End
Begin VB.Frame Purchase
Caption = "Purchase materials"
Height = 1575
```

```
Left = 0
TabIndex = 6
Top = 3120
Width = 2655
Begin VB.TextBox txtsaftystock
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 35
Top = 1200
Width = 855
End
Begin VB.TextBox txttimeordermaterial
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 34
Top = 960
Width = 855
End
Begin VB.TextBox txtLot orderper
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 33
Top = 720
Width = 855
End
Begin VB.TextBox txtLot ordercost
Enabled = 0 'False
Height = 285
Form1 - 3
Left = 1680
TabIndex = 32
Top = 480
Width = 855
End
Begin VB.TextBox txtLot order
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 31
Top = 240
Width = 855
End
Begin VB.Label Label27
```

```
BorderStyle = 1 'Fixed Single
Caption = "Lot order"
Height = 255
Left = 120
TabIndex = 30
Top = 240
Width = 1575
End
Begin VB.Label Label18
BorderStyle = 1 'Fixed Single
Caption = "Lot ordercost"
Height = 255
Left = 120
TabIndex = 29
Top = 480
Width = 1575
End
Begin VB.Label Label25
BorderStyle = 1 'Fixed Single
Caption = "SaftyStock"
Height = 255
Left = 120
TabIndex = 18
Top = 1200
Width = 1575
End
Begin VB.Label Label19
BorderStyle = 1 'Fixed Single
Caption = "Order Cost Per Unit"
Height = 255
Left = 120
TabIndex = 13
Top = 720
Width = 1575
End
Begin VB.Label Label11
BorderStyle = 1 'Fixed Single
Caption = "TimeOrderMaterial"
Height = 255
Left = 120
TabIndex = 7
Top = 960
Width = 1575
End
End
Begin VB.Frame Frame3
```

```
Caption = "Sale & Marketing"
Height = 2055
Left = 2760
TabIndex = 2
Top = 0
Width = 2895
Begin VB.TextBox txtcash sale percentage
Enabled = 0 'False
Height = 285
Form1 - 4
Left = 1920
TabIndex = 46
Top = 1680
Width = 855
End
Begin VB.TextBox txtcheck sale percentage
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 45
Top = 1440
Width = 855
End
Begin VB.TextBox txtproduct price
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 44
Top = 1200
Width = 855
End
Begin VB.TextBox txtdelivery capability
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 43
Top = 960
Width = 855
End
Begin VB.TextBox txtdelivery man
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 42
Top = 720
Width = 855
```

```
End
Begin VB.TextBox txtwait capability
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 41
Top = 480
Width = 855
End
Begin VB.TextBox txtwaiter
Enabled = 0 'False
Height = 285
Left = 1920
TabIndex = 40
Top = 240
Width = 855
End
Begin VB.Label Label2
BorderStyle = 1 'Fixed Single
Caption = "waiter"
Height = 255
Left = 120
TabIndex = 53
Top = 240
Width = 1815
End
Begin VB.Label Label3
BorderStyle = 1 'Fixed Single
Caption = "delivery man"
Height = 255
Left = 120
TabIndex = 52
Top = 720
Form1 - 5
Width = 1815
End
Begin VB.Label Label4
BorderStyle = 1 'Fixed Single
Caption = "waiter capability"
Height = 255
Left = 120
TabIndex = 51
Top = 480
Width = 1815
End
Begin VB.Label Label5
```

```
BorderStyle = 1 'Fixed Single
Caption = "delivery capability"
Height = 255
Left = 120
TabIndex = 50
Top = 960
Width = 1815
End
Begin VB.Label Label26
BorderStyle = 1 'Fixed Single
Caption = "Product price"
Height = 255
Left = 120
TabIndex = 49
Top = 1200
Width = 1815
End
Begin VB.Label Label12
BorderStyle = 1 'Fixed Single
Caption = "check sale percentage"
Height = 255
Left = 120
TabIndex = 48
Top = 1440
Width = 1815
End
Begin VB.Label Label13
BorderStyle = 1 'Fixed Single
Caption = "cash sale percentage"
Height = 255
Left = 120
TabIndex = 47
Top = 1680
Width = 1815
End
End
Begin VB.Frame Frame2
Caption = "Production"
Height = 1815
Left = 0
TabIndex = 1
Top = 1320
Width = 2655
Begin VB.TextBox txtmaterials
Enabled = 0 'False
Height = 285
```

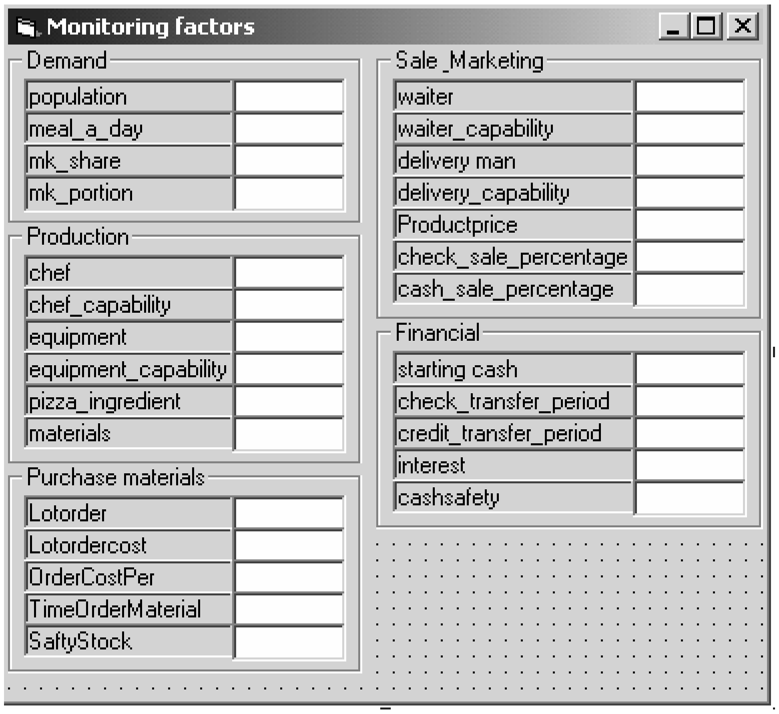
```
Left = 1680
TabIndex = 22
Top = 1440
Width = 855
End
Begin VB.TextBox txtpizza ingredient
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 21
Top = 1200
Form1 - 6
Width = 855
End
Begin VB.TextBox txtequipment capability
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 20
Top = 960
Width = 855
End
Begin VB.TextBox txtequipment
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 19
Top = 720
Width = 855
End
Begin VB.TextBox txtchef capability
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 17
Top = 480
Width = 855
End
Begin VB.TextBox txtchef
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 16
Top = 240
Width = 855
End
```

```
Begin VB.Label Label1
BorderStyle = 1 'Fixed Single
Caption = "chef"
Height = 255
Left = 120
TabIndex = 28
Top = 240
Width = 1575
End
Begin VB.Label Label6
BorderStyle = 1 'Fixed Single
Caption = "equipment"
Height = 255
Left = 120
TabIndex = 27
Top = 720
Width = 1575
End
Begin VB.Label Label7
BorderStyle = 1 'Fixed Single
Caption = "pizza ingredient"
Height = 255
Left = 120
TabIndex = 26
Top = 1200
Width = 1695
End
Begin VB.Label Label8
BorderStyle = 1 'Fixed Single
Caption = "chef capability"
Height = 255
Left = 120
TabIndex = 25
Top = 480
Width = 1695
Form1 - 7
End
Begin VB.Label Label9
BorderStyle = 1 'Fixed Single
Caption = "materials"
Height = 255
Left = 120
TabIndex = 24
Top = 1440
Width = 1575
End
```

```
Begin VB.Label Label10
BorderStyle = 1 'Fixed Single
Caption = "equipment capability"
Height = 255
Left = 120
TabIndex = 23
Top = 960
Width = 1575
End
End
Begin VB.Frame Frame1
Caption = "Demand"
Height = 1335
Left = 0
TabIndex = 0
Top = 0
Width = 2655
Begin VB.TextBox txtmk portion
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 39
Top = 960
Width = 855
End
Begin VB.TextBox txtmk share
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 38
Top = 720
Width = 855
End
Begin VB.TextBox txtmeal a day
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 37
Top = 480
Width = 855
End
Begin VB.TextBox txtpopulation
Enabled = 0 'False
Height = 285
Left = 1680
TabIndex = 36
```

```
Top = 240
Width = 855
End
Begin VB.Label Label23
BorderStyle = 1 'Fixed Single
Caption = "mk portion"
Height = 285
Left = 120
TabIndex = 15
Top = 960
Width = 1575
End
Begin VB.Label Label22
Form1 - 8
BorderStyle = 1 'Fixed Single
Caption = "meal a day"
Height = 255
Left = 120
TabIndex = 5
Top = 480
Width = 1575
End
Begin VB.Label Label21
BorderStyle = 1 'Fixed Single
Caption = "population"
Height = 255
Left = 120
TabIndex = 4
Top = 240
Width = 1575
End
Begin VB.Label Label24
BorderStyle = 1 'Fixed Single
Caption = "mk share"
Height = 255
Left = 120
TabIndex = 3
Top = 720
Width = 1575
End
End
End
```

Figure A.6. Monitoring Factor Window



Monitoring factors

Demand

population	
meal_a_day	
mk_share	
mk_portion	

Production

chef	
chef_capability	
equipment	
equipment_capability	
pizza_ingredient	
materials	

Purchase materials

Lotorder	
Lotordercost	
OrderCostPer	
TimeOrderMaterial	
SaftyStock	

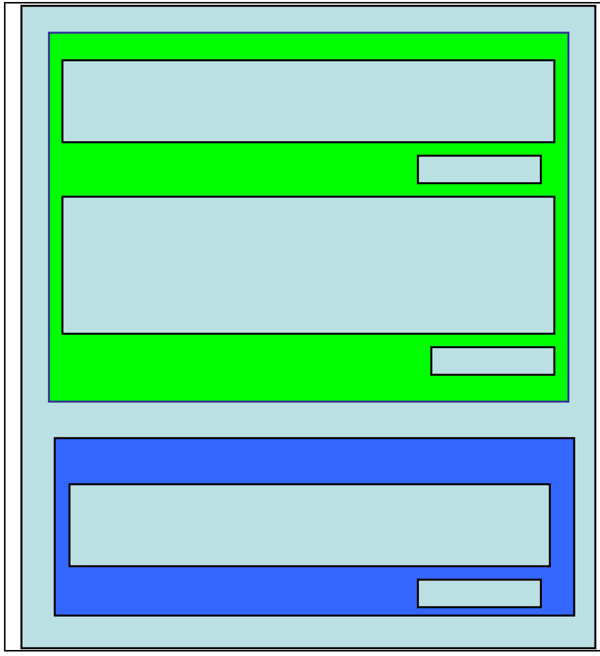
Sale_Marketing

waiter	
waiter_capability	
delivery man	
delivery_capability	
Productprice	
check_sale_percentage	
cash_sale_percentage	

Financial

starting cash	
check_transfer_period	
credit_transfer_period	
interest	
cashsafety	

Table A.8. Report statements

	1. Green color area is monthly income statement 2. Blue color is Statement of Owner Equity

*The first statement contains revenue item and expense items form business operation the different of those summation will show net profit or net loss.

*The second statement contains retain earning number the restaurant.

Table A.9. Object script and properties

```
DataReport – 1
VERSION 5.00
Begin {78E93846-85FD-11D0-8487-00A0C90DC8A9} DataReport
Caption = "DataReport"
ClientHeight = 6975
ClientLeft = 165
ClientTop = 450
ClientWidth = 10845
StartUpPosition = 3 'Windows Default
 ExtentX = 19129
 ExtentY = 12303
 Version = 393216
 DesignerVersion= 100685828
ReportWidth = 8775
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
```

```
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
GridX = 10
GridY = 10
LeftMargin = 1440
RightMargin = 1440
TopMargin = 1000
BottomMargin = 1140
NumSections = 7
SectionCode0 = 1
BeginProperty Section0 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Section4"
NumControls = 0
EndProperty
SectionCode1 = 2
BeginProperty Section1 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Section2"
Object.Height = 555
NumControls = 2
ItemType0 = 3
BeginProperty Item0 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Label1"
Object.Left = 2592
Object.Width = 3312
Object.Height = 432
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 18
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Object.Caption = "Income Statement"
EndProperty
ItemType1 = 5
BeginProperty Item1 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
```

```
Name = "Line3"
Object.Top = 432
Object.Width = 8352
EndProperty
EndProperty
SectionCode2 = 3
BeginProperty Section2 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
DataReport - 2
 Version = 393216
Name = "Section6"
Object.Height = 300
NumControls = 1
ItemType0 = 4
BeginProperty Item0 {1C13A8E2-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "LrpTopic"
Object.Left = 864
Object.Width = 2880
Object.Height = 285
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 9.75
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
DataField = "TopicGroup"
BeginProperty DataFormat {6D835690-900B-11D0-9484-00A0C91110ED}
Type = 0
Format = ""
HaveTrueFalseNull= 0
FirstDayOfWeek = 0
FirstWeekOfYear = 0
LCID = 1033
SubFormatType = 0
EndProperty
EndProperty
EndProperty
SectionCode3 = 4
BeginProperty Section3 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Section1"
Object.Height = 288
NumControls = 2
```

```
ItemType0 = 4
BeginProperty Item0 {1C13A8E2-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Text1"
Object.Left = 1440
Object.Width = 2160
Object.Height = 288
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
DataField = "Name"
BeginProperty DataFormat {6D835690-900B-11D0-9484-00A0C91110ED}
Type = 0
Format = ""
HaveTrueFalseNull= 0
FirstDayOfWeek = 0
FirstWeekOfYear = 0
LCID = 1033
SubFormatType = 0
EndProperty
DataMember = "Child"
EndProperty
ItemType1 = 4
BeginProperty Item1 {1C13A8E2-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
DataReport - 3
Name = "Text2"
Object.Left = 4464
Object.Width = 1290
Object.Height = 285
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
DataField = "Amount"
```

```
BeginProperty DataFormat {6D835690-900B-11D0-9484-00A0C91110ED}
Type = 1
Format = """$""#,##0.00;(""$""#,##0.00)"
HaveTrueFalseNull= 0
FirstDayOfWeek = 0
FirstWeekOfYear = 0
LCID = 1033
SubFormatType = 2
EndProperty
DataMember = "Child"
EndProperty
EndProperty
SectionCode4 = 5
BeginProperty Section4 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Section7"
Object.Height = 288
NumControls = 2
ItemType0 = 13
BeginProperty Item0 {49FF6930-2B8C-11D1-8DA9-00A0C90FFFC2}
 Version = 393216
Name = "Function1"
Object.Left = 5616
Object.Width = 1584
Object.Height = 285
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
DataField = "Amount"
BeginProperty DataFormat {6D835690-900B-11D0-9484-00A0C91110ED}
Type = 1
Format = """$""#,##0.00;(""$""#,##0.00)"
HaveTrueFalseNull= 0
FirstDayOfWeek = 0
FirstWeekOfYear = 0
LCID = 1033
SubFormatType = 2
EndProperty
DataMember = "child"
EndProperty
```

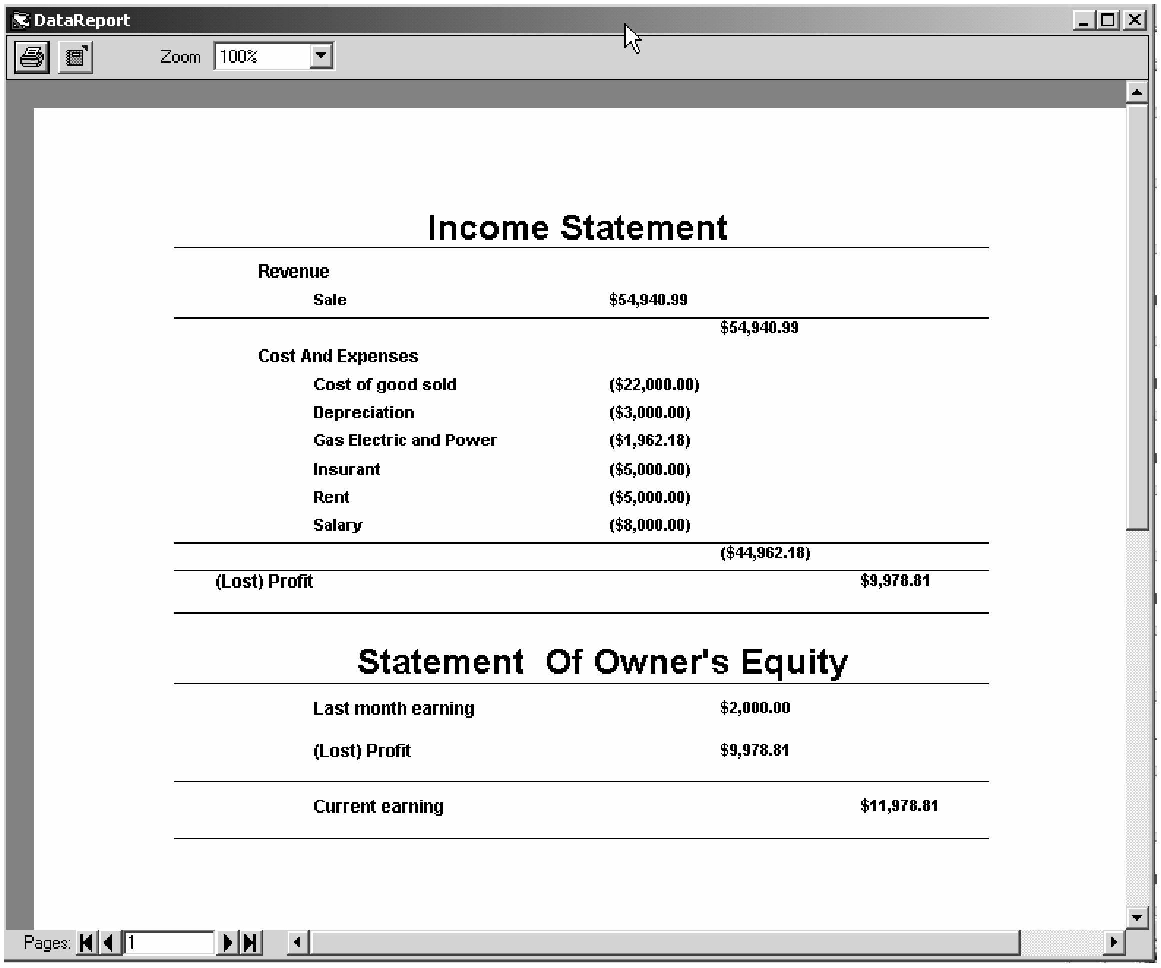
```
ItemType1 = 5
BeginProperty Item1 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Line1"
Object.Width = 8352
EndProperty
EndProperty
SectionCode5 = 7
BeginProperty Section5 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
DataReport - 4
 Version = 393216
Name = "Section3"
NumControls = 0
EndProperty
SectionCode6 = 8
BeginProperty Section6 {1C13A8E0-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Section5"
Object.Height = 2859
NumControls = 14
ItemType0 = 3
BeginProperty Item0 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Label2"
Object.Left = 432
Object.Width = 2295
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 9.75
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Object.Caption = "(Lost) Profit"
EndProperty
ItemType1 = 3
BeginProperty Item1 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "GT"
Object.Left = 7056
Object.Width = 1155
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
```

```
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
EndProperty
ItemType2 = 5
BeginProperty Item2 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Line2"
Object.Width = 8355
EndProperty
ItemType3 = 5
BeginProperty Item3 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Line4"
Object.Top = 432
Object.Width = 8355
EndProperty
ItemType4 = 3
BeginProperty Item4 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Label3"
Object.Left = 1872
Object.Top = 720
Object.Width = 5184
Object.Height = 435
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 18
DataReport - 5
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Object.Caption = "Statement Of Owner's Equity"
EndProperty
ItemType5 = 5
BeginProperty Item5 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Line5"
```

```
Object.Top = 1152
Object.Width = 8355
EndProperty
ItemType6 = 3
BeginProperty Item6 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Label4"
Object.Left = 1440
Object.Top = 1296
Object.Width = 2295
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 9.75
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Object.Caption = "Last month earning"
EndProperty
ItemType7 = 3
BeginProperty Item7 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Label5"
Object.Left = 1440
Object.Top = 1728
Object.Width = 2295
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 9.75
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Object.Caption = "(Lost) Profit"
EndProperty
ItemType8 = 3
BeginProperty Item8 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Label6"
Object.Left = 1440
Object.Top = 2304
Object.Width = 2295
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 9.75
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
DataReport - 6
EndProperty
Object.Caption = "Current earning"
EndProperty
ItemType9 = 5
BeginProperty Item9 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Line6"
Object.Top = 2160
Object.Width = 8355
EndProperty
ItemType10 = 5
BeginProperty Item10 {1C13A8E3-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "Line7"
Object.Top = 2736
Object.Width = 8355
EndProperty
ItemType11 = 3
BeginProperty Item11 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "LmOeq"
Object.Left = 5616
Object.Top = 1296
Object.Width = 1155
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
```

```
EndProperty
ItemType12 = 3
BeginProperty Item12 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "LPOEq"
Object.Left = 5616
Object.Top = 1728
Object.Width = 1155
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
EndProperty
ItemType13 = 3
BeginProperty Item13 {1C13A8E1-A0B6-11D0-848E-00A0C90DC8A9}
 Version = 393216
Name = "COEq"
Object.Left = 7056
Object.Top = 2304
Object.Width = 1155
Object.Height = 300
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Arial"
Size = 8.25
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
DataReport - 7
EndProperty
EndProperty
EndProperty
End
```

Figure A.7. Statements Report Window



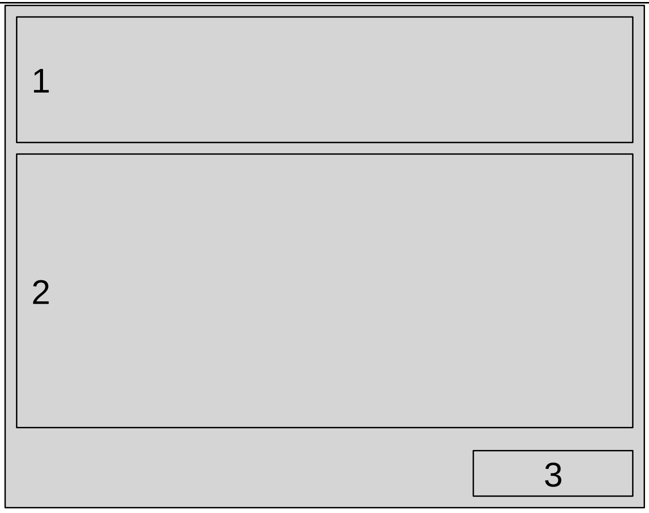
Income Statement

Revenue			
Sale	$54,940.99		
		$54,940.99	
Cost And Expenses			
Cost of good sold	($22,000.00)		
Depreciation	($3,000.00)		
Gas Electric and Power	($1,962.18)		
Insurant	($5,000.00)		
Rent	($5,000.00)		
Salary	($8,000.00)		
		($44,962.18)	
(Lost) Profit			$9,978.81

Statement Of Owner's Equity

Last month earning	$2,000.00	
(Lost) Profit	$9,978.81	
Current earning		$11,978.81

Table A.10. Situational Decision Window

	1. The situation detail 2. Alternative of situations 3. Selection button

* The situation detail is text box containing description of business situation.

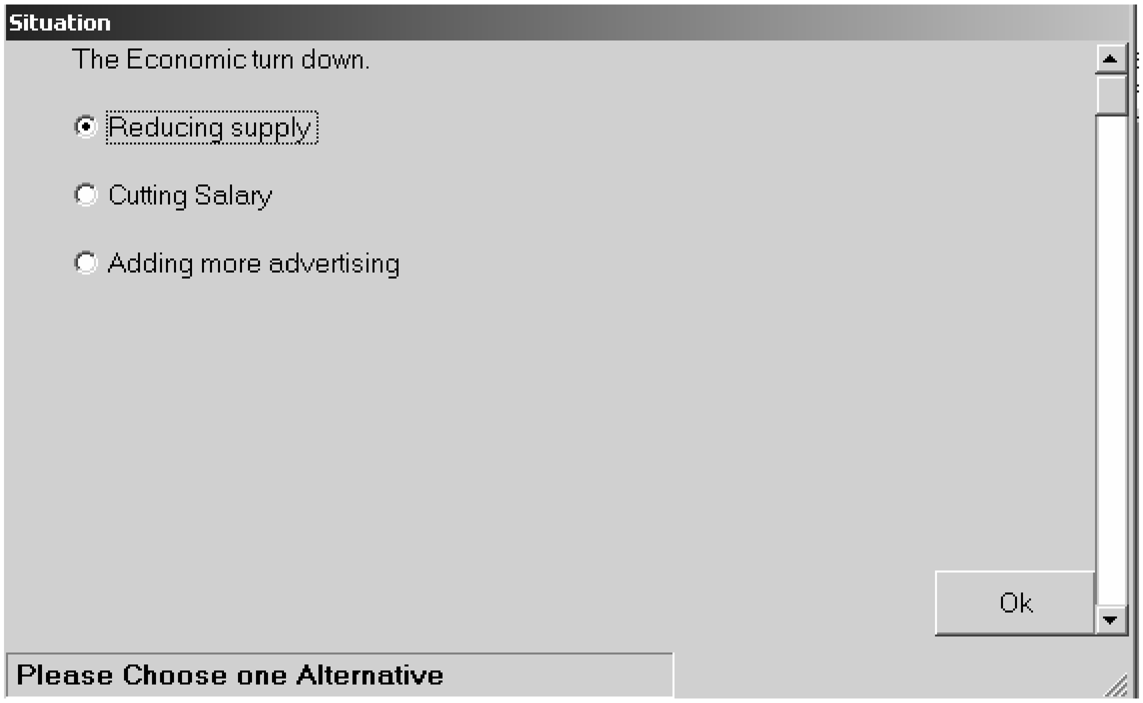
* The alternative situation is a set option choice of strategic planning to dual with the situation.

Table A.11 Object Script and Properties

```
Situation – 1
VERSION 5.00
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0";
"MSCOMCTL.OCX"
Begin VB.Form Situation
BorderStyle = 1 'Fixed Single
Caption = "Situation"
ClientHeight = 4920
ClientLeft = 4995
ClientTop = 3090
ClientWidth = 8415
ControlBox = 0 'False
BeginProperty Font
Name = "MS Sans Serif"
Size = 9.75
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
LinkTopic = "Form1"
MaxButton = 0 'False
MinButton = 0 'False
ScaleHeight = 4920
ScaleWidth = 8415
Begin VB.VScrollBar VScroll1
Height = 4455
Left = 8160
TabIndex = 2
Top = 0
Width = 255
End
Begin MSComctlLib.StatusBar StatusBar1
Align = 2 'Align Bottom
Height = 375
Left = 0
TabIndex = 1
```

```
Top = 4545
Width = 8415
 ExtentX = 14843
 ExtentY = 661
 Version = 393216
BeginProperty Panels {8E3867A5-8586-11D1-B16A-00C0F0283628}
NumPanels = 1
BeginProperty Panel1 {8E3867AB-8586-11D1-B16A-00C0F0283628}
Object.Width = 8819
MinWidth = 8819
Text = "Please Choose one Alternative"
TextSave = "Please Choose one Alternative"
EndProperty
EndProperty
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "MS Sans Serif"
Size = 9.75
Charset = 0
Weight = 700
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
OLEDropMode = 1
End
Begin VB.CommandButton Command1
Caption = "Ok"
Height = 495
Left = 6960
TabIndex = 0
Top = 3960
Width = 1215
End
Situation - 2
End
```

Figure A.8. Situation Window


Situation

The Economic turn down.

[✓ Reducing supply]

Cutting Salary

Adding more advertising

Please Choose one Alternative

B. Analysis and Design Section 2: Situation Simulation

Structure Chart and Codes

Figure A.9. Situation Simulation: Data Flow Diagram

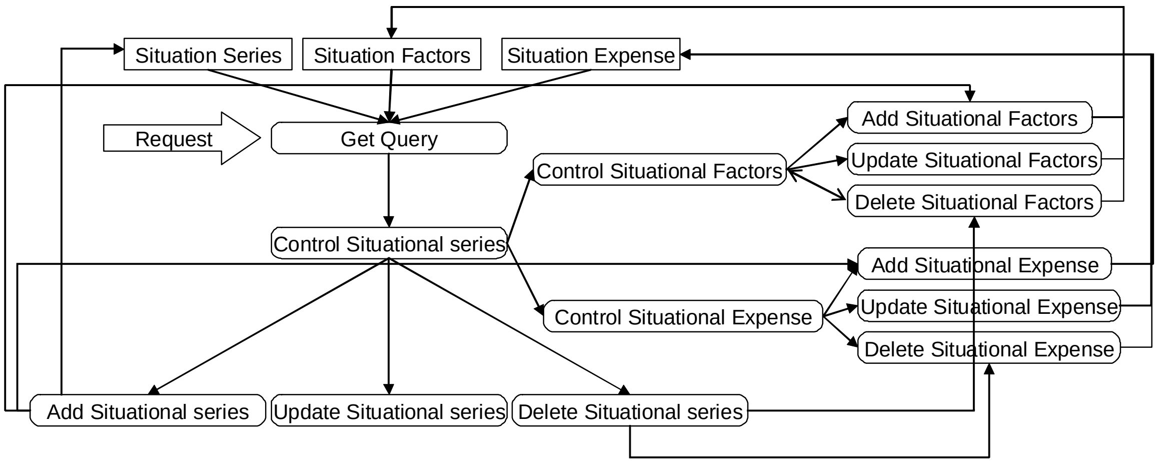

Situation Series
Situation Factors
Situation Expense
Request
Get Query
Control Situational series
Control Situational Factors
Add Situational Factors
Update Situational Factors
Delete Situational Factors
Control Situational Expense
Add Situational Expense
Update Situational Expense
Delete Situational Expense
Add Situational series
Update Situational series
Delete Situational series

Table A.12. Situation Simulation Modules

Process Item	Code
Get Query	Dim rs, db Set db = DBSituation.DBSituation

	'db.Open Set rs = db.Execute("select * from factors where Type ='ch'; rs.MoveFirst Do While Not rs.EOF ' ComboFactor.AddItem rs.Fields("var name rs.MoveNext Loop load factors load Expense
Control Situational Series	'Data.Refresh If TxtSerialType.Text = "Q" Then txtAnswer.Visible = True Else txtAnswer.Visible = False End If Data.Caption = "Series " & CStr(Data.Recordset.AbsolutePosition + 1) load factors load Expense
Control Situational Factor	Dim db, strsql, rs, i Set db = DBSituation.DBSituation If Data.Recordset.AbsolutePosition = -1 Then Data.Recordset.MoveLast strsql = "select * from situation factor where situation number = " & Data.Recordset("situation number Set rs = db.Execute(strsql) If ListofFactors.ListCount > 0 Then ListofFactors.Clear End If Do While Not rs.EOF ListofFactors.AddItem rs.Fields("var name & " " & rs.Fields("operator & " " & rs.Fields("var num rs.MoveNext Loop
Control Situational Expense	Dim db, strsql, rs, i Set db = DBSituation.DBSituation If Data.Recordset.AbsolutePosition = -1 Then Data.Recordset.MoveLast strsql = "select * from situation Expense where situation number = " & Data.Recordset("situation number Set rs = db.Execute(strsql) If ListExpense.ListCount > 0 Then ListExpense.Clear End If Do While Not rs.EOF

	ListExpense.AddItem rs.Fields("detail & " = " & rs.Fields("Amount & " * " & rs.Fields("Base rs.MoveNext Loop
Add Situational Series	Dim newnum If Data.Recordset.RecordCount <> 0 Then Data.Recordset.MoveLast newnum = Data.Recordset("number + 1 Data.Recordset.AddNew Data.Recordset("number = newnum Data.Recordset.Update Data.Recordset.MoveLast Data.Caption = "Series " & CStr(Data.Recordset.RecordCount) Else Data.Recordset.AddNew Data.Recordset("number = 1 Data.Recordset.Update Data.Recordset.MoveLast End If
Edit Situational Series	If Data.Recordset.RecordCount <> 0 Then Data.Recordset.Edit Data.Recordset.Update End If
Delete Situational Series	If Data.Recordset.RecordCount <> 0 Then DBSituation.DBSituation.Execute ("delete * from situation factor where situation number=" + CStr(Data.Recordset("Number) + "; DBSituation.DBSituation.Execute ("delete * from situation expense where situation number=" + CStr(Data.Recordset("Number) + "; Data.Recordset.Delete If Data.Recordset.RecordCount = Data.Recordset.AbsolutePosition + 1 Then Data.Recordset.MovePrevious Else Data.Recordset.MoveNext End If End If
Add Situational Factor	frmFactors.Show frmFactors.ZOrder frmFactors.Caption = "Insert Factor" …. …. If Caption = "Insert Factor" Then frmSeries.ListofFactors.AddItem ComboFactor.Text + " " + ComboListOp.Text + " " + txtvalue.Text Set db = DBSituation.DBSituation strsql = "INSERT INTO situation factor(situation number,var

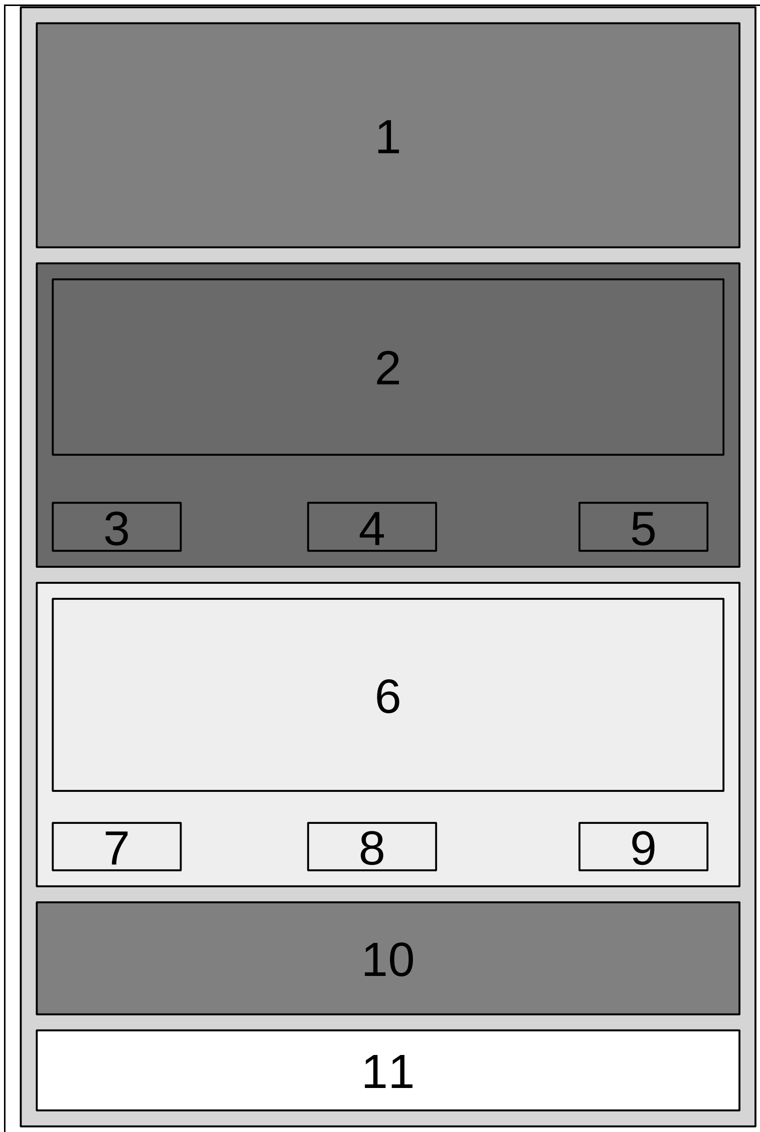
	name,operator,var num) values('" & frmSeries.Data.Recordset("situation
	number & "','" & ComboFactor.Text & "','" & ComboListOp.Text &
	"','" & txtvalue.Text & "');"
	db.Execute (strsql)
	End If
Edit Situational Factor	frmFactors.Show
	frmFactors.ZOrder
	frmFactors.Caption = "Update Factor"
	….
	….
	If Caption = "Update Factor" Then
	If frmSeries.ListofFactors.ListIndex <> -1 Then
	frmSeries.ListofFactors.RemoveItem
	(frmSeries.ListofFactors.ListIndex)
	frmSeries.ListofFactors.AddItem frmFactors.ComboFactor.Text + " " +
	frmFactors.ComboListOp.Text + " " + frmFactors.txtvalue.Text
	Set db = DBSituation.DBSituation
	strsql = "INSERT INTO situation factor(situation number,var
	name,operator,var num) values('" & frmSeries.Data.Recordset("situation
	number & "','" & ComboFactor.Text & "','" & ComboListOp.Text &
	"','" & txtvalue.Text & "');"
	db.Execute (strsql)
	'db.Refresh
	Else
	MsgBox "Please select factor"
	End If
	End If
Delete Situational Factor	frmFactors.Show
	frmFactors.ZOrder
	frmFactors.Caption = "Delete Factor"
	….
	….
	If Caption = "Delete Factor" Then
	If frmSeries.ListofFactors.ListIndex <> -1 Then
	frmSeries.ListofFactors.RemoveItem
	(frmSeries.ListofFactors.ListIndex)
	Set db = DBSituation.DBSituation
	strsql = "Delete * from situation factor where (situation number =" &
	frmSeries.Data.Recordset("situation number & and (var name='" &
	ComboFactor.Text & "') and (operator='" & ComboListOp.Text & "')
	and (var num='" & txtvalue.Text & "')"
	Set rs = db.Execute(strsql)
	Else
	MsgBox "Please select factor"
	End If
	End If

Add Situational Expense	frmExpense.Show frmExpense.ZOrder frmExpense.Caption = "Insert Expense" …. …. If Caption = "Insert Expense" Then frmSeries.ListExpense.AddItem txtDetail.Text + " = " + txtvalue.Text + " * " + ComboFactor.Text Set db = DBSituation.DBSituation strsql = "INSERT INTO situation expense(situation number,Detail,Amount,base) values('" & frmSeries.Data.Recordset("number & "','" & txtDetail.Text & "','" & txtvalue.Text & "','" & ComboFactor.Text & "');" db.Execute (strsql) End If
Edit Situational Expense	frmExpense.Show frmExpense.ZOrder frmExpense.Caption = "Update Expense" …. …. If Caption = "Update Expense" Then If frmSeries.ListExpense.ListIndex <> -1 Then frmSeries.ListExpense.RemoveItem (frmSeries.ListExpense.ListIndex) frmSeries.ListExpense.AddItem txtDetail.Text + " = " + txtvalue.Text + " * " + ComboFactor.Text Set db = DBSituation.DBSituation strsql = "INSERT INTO situation expense(situation number,Detail,Amount,base) values('" & frmSeries.Data.Recordset("situation number & "','" & txtDetail.Text & "','" & txtvalue.Text & "','" & ComboFactor.Text & "');" db.Execute (strsql) 'db.Refresh Else MsgBox "Please select factor" End If End If
Delete Situational Expense	frmExpense.Show frmExpense.ZOrder frmExpense.Caption = "Delete Expense" …. …. If Caption = "Delete Expense" Then If frmSeries.ListExpense.ListIndex <> -1 Then frmSeries.ListExpense.RemoveItem (frmSeries.ListExpense.ListIndex) Set db = DBSituation.DBSituation strsql = "Delete * from situation Expense where (situation number =" &

	frmSeries.Data.Recordset("number & and (Base='" & ComboFactor.Text & "') and (Detail='" & txtDetail.Text & "') and (Amount=" & txtvalue.Text & " Set rs = db.Execute(strsql) Else MsgBox "Please select factor" End If End If

Graphic User Interfaces

Table A.13.Situational Series Window

	1. The gray color area in number 1 and 10 contain situation series' fields. 2. The green color area contain situational factor and object interfaces control 3. The yellow color area contain situational expense and object interface control 4. The white color area are the situational series record control and object interfaces

	control.

* The first area is consisted of text box and combo boxes that are the details of situational series

*The second area is consisted of: 2 as a list box, 3 4, and 5 as the button box

*The third area is consisted of 6 as a list box, 7, 8, and 9 as the button box

*The last area is consisted of record control object and button boxes.

Table A.14.Object Script and properties

```
VERSION 5.00
Begin VB.Form frmSeries
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Series"
   ClientHeight    =   8805
   ClientLeft      =   1860
   ClientTop       =   1665
   ClientWidth     =   6675
   ControlBox      =   0   'False
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MDIChild        =   -1  'True
   MinButton       =   0   'False
   ScaleHeight     =   8805
   ScaleWidth      =   6675
   Begin VB.Frame Frame3
      Caption         =   "Answer"
      Height          =   975
      Left            =   0
      TabIndex        =   27
      Top             =   7080
      Width           =   6375
      Begin VB.TextBox txtAnswer
         DataField       =   "situation answer"
         DataSource      =   "Data"
         Height          =   615
         Left            =   120
         TabIndex        =   28
         Top             =   240
         Width           =   6135
      End
```

```
End
Begin VB.ComboBox Comboenv
   DataField       =   "Situation env"
   DataSource      =   "Data"
   Height          =   315
   Left            =   1560
   TabIndex        =   25
   Top             =   1320
   Width           =   1335
End
Begin VB.Frame Frame2
   Caption         =   "Schedule"
   Height          =   615
   Left            =   3000
   TabIndex        =   20
   Top             =   960
   Width           =   3255
   Begin VB.TextBox Text3
      DataField       =   "situation year"
      DataSource      =   "Data"
      Height          =   285
      Left            =   2280
      TabIndex        =   24
      Top             =   240
      Width           =   855
   End
   Begin VB.TextBox Text2
      DataField       =   "situation month"
      DataSource      =   "Data"
      Height          =   285
      Left            =   720
      TabIndex        =   21
      Top             =   240
      Width           =   855
   End
   Begin VB.Label Label3
      Caption         =   "Year"
      Height          =   255
      Left            =   1800
      TabIndex        =   23
      Top             =   240
      Width           =   495
   End
   Begin VB.Label Label2
      Caption         =   "Month"
      Height          =   255
      Left         =  120
      TabIndex     =  22
      Top          =  240
      Width        =  495
    End
  End
  Begin VB.TextBox TxtSerialType
    DataField      =  "Situation Type"
    DataSource     =  "Data"
    Height         =  285
    Left           =  1560
    TabIndex       =  18
    Top            =  960
    Width          =  1335
  End
  Begin VB.Frame Frame1
    Caption        =  "Expenses"
    Height         =  2655
    Left           =  0
    TabIndex       =  9
    Top            =  4440
    Width          =  6375
    Begin VB.CommandButton CmdDelExpense
      Caption      =  "Delete Expense"
      Height       =  375
      Left         =  4920
      TabIndex     =  17
      Top          =  2160
      Width        =  1335
    End
    Begin VB.CommandButton CmdUpdateExpense
      Caption      =  "Update Expense"
      Height       =  375
      Left         =  2520
      TabIndex     =  16
      Top          =  2160
      Width        =  1335
    End
    Begin VB.CommandButton CmdAddExpense
      Caption      =  "Add Expense"
      Height       =  375
      Left         =  120
      TabIndex     =  15
      Top          =  2160
      Width        =  1215
    End
```

```
      Begin VB.TextBox TxtexoExpense
         Height          =   1815
         Left            =   3960
         TabIndex        =   11
         Top             =   240
         Width           =   2295
      End
      Begin VB.ListBox ListExpense
         Height          =   1815
         Left            =   120
         TabIndex        =   10
         Top             =   240
         Width           =   3855
      End
   End
   Begin VB.Frame FrameFactor
      Caption         =   "Factors"
      Height          =   2655
      Left            =   0
      TabIndex        =   6
      Top             =   1680
      Width           =   6375
      Begin VB.CommandButton cmdaddfactor
         Caption         =   "Add Factor"
         Height          =   375
         Left            =   120
         TabIndex        =   14
         Top             =   2160
         Width           =   1215
      End
      Begin VB.CommandButton delfactor
         Caption         =   "Delete Factor"
         Height          =   375
         Left            =   5040
         TabIndex        =   13
         Top             =   2160
         Width           =   1215
      End
      Begin VB.CommandButton cmdupdatefactor
         Caption         =   "Update Factor"
         Height          =   375
         Left            =   2760
         TabIndex        =   12
         Top             =   2160
         Width           =   1215
      End
```

```
    Begin VB.TextBox txtexpFactor
      Height          =   1815
      Left            =   3960
      TabIndex        =   8
      Top             =   240
      Width           =   2295
    End
    Begin VB.ListBox ListofFactors
      DataMember      =   "Situation Factors"
      DataSource      =   "DBSituation"
      Height          =   1815
      Left            =   120
      TabIndex        =   7
      Top             =   240
      Width           =   3855
    End
  End
  Begin VB.PictureBox picButtons
    Align           =   2  'Align Bottom
    Appearance      =   0  'Flat
    BorderStyle     =   0  'None
    ForeColor       =   &H80000008&
    Height          =   660
    Left            =   0
    ScaleHeight     =   660
    ScaleWidth      =   6675
    TabIndex        =   1
    Top             =   8145
    Width           =   6675
    Begin VB.Data Data
      Caption         =   "Data1"
      Connect         =   "Access 2000;"
      DatabaseName    =   "C:\thesis\base.mdb"
      DefaultCursorType=   0  'DefaultCursor
      DefaultType     =   2  'UseODBC
      Exclusive       =   0   'False
      Height          =   375
      Left            =   360
      Options         =   0
      ReadOnly        =   0   'False
      RecordsetType   =   1  'Dynaset
      RecordSource    =   "situation"
      Top             =   0
      Width           =   5415
    End
    Begin VB.CommandButton cmdRefresh
```

```
      Caption         =   "&Refresh"
      Height          =   300
      Left            =   3600
      TabIndex        =   5
      Top             =   360
      Width           =   1095
    End
    Begin VB.CommandButton cmdDelete
      Caption         =   "&Delete"
      Height          =   300
      Left            =   2520
      TabIndex        =   4
      Top             =   360
      Width           =   1095
    End
    Begin VB.CommandButton cmdUpdate
      Caption         =   "&Update"
      Height          =   300
      Left            =   1440
      TabIndex        =   3
      Top             =   360
      Width           =   1095
    End
    Begin VB.CommandButton cmdAdd
      Caption         =   "&Add"
      Height          =   300
      Left            =   360
      TabIndex        =   2
      Top             =   360
      Width           =   1095
    End
  End
  Begin VB.TextBox TxtSituation
    DataField       =   "SituationDetail"
    DataSource      =   "Data"
    Height          =   855
    Left            =   0
    MultiLine       =   -1  'True
    ScrollBars      =   2  'Vertical
    TabIndex        =   0
    Top             =   0
    Width           =   6375
  End
  Begin VB.Label Label4
    Caption         =   "External Environment"
    Height          =   375
    Left          =   0
    TabIndex      =   26
    Top           =   1320
    Width         =   1575
  End
  Begin VB.Label Label1
    Caption       =   "SeriesType"
    Height        =   255
    Left          =   0
    TabIndex      =   19
    Top           =   960
    Width         =   1095
  End
End
```

Figure A.10. Situation Series Window

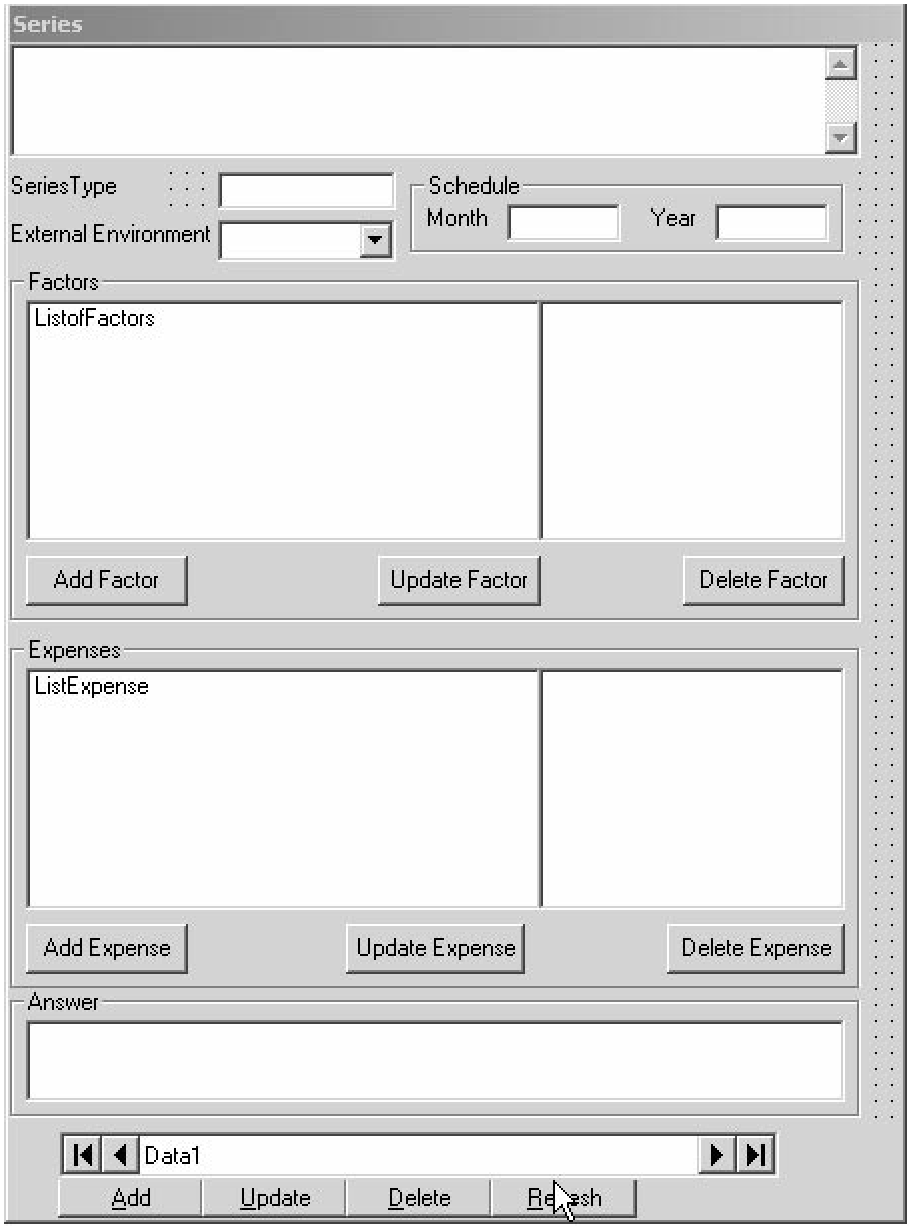
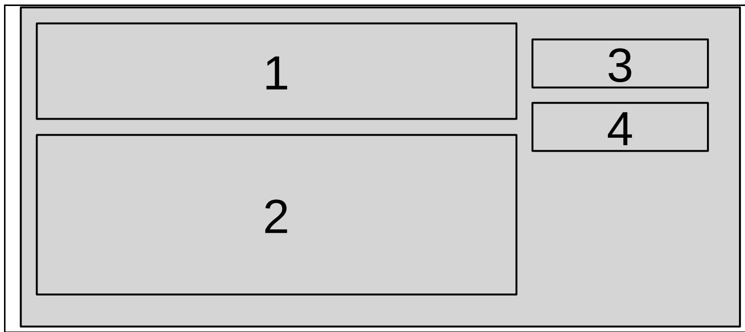


Table A.15. Situational Factor Window

	1. The mathematical function area of situational factor. 2. The description of the

	business factor 3. 3 and 4 are the object interfaces

* The first and areas are details, which are in text boxes and combo box, of situational factor.

* The third and fourth areas are button boxes.

Table A. 16. Object Script and properties

```
VERSION 5.00
Begin VB.Form frmExpense
  Caption        =   "Expense"
  ClientHeight    =   1515
  ClientLeft      =   60
  ClientTop       =   345
  ClientWidth     =   5115
  ControlBox      =   0   'False
  LinkTopic       =   "Form"
  MDIChild        =   -1  'True
  ScaleHeight     =   1515
  ScaleWidth      =   5115
  Begin VB.CommandButton cmdOK
    Caption        =   "OK"
    Height         =   315
    Left           =   3840
    TabIndex       =   7
    Top            =   240
    Width          =   1095
  End
  Begin VB.CommandButton cmdcancel
    Caption        =   "Cancel"
    Height         =   315
    Left           =   3840
    TabIndex       =   6
    Top            =   600
    Width          =   1095
  End
  Begin VB.TextBox txtDetail
    Height         =   735
    Left           =   840
    TabIndex       =   2
    Top            =   360
    Width          =   2775
  End
  Begin VB.TextBox txtvalue
    Height         =   315
    Left           =   840
    TabIndex       =   1
    Top            =   1080
    Width          =   855
  End
  Begin VB.ComboBox ComboFactor
    Height         =   315
    Left           =   840
    TabIndex       =   0
    Top            =   0
    Width          =   1815
  End
  Begin VB.Label Label3
    Caption        =   "Value"
    Height         =   255
    Left           =   0
    TabIndex       =   5
    Top            =   1080
    Width          =   855
  End
  Begin VB.Label Label2
    Caption        =   "Detail"
    Height         =   255
    Left           =   0
    TabIndex       =   4
    Top            =   360
    Width          =   855
  End
  Begin VB.Label Label1
    Caption        =   "Base"
    Height         =   255
    Left           =   0
    TabIndex       =   3
    Top            =   0
    Width          =   855
  End
End
```

Figure A.11. Situation Factor Window

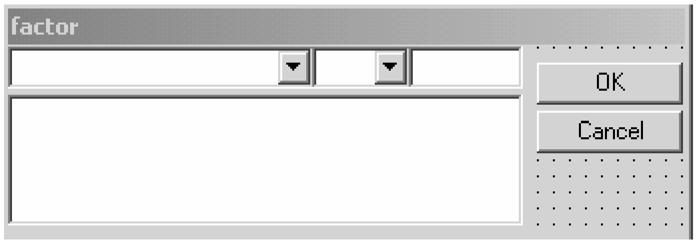
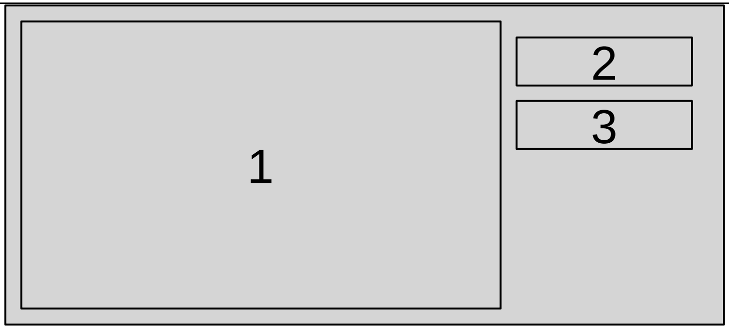


Table A.17. Situational Expense Window

	1. The mathematical function area of situational expense. 2. 2 and 3 are the object interface control

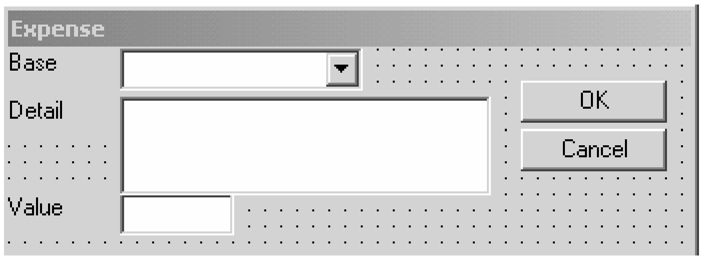
*The first area is text boxes and combo box which

* The number 2 and 3 are button boxes

Table A.18. Object Script and properties

```
VERSION 5.00
Begin VB.Form frmFactors
  Caption         =   "factor"
  ClientHeight    =   1410
  ClientLeft      =   60
  ClientTop       =   345
  ClientWidth     =   5070
  ControlBox      =   0   'False
  LinkTopic       =   "Form1"
  MDIChild        =   -1  'True
  ScaleHeight     =   1410
  ScaleWidth      =   5070
  Begin VB.TextBox txtExp
    Height          =   975
    Left            =   0
    TabIndex        =   5
    Top             =   360
    Width           =   3855
  End
  Begin VB.CommandButton cmdcancel
    Caption         =   "Cancel"
    Height          =   315
    Left            =   3960
    TabIndex        =   4
    Top             =   480
    Width           =   1095
  End
  Begin VB.CommandButton cmdOK
    Caption         =   "OK"
    Height          =   315
    Left            =   3960
    TabIndex        =   3
    Top             =   120
    Width           =   1095
  End
  Begin VB.TextBox txtvalue
    Height          =   315
    Left            =   3000
    TabIndex        =   2
    Top             =   0
    Width           =   855
  End
  Begin VB.ComboBox ComboListOp
    Height          =   315
    Left            =   2280
    TabIndex        =   1
    Top             =   0
    Width           =   735
  End
  Begin VB.ComboBox ComboFactor
    Height          =   315
    Left            =   0
    TabIndex        =   0
    Top             =   0
    Width           =   2295
  End
End
```

Figure A.12. Situation Expense Window


Expense
Base
Detail
Value
OK
Cancel

C. Database Analysis

Database Object

Table A.19. SQL Command in The Data Object

Connection	SQL Command
Factors	Select * From Currentdata;
Situation Factor	Select * From Situation;
Situation	Select * from situation factor;
MonthlyExp	Select * from situation expense

DataBase Tables' Structure

Table A.20. The Structure of CurrentFactor

CurrentFactor		
Field Name	Type	Description
Event	Text	Identify process module of factor.
Name	Text	Identify name of factor
Value	Number	Value of the factor
Type	Text	Identify type of factor if it changeable or calculated
Desc	Text	Description of the factor

Table A.21. The Structure of Situation

Situation		
Field Name	Type	Description
Number	Number	Identify situation series
SituationDetail	Memo	Describe of the situation
Situation Type	Text	Situation Type as Question or Answer
Situation env	Text	Business environment that situation is in
Situation month	Number	The scheduling month in the game, the number identify which month the situation is
Situation year	Number	The scheduling year in the game, the number identify which year the situation is
Situation answer	Memo	Text describe the best alternative in the situation

Table A.22. The Structure of Expense

Situation Expense		
Field Name	Type	Description
Situation number	Number	Identify the situation cause impacted factor
Base	Text	A factors base of expense, referring to factor’s value
Detail	Text	Describe of expense
Amount	Number	Expense value for item per base factor

Table A.23. The Structure of Situation Factors

Situation Factor		
Field Name	Type	Description
Situation number	Number	Identify the situation cause impacted factor
Var name	Text	A factor for situation impact
Operation	Text	Symbol of operation the factor such as subtract
Amount	Number	The number to operate to factor and give value to that factor

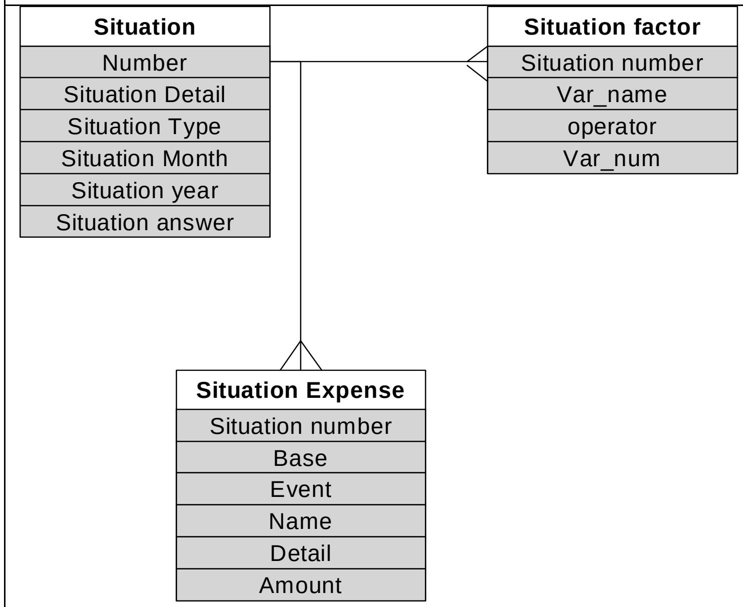
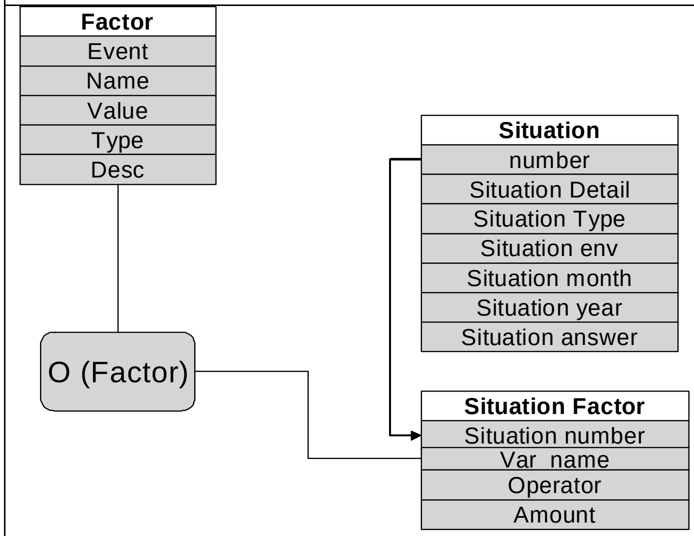
Table A.24. The Structure of ExpenseBase

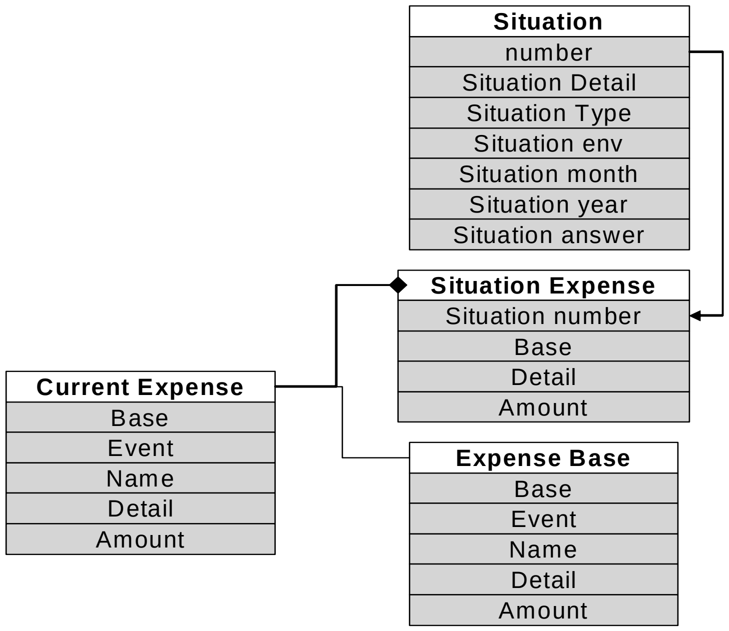
ExpenseBase		
Field Name	Type	Description
Base	Text	A factors base of expense referring to factor’s value
Event	Text	Event process calculated factor
Name	Text	Process expense name
Detail	Text	Expense text detail
Amount	Number	Expense value

Table A.25 The Structure of CurrentExpense

CurrentExpense		
Field Name	Type	Description
Base	Text	A factors base of expense referring to factor’s value
Event	Text	Event process calculated factor
Name	Text	Process expense name
Detail	Text	Expense text detail
Amount	Number	Expense value

Table A.26. Database Relationship

	• There are two relationship from situation table • A situation can have many impacted factors. • A situation can have many expenses
	• The Factor table is mapped to data object variable. • Situation factor table is a child table from Situation table. • The situation factor has a join relation with data object variable

 Situation number Situation Detail Situation Type Situation env Situation month Situation year Situation answer Situation Expense Situation number Base Detail Amount Current Expense Base Event Name Detail Amount Expense Base Base Event Name Detail Amount	• There are relationships among three tables. • The current expense table is the result of joining between situation expense and expense base. • The situation expense is linked to the parented table where situation month field equates simulation month in the game.

D. User Manual.

Table A.27. Student Manual

Students

Install

1. Double click on CD drive where The Simulation Game's CD has been inserted
2. Double click on the folder "The Simulation Game"
3. Run "Setup.exe" file (double click on "Setup.exe")

Running the Simulation Game

1. Before playing, make sure that you copy "base.mdb" from an instructor to the folder that contains the game application (for example checking on "c:\Program Files\The Simulation Game"). The original "base.mdb" file will be included in the installed CD.
2. Go to Window "Start" bar, "Programs" and "The Simulation Game"
3. Run "The Simulation Game"
4. Wait for the simulation game routine run for a month and get the monthly report and situational simulation for next month.
5. Select the base alternative for next month situation and close the situation window to continuously run the game

* The goal of the game is the make a managerial knowledge to select the base alternative for each month situation.

* Tip of playing the simulation game is to click on the CATscanner model's internal environment and observes the factors' variables.

Table A.28. Instructor Manual

Instructors

Install

1. Double click on CD drive where The Simulation Game's CD has been inserted
2. Double click on the folder "The Situational Simulation"
3. Run "Setup.exe" file (double click on "Setup.exe")

Running the Situational Simulation

1. Before running, make sure that you have "base.mdb" on the folder that contains the application (for example checking on c:\Program Files\The Situational Simulation"). The original "base.mdb" file will be included in the installed CD.
2. Go to Window "Start" bar, "Programs" and "The Situational Simulation"
3. Run "The Simulation Game"

*The user can use the recordset object go to any situation record and follow the controlling situation direction.

Controlling Situation

Create Situation

1. Click "Add" button on the bottom of the window to add the new record situation's record.
2. Add data into the text boxes.
3. Insert data on the window and click "Update" button on the bottom of the window.

Update Situation

1. Go to the record that you want to change the data.
2. Go to a text box area to change data
3. Click "Update" button on the button of the window

Delete Situation

1. Go to the record that you want to delete
2. Click "Delete" button on the button of the window.

Create Factor in The Situation

1. In the factor section, Click the "Add factor"
2. After "Insert Factor" window pop-upped, Insert the data to text boxes
3. Click "OK" to confirm the inserted data or "Cancel" to not save the data.

Update Factor in The Situation

1. Select the factor item in the factor list box that you want to edit data
2. In the factor section, click the "Update factor"
3. After "Update Factor" window pop-upped, edit the data on the window.
4. Click "OK" to confirm updating or "Cancel" to not save the data.

Delete Factor in The Situation

1. Select the factor item in the factor list box that you want to delete
2. In the factor section, click the "Delete factor."

3. Click “OK” to confirm deleting or “Cancel” to not delete

*To control factors, make sure that you are on the correct situation record that you want to work with

Create Expense in The Situation

1. In the expense section, Click the “Add expense.”
2. After “Insert Expense” window pop-upped, Insert the data to text boxes.
3. Click “OK” to confirm the inserted data or “Cancel” to not save the data.

Update Expense in The Situation

1. Select the expense item in the expense list box that you want to edit data
2. In the expense section, click the “Update expense.”
3. After “Update Expense” window pop-upped, edit the data on the window.
4. Click “OK” to confirm updating or “Cancel” to not save the data.

Delete Expense in The Situation

1. Select the expense item in the factor list box that you want to delete.
2. In the expense section, click the “Delete expense.”
3. Click “OK” to confirm deleting or “Cancel” to not delete.

*To control expense, make sure that you are on the correct situation record that you want to work with

* After creating the situations and saving them on the database, make sure to check the reasonable answer and send the “base.mdb” to students