

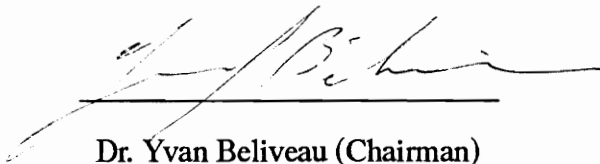
# Visual Schedule Simulation System (VSS)

By  
Jeffrey F. Skolnick

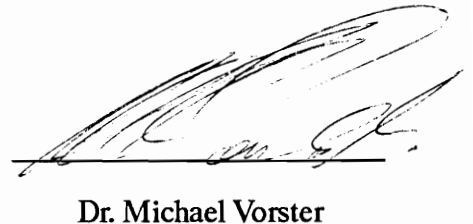
Thesis submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

Master of Science  
in  
Civil Engineering

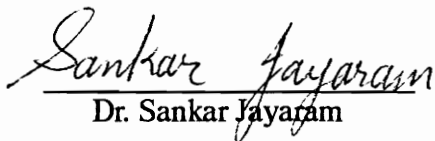
APPROVED



Dr. Yvan Beliveau (Chairman)



Dr. Michael Vorster



Dr. Sankar Jayaram



Vernon Francisco

*August 1990*  
*Blacksburg, Virginia*

LD  
5655

V855

1990

S589

0.2

# **The Visual Schedule Simulation System - VSS**

By

Jeffrey F. Skolnick

Yvan J. Beliveau, Chairman

The Charles Edward Via, Jr. Department of Civil Engineering

## **(ABSTRACT)**

Current planning and scheduling techniques are carried out in an unstructured form with considerable reliance on planners judgement, imagination and intuition. The final product of such techniques is typically a lengthy textural and tabular report and/or symbolic network. This serves as an abstract rather than visual modeling of the real construction process. The availability of advanced computer hardware and software allows us, today, to develop new planning and scheduling techniques to overcome the current limitations.

Computer-Aided Design (CAD) is a computing system which makes extensive use of computer graphics. The use of CAD systems in construction presents a great opportunity for integrating engineering and construction processes in a more cost effective way. The combination of computer graphics, animation, and 3D computer modeling can be extremely effective for real-time simulation and visualization to support engineering and construction from the conceptual design to the construction process.

This thesis presents a new planning and scheduling system. The system combines a construction scheduling network with 3D computer models to form a Visual Scheduling

Simulation (VSS) of the construction process. The VSS system simulates, or put into motion, construction activities so they can be viewed at a graphics display.

The VSS system allows the user to view the actual and planned construction sequence. The user has the option to view either: planned schedule; actual schedule; or both schedules shown side by side for quick visual comparison. The user has the option to visually simulate the entire construction project, or any specified time period. This simulation can be viewed for a partial segment or the entire configuration of the project.



## ACKNOWLEDGEMENTS

My committee made significant contributions to this thesis. Dr. Yvan Beliveau oversaw my work and provided guidance. Without his help this thesis would be of lesser quality. Dr. Michael Vorster continually reviewed and offered suggestions on the structure of this thesis. His help made this thesis more complete. Dr. Sankar Jayaram taught me a great deal on the technical aspects of this thesis and provided assistance whenever needed. Vernon Francisco helped me with the ideas and development of the Visual Simulator. I feel fortunate to have had a committee of such fine people. I thank them all.

I would also like to thank Robert Lewis, Eric Lundberg, and Walid Thabet for their help preparing the thesis presentation. I would like to thank Aiman Morad for his aid in developing the ideas for this thesis.

Finally, I would like to thank my parents Richard and Phyllis Skolnick and my girlfriend Beth Bryant. Without their continual support and encouragement I would have had a more difficult time completing my Master's degree.

# ***TABLE OF CONTENTS***

<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 The Visual Schedule Simulation System .....	2
1.2 System Objectives .....	4
1.3 System Description .....	5
1.4 Design Methodology .....	9
1.5 Scope of Work Performed .....	10
1.6 System Limitations .....	12
1.7 Presentation of Thesis .....	13
<b>2. TRADITIONAL SCHEDULING TECHNIQUES .....</b>	<b>16</b>
2.1 Historical Backgroud .....	16
2.2 Current Practices with CPM Technology .....	18
2.2.1 Network Use in Construction .....	18
2.2.2 Planning and Scheduling in the Construction Industry .....	19
2.3 Problems With CPM Techniques and a Proposed Solution .....	20
2.3.1 Problems and Deficiencies with CPM Output for Controlling Construction Projects .....	20
2.3.2 A Proposed Solution .....	21
<b>3. TECHNOLOGY REVIEW .....</b>	<b>23</b>
3.1 Primavera Project Planner .....	24
3.1.1 System Description .....	24
3.1.2 Primavera Project Planner and the VSS System .....	24
3.2 Computer Aided Design .....	25
3.2.1 Interactive Computer Graphics .....	25
3.2.2 Geometric Modeling .....	26
3.2.3 Graphical Capabilities of CAD Systems .....	28
3.2.4 CAD and the VSS System .....	29
3.3 Relational Database Management Systems .....	29

3.3.1	Relational Database Filing Systems .....	30
3.3.2	dBASE IV and the VSS System .....	30
3.4	WALKTHRU .....	32
3.4.1	System Description .....	32
3.4.2	System Features .....	33
3.4.3	WALKTHRU and the VSS System .....	35
<b>4.</b>	<b>REVIEW OF SIMILAR SYSTEMS .....</b>	<b>36</b>
4.1	Simulation System for Construction Planning and Scheduling .....	37
4.2	Three-Dimensional Solid Modeling and Database Integration for Construction Project Management .....	38
4.3	KNOW-PLAN .....	39
4.4	CAD Based Schedule Representation .....	41
4.5	Summary .....	41
<b>5.</b>	<b>VISUAL SCHEDULE SIMULATION SYSTEM .....</b>	<b>45</b>
5.1	System Overview .....	45
5.2	Data Preprocessor .....	48
5.2.1	The Construction Network .....	48
5.2.2	The 3D Computer Modeling System .....	51
5.3	The Database Manager .....	53
5.3.1	"Build Model File" Function .....	54
5.3.2	"Map Model Objects to Activities" Function .....	54
5.3.3	"Compute Object Dates" Function .....	56
5.3.4	"Export dBASE File to ASCII File" Function .....	59
5.3.5	"Exit the VSS System" Function .....	59
5.4	The Visual Simulator .....	59
5.4.1	System Description .....	60
5.4.2	System Features .....	61
<b>6.</b>	<b>VSS SYSTEM IMPLEMENTATION .....</b>	<b>67</b>
6.1	Phase II - The Database Manager .....	68
6.1.1	"Build Model File" .....	71
6.1.2	"Map Model Objects to Activities" .....	73
6.1.3	Compute Object Dates" .....	75
6.1.4	"Export dBASE File to ASCII File" .....	78
6.2	Phase III - The Visual Simulator .....	79

6.2.1	Creating the WALKTHRU File from the Original 3D Computer Model .....	83
6.2.1.1	WALKIGDS and WALK3DM .....	83
6.2.1.2	WALKIGES .....	84
6.2.1.3	WALKPRE .....	85
6.2.1.4	Merging WALKTHRU model files .....	85
6.2.2	The "File -" Function .....	86
6.2.2.1	The Input File .....	88
6.2.2.2	Project Start Date .....	89
6.2.2.3	Object Name File .....	90
6.2.3	The "Time -" Function .....	96
6.2.4	The "Early Dates/Late Dates/Early & Late Dates" Function .....	96
6.2.5	The "Planned/Actual/Planned & Actual" Function .....	97
6.2.6	The "Start Date -" and "Finish Date -" Functions .....	98
6.2.7	The "Execute" Function .....	100
7.	CONCLUSION .....	105
7.1	Summary .....	105
7.1.1	Computer Software .....	107
7.1.2	Computer Hardware .....	108
7.2	Recommendations and Benefits .....	108
7.3	Future Extensions .....	111
7.4	Conclusion .....	113
 <b><u>APPENDIX</u></b>		
A.	USER'S GUIDE .....	115
A.1	Stage 1 .....	116
A.1.1	Exporting the CPM Schedule .....	117
A.1.2	Naming 3D Computer Model Objects .....	123
A.1.3	Creating the WALKTHRU Model File .....	124
A.2	Stage 2 .....	126
A.2.1	Step 1 - "Build Model File" .....	128
A.2.2	Step 2 - "Map Model Objects to Activities .....	130
A.2.3	Step 3 - "Compute Object Dates" .....	133
A.2.4	Step 4 - "Exporting dBASE File to ASCII File .....	135
A.3	Stage 3 .....	136

A.3.1	Part 1 - Preparing the 3D Computer Model .....	137
A.3.2	Part 2 - Creating the Object Name File .....	139
A.3.3	Part 3 - Operating the Visual Simulator .....	140
A.3.3.1	Menu Item 1 .....	142
A.3.3.2	Menu Item 2 .....	143
A.3.3.3	Menu Item 3 .....	144
A.3.3.4	Menu Item 4 .....	144
A.3.3.5	Menu Item's 5 and 6 .....	145
A.3.3.6	Menu Item 7 .....	146
A.3.3.7	Menu Item 8 .....	147
A.3.4	Quick Reference .....	148
<b>B.</b>	<b>FLOWCHARTS .....</b>	<b>151</b>
B.1	The Database Manager - Phase II .....	152
B.2	The Visual Simulator - Phase III .....	161
<b>C.</b>	<b>SOURCE CODE .....</b>	<b>175</b>
C.1	The Database Manager - Phase II .....	176
C.2	The Visual Simulator - Phase III .....	199
<b>REFERENCES .....</b>		<b>235</b>
<b>VITA .....</b>		<b>237</b>

## ***LIST OF FIGURES***

Figure 1.1: The VSS System .....	6
Figure 3.1: Geometric Modeling Process .....	27
Figure 3.2: Example of a Typical Database File .....	31
Figure 4.1: Comparison of Relevant Systems .....	44
Figure 5.1: Flow of Data in the VSS System .....	47
Figure 5.2: Typical scheduling file exported from Primavera .....	50
Figure 5.3: The Mapping Process .....	55
Figure 5.4: Activity/Object Mapping .....	56
Figure 5.5: Activity Schedule .....	57
Figure 5.6: Activity/Object Mapping .....	58
Figure 5.7: Object Schedule .....	58
Figure 5.8: The Execution Process .....	66
Figure 6.1: Database Manager's Main Menu .....	69
Figure 6.2: Compute Object Dates process .....	76
Figure 6.3: Visual Simulator's Main Menu .....	80
Figure 6.4: Positioning of Primary and Secondary 3D computer Models .....	87
Figure 6.5: Example of a typical Object Name File .....	93
Figure 6.6: Example of information contained in the schedule structure .....	95
Figure 6.7: Information passed into the simulation function .....	102
Figure 6.8: Simulation algorithm .....	104

Figure A.1: The "EXPORT DATA FILES" Screen ..... 119

Figure A.2: The "EXPORT DATA FILES" Screen (window/format) ..... 121

Figure A.3: The Database Manager's Main Menu ..... 127

Figure A.4: The Map Model Objects to Activities Screen ..... 132

Figure A.5: The Visual Simulator's Main Menu ..... 141

## ***1. INTRODUCTION***

Each year new and innovative construction materials and methods are being discovered allowing architects and engineers to design increasingly complex construction projects. These complex projects add to the already complex bidding operation. Planners are using many different tools to aid in planning and scheduling these projects in order to remain competitive and secure construction contracts. Perhaps the most powerful tools used are computer based project management techniques.

The most widely used project management technique employs network based methodology, such as the Critical Path Method (CPM). CPM is used as the basis for activity tracking, scheduling, resource utilization planning, and work breakdown structure [Simons et al. 1988]. CPM assists estimators, planners and other project management personnel to ensure that a construction project can be completed within budgeted costs and time constraints. CPM techniques are also used to control the project once construction begins. CPM schedules are easier to update and manage using these computer based techniques thus alleviating the manual, more time consuming methods previously used.

Even though significant advances have been made in the area of project management, construction projects still incur cost overruns and delays. These overruns and delays may be due to improper use of current project management systems; however, much of the



problem lies in the deficiencies of these systems.

Project management systems often require that the planner form a mental picture of configurations, characteristics, and spatial relationships among the various components of the construction project. This becomes extremely difficult as projects increase in size and complexity. Another problem with current project management systems is the way information is presented. Output consists of lengthy textual reports and/or symbolic networks. This forces construction personnel to mentally visualize the construction process, thus causing communication problems if the same information is perceived differently [Morad & Beliveau 1990].

Technology is now available to develop systems which enhance current project management techniques. Recent advances in computer hardware and software allow problems with construction planning, scheduling and controlling to be addressed in entirely new ways. These advances include the following technologies relevant to project management:

1. 3D computer modeling systems
2. Development of new graphics superworkstations
3. Relational database management systems
4. Enhancements to commercial project management systems.

This technology can be combined to create a system which alleviates the problem of mentally visualizing the construction sequence.

### ***1.1 The Visual Schedule Simulation System***

This thesis describes the development of the Visual Scheduling Simulation system (VSS). This system expands traditional CPM scheduling output by providing a new method of viewing the sequence of construction activities. The VSS system is designed to integrate CPM scheduling technology with 3D computer modeling technology. This will aid planners and other construction personnel view the construction sequence on a graphics display. This system overcomes the visual limitations of current scheduling systems. This allows the user to view the sequence of construction activities using 3D computer models rather than symbolic networks.

The VSS system extracts information from both a CPM schedule of a construction project and a 3D computer model of the construction project. The start and finish dates assigned to each construction activity are associated or mapped to related objects in the 3D computer model. This gives start and finish dates to each object thus producing a schedule of object dates (object schedule). The VSS system then provides for a visual simulation of the construction sequence. The VSS system uses a 3D computer model of the construction project and displays the construction sequence on a graphics display.

The VSS system also provides the user with the ability to customize the visual simulation. To accomplish this, the VSS system uses the object schedule information which defines the start and finish dates associated with each computer model object. The user is allowed to modify the simulation. The user can visually simulate the construction process for any specified time period using: early or late date scheduling logic, and planned and/or actual construction progress.

## ***1. INTRODUCTION***

## ***1.2 System Objectives***

The primary objective in developing this thesis of study was to produce a tool for use by planners and designers to visualize construction schedules. Planners and designers are able to view the sequence of construction activities on a graphics display. This helps to determine if there are any scheduling errors before construction begins.

The secondary objective in developing this thesis was to provide for improvement in total project management. The VSS system provides a tool to improve communication between construction personnel involved with a construction project. The VSS system allows project management personnel to experiment with different scenarios to determine the optimum construction sequence. The VSS system provides an innovative method for controlling the construction project. It lets construction management visually compare the planned sequence of construction activities with actual progress.

The VSS system was designed to combine several technologies which are commonly used by construction and engineering firms. The VSS system will help these firms better cost justify using systems such as computer based project management systems and Computer Aided Design (CAD) system. The ultimate goal of the VSS system was to provide a new method which will better aid construction management complete projects on time and within budget.

### ***1.3 System Description***

This section gives an overview of the VSS system. Chapters 5 and 6 describe the VSS system in greater detail. The VSS system combines three separate phases in order to produce a visual simulation of the construction process. Phase I is the Data Preprocessor, Phase II is the Database Manager, and Phase III is the Visual Simulator. Figure 1.1 depicts each of these phases and illustrates how they interact with one another. These three phases are discussed in turn below.

Phase I of the VSS system is the Data Preprocessor. The Data Preprocessor involves using a computer based project management system such as Primavera Project Planner to plan and schedule the construction project. Phase I also involves the creation or use of an existing 3D computer model of the construction project. This 3D model can be created on a 3D computer modeling system such as CADAM.

The information regarding the construction schedule developed using Primavera can be exported to a file format and transferred to Phase II of the VSS system. The names of each object in the 3D computer model must be entered manually by the user into Phase II of the VSS system. The geometric parameters of the 3D computer model are contained in a file and transferred to Phase III of the VSS system.

Phase II of the VSS system is the Database Manager. This phase allows the user to link the information from the CPM schedule with the 3D computer model. Phase II uses dBASE IV which is a Relational Database Management System to manipulate the CPM schedule

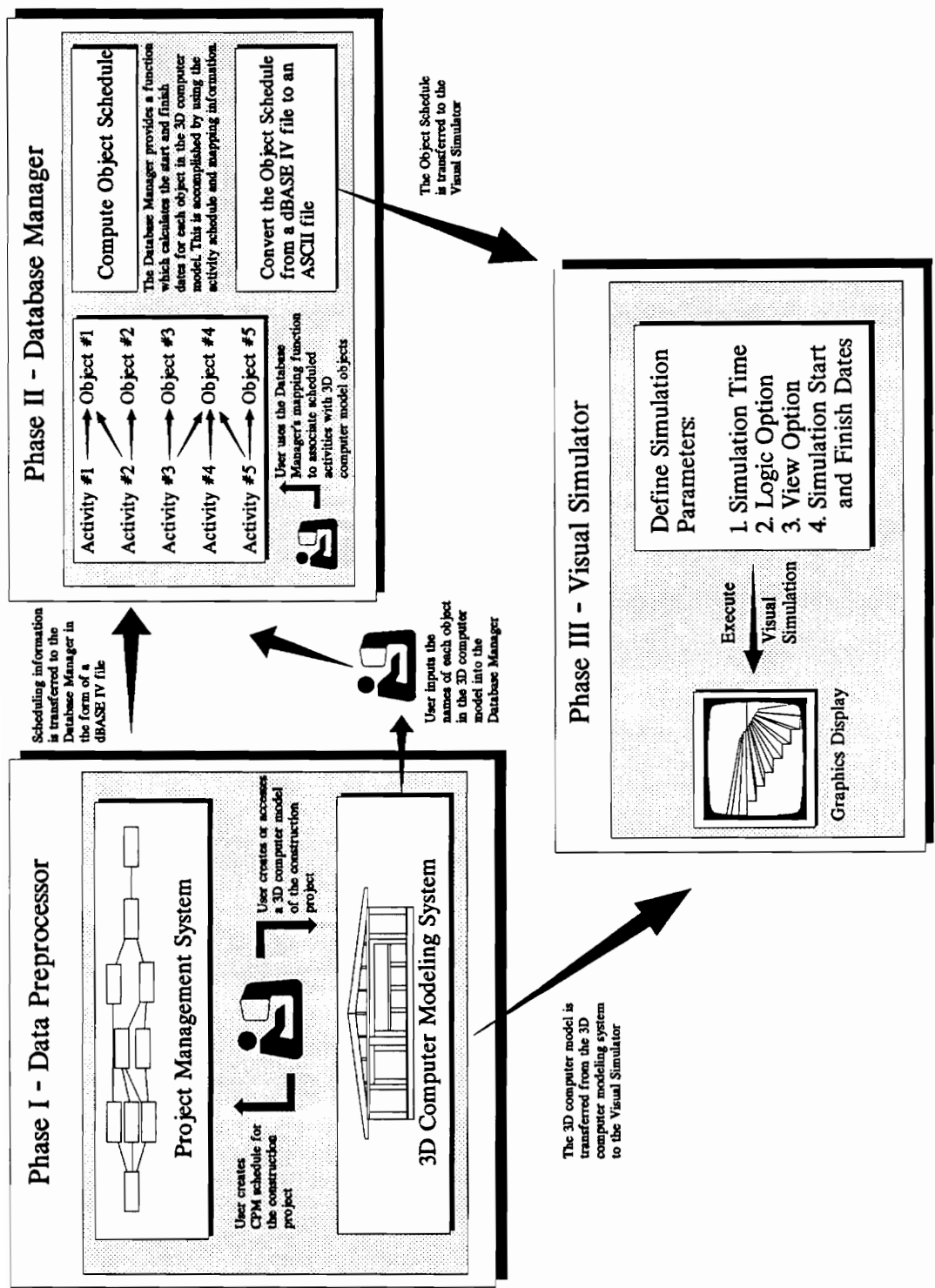


Figure 1.1: The VSS System

information and 3D computer model information.

The Database Manager allows the user to define the schedule file for a construction project which has been imported from Phase I, the Data Preprocessor. It also allows the user to input the names for each object in the 3D computer model for a construction project developed during Phase I.

The Database Manager allows the user to define the relationships between the construction activities in the CPM schedule and the objects defined in the 3D computer model. These relationships are linked using the Database Manager's mapping function. The mapping function lets the user enter all of the activities associated with each of the 3D computer model objects.

Start and finish dates associated with scheduled activities must be assigned to the 3D computer model objects for which these activities are associated. This is necessary in order to provide a visual simulation of the construction sequence. The Database Manager provides a function which computes object start and finish dates. This function uses the mapping information previously defined to determine which activity start and finish dates to assign to each object in the 3D computer model. The result of Phase II is an object schedule which defines the start and finish date for each object in the 3D computer model.

The object schedule is stored in a dBASE IV file. The Database Manager provides a function which converts the object schedule into an ASCII file format. This file is transferred to Phase III of the VSS system.

## **1. INTRODUCTION**

Phase III of the VSS system is the Visual Simulator. The Visual Simulator visually simulates or put into motion the construction sequence of activities. This phase uses the WALKTHRU simulation software. WALKTHRU is a 3D simulation and visualization system developed by Bechtel Software Corporation. WALKTHRU is discussed in detail in Section 3.4 of Chapter 3.

The Visual Simulator uses the 3D computer model developed during Phase I. The 3D computer model must first be transferred from the 3D computer modeling system to the WALKTHRU system.

The Visual Simulator allows the user to customize the simulation process. It provides a set of user interactive functions which permit the user to define the following simulation parameters:

1. Simulation Time - This allows the user to adjust the speed of simulation.
2. Logic Option - This allows the user to select between early start and early finish dates, late start and late finish dates, or early start and late finish dates. These dates are used when simulating the planned schedule.
3. View Option - This allows the user to select how to view the simulation. The user can choose to simulate the planned schedule, actual schedule, or both planned and actual schedule at the same time.
4. Simulation Start and Finish Dates - This allows the user to define the time period for simulation.

The user can execute visual simulation once the customized simulation parameters have

been defined. The Visual Simulator uses the 3D computer model of a construction project shown on the graphics display, and has the ability to turn its objects off and on. This permits the user to determine which objects are under construction and/or have been completed for each day during the simulation process. The result is a time scaled simulation of the construction sequence

### ***1.4 Design Methodology***

The VSS system was developed to provide a visual method of representing the CPM schedule for a construction project. The VSS system uses several computer based technologies most of which are already widely used in the construction industry. The VSS system combines these technologies to produce a visual schedule simulation of the construction process.

Primavera Project Planner was chosen as the project management system used by the VSS system. This system was chosen because it is widely used in the construction industry. Primavera is also capable of exporting scheduling data into a dBASE IV file format. The dBASE IV file format allows Phase II of the VSS systems, the Database Manager, to easily manipulate scheduling information.

CADAM, marketed by IBM, was used in the VSS system as the 3D computer modeling system. This system was chosen because of the extensive availability of CADAM at Virginia Tech. CADAM is widely used in the engineering and construction industry. CADAM is a product which is sold by IBM Corporation. It runs on IBM mainframe

## ***1. INTRODUCTION***



computers or on IBM stand alone workstations.

dBASE IV was used to create Phase II of the VSS system, the Database Manager. The Database Manager is a user interactive system developed using dBASE IV's programming language. This system allows the user to easily map scheduled construction activities to 3D computer model objects. The Database Manager runs on any IBM PC or compatible system.

Phase III of the VSS system, the Visual Simulator, was developed within an existing software called WALKTHRU. WALKTHRU is a 3D visual simulation and animation system. It is capable of interacting with several different 3D computer modeling systems. WALKTHRU runs on Silicon Graphics IRIS 4D series graphics superworkstation. The Visual Simulator was developed using the "C" programming language and is accessible from WALKTHRU's main menu system. The Visual simulator is menu driven. This allows the user to create a visual simulation of the construction process using the 3D computer model developed on CADAM and transferred to the WALKTHRU system.

### ***1.5 Scope of Work Performed***

The VSS system combines several different hardware and software systems. The following specific tasks were performed to complete this research and produce a working prototype proof of concept version of the VSS system:

## ***1. INTRODUCTION***

1. Conceptual design to determine how the VSS system would be created and used. The following systems were chosen to develop the VSS system:
  - a. Primavera Project Planner - Scheduling system.
  - b. CADAM - 3D computer modelling system.
  - c. dBASE IV - Database Manager.
  - d. WALKTHRU - Visual Simulator.
2. Development of Phase II of the VSS system, the Database Manager, using the dBASE IV system. This involved writing source code using the dBASE IV programming language. This source code provides a user interactive environment using menu structures and produces an object schedule which can be used by the Visual Simulator.
3. Development of Phase III of the VSS system, the Visual Simulator, using the WALKTHRU system. This involved writing source code using the "C" programming language. This source code provides a user interactive environment using menu structures. The Visual Simulator source code can be accessed from WALKTHRU's main menu as Bechtel has provided a "VSS" menu choice.
4. Developing documentation for the VSS system. This includes the following:
  - a. VSS system description and implementation (Chapter's 5 & 6 ).
  - b. User's Manual (Appendix A).
  - c. Flow charts (Appendix B).
  - d. Source Code (Appendix C).

## ***1.6 System Limitations***

The VSS system is a highly interactive system. It uses menu structures so that the user can easily input data and change system parameters. However, the system has limitations as do all software systems. The following is a list of the limitations of the VSS system:

1. The Visual Simulator assumes that all construction projects begin on a Monday.
2. The current version will only simulate construction schedules using a five day work week.
3. The VSS system uses several different software and hardware systems. Therefore, the information must be transferred from system to system.
4. The VSS system can only use project management software capable of exporting activity information in a .dbf format.
5. Scheduling dates must be exported from the project management systems using working days, not calendar days.
6. The VSS system's Visual Simulator is dependent on the resolution of the graphics display and the size of the 3D computer model. For example, if the user wants to visually simulate construction activities for the entire construction project, then the entire model must be shown on the graphics display. The user will be able to determine the scheduled start and completion dates of only those object which can be seen. Small objects will go unnoticed. Conversely, if the user wants to visually simulate construction activities for only a small portion of the project, then only the desired portion of the model should be shown on the graphics display. This

causes the user to loose sight of activities which may be occurring elsewhere in the 3D computer model.

## ***1.7 Presentation of Thesis***

This thesis introduces the problems with current computer based project management systems and more specifically the way CPM scheduling information is presented and used by construction personnel. The thesis describes the development of the VSS system and how the VSS system can be used in the construction industry. The discussion of this material is presented in the following manner.

Chapter 2 discusses the historical background of CPM technology. The chapter also examines how CPM technology has evolved and is used today. Finally the chapter addresses the problems with current CPM technology and offers a solution to the problem.

Chapter 3 discusses technology relevant to project planning and scheduling. The chapter starts by describing Primavera Project Planner. This is the project management system used in the development of the VSS system. The chapter also describes computer aided design and how it is used by the VSS system. Relational database systems are also described and related to the VSS system. Finally, chapter 3 describes the WALKTHRU system and its features and relates the WALKTHRU system to the VSS system.

Chapter 4 is the literature review. This chapter examines systems which have been created to provide a better method for viewing scheduled activities. These systems all use computer modeling technology. This chapter concludes with a summary describing how

### ***1. INTRODUCTION***

each system relates to and differs from the VSS system.

Chapter 5 gives a description of the VSS system's functionality. It discusses the creation of the CPM schedule and the 3D computer model for the construction project. Chapter 5 continues by describing how this information is used by Phase II of the VSS system, the Database Manager. The use of each item on the Database Manager's main menu is described. The development of the object schedule is also examined. Chapter 5 concludes by describing how Phase III of the VSS system, the Visual Simulator, uses the object schedule to produce a visual simulation of the construction sequence. Each item on the Visual Simulator's main menu is discussed. Chapter 5 describes how the user can change the parameters on the main menu to customize the visual simulation process.

Chapter 6 reviews the creation of the VSS system. Chapter 6 examines the design and implementation of the VSS system's Database Manager and Visual Simulator. The reaction of respective programs to menu item selection is described. Chapter 6 details how the programs process information entered by the user, and how procedures or functions making up the program interact with each other and the user.

Chapter 7 is the closing chapter. A summary of the information presented in chapters 1 - 6 is presented. Recommendations are offered as to how the system can be utilized. Chapter 7 ends with a discussion on how the VSS system can be enhanced to provide a more complete project management system.

Three appendices are included. Appendix A provides a user's guide. This gives the user of the VSS system a step-by-step guide on how to operate the VSS system. Appendix B

## ***1. INTRODUCTION***

contains flow charts which describe the program development of the VSS system. Finally, appendix C contains the dBASE IV language source code for the Database Manager and the "C" language source code for the Visual Simulator.

## ***2. TRADITIONAL SCHEDULING TECHNIQUES***

CPM scheduling has gone through little change since the mid 1960's. The primary change is that computer processors have been used to enhance the scheduling process by providing a method to efficiently calculate networks for large construction projects having many activities. This chapter presents the historical background of CPM technology, current uses in the construction industry, problems with today's systems, and a look at how the VSS system can alleviate some of these problems.

### ***2.1 Historical Background***

Prior to Critical Path Methods (CPM), most project managers used bar charts as a method for grouping individual activities together to graphically portray the plan, schedule, and progress of a construction project. They were, however, ineffective in showing the interdependencies and logic between each activity. Project planners and managers were in need of new techniques which could do the following [Moder et al. 1983]:

1. Simultaneously plan and schedule construction activities.
2. Show enough detail to allow for detection of scheduling errors by showing dependency relationships.
3. Make it easy to determine the effects of progress delays in individual

activities.

4. Efficiently set-up, maintain, and update schedules for large construction projects.

As a result of these needs CPM technology was developed.

[Bent & Thumann 1989] discuss two network based methodologies commonly used today. "CPM scheduling was developed in the late 1950's. It was introduced into the construction industry as a tool to improve planning and scheduling of construction projects. Concurrent with the industrial development of CPM, the U.S. Navy introduced a similar method of scheduling call PERT. PERT is an acronym for Program Evaluation and Review Technique. The Navy developed this method to evaluate and monitor progress of the Polaris Missile Program. The major difference between CPM and PERT is that PERT is a more probabilistic approach that lends itself to activities for which there is little or no historical experience, whereas CPM uses historical information for establishing durations."

CPM and PERT are network-based planning methods which graphically portray the sequence of activities necessary to complete a construction project. The output from these methods is a graph or network showing the dependency relationship between activities.

CPM and PERT networks were originally manually generated. During the 1960's construction projects became more complex. Manually calculating and generating construction networks was no longer feasible for planning and scheduling. Digital main frame computers were implemented as CPM processors to replace the traditional paper and pencil techniques of the past [Bent & Thumann 1989].

## **2. TRADITIONAL SCHEDULING TECHNIQUES**



Rapid advancement in computer technology and project management software has provided for the development of powerful scheduling tools. CPM processing is done almost entirely on microcomputers and personal computers. This development has given project managers and other construction personnel a readily available method to generate construction networks.

## ***2.2 Current Practices with CPM Technology***

Network planners in the construction industry vary significantly in their level and style of network use. The following sections describe how CPM networks are currently used in the construction industry as well as current problems and deficiencies.

### ***2.2.1 Network Use in Construction***

Large construction management and engineering firms often have elaborate planning and scheduling departments with trained personnel and computer systems. These large firms use CPM techniques out of necessity due to the large and complex jobs they attract. Many firms, on the other hand, often employ scheduling consultants. These consultants are sometimes hired because the firm wants to efficiently plan and schedule projects; but more often because of contractual requirements which state that network methods must be used to report the plans and progress of construction [Moder et al. 1983].

Even though powerful personal computers are available which run state-of-the-art CPM

## ***2. TRADITIONAL SCHEDULING TECHNIQUES***

software to generate networks, it is still common place for planners and schedulers to use manual methods to generate CPM networks. Networks are sometimes still manually prepared on tracing paper using adhesive mylar labels depicting activities [Moder et al. 1983].

Manual techniques, rather than computer processing, are still used in order to keep the schedule simple. If a project is to be efficiently maintained, clear lines of communication between project management and field personnel must be established. Lengthy textual reports and cluttered complex networks generated by today's project management systems tend to cloud communication lines. Field personnel are typically burdened by details that may confuse the most important facts of the schedule.

### ***2.2.2 Planning and Scheduling in the Construction Industry***

The difference between project management in the construction industry compared to other industries is the element of uniqueness. Activities are usually carried out only once during a construction project versus repetitive assembly line procedures present in many other industries. To complicate matters further, the same activity is usually not performed in the same manner from project to project due to varying site conditions, weather conditions and other factors. If the planner of such projects is not experienced in dealing with these ever changing conditions then costly errors and delays will occur making project control a rigorous and difficult task [Jackson 1986].

The better a construction project is planned and scheduled the less of a problem there will

## **2. TRADITIONAL SCHEDULING TECHNIQUES**

be in controlling the project. Inexperienced planners must rely on information from outside sources in order to reduce planning and scheduling errors which can only be discovered once construction has begun. This is time consuming and costly. If the constructor is on a tight budget, as most are, then they must prepare the schedule and wait until construction begins to determine if the schedule is workable. This can result in costly errors which are sometimes impossible to overcome.

### ***2.3 Problems With CPM Techniques and a Proposed Solution***

Current use of CPM techniques has some limitations in use, especially during the control phase of a construction project. A major problem with today's project management systems is the way output is generated. The following section outlines the problems with output generated from project management systems. Section 2.3.2 introduces a proposed solution to this problem.

#### ***2.3.1 Problems and Deficiencies with CPM Output for Controlling Construction Projects***

In addition to graphical output, computer-based CPM scheduling systems generate lengthy textual and tabular reports. Many of these systems provide information that is not required to monitor construction progress. Field personnel and even project management often do not have the time or patience to sift through the large amount of information generated in order to extract the information they need.

## ***2. TRADITIONAL SCHEDULING TECHNIQUES***

Many construction companies hang large network diagrams in their site trailers. These diagrams often cover entire walls showing hundreds and even thousands of activities. As construction begins this network serves as little more than wall paper. Construction personnel are forced to try and visualize how activities relate to one another and everyone who tries to interpret this network has a different perception as to whether it is correct or incorrect.

Updating the schedule is a necessary function in order to control construction progress. Field personnel have trouble comparing original or updated networks with actual construction progress. This makes it difficult to determine the state of the construction project.

Many companies have taken a step backward by using manual methods for generating CPM schedules in order to keep them simple [Moder et al. 1983]. Technology is now available allowing the construction industry to take a step forward in order to overcome the deficiencies in the representation of CPM network information.

### ***2.3.2 A Proposed Solution***

Computer technology is at an advanced level allowing development of systems to enhance the output of typical project management systems. This thesis presents a system, the Visual Schedule Simulation (VSS) system, which takes traditional CPM output technology one step further. The VSS system uses 3D computer models to simulate the construction process thus eliminating the shortcomings present in current scheduling systems.

## ***2. TRADITIONAL SCHEDULING TECHNIQUES***

The VSS system will enable construction personnel to better monitor construction activities by providing visual comparison of actual construction progress with planned progress.

Field personnel no longer have to try to visualize how construction activities relate to one another. They can view, at a computer graphics display, the sequence of the construction process. Planners will benefit from this system by using visual simulation to test different construction scenarios, thus eliminating costly errors once construction has begun.

Current technology provides for the ability to develop the VSS system. This technology has become available within the past few years with the advent of improved software, microcomputers, and superworkstations. A look at the technology needed to make VSS possible is presented in the following chapter.

### ***3.0 TECHNOLOGY REVIEW***

Computers have been used in the engineering and construction industries for many years. Most of the early applications dealt with engineering analysis (such as structural analysis). Real-time interaction with the computer was limited or non-existent and graphics had limited use, except to show the results of the analysis [Cleveland & Francisco 1988]. Computer technology has advanced rapidly, especially in the past decade. This has enabled use of more advanced systems, which overcome the deficiencies of past systems, for real-time interaction and graphics.

This chapter describes the following software systems:

1. Primavera Project Planner
2. Computer Aided Design
3. Relation Database Management Systems
4. WALKTHRU

This represents the current state of software systems used in the development of the VSS system.

### ***3.1 Primavera Project Planner***

Primavera Project Planner is a computer based CPM scheduling processor. The following section 3.1.1 gives a brief description of the system. Section 3.1.2 gives a description of how it is used with the VSS system.

#### ***3.1.1 System Description***

Phase I of the VSS system, the Data Preprocessor, uses Primavera Project Planner to allow the user to generate a construction schedule. Primavera Project Planner is a comprehensive project management and control software system that allows managers to control large and small construction projects. Primavera's capabilities include critical path scheduling, resource allocation and leveling, and cost control. Primavera is interactive and command driven. It can run on most IBM PC's. Primavera can handle up to 10,000 activities for a single project [Primavera Project Planner 1987].

#### ***3.1.2 Primavera Project Planner and the VSS System***

Primavera was chosen as the project management system for the VSS system because of its popularity in the construction industry. By using Primavera's export capability, information concerning the project, such as: activity id's, planned scheduling dates, and actual progress dates, can be extracted from the CPM schedule of construction activities. This information is transported to Phase II of the VSS system, the Database Manager. The Database manager associates this information with information related to the 3D computer

model of the construction project. This information is ultimately used to form a visual simulation of the construction project.

### ***3.2 Computer Aided Design***

[Groover & Zimmers 1984] define Computer Aided Design (CAD) in the following manner: "CAD is the use of computer systems to assist in the creation, modification, analysis, or optimization of a design." Typical CAD hardware includes a computer to process information, keyboards, digitized tablets, a mouse, or function boxes for input, display terminals, plotters, and printers for output, and other peripheral equipment. CAD software consists of computer programs which make use of computer graphics and application programs in order to accomplish a task .

#### ***3.2.1 Interactive Computer Graphics***

Today's CAD systems are based on interactive computer graphics. Groover and Zimmers define interactive computer graphics as "a user-oriented system which the computer is employed to create, transform, and display data in the form of pictures or symbols." The user communicates with the CAD system using input devices to enter data and commands. The computer communicates with the user via a graphics display monitor.

Interactive computer graphics increases the utility of the CAD system and enhances the design process. The user performs tasks suitable for human thinking while the computer performs tasks best suited to its capabilities. These tasks include calculations, visual

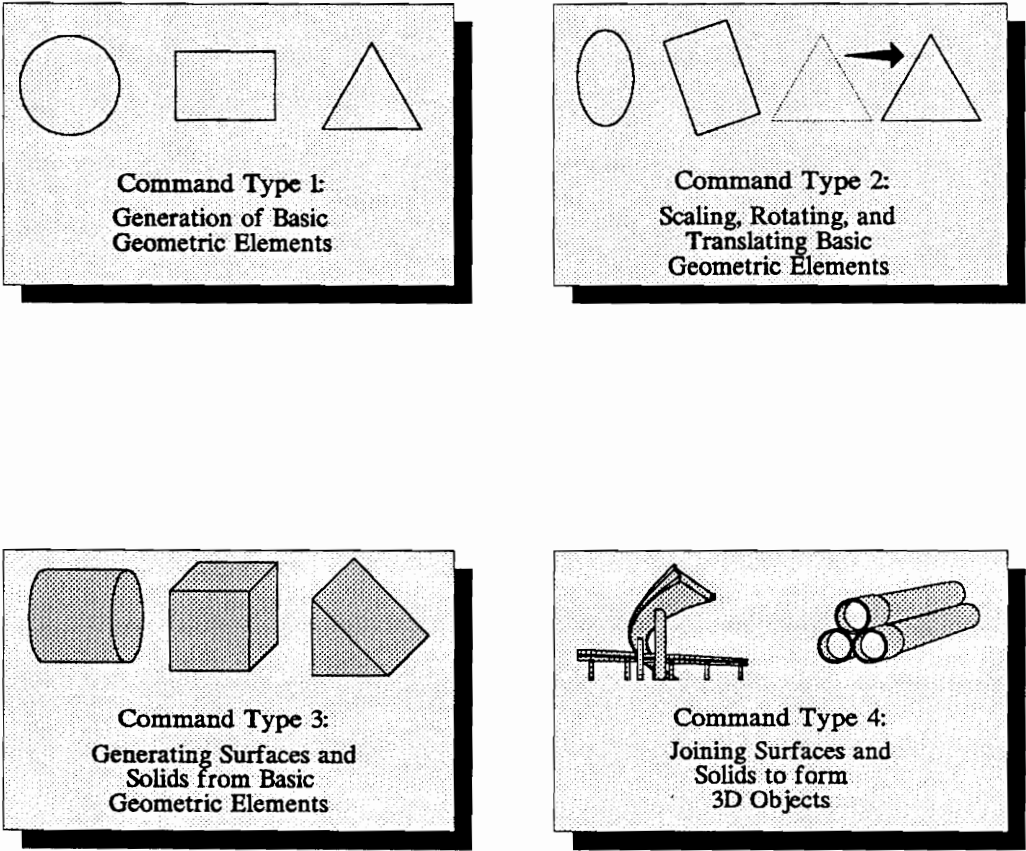


display, and storing and manipulating large amounts of data [Groover & Zimmers 1984].

### ***3.2.2 Geometric Modeling***

In CAD, geometric modeling is concerned with the mathematical description of the geometry of computer-based objects. This mathematical description allows the image of the object to be displayed and manipulated on a graphics display [Groover & Zimmers 1984].

"To use geometric modeling, the designer constructs the graphical image of an object on the CRT screen" of the CAD system. Using interactive graphics within the CAD system, objects can be created by inputting four types of commands to the system. The first type of command lets the user generate basic geometric elements. These geometric elements include points, lines, and circles. The second command type allows the user to scale, rotate, or translate these geometric elements. The third command type generates surfaces and solids from geometric elements. The fourth type of command allows these elements to be joined into the desired shapes to form objects. During the geometric modeling process, the computer converts the commands into mathematical models, stores this information in the computer data files, and displays this information as an image on the graphics display monitor [Groover & Zimmers 1984]. This process is illustrated in Figure 3.1.



**Figure 3.1: Geometric Modeling Process**

Objects can be represented by using several different techniques common to geometric modeling systems. These are as follows:

1. Wire frames - Objects can be represented by interconnecting lines. Wire frame models can be:

- o 2D representations - Used for flat objects
  - o 3D representations - Used for more complex geometry
- [Groover & Zimmers 1984].

2. Surface models - The outer surfaces of objects are represented mathematically as solid faces. The interior of these objects remain hollow.

3. Solid models - This is the most advanced method of geometric modeling. Models are represented mathematically and visually as solids. This enables users to define objects giving them physical characteristics of real materials from which they may be constructed [Groover & Zimmers 1984].

### ***3.2.3 Graphical Capabilities of CAD Systems***

Surface and solid modeling allow the user to create color-shaded images of objects with near-photographic qualities. Graphic systems allow the user to define light sources and other characteristics which effect the way the objects are displayed [Hordeski 1986].

Parameters can be defined which show surface textures, color gradations, shadows, reflections, and slight irregularities. This adds clarity to computer generated images helping these systems achieve visual realism. Surface and solid models can show objects in realistic detail. This allows the user to visualize and improve products before they are built [Hordeski 1986].

### ***3.2.4 CAD and the VSS System***

The VSS system requires a 3D computer model of a construction project to be generated using a CAD system. The user utilizes the interactive computer graphics capabilities of a CAD system to surface model all objects related to the design. These objects can include foundations, walls, floors, electrical systems, mechanical systems and virtually anything else which can be modeled. The developer of the 3D computer model should use the CPM network of activities, if it is available, as a guide for selecting which objects should be separately distinguishable. The ideal situation is to have one model object for each activity in the CPM network. This model, when completed, is ready to be transferred to the VSS system's simulator (see Section 3.4).

## ***3.3 Relational Database Management Systems***

Relational Database Management Systems (RDBMS) are powerful tools for managing data. RDBMS's are electronic data processors capable of storing, relating, manipulating, and retrieving large amounts of data quickly and efficiently. RDBMS's are used to connect these data to application programs [Robbins 1989].

## ***3. TECHNOLOGY REVIEW***

### ***3.3.1 Relational Database Filing Systems***

Databases are used to organize information for ease of reference. An efficient database will organize its information by filing it in related categories. This is similar to a filing cabinet in which files are stored alphabetically. Each file might hold a paper form containing more details. The same form may be used for every file, although the information on each form is different. This information is organized in repeated groups of data.

A database file may contain many lines of detailed information. Each line is made up of smaller items of information called fields. These fields define the structure of a database file [Learning dBASE IV 1988]. Figure 3.2 shows a typical database file.

### ***3.3.2 dBASE IV and the VSS System***

dBASE IV was chosen as the VSS system's RDBMS because of its high power and ease of use. dBASE IV automates the storage of activity names, activity start dates and activity finish dates imported from the VSS system's project management system.

dBASE IV contains a complete programming language. This language was used to develop a complete user interface which acts as both a translator and data processor.

dBASE IV programming functions make menu generation and data retrieval and manipulation possible [Robbins 1989]. The dBASE IV program reads data exported from the VSS system's CPM processor, manipulates this data according to user defined parameters, and produces a file that the visual simulator can understand.

Typical Database Field

File Structure →

Activity Id	Scheduled Start Date	Scheduled Finish Date	Actual Start Date	Actual Finish Date
A101	2/1/90	2/4/90	2/1/90	2/3/90
A102	2/3/90	2/6/90	2/3/90	2/5/90
A105	2/5/90	2/8/90	2/5/90	2/8/90
A202	2/7/90	2/10/90	2/7/90	2/12/90
A302	2/9/90	2/12/90	2/9/90	2/14/90
A303	2/11/90	2/14/90	2/13/90	2/16/90
A304	2/13/90	2/16/90	2/15/90	2/16/90
A305	2/15/90	2/18/90	2/17/90	2/19/90
A307	2/16/90	2/20/90	2/17/90	2/19/90
A309	2/17/90	2/22/90	2/18/90	2/21/90
A401	2/19/90	2/24/90	2/19/90	2/22/90
B101	2/25/90	2/26/90	2/25/90	2/27/90
B104	3/1/90	3/5/90	3/3/90	3/5/90
B105	3/2/90	3/7/90	3/4/90	3/7/90
B202	3/6/90	3/9/90	3/7/90	3/9/90
B203	3/9/90	3/12/90	3/9/90	3/11/90
B303	3/11/90	3/14/90	3/11/90	3/13/90
B304	3/13/90	3/16/90	3/16/90	3/19/90
C101	3/15/90	3/18/90	3/18/90	3/22/90
C102	3/17/90	3/20/90	3/19/90	3/24/90
C201	3/21/90	3/23/90	3/24/90	3/26/90
C204	3/25/90	3/27/90	3/25/90	3/30/90
C205	4/1/90	4/4/90	4/4/90	4/6/90
C301	4/3/90	4/6/90	4/7/90	4/10/90

Typical Database Record →

Figure 3.2: Example of a Typical Database File

### **3.4 WALKTHRU**

In order to simulate and present the 3D computer model data, a simulation system is required. WALKTHRU is used within this thesis as the simulation and visualization system, for Phase III, the Visual Simulator.

#### **3.4.1 System Description**

WALKTHRU is a real-time 3D animation and visualization system developed by Bechtel Corporation. It was developed to electronically replace plastic models as a method for visualizing real world objects. WALKTHRU allows users to interact with existing 3D computer models in a lifelike manner. With WALKTHRU and a Silicon Graphics IRIS 4D Workstation, the user interacts with the system through a control panel to control body and head motion. This allows the user to move through the computer model, seeing its physical objects much as they would appear in the real world [Cleveland & Francisco 1988].

The WALKTHRU system runs on a Silicon Graphics IRIS 4D series workstation. The Silicon Graphics IRIS 4D workstation uses state-of-the-art graphics architecture for 3D computing applications. By using RISC technology and VLSI graphics processors, this workstation provides high performance computing and graphics allowing the user to manipulate complex objects in real-time, under direct input-device control.

An IRIS 4D Graphics Subsystem may include:

- o 64 image bit-planes (including 24 color bit-planes and 8 alpha bit-planes, double buffered)
- o 24-bit Z-buffer for hidden surface removal
- o 4 bit-planes for overlay or underlay
- o access to 16.7 million colors in double-buffered RGB mode
- o 1280x1024 screen resolution

Hardware support is included for:

- o Alpha-blending capability for transparency effects
- o Advanced lighting models
  - Multiple colored light sources (up to 8)
  - Ambient, diffuse, and specular lighting models
  - Phong lighting
  - Local viewer and infinite light source positioning
- o Pan and zoom capabilities
- o Multimode windowing environment
- o Flat and Gouraud shading
- o Anti-aliased lines

### ***3.4.2 System Features***

WALKTHRU uses a dial/button box, and a mouse as its two primary input devices. The dial/button box allows the user to control motion through the model. The mouse allows the user to access menus to control all of the functions available on the dial/button box.

## **3. TECHNOLOGY REVIEW**



WALKTHRU will perform the following basic functions [WALKTHRU User's Manual 1988]:

1. **View Control** - By turning dials or using pop-up menus controlled by the mouse, the user can control all aspects of viewing the model. The user has the freedom to control the speed of travel through the model, viewing angle of the model, and viewing position.
2. **Display Control** - At any point, the user can switch between wire frame and shaded images by simply pushing a single button. The control panel shows x,y, and z positions so the user always knows relative positions within the model.
3. **Object Motion** - WALKTHRU allows the user to move and position objects within the model. The user can define object hierarchy so that motion of one or more objects is dependent on the motion of other objects.
4. **Interference Detection** - While moving objects within WALKTHRU, the system is able to check and report any interferences with other objects in the model.
5. **Measuring** - The user can measure between any two point by using the mouse. The system reports values for distance, angle, and delta x,y, or z.

### **3. TECHNOLOGY REVIEW**

6. "Record/Playback - Walkthru can record a sequence of key frames that contain the current view parameters, object positions, orientation, and the change in time in between frames. The frames are stored in a "Record" file that can be read by WALKTHRU's REPLAY function. REPLAY interpolates between the key frame parameters, and replays the sequence in real time."

7. "Shaded Image Animation - WALKTHRU can replay the viewer and object motion one frame at a time, shading each frame and sending the shaded frame to a video recorder. The final product is a videotape of the recorded motion, displayed entirely in shaded image."

### ***3.4.3 WALKTHRU and the VSS System***

The VSS system uses WALKTHRU as its simulation/animation system. Simulation is the process of designing a model (mathematical, logical, or graphical) of a real system and experimenting with this model on a computer. Simulation uses animation to produce visual movement through the use of a computer system. The visual simulation module of the VSS system was built on top of the WALKTHRU application program making extensive use of WALKTHRU's features. It uses the 3D computer model generated on a CAD system and the information about the schedule of activities extracted from its database management system to produce a visual simulation of a construction project.

## ***4. REVIEW OF SIMILAR SYSTEMS***

Computers have been used in the construction industry for well over 20 years. As construction projects have become more advanced and complicated it has become necessary for the construction industry to use computers as Critical Path Method (CPM) processors to generate construction project schedules.

CPM processors have gone through little change over the last 20 years. Just about every software package uses network or graph-based methodology to depict scheduled activity progress, resource utilization, and work breakdown structures [Cleveland & Francisco 1988]. The output from these software packages are graph-based arrow and precedence diagrams, and tabular and textual reports. These methods are certainly not incorrect, however, they are at times difficult to analyze and interpret. This makes it difficult for the planner or scheduler to visualize the construction process.

Recent advances in computer hardware and software allow some of these problems to be addressed in entirely new ways. Computer Aided Design (CAD) is an application which is highly interactive and makes extensive use of computer graphics. New graphics superworkstations provide visualization and simulation capabilities unimagined just a few years ago. This breakthrough in technology has generated a new wave of thinking and development in the construction industry. Large construction companies are now

developing systems which apply CAD technology to plan, schedule and control construction projects.

This chapter presents a review of systems developed, using CAD technology, for the purpose of either planning, scheduling or controlling the construction process. Four different systems are summarized below.

#### ***4.1 Simulation System for Construction Planning and Scheduling***

This system was developed by Bechtel Western Power Co. and Bechtel Construction Inc. It uses intelligent 3D CAD plant models to aid in efficiently planning and scheduling construction and/or maintenance activities. This system combines state-of-the-art graphics capabilities to visualize and simulate construction activities, knowledge-based decision support features, and a CPM schedule processor to automatically generate a construction schedule. The system consists of four separate programs to accomplish its goal.

Program 1 is the preprocessor. The preprocessor translates 3D CAD model data into a format that can be loaded into and understood by the central project management database. The preprocessor automatically analyzes and generates tag identifiers which are used to uniquely identify separately installable objects.

Program 2 is the handling equipment librarian. This program accepts 3D CAD models of handling equipment such as cranes, monorails, fork lifts, chain hoists, etc., and converts

this data into a simplified format that can be loaded into the handling equipment database. The equipment defined is then available for use within the construction simulator.

Program 3 is the construction simulator. This software allows the user to "pre-build" the plant in the sequence it should actually be constructed. A set of interactive tools for simulation and planning are provided using 3D computer models of the modified plant. The user interacts with this environment by using a mouse to point at and pick objects from screen displays. The construction simulator utilizes automatic rule-based sequencing using a built in expert system to aid the user in decision making. Manual sequencing is also allowed giving the user the opportunity to draw on personal experiences. The final sequence of activities is written into the systems project management database.

Program 4 is the CPM schedule processor. The construction simulator allows planners to develop, define, and playback a sequence of construction activities. The results of this simulation are written to the project management database. The CPM processor uses this data along with other data from the database to generate and process a CPM schedule for the construction project [Simons et al. 1988].

## ***4.2 Three-Dimensional Solid Modeling and Database***

### ***Integration for Construction Project Management***

This system was developed by Stone & Webster Engineering Corporation. It uses advanced IBM hardware and software capabilities to form an integrated database which combines interactive graphics, engineering data, and all other information necessary for the

engineering, design, construction, operation, and management of a construction project.

The database represents a computer model of all stages of the construction project. Each object in the computer model is uniquely defined so that it can be accessed and manipulated independently. All parties involved in the construction project's development can deposit and extract information to and from this database, thus functioning as an interface between all participants in the project.

The construction planner can use this information to aid in planning and scheduling construction activities. Each step in the construction sequence model represents an activity in the construction network. The construction planner calculates the duration of each activity interactively using the graphics model and the database. The result is the sequence and duration of every path in the network. Parallel paths and dependencies between paths can be defined to form an entire network [Zabilski 1988].

### ***4.3 KNOW-PLAN***

KNOW-PLAN is a knowledge based planning system developed at Virginia Tech. This system integrates Artificial Intelligence (AI) technology with Computer Aided Design technology to generate and simulate the construction schedule of activities [Morad 1990 & Morad & Beliveau 1991].

The KNOW-PLAN system generates a construction schedule by making use of captured knowledge, extracted geometric data, and other information entered by the user of the system. The overall system is divided into six stages:

#### ***4. REVIEW OF SIMILAR SYSTEMS***

1. **Generation of a 3D computer model** - A 3D computer model of the designed facility is generated using a 3D CAD system.
2. **Database and Knowledge-base creation and manipulation** - Geometric data from the 3D computer model stored in the central database is appended. The user interactively inputs attributes related to the various objects of the model. The attributes include class listings, zoning data, connection type, resource requirements, procurement dates and time constraints.
3. **Dynamic sequencing process** - The dynamic sequencer uses knowledge-based concepts and other AI concepts to form a logic network of activities based on geometric data. The final product of this process calculates the schedule dates for different activities.
4. **Interactive sequence modification** - This stage lets the user interactively modify and customize the generated schedule of activities.
5. **Conventional scheduling and reporting** - The system will generate schedule reports and interact with a CPM processor to generate bar charts, logic networks, and time-scaled networks.
6. **Visual simulation** - The system visually simulates the construction process based on the generated sequence of activities. The user can view the step-by-step sequence of construction using the 3D computer model of the designed facility.

#### ***4.4 CAD Based Schedule Representation***

This system was developed by Beiswenger Hoch and Associates. This system does not have a formal name but can best be described as a CAD Based Schedule Representation. This system provides a tool to monitor and document the progress of construction using CAD technology. It uses CAD technology to create and update its construction plans using 2-D capabilities of a graphics workstation.

This system can provide, at any time during the construction project, a paper plot of the actual plans drawn in colors with each color signifying the status of construction. Color paper plots are created using the actual plans to show construction status for a specified time period. This system will allow for a static 2-D display, on a computer screen or on a hard copy, to show construction progress [Norona 1988].

#### ***4.5 Summary***

The systems reviewed in the previous section all take a new and different approach to planning and scheduling compared to traditional CPM techniques. The following is a brief synopsis of each system and how they relate to the VSS system.

The systems developed by Bechtel and Stone & Webster use the objects of a 3D computer model of a construction project as "building blocks" for generating a construction schedule. This allows planners and schedulers to understand and create CPM schedules easily and

#### ***4. REVIEW OF SIMILAR SYSTEMS***



efficiently versus traditional techniques. These systems, however, offer little support for controlling activities once construction has begun. Conversely, the VSS system concentrates on controlling construction activities once the CPM schedule has been developed. The VSS system is intended to help project managers, superintendents, and other construction personnel control construction activities by allowing them to visually compare planned and actual schedules on a graphics display.

KNOW-PLAN offers a new and innovative approach compared to traditional CPM processors. This system generates a schedule of construction activities based on knowledge extracted from 3D computer models as well as other knowledge sources. The last stage of the KNOW-PLAN system (stage 6) visually simulates construction activities, from start to finish, based on the dynamic sequence generated in stages 1-4. The VSS system is an extension to the visual simulation produced by KNOW-PLAN. VSS differs, however, by allowing the user more freedom to interact and customize the simulation. The VSS system lets the user define parameters such as the time period, viewing angle, network logic, speed of simulation, and the opportunity to simulate both planned and actual construction progress. These options, not available in the KNOW-PLAN system, give the user much more power for controlling construction activities.

The system developed by Beiswenger Hoch and Associates utilizes 2D CAD technology to show actual progress of construction activities. While this is similar in concept to the VSS system it does not utilize 3D graphics capabilities available on today's CAD workstations. It does little more than color code 2D construction plans showing the percent completion of construction. The VSS system utilizes 3D simulation capabilities available in today's advanced computer hardware and software. This allows the user to visualize exactly what

#### **4. REVIEW OF SIMILAR SYSTEMS**

is taking place at the construction site.

A comparison between the systems presented in this chapter are graphically represented in Figure 4.1. This Figure represents the flow of data in the various systems.

The VSS system provides many functions which allow for user interactivity. The functionality of the VSS system is described in the following chapter.

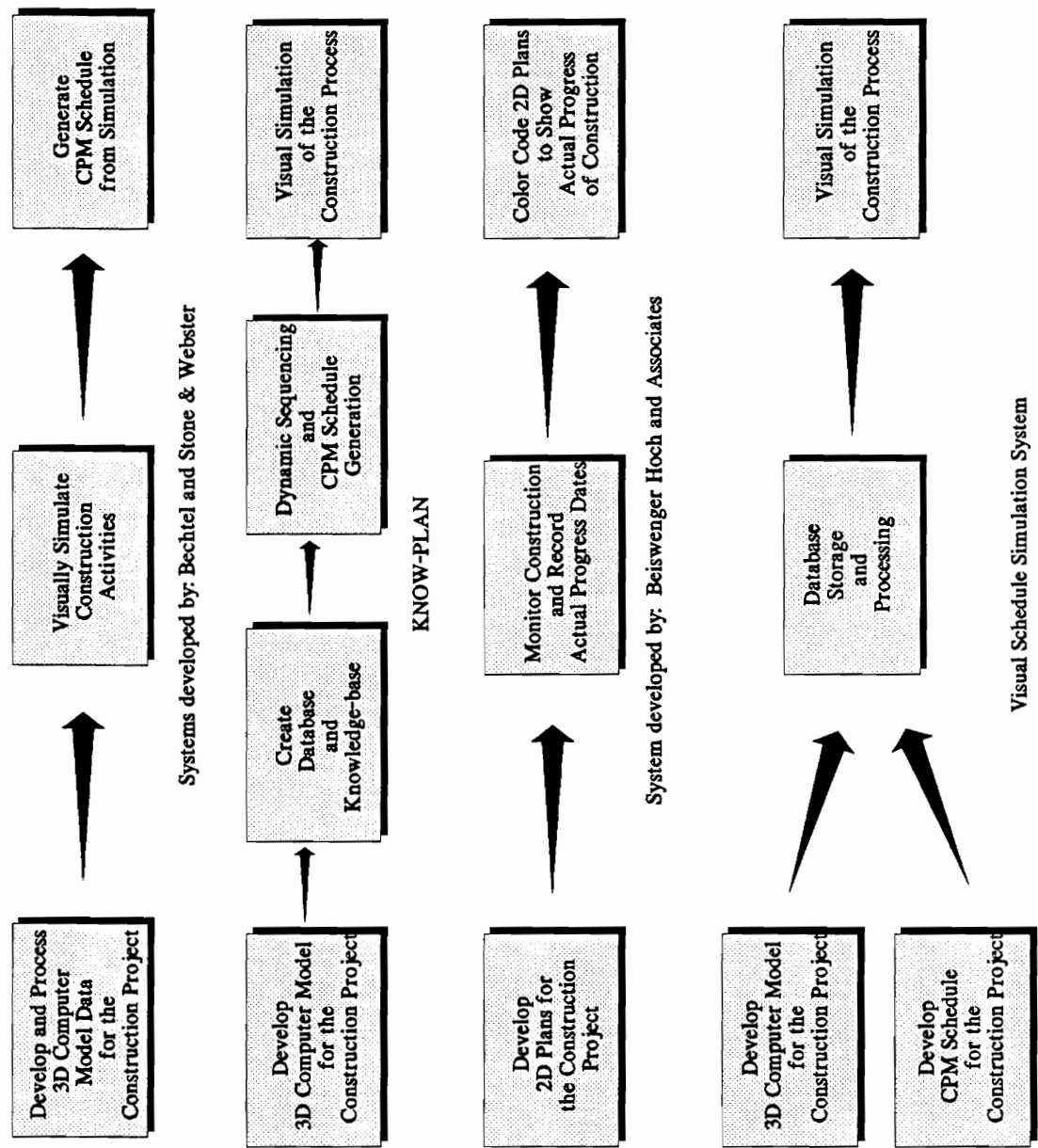


Figure 4.1: Comparison of Relevant Systems

## ***5. VISUAL SCHEDULE SIMULATION SYSTEM***

This chapter discusses the various phases of the VSS system and how these phases work. The operation and interaction of each component within the VSS system is discussed. This chapter is intended to present the VSS system using minimal technical detail. Chapter 1 provided a brief description of the VSS system. This chapter explains the VSS system in greater detail. Chapter 6 describes the technical aspects of how the VSS system was created and emphasizes the programming methodology and development.

### ***5.1 System Overview***

The Visual Schedule Simulation System (VSS) has been developed to enhance and overcome deficiencies in the output of traditional planning and scheduling techniques. The VSS system combines a CPM schedule and a 3D computer model of a construction project to produce a visual simulation of the construction sequence. The simulation takes place on a graphics display.

The VSS system uses a CPM schedule and a 3D computer model of the construction project. The CPM schedule and 3D computer model are generated in Phase I, the Data Preprocessor.

The VSS system has an internal data management system called the Database Manager. This is Phase II of the VSS system. The Database Manager obtains activity data from a project management system, such as Primavera. This information is generated in Phase I, the Data Preprocessor. This information is stored in a schedule file within the Database Manager. The Database Manager also allows the user to enter all relevant information concerning the 3D computer model and stores this data in a model file. This information relates to the 3D computer model generated in Phase I, the Data Preprocessor.

The primary function of the Database Manager is to link the information contained in the schedule file and model file. The Database Manager allows the user to input the relationships between 3D model objects and scheduled activities and converts this information into an object schedule. This information is exported to the VSS system's Visual Simulator.

The Visual Simulator allows the user to customize simulation of the construction sequence. The user can visually simulate this process using: early, late, or early/late date scheduling logic, and planned and/or actual construction progress dates. The Visual Simulator produces an animated simulation using these parameters for any user specified time period during the construction sequence.

The VSS system consists of three distinct phases which can interact with each other via computer networking. Phase I is the Data Preprocessor. Phase II is the Database Manager. Phase III is the Visual Simulator. The flow of data within the VSS system is shown in Figure 5.1. A detailed description of these three phases is presented in the following sections.

## **5. VISUAL SCHEDULE SIMULATION SYSTEM**

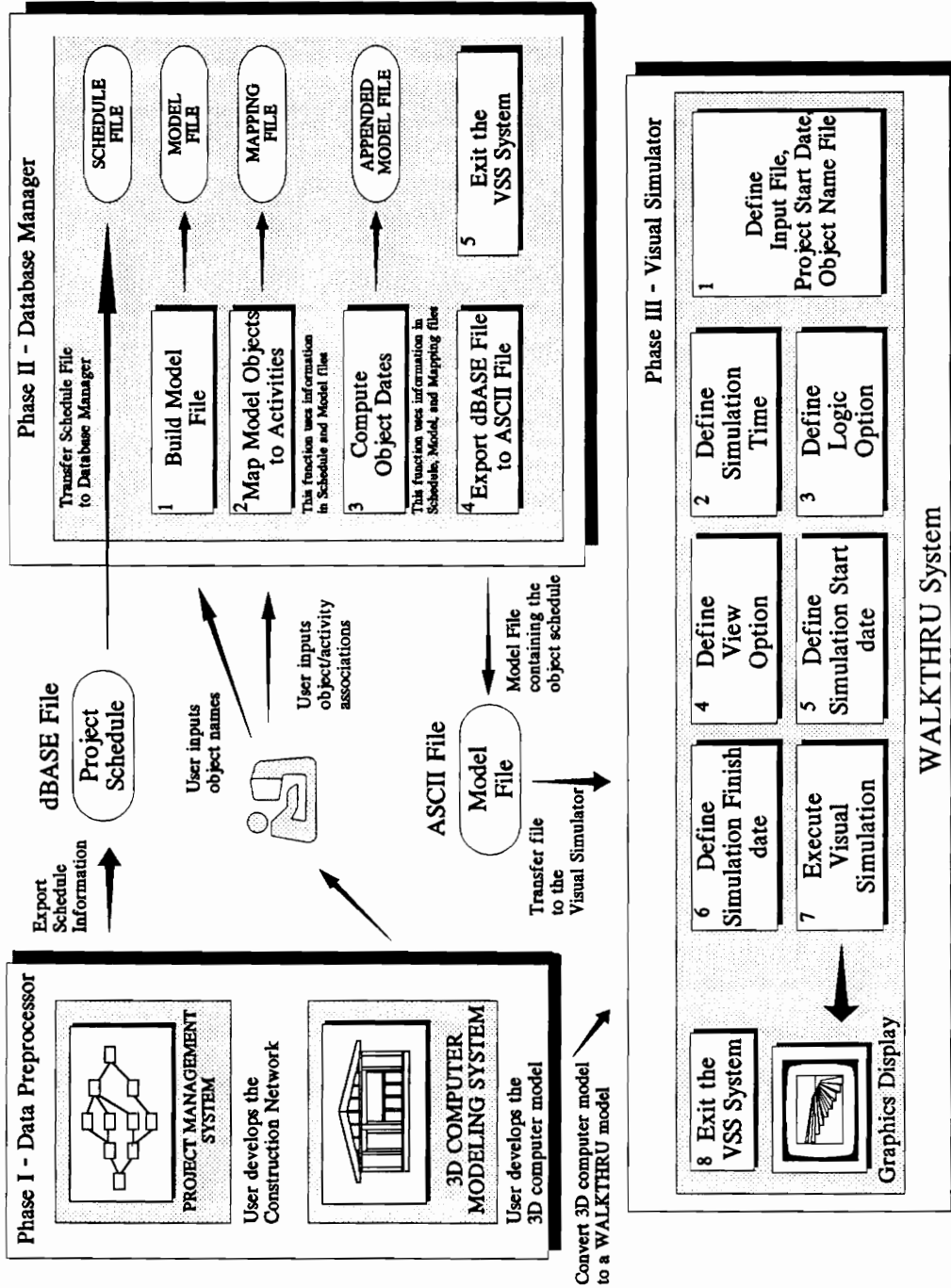


Figure 5.1: Flow of Data in the VSS System

## ***5.2. Data Preprocessor***

Phase I of the VSS system is the Data Preprocessor. The VSS system relies on two outside sources of information before it can produce a visual simulation. The Data Preprocessor involves the development of 1) the construction network, and 2) a 3D computer model. Each of these two are further described in the following two subsections.

### ***5.2.1 The Construction Network***

It is common in the construction industry for planners to generate a schedule of construction activities using a computer based project management system. Several project management systems such as Primavera Project Planner, MacProject, Open-Plan, and Microsoft Project are frequently used by planners. These systems are capable of generating a CPM schedule for a construction project using a personal computer. The VSS system can interact with any project management system as long as they are able to export the correct scheduling data in the proper file format.

The VSS system was developed using Primavera Project Planner as its project management system. Primavera was chosen because of its popularity in the construction industry as well as its ability to generate a .dbf file format. The Database Manager was designed to read scheduling information presented in this format (this will be discussed in more detail in Section 5.3).

## ***5. VISUAL SCHEDULE SIMULATION SYSTEM***

The Database Manager relies on seven items of information associated with each activity in the construction project. The .dbf file exported from the project management system must contain the following information in the following order for each activity record:

1. Activity ID
2. Early Start Date
3. Early Finish Date
4. Late Start Date
5. Late Finish Date
6. Actual Start Date
7. Actual Finish Date

Even though the project may not have started yet, the Database Manager still requires that zero's are entered for actual start and finish dates. An example of a typical CPM schedule file is shown in Figure 5.2. The schedule file exported from Primavera in a .dbf file format has a distinct file structure. It uses the letter "W" in its date fields to denote that these dates are in working days. For example, ESW stands for the early start date in working days. Although Primavera was used to develop the VSS system, any project management system capable of generating a .dbf file with the appropriate information may be used.

The planner, when preparing the CPM schedule of construction activities, should keep in mind the level of detail needed for visual simulation. They can schedule activities as detailed or as general as they wish. For example, if the planner wants to visually simulate a room being built brick-by-brick then each brick placement must be scheduled independently. If the planner wants to visually simulate the room being built



Structure for  
Scheduling File

ACT	ESW	EFW	LSW	LFW	ASW	AFW
ACT01	01	38	01	38	01	37
ACT02	39	41	39	41	37	40
ACT03	41	43	55	57	54	57
ACT04	53	62	60	69	60	68
ACT05	43	45	67	69	00	00
ACT06	42	45	56	59	57	60
ACT07	40	43	40	43	40	43
ACT08	42	52	49	59	48	59
ACT09	63	66	76	79	00	00
ACT10	63	72	70	79	00	00
ACT12	53	56	72	75	00	00
ACT16	64	66	93	95	00	00
ACT17	80	83	92	95	00	00
ACT18	80	82	93	95	00	00
ACT19	80	91	80	91	00	00
ACT20	84	86	96	98	00	00
ACT21	92	95	92	95	00	00
ACT22	96	98	96	98	00	00
ACT23	99	99	99	99	00	00
ACT11A	44	47	44	47	45	48
ACT11B	48	51	52	55	51	56
ACT13A	52	55	60	63	59	62
ACT13B	56	59	68	71	00	00
ACT14A	48	55	48	55	48	56
ACT14B	56	63	56	63	00	00
ACT15A	64	71	64	71	00	00
ACT15B	72	79	72	79	00	00

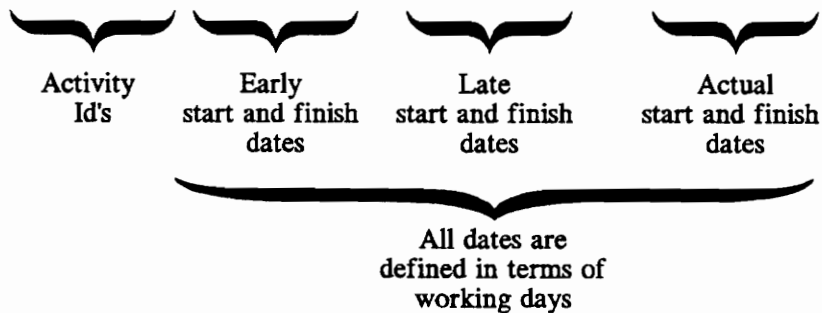


Figure 5.2: Typical scheduling file exported from Primavera

wall-by-wall then each wall placement should be scheduled as a separate activity.

The schedule should also reflect the precision for which the construction management team wishes to control the project. The construction schedule should be perpetually updated throughout the life of the construction project to reflect updates in the planned schedule as well as actual start and finish dates for each activity. Once this information has been transferred to the Database Manager in the form of a scheduling file it is ready to be used for activity/object mapping and object date processing.

### ***5.2.2 The 3D Computer Modeling System***

The 3D computer model needs to capture a certain level of detail when it is created. It must be able to reflect the CPM schedule related to the construction project. This is necessary to produce a realistic visual simulation of the construction process. To accomplish this the 3D computer model must be subdivided into individual objects. An object is a part of the model that can be referred to independently. It can be turned on and off, translated, and have its color changed while not affecting the other model objects.

The Visual Simulator, described in Section 5.4, uses the WALKTHRU system.

WALKTHRU interacts with existing 3D computer models created on a 3D CAD system.

Before WALKTHRU can interact with a 3D computer model it must first be converted into a WALKTHRU model file. WALKTHRU currently supports several file formats including: CADAM and CATIA model files, Intergraph and Bechtel 3DM Intergraph design files, IGES files, and WALKTHRU ASCII files [WALKTHRU User's Guide 1988].

Objects must be created within 3D computer models before they are translated into WALKTHRU files. This can be accomplished by grouping all geometric elements which comprise a specific object. CAD systems allow the user to separate objects by assigning all elements comprising them to either a layer, set, or overlay. When the 3D computer model is transferred to the WALKTHRU system, the WALKTHRU system groups all elements contained in each separate layer, set, or overlay to define objects.

The appearance of the 3D computer model once it has been transferred to the WALKTHRU system is also of prime importance. In order for objects to appear as solids, rather than wire frames, within the WALKTHRU environment, they must be surface modeled (WALKTHRU also supports CATIA solids). This allows WALKTHRU to shade the 3D computer model giving it a solid and realistic appearance. Objects can be modeled in any color except pink. Pink is reserved by the Visual Simulator to show objects for which construction activities are currently under construction (see Section 5.4.2).

The schedule developer must keep in mind the CPM schedule of construction activities when creating the 3D computer model used for visual simulation. The ideal 3D computer model defines one object for each activity in the construction schedule. This enables the VSS system's Visual Simulator to truly simulate each step in the construction process. The 3D computer model, however, often will contain either more detail or less detail than the CPM schedule. The Database Manager handles this situation using its activity/object mapping facility (see Section 5.3.2. for more details). Using the previous example, if the user wishes to produce visual simulation of a room being built brick-by-brick, then each brick must be modeled as a separate object. If the user wishes to produce visual simulation of the room being built wall-by-wall, then each wall should be modeled as single objects.

##### **5. VISUAL SCHEDULE SIMULATION SYSTEM**

### ***5.3 The Database Manager***

Phase II of the VSS system is the Database Manager. The Database Manager links the construction CPM schedule with the 3D computer model for the construction project, giving time values (start and finish dates) to each model object. This phase is a computer program developed using dBASE IV programming language. It operates on an IBM PC or any IBM PC compatible that supports dBASE IV.

The Database Manager maps the schedule file with the model file. The schedule file is generated from the CPM scheduling processor and transferred to the Database Manager. The user must also input into the Database Manager the names of each object created in the 3D computer model.

The final product of the Database Manager is an ASCII file defining the start and finish dates for each model object. The ASCII file is transferred to Phase III of the VSS system, the Visual Simulator, to produce a visual simulation of the construction sequence.

The following functions are provided within the Database Manager to aid the user in producing the schedule of object durations:

1. Build Model File.
2. Map Model Objects to Activities.
3. Compute Object Dates.
4. Export dBASE file to ASCII File.
5. Exit the VSS System.

### ***5. VISUAL SCHEDULE SIMULATION SYSTEM***

These functions appear and are accessible from the Database Manager's main menu. The following sections describe these five functions. These five functions are shown in Figure 5.1 as executable menu items.

### ***5.3.1 "Build Model File" Function***

Selecting the "Build Model File" function allows the user to define the names of all objects contained in the 3D computer model of the construction project. This function prompts the user to enter the name of each object in the 3D computer model. These names are stored in a user defined model file. The object names need to be entered only once for a given 3D computer model, unless the model has been revised by adding or deleting objects. In this case an updated model file must be created.

### ***5.3.2 "Map Model Objects to Activities" Function***

Selecting the "Map Model Objects to Activities" function allows the user to logically associate schedule activities with 3D model objects. The model file for the construction project must be built before the user can utilize this function. Figure 5.3 illustrates the mapping process. Some of the items in Figure 5.3 are not described in this chapter. Section 6.1.2 describes these items in more detail.

This function prompts the user to enter the name of the scheduling file and the model file to be retrieved. Each object name in the model file is then displayed, one at a time. The user

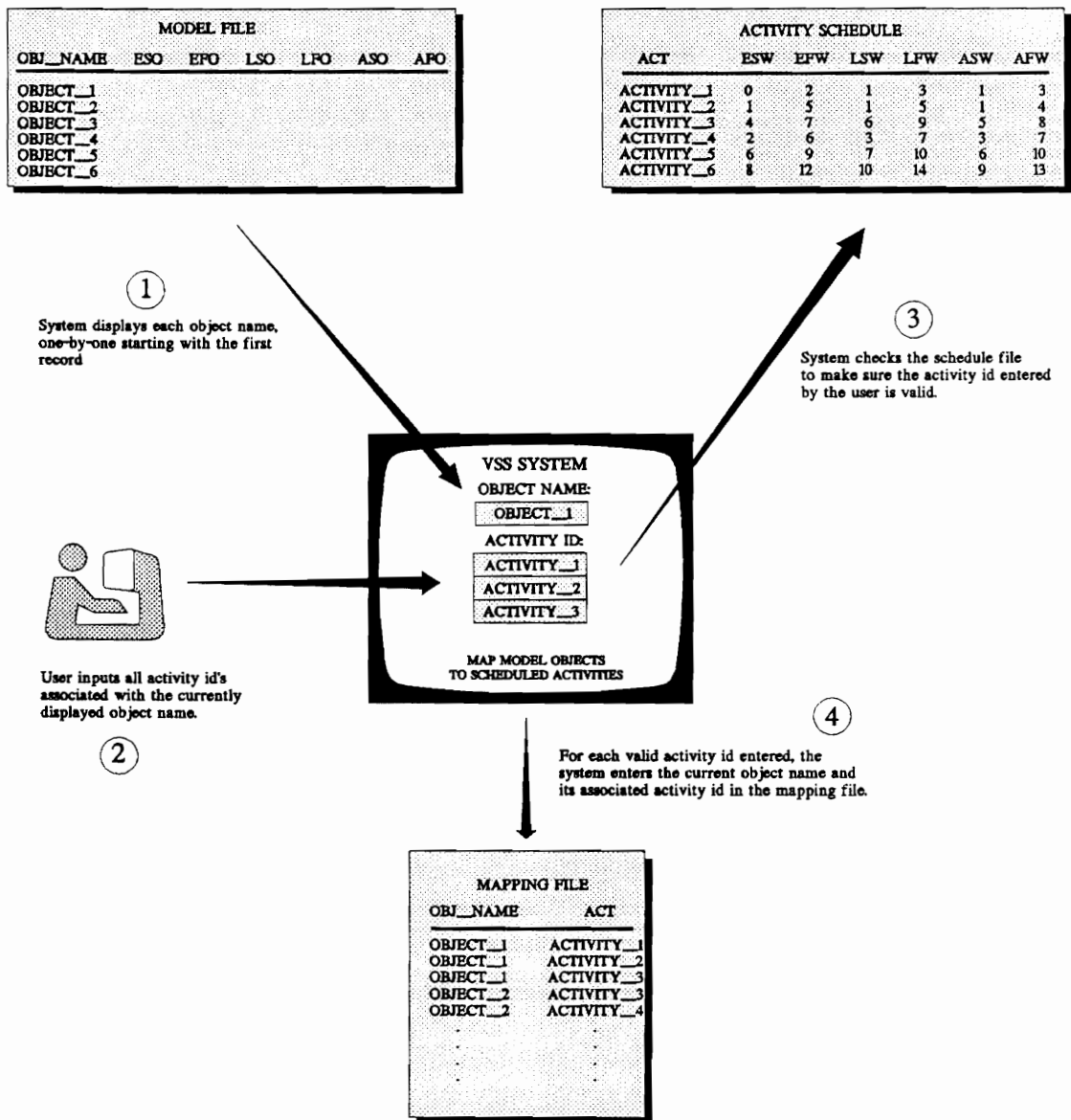
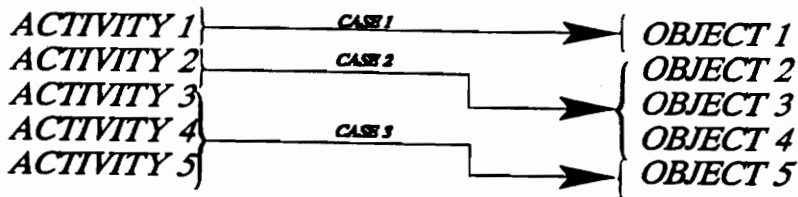


Figure 5.3: The Mapping Process

must enter all activities associated with each object name. Three types of cases can occur:

1. one activity is mapped into one object.
2. one activity is mapped into many objects.
3. many activities are associated with one object.

(see Figure 5.4)



**FIGURE 5.4: Activity/Object Mapping**

This mapping information is stored in a user defined mapping file for later use.

### **5.3.3 "Compute Object Dates" Function**

The "Compute Object Dates" function uses the mapping file to calculate object start and finish dates. This is accomplished by using the relationships developed between model objects and scheduled activities for a construction project. This function will prompt the user to enter the names of the schedule file, model file, and mapping file associated with the construction project. These files are retrieved and opened. The Database Manager then performs the following:

1. Reads the mapping file to determine which activities are associated with each model object.
2. Extracts the earliest start date and latest finish date from each group of activities associated with each model object. This information is retrieved from the schedule file for each of the view options: early date, late date, and actual date fields.
3. Appends the model file by assigning these durations to the appropriate objects.

To illustrate this algorithm, consider the following simple example:

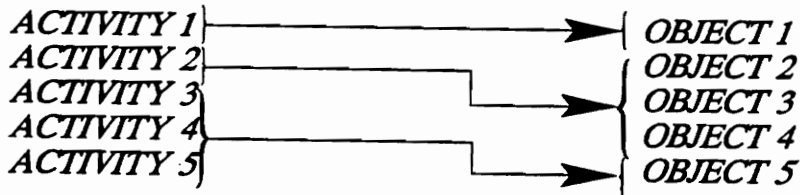
The following information for a construction project is stored in the scheduling file:

<i>ACTIVITY #</i>	<i>Early Start Date</i>	<i>Early Finish Date</i>	<i>Late Start Date</i>	<i>Late Finish Date</i>	<i>Actual Start Date</i>	<i>Actual Finish Date</i>
<i>1</i>	<i>0</i>	<i>3</i>	<i>0</i>	<i>3</i>	<i>0</i>	<i>3</i>
<i>2</i>	<i>3</i>	<i>8</i>	<i>4</i>	<i>9</i>	<i>3</i>	<i>9</i>
<i>3</i>	<i>3</i>	<i>5</i>	<i>5</i>	<i>7</i>	<i>4</i>	<i>6</i>
<i>4</i>	<i>8</i>	<i>14</i>	<i>10</i>	<i>16</i>	<i>9</i>	<i>15</i>
<i>5</i>	<i>14</i>	<i>18</i>	<i>14</i>	<i>18</i>	<i>15</i>	<i>18</i>

**FIGURE 5.5: ACTIVITY SCHEDULE**



Using the following activity/object relationships are stored in a mapping file:



**FIGURE 5.6: Activity/Object Mapping**

The following object dates can be calculated:

<i>OBJECT #</i>	<i>Early Start Date</i>	<i>Early Finish Date</i>	<i>Late Start Date</i>	<i>Late Finish Date</i>	<i>Actual Start Date</i>	<i>Actual Start Date</i>
<i>1</i>	<i>0</i>	<i>3</i>	<i>0</i>	<i>3</i>	<i>0</i>	<i>3</i>
<i>2</i>	<i>3</i>	<i>8</i>	<i>4</i>	<i>9</i>	<i>3</i>	<i>9</i>
<i>3</i>	<i>3</i>	<i>8</i>	<i>4</i>	<i>9</i>	<i>3</i>	<i>9</i>
<i>4</i>	<i>3</i>	<i>8</i>	<i>4</i>	<i>9</i>	<i>3</i>	<i>9</i>
<i>5</i>	<i>3</i>	<i>18</i>	<i>5</i>	<i>18</i>	<i>4</i>	<i>18</i>

**FIGURE 5.7: OBJECT SCHEDULE**

This information is appended to the model file and stored.

The model file now contains the early start and finish dates, late start and finish dates, and actual start and finish dates for each 3D model object created for a construction project.

## 5. VISUAL SCHEDULE SIMULATION SYSTEM

#### ***5.3.4 "Export dBASE File to ASCII File" Function***

All files used by the Database Manager are in the .dbf format. The Visual Simulator (Phase III of the VSS system) uses the model file to determine object durations in order to visually simulate the construction sequence. The Visual Simulator, however, can only read files in the ASCII format. Therefore, the model file must first be converted from the .dbf format to ASCII format before it can be used.

The "Export dBASE File to ASCII File" function allows the user to specify a model file name and converts this file from a .dbf format to an ASCII format. The model file is now ready to be transferred to the Visual Simulator to produce a visual simulation of the construction process.

#### ***5.3.5 "Exit the VSS System" Function***

The "Exit the VSS System" function allows the user to leave the Database Manager. This function returns the user to the dBASE IV environment.

### ***5.4 The Visual Simulator***

Phase III in the VSS system is the Visual Simulator. The Visual Simulator allows the user to produce a visual simulation of the construction process. The Visual Simulator uses the time values assigned to each 3D computer model object, during Phase II, to define the parameters to visually simulate construction activities.

## ***5. VISUAL SCHEDULE SIMULATION SYSTEM***

The user has the flexibility to simulate early start and finish dates, late start and finish dates, or early start and late finish dates for planned construction. The user can also visually simulate actual construction progress. Both planned and actual construction progress can be viewed individually, or at the same time using two identical models of the construction project. The user can simulate the entire project or any segment desired.

#### ***5.4.1 System Description***

The Visual Simulator incorporates several "C" programming language functions which can be accessed using the WALKTHRU system. In order to use the Visual Simulator, the ASCII model file generated by the Database Manager must be transferred to the platform running the WALKTHRU system. Also, the 3D computer model of the construction project must be converted into a WALKTHRU file. Two copies of the same model should be translated into the same WALKTHRU file using WALKTHRU's merge facility (see Section 6.2.1.4).

The Visual Simulator denotes one model as the primary model and the second identical model as the secondary model. The primary model is used to show simulation for either planned or actual construction progress. It is also used to show simulation for planned construction when the user wishes to view both planned and actual construction progress at the same time. The secondary model is only used to show actual construction progress when the user wishes to view both planned and actual construction at the same time.

In order to customize visual simulation of the construction process the user must enter the WALKTHRU environment and display the 3D computer models of the construction

### ***5. VISUAL SCHEDULE SIMULATION SYSTEM***

project. The Visual Simulator is menu driven. Its main menu can be accessed by selecting the VSS function from WALKTHRU's main menu system. The following features are made available to the user in order to produce and customize the visual simulation process.

1. Define the Input File, Project Start Date, and Object Name File
2. Simulation Time
3. Logic Options
4. View Options
5. Simulation Start Date
6. Simulation Finish Date
7. Execution
8. Exit the VSS System

These are found in Figure 5.1 as executables 1 - 8. The following subsection describes each of these features in detail.

### ***5.4.2 System Features***

**Define the Input File, Project Start Date, and Object Name File** - The first steps in the Visual Simulator requires the user to define the input file, the project start date and the object name file. Upon selection of this feature, the Visual Simulator prompts the user to enter the name of the input file. This is the model file containing the object dates. This file is generated by the Database Manager and transferred to the Visual simulator. The Visual Simulator reads the data contained in this file and stores this information for later use. This data contains information on the early, late, and actual start and finish dates for

each 3D computer model objects (see Section 6.2.2.1).

The Visual Simulator then prompts the user to enter the project start date. The Visual Simulator assumes the project starts on a monday and that work weeks are five days long. It calculates the next 700 calendar dates using this logic (see Section 6.2.2.2).

The Visual Simulator then prompts the user to enter the name of the object name file. This file defines the names of each object in the WALKTHRU model file (see Section 6.2.2.3).

**Simulation Time** - This feature allows the user to adjust the speed for visual simulation. The user may select a speed ranging from 0 to 20. These numbers represent the time, in quarter seconds, between each working day during the simulation process. For example, if the user selects a speed of 6, this represents  $6/4$  or 1.5 seconds between each day during the simulation process. If the user wants to simulate construction for 60 actual working days, then visual simulation will last for 90 seconds (60 days x 1.5 seconds/day). If the user selects a speed of 0, then the simulation speed is limited by the processing speed of the computer.

**Logic Options** - This feature allows the user to select which type of scheduling logic to use for visual simulation. The following options are available:

1. **Early Dates:** Early start and finish dates assigned to each model object are used for visual simulation of planned construction.
2. **Late Dates:** Late start and finish dates assigned to each model object are used for visual simulation of planned construction.
3. **Early/Late Dates:** Early start and late finish dates assigned to each model

## 5. VISUAL SCHEDULE SIMULATION SYSTEM

object are used for visual simulation of planned construction.

**View Options** - This feature gives the user the flexibility to choose the type of schedule to use to view visual simulation. The following is a list of available choices:

1. **Planned Schedule:** Visual simulation is prepared using the primary 3D computer model to show planned progress of construction using the appropriate logic option.
2. **Actual Schedule:** Visual simulation is prepared using the primary 3D computer model to show actual progress of construction.
3. **Planned and Actual Schedule:** Visual Simulation is prepared using a) the primary 3D computer model to show planned progress of construction using the appropriate logic option, and b) the secondary 3D computer model to show actual progress of construction. Planned and actual construction simulation occur coincidentally.

**Simulation Start and Finish Dates** - These two features allows the user to define the time period for simulation. Specifying the true planned or actual construction start and finish dates as the simulation start and finish dates allows the user to visually simulate the construction process in its entirety, from beginning to end. The user, however, may simulate any time period during the true construction period. The user merely needs to specify simulation start and finish dates accordingly.

For example, consider a construction project that is scheduled to begin February 1, 1990 and end December 15, 1990. If the user wants to view construction activities between

April 1, 1990 and April 15, 1990, then the user must enter these dates for the simulation start and finish dates. The Visual Simulator will only simulate construction activities between these two dates.

The Visual Simulator follows the following rules when simulating construction for a given time period:

1. All construction activities completed before the simulation start date will not be shown. If the user wishes to see activities completed before the simulation start date then they must first simulate activities from the construction start date to the desired simulation start date. The user can then proceed with the visual simulation as planned.
2. All construction activities started before the simulation start date but completed after the simulation start date will be shown.
3. All construction activities started between the simulation start and finish dates will be shown.
4. All construction activities started after the simulation finish date will not be shown.

**Execution** - Execution allows the user to view, on the graphics display, a simulation of the construction process using all of the previously defined parameters. Simulation proceeds in the following manner.

1. All objects are first turned off so that a blank screen is displayed.
2. Simulation proceeds in a day-by-day manner, from simulation start date

##### **5. VISUAL SCHEDULE SIMULATION SYSTEM**

to simulation finish date.

3. Each day, the Visual Simulator checks to see if the activities associated with each 3D model object have started on that day, are still in progress on that day, or have finished on that day.

4. If construction for an object begins on the given day, then the object is turned on and is represented using the color pink. If an object is displayed pink then the viewer knows the object is still under construction.

5. If activities related to an object are still in progress on the given day, then the object remains pink. 6. If activities related to an object finish on the given day, then the object is changed from pink back to its original color.

This lets the view know that construction on that object has been completed.

This process is illustrated in Figure 5.8.

**Exit the VSS System** - This function allows the user to leave the Visual Simulator.

This function returns the user to the WALKTHRU environment.

This chapter has described the functionality of the VSS system. It does not, however, describe what happens with the VSS system when menu items are selected and the user inputs information. The next chapter describes how the VSS system's programs and sub-programs interact with each other and the user to provide a user interactive environment.



The following diagram is an example of how the VSS system reacts during the execution phase.

The Visual Simulator simulates construction activities from simulation start date to simulation finish date.

The Visual Simulator checks the status of every object for each simulation day and determines which objects are to be turned on and how they are to appear when they are on. It determines this by adhering to the rules in the following diagram:

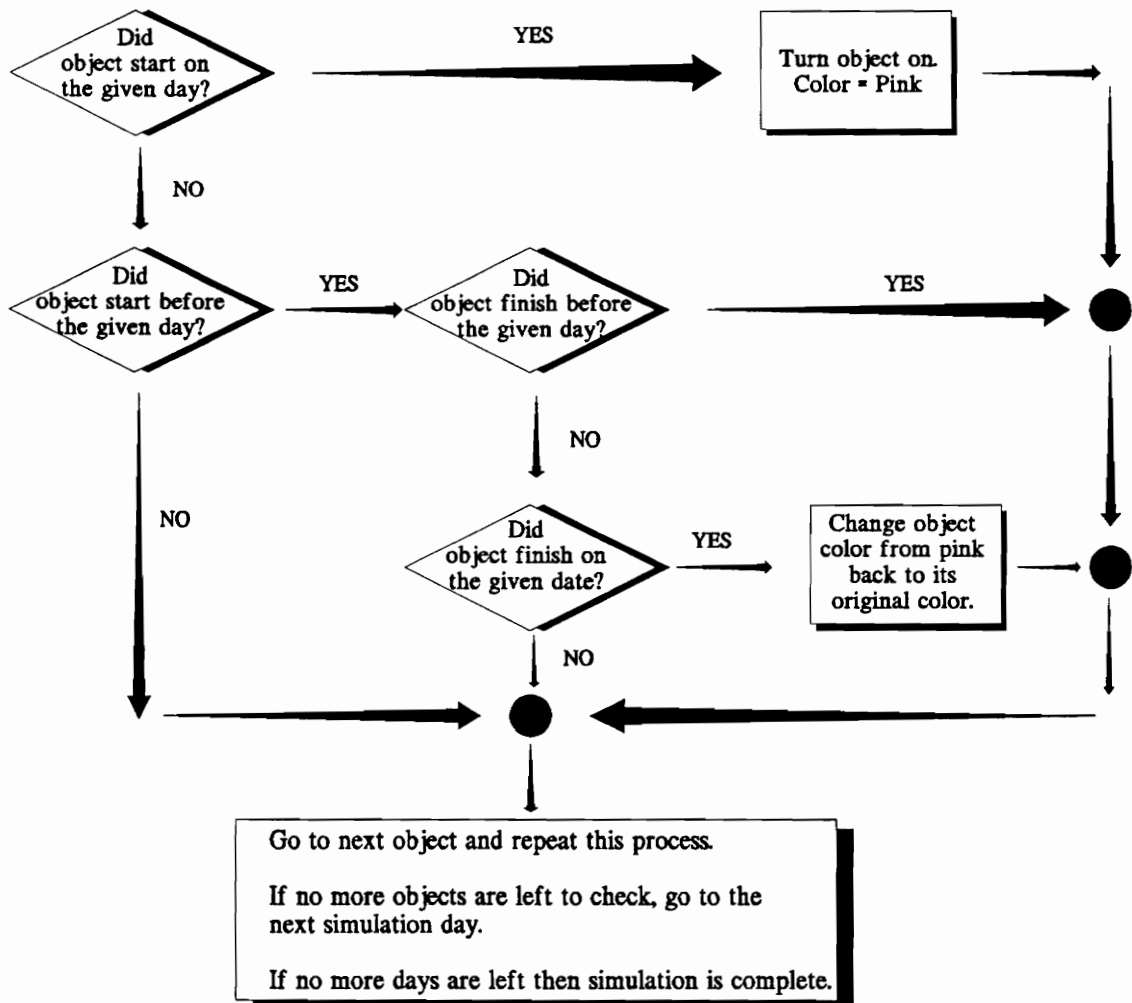


Figure 5.8: The Execution Process

## ***6. VSS SYSTEM IMPLEMENTATION***

This chapter discusses the technical aspects of the VSS system. It describes how both Phase II, the Database Manager and Phase III, the Visual Simulator were constructed. This chapter addresses the programming methodology and design.

In addition to Phase 1, the Data Preprocessor, the VSS system is made up of the Database Manager and the Visual Simulator software systems. The Database Manager is a collection of procedures developed using the dBASE IV programming language and runs on an IBM PC or IBM PC compatible capable of running the dBASE IV software. The Visual Simulator is a collection of "C" programming language functions which can be accessed from within the WALKTHRU system. It runs on a Silicon Graphics IRIS 4D series workstation.

The Database Manager system needs the dBASE IV software installed on the PC before it can be used. The Database Manager software must be installed in a directory on the PC which can access the dBASE IV system. The Database Manager software includes the source code, object.dbf file, and map.dbf file (these files are discussed in Sections 6.1.1 and 6.1.2 respectively).

The Visual Simulator system requires that the WALKTHRU software be installed on a Silicon Graphics IRIS 4D workstation before it can be used. The Visual Simulator software must be installed in a directory which can access the WALKTHRU system.

Section 6.1 discusses the program development of the Database Manager. The procedures making up the Database Manager and how they interact with one another to ultimately produce an object schedule is presented.

Section 6.2 discusses the program development of the Visual Simulator. The functions defined by the Visual Simulators main menu is presented. How these functions interact with one another to ultimately produce a visual simulation of the construction sequence is discussed.

The terms procedure and function are used interchangeably throughout this chapter. This is to preserve the terminology used by the dBASE IV programming language, which uses "procedure", and the "C" programming language, which uses "function". These terms refer to sub-programs within main programs of the Database Manager and Visual Simulator.

## ***6.1 Phase II - The Database Manager***

To access the Database Manager, the user must enter the dBASE IV system and exit the Control Center to the dot prompt. Typing the command "DO VSS" at the dot prompt lets the user access the Database Manager's main menu. The following menu appears:

### **6. VSS SYSTEM IMPLEMENTATION**

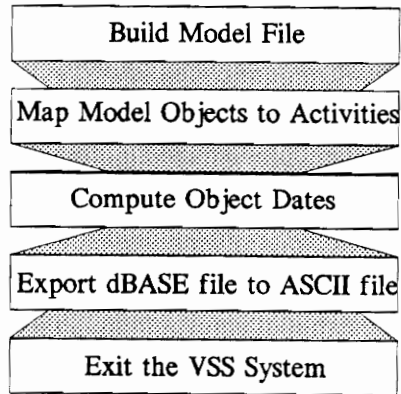


Figure 6.1: Database Manager's Main Menu

The main menu allows the user to interact with the Database Manager. These menu options have been previously described in detail in Section 5.3.

Before the Database Manager can be described, the program's procedures must be discussed. How these procedures are used is further described in the subsequent subsections. The following is a description of each procedure developed and used by the Database Manager:

1. **VSS procedure** - This is the main procedure of the Database Manager. It sets up the Database Manager's main menu, the dBASE environment, and calls the appropriate procedure when the user selects an item from the main menu.
2. **BUILD\_OB procedure** - This procedure allows the user to define the name of the model file and the name for each object in the 3D computer model of the construction project. It is called from the VSS procedure.

3. **MAPS procedure** - This procedure prompts the user for the names of the scheduling file and model file to use for activity/object mapping, and the name of the mapping file to store the mapping information. This procedure then prompts the user to enter the activity id's associated with each 3D computer model object. It is called from the VSS procedure.
4. **COM\_LOG procedure** - This procedure prompts the user for the names of the schedule file, object file, and mapping files to retrieve. It uses the information contained in these files to generate an object schedule. It is called from the VSS procedure.
5. **EXPORT procedure** - This procedure prompts the user for the name of a .dbf model file and transforms this file into an ASCII format. It is called from the VSS procedure.
6. **RET\_SCH procedure** - This procedure prompts the user for the name of the scheduling file to be used and retrieves this file. It is called from the MAPS and COM\_LOG procedures.
7. **RET\_OBJ procedure** - This procedure prompts the user for the name of the model file to be used and retrieves this file. It is called from the MAPS and COM\_LOG procedures.
8. **RET\_MAP procedure** - This procedure prompts the user for the name of the mapping file to be used and retrieves this file. It is called from the COM\_LOG procedure.
9. **HEADING procedure** - This procedure creates the "VSS System" heading at the top of each screen in the Database Manager system. It is called from the VSS, BUILD\_OB, MAPS, COM\_LOG, and EXPORT procedures.

## **6. VSS SYSTEM IMPLEMENTATION**

The following sections describe the procedures which are utilized directly from the Database Manager's main menu. Appendix B.1 and C.1 provide the flow charts and source code for the Database Manager.

### ***6.1.1 "Build Model File"***

When the user wants to enter the names of the objects in the 3D computer model, they must select the "Build Model File" menu item. The Database Manager's VSS procedure reacts by calling the BUILD\_OB procedure and displaying the BUILD MODEL FILE screen.

The BUILD\_OB procedure first prompts the user to enter the file name to assign to the model file. Once the file name has been entered, the system checks the current directory to ensure a file with this name does not already exist. If it does, then the system gives the user the choice to overwrite the file or enter a new file name.

Once a valid model file name has been defined, the system creates a .dbf file. In order for the Database Manager to use a .dbf file, it must have a file structure which defines the fields of information the file will contain (see Section 3.3.1). The model file structure is permanently stored in a file named OBJECT.DBF. This file does not contain any records of information it merely defines the following structure:

<u>FIELD</u>	<u>FIELD NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DEFINITION</u>
1	OBJ_NAME	CHARACTER	15	OBJECT NAME
2	ESO	NUMERIC	5	OBJ. EARLY START DATE
3	EFO	NUMERIC	5	OBJ. EARLY FINISH DATE
4	LSO	NUMERIC	5	OBJ. LATE START DATE
5	LFO	NUMERIC	5	OBJ. LATE FINISH DATE
6	ASO	NUMERIC	5	OBJ. ACTUAL START DATE
7	AFO	NUMERIC	5	OBJ. ACTUAL FINISH DATE

The BUILD\_OB procedure copies the structure from the OBJECT.DBF file to the user defined model file.

After the model file structure has been defined, the system prompts the user to enter the names of each object in the 3D computer model of the construction project. These names must be identical to the object names defined in the WALKTHRU "Object Name File" (this is discussed in Section 6.2.2.3). The names the user assigns to each of the 3D computer model objects must be unique. Therefore, each time an object name is entered the system checks the model file to ensure it does not duplicate an object name previously entered. If a duplicate name is found, the system warns the user and provides a prompt to enter a new object name. If the name is valid it is stored sequentially, starting with the first record, under the OBJ\_NAME field. When the user wants to terminate entering object names the escape key must be pressed. This signals the BUILD\_OB procedure to close the model file, return to the VSS procedure, and display the main menu screen. The model file now contains the names of each object in the 3D computer model of the construction project.

## **6. VSS SYSTEM IMPLEMENTATION**

### 6.1.2 "Map Model Objects to Activities"

When the user wants to define which construction activities are associated with each 3D computer model object the "Map Model Objects to Activities" menu item must be selected. The Database Manager's VSS procedure calls the MAPS procedure and displays the MAP MODEL OBJECTS TO SCHEDULED ACTIVITIES screen. Figure 5.3 illustrates the mapping process and shows an example of a typical mapping file created by the MAPS procedure. This Figure should be referred to when reading this section.

The MAPS procedure calls, 1) the RET\_SCH procedure to retrieve a user defined schedule file, and 2) the RET\_OBJ procedure to retrieve a user defined model file. The schedule file is the .dbf file created by the CPM processor. The schedule file has the following file structure:

FIELD	FIELD NAME	TYPE	WIDTH	DEFINITION
1	ACT	CHARACTER	15	ACTIVITY ID
2	ESW	NUMERIC	5	ACT. EARLY START DATE
3	EFW	NUMERIC	5	ACT. EARLY FINISH DATE
4	LSW	NUMERIC	5	ACT. LATE START DATE
5	LFW	NUMERIC	5	ACT. LATE FINISH DATE
6	ASW	NUMERIC	5	ACT. ACTUAL START DATE
7	AFW	NUMERIC	5	ACT. ACTUAL FINISH DATE

Once the schedule and model files have been retrieved, the system prompts the user for the file name to store mapping information. A .dbf file is created with the user defined file

## 6. VSS SYSTEM IMPLEMENTATION



name. The structure for the mapping file has been stored in a file named MAP.DBF. This file defines the following structure:

<u>FIELD</u>	<u>FIELD NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DEFINITION</u>
1	OBJ_NAME	CHARACTER	15	OBJECT NAME
2	ACT	CHARACTER	15	ACTIVITY ID

The MAPS procedure copies the structure from the MAP.DBF file to the user defined mapping file.

Once the mapping file has been defined and the schedule file and model file have been retrieved, the MAPS procedure opens each of these files. The system reads each record in the model file and displays the object names from the OBJ\_NAME field, one at a time. For each object name displayed, the user must enter all associated activity ID's. The Database Manager allows up to eight activity ID's to be associated with each object name. Each time an activity ID is entered, the system checks the ACT field in the schedule file to make sure it exists. If it does not, the system warns the user and prompts them to enter a valid activity ID. When a valid activity ID is entered the MAPS procedure sequentially stores, 1) the object name, under the OBJ\_NAME field, and 2) its associated activity id, under the ACT field, of the user defined mapping file. When all activity ID's have been entered for the last object in the model file, the system closes all open files, returns to the VSS procedure, and displays the main menu screen.

### ***6.1.3 "Compute Object Dates"***

When the user wants to produce the object schedule the "Compute Object Dates" menu item must be selected. The VSS procedure calls the COM\_LOG procedure and displays the COMPUTE OBJECT DATES screen. The COM\_LOG procedure calls, 1) the RET\_SCH procedure to retrieve the schedule file, 2) the RET\_OBJ procedure to retrieve the model file, and 3) the RET\_MAP procedure to retrieve the mapping file. These files must all be related to the same construction project or an error will result. Once these files have been retrieved the COM\_LOG procedure opens each of them.

The next step in the COM\_LOG procedure initializes the date fields in the model file (see Section 6.1.1 for the file structure of the model file). Each record for the ESO, LSO, and ASO fields are assigned the value 10,000. Each record for the EFO, LFO, and AFO fields are assigned the value 0. The reason these values were chosen will become apparent.

Figure 6.2 illustrates the "Compute Object Dates" process. This Figure should be referred to when reading the rest of this section.

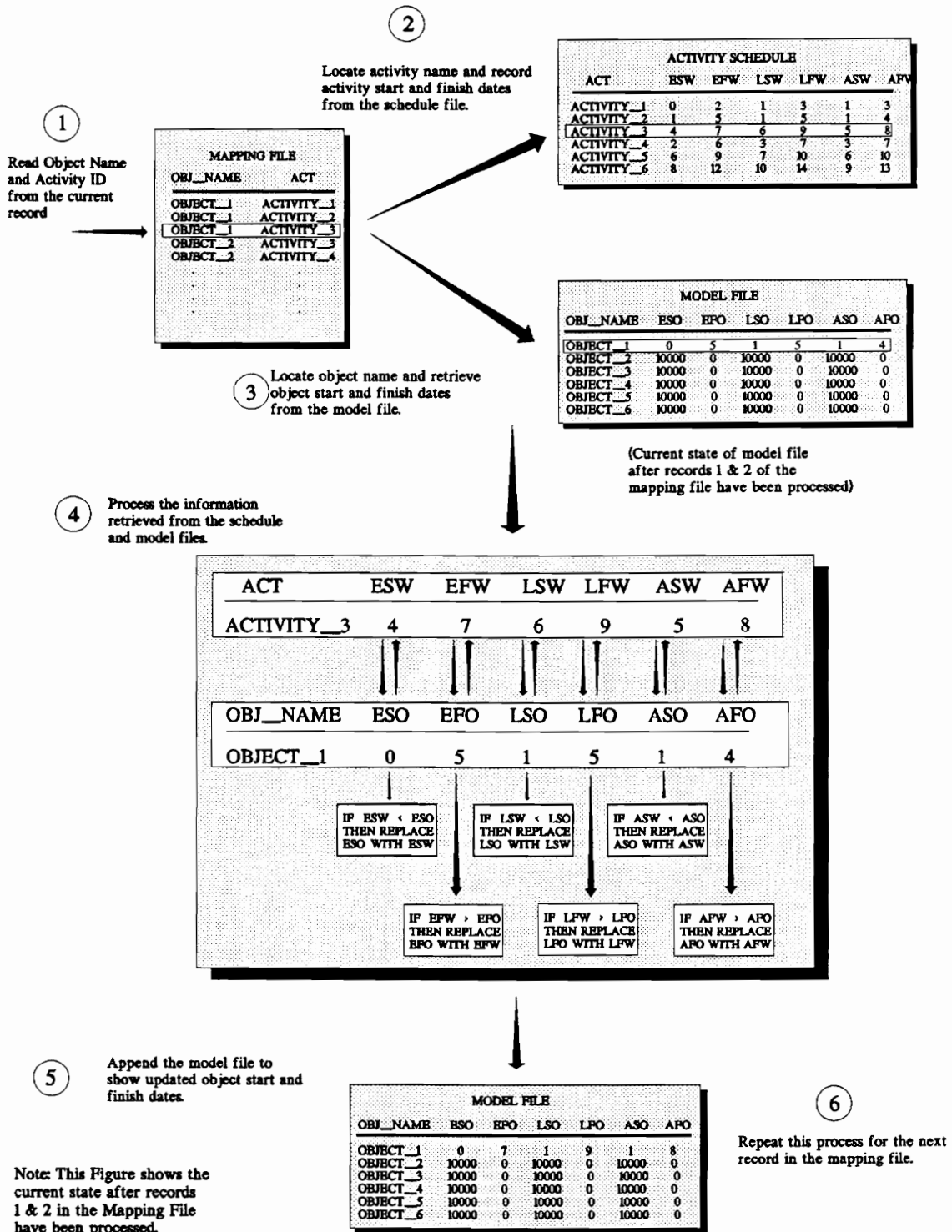


Figure 6.2: Compute Object Dates process

## 6. VSS SYSTEM IMPLEMENTATION

The COM\_LOG procedure is now ready to assign object durations to each object name in the model file. Each object name has a group of associated activity ID's defined by the mapping file. The activity from this group which has the earliest start date defines the object start date. The activity which has the latest finish date defines the object finish date. For example:

If activity 1, 2, and 3 are associated with object 1 and these activities are scheduled as follows:

ACT	ESW	EFW	LSW	LFW	ASW	AFW
ACTIVITY1	0	2	1	3	0	3
ACTIVITY2	2	8	2	8	3	8
ACTIVITY3	3	7	4	9	2	6

Then the following values are assigned to object 1 in the model file:

OBJ NAME	ESO	EFO	LSO	LFO	ASO	AFO
OBJECT1	0	8	1	9	0	8

The COM\_LOG procedure processes the object schedule as follows:

Step 1 - Read the object name and activity ID from the current record (start with the first record).

Step 2 - Locate the activity ID in the schedule file and record its early, late, and actual start and finish dates.

## 6. VSS SYSTEM IMPLEMENTATION

Step 3 - Locate the object name in the model file.

Step 4 - If the activity start date is less than the object start date, then replace the object start date with the activity start date. If the activity finish date is greater than the object finish date, then replace the object start date with the activity start date (it is assumed that the first time an object's start and finish date is checked the activity start date will be less than its initial value 10,000 and the activity finish date will be greater than or equal to its initial value 0).

Step 5 - Append the model file to show the updated object start and finish dates.

Step 6 - Go to the next record in the mapping file and repeat steps 2 - 6 until the last record has been processed.

When the last record in the mapping file has been processed, logic computation is complete. The COM\_LOG procedure closes all open files, returns to the VSS procedure, and displays the main menu screen. The model file now contains the object schedule.

#### ***6.1.4 "Export dBASE File to ASCII File"***

When the user wants to convert a file from .dbf format to ASCII format the "Export dBASE File to ASCII File" menu item must be selected. The VSS procedure calls the EXPORT procedure and displays the EXPORT FILE screen. The EXPORT procedure prompts the user to enter the name of the file to export and ensures this is a valid dBASE file.

The EXPORT procedure retrieves the dBASE file entered and uses the

dBASE IV "COPY" command to duplicate this file into ASCII format. The COPY command uses the SDF file option. The SDF option creates a System Data Format ASCII (.txt) file. Character data in the file is not delimited and fields are not separated with a character. Records are fixed length, the same length as the records in the dBASE file, and every record ends with a carriage return and line feed.

After the ASCII file has been created, the system returns to the VSS procedure and main menu screen. When the user exports the model file, containing the object schedule, to an ASCII format, it is ready to be transferred to the Visual Simulator for Phase III of the VSS system.

## ***6.2 Phase III - The Visual Simulator***

The Visual Simulator runs on the Silicon Graphics IRIS 4D series workstation. In order to transfer the file containing the object schedule from the PC running the Database Manager to the Silicon Graphics workstation the user must either:

1. Down load the file onto a disk or tape and upload the file onto the Silicon Graphics workstation.
2. Transfer the file from the PC to the Silicon Graphics workstation via an ethernet link.

The Visual Simulator becomes accessible to the user by selecting the VSS menu item from WALKTHRU's main menu. The main menu has been altered in a minor way to support the VSS system. A menu choice entitled "VSS" has been added. Once this item is selected

the Visual Simulators main menu appears in the bottom left hand corner of the screen giving the user power to produce and customize visual simulation of construction activities for a construction project. The following menu appears:

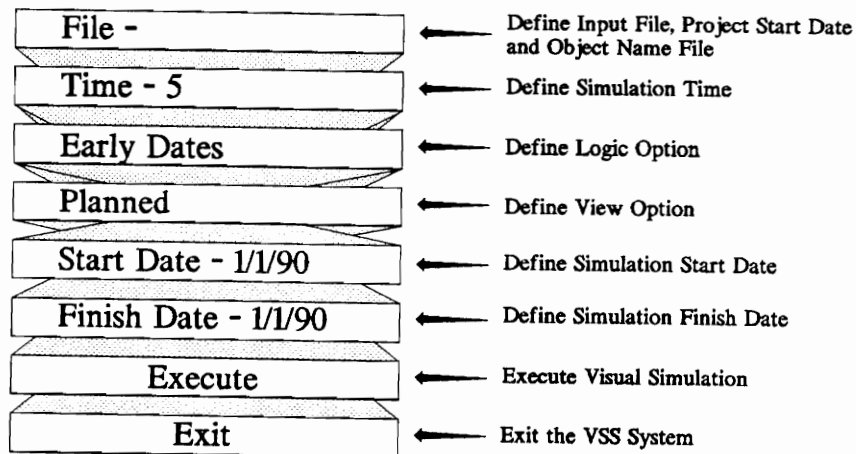


Figure 6.3: Visual Simulator's Main Menu

The Visual Simulator allows the user to:

1. Define the following initial information:
  - a. The model file containing the object schedule. This is the ASCII file transferred from the Database Manager.
  - b. The project start date.
  - c. The Object Name File.

(see Section 6.2.2)
2. Define the simulation time between construction working days (see Section 6.2.3).

3. Define the logic options (see Section 6.2.4):
  - a. Early Dates
  - b. Late Dates
  - c. Early/Late Dates
4. Define the view options (see Section 6.2.5):
  - a. Planned
  - b. Actual
  - c. Planned and Actual
5. Define simulation start date (see Section 6.2.6).
6. Define simulation finish date (see Section 6.2.6).
7. Execute visual simulation (see Section 6.2.7).
8. Exit the VSS system.

These options have all been described in detail in Section 5.4.2.

Before the Visual Simulator can be described, the program's functions must be discussed. The following is a description of each function developed and used by the Visual Simulator:

1. **vss function** - This is the main function of the Visual Simulator. This function sets up the Visual Simulators main menu, initializes menu values, and calls the appropriate function when the user places the cursor of the mouse over a menu item.
2. **get\_file function** - This function allows the user to define the model file containing the object schedule, construction start date, and Object Name



File. It is called from the vss function.

3. **calc\_date** function - This function defines the calendar dates starting with the construction start date and ending 700 days from this date. It assumes the construction project begins on a monday and the project has a five day work week. It is called from the **get\_file** function.

4. **date\_update** function - This function calculates the 700 calendar dates for the **calc\_date** function. It is called from the **calc\_date** function.

5. **number\_of\_days** function - This function defines the number of days for each month so the **date\_update** function can calculate the correct calendar date. It is called from the **calc\_date** function.

6. **is\_leap\_year** function - This function determines whether a given calendar year is a leap year so that the number of days function can determine if February has either 28 or 29 days. It is called from the **number\_of\_days** function.

7. **get\_time** function - This function allows the user to define the simulation time between working days. It is called from the **vss** function.

8. **get\_logic** function - This function allows the user to define the logic option to use for visual simulation. It is called from the **vss** function.

9. **get\_view** function - This function allows the user to define the view option to use for visual simulation. It is called from the **vss** function.

10. **get\_start\_date** function - This function allows the user to define the simulation start date. It is called from the **vss** function.

11. **get\_finish-date** function - This function allows the user to define the simulation finish date. It is called from the **vss** function.

12. **simulate** function - This function uses the parameters defined by the

## **6. VSS SYSTEM IMPLEMENTATION**

previous functions to execute the visual simulation of the construction project. It is called from the vss function.

13. **sim\_act** function - This function produces a visual simulation of the actual construction, using the secondary model, when the "Planned & Actual" view option is selected. It is called from the simulate function.

The following sections describe the functions which are utilized directly from the Visual Simulators main menu. Appendix B.2 and C.2 provide the flow charts and source code for the Visual Simulator.

### ***6.2.1 Creating the WALKTHRU File from the Original 3D Computer Model***

As discussed in Sections 5.2.2 and 5.4.1, two copies of the 3D computer model need to be converted into a single WALKTHRU model file. The 3D computer model can be converted into a WALKTHRU file using one of four conversion programs available. The options available to each of these programs can be found in Chapter 2 of the WALKTHRU User's Guide.

#### ***6.2.1.1 WALKIGDS and WALK3DM***

If the 3D computer model of the construction project has been developed using either InteRgraph or Bechtel 3DM Intergraph CAD systems, the user will use slightly different versions of the same conversion program: WALKIGDS and WALK3DM. The only

difference between the two is WALK3DM extracts 3DM labels associated with the graphic elements and puts them into the WALKTHRU model file. The syntax for converting Intergraph design files and Bechtel 3DM Intergraph design files using WALKIGDS and WALK3DM programs are as follows:

```
walkigds [options] [IGDSfile1 IGDSfile2 ...]
```

or

```
walk3dm [options] [3DMfile1 3DMfile2 ...]
```

The result of running the above programs is a WALKTHRU model file with the .wlk file extension. This file can now be used by the WALKTHRU system to display the 3D computer model.

#### **6.2.1.2 WALKIGES**

If the 3D computer model of the construction project has been designed on a CAD system whose model file cannot be directly converted into a WALKTHRU model file, the model file can first be converted to an IGES data file using the Initial Graphics Exchange Specification. This file can be converted into a WALKTHRU model file using the WALKIGES program. The syntax for converting the IGES data file into a WALKTHRU model file is as follows:

```
walkiges [options] [IGESfile1 IGESfile2 ...]
```

The result of running the above program is a WALKTHRU model file with the .wlk file

extension.

### **6.2.1.3 WALKPRE**

If the 3D computer model of the construction project is in a WALKTHRU ASCII file format, then the model file must be converted into a WALKTHRU model file using the walkpre program (see appendix B of the WALKTHRU User's Guide for the format of WALKTHRU ASCII files). The syntax for converting WALKTHRU ASCII files into WALKTHRU model files is as follows:

```
walkpre [options] [ASCIIfile1 ASCIIfile2 ...]
```

CADAM and CATIA both convert model information into a WALKTHRU ASCII file format. When converting these files the -ll option must be used. The -ll option allows the WALKPRE program to differentiate between separate objects in the 3D computer model. The result of running the WALKPRE program is a WALKTHRU model file with the .wlk file extension.

### **6.2.1.4 Merging WALKTHRU model files**

Two identical 3D computer models are needed to view both planned and actual construction progress. In order for the WALKTHRU file to contain two identical 3D computer models of the construction project, the user must use WALKTHRU's merge facility. The user must first translate the model using the proper translation program. The user must next

translate the same model using the -m option. The -m option allows the WALKTHRU system to add additional objects to an existing WALKTHRU file.

Once both models of the construction project have been merged into the same WALKTHRU model file the user must position each of the models to allow for the most advantageous view for simulation. They should be positioned next to each other as shown in Figure 6.4. This position can be saved by using WALKTHRU record function. The VSS system defines one of the models as the primary model and the other as the secondary model. Both models are necessary to show both planned and actual construction simulation at the same time.

### ***6.2.2 The "File -" Function***

Defining the model file containing the object schedule is the first menu item on the Visual Simulator's main menu. This is also the first step to producing a visual simulation of the construction sequence. To access this function the user needs to place the cursor over the "File -" menu item and press the right mouse button. This activates the get\_file function.

The get\_file function retrieves and stores all of the attributes related to each of the objects in the WALKTHRU model file. This information is stored within the Visual Simulator to produce visual simulation. This function prompts the user to enter the input file name, the project start date, and the name of the Object Name File. The next three Sections describe how this information is processed and used within the Visual Simulator.

## **6. VSS SYSTEM IMPLEMENTATION**

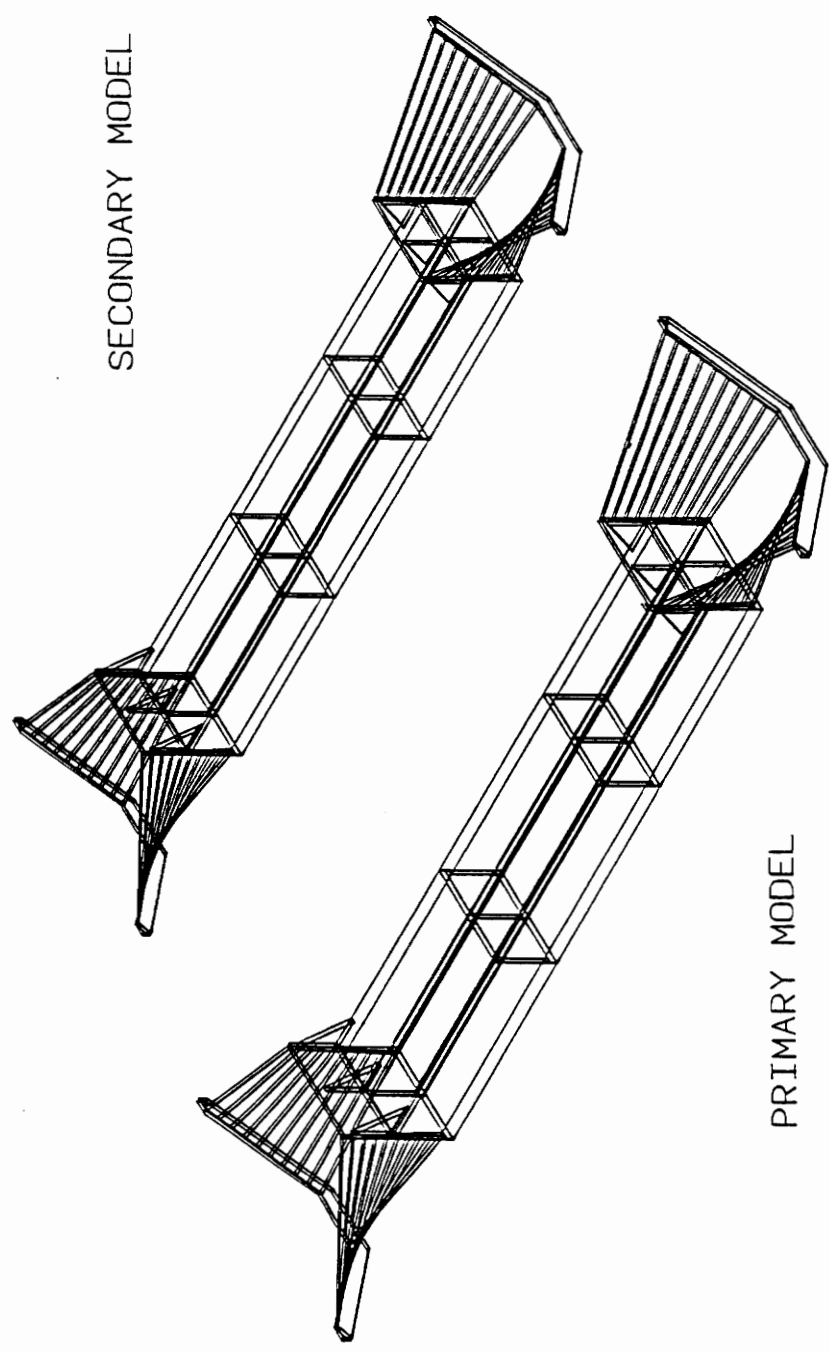


Figure 6.4: Positioning of Primary and Secondary 3D Computer Models

### ***6.2.2.1 The Input File***

The `get_file` function first provides a prompt to enter the name of the input file. The user must key in the name of the model file containing the object schedule. The system searches the current directory for the file (unless another directory is defined when specifying the file name). This file is the ASCII file transferred from the Database Manager. If this file cannot be found, the system will warn the user and provides a prompt to enter another file name. If the file is found, the system opens the file, reads each line or record, and stores the information contained in each record in a "C" structure. The "C" structure is an array of elements capable of handling up to 200 records of information. Each element in the structure stores the following information:

1. Object Name - This is an array of 15 characters defining the name for each 3D computer model object.
2. Early Start Date - This is an integer defining the early start date, in working days, for each object.
3. Early Finish Date - This is an integer defining the early finish date, in working days, for each object.
4. Late Start Date - This is an integer defining the late start date, in working days, for each object.
5. Late Finish Date - This is an integer defining the late finish date, in working days, for each object.
6. Actual Start Date - This is an integer defining the actual start date, in working days, for each object.

7. Actual Finish Date - This is an integer defining the actual finish date, in working days, for each object.
8. Physical Object Number - This is an integer defining the physical object number, corresponding to the object name, for each object.
9. Color - This is an integer defining the color for each object.

This structure is defined as "struct schedule" within the visual simulator. Structure elements 1-7 are extracted from the model file containing the object schedule defined by the user. Structure element 8 is extracted from the Object Name File. Structure element 9 is extracted from the surf.color structure within the WALKTHRU system.

#### ***6.2.2.2 Project Start Date***

After the Input File has been defined and the information for elements 1-7 have been stored in the schedule structure, the system prompts the user to enter the project start date. The user must enter the calendar start date for the planned start of construction. Since all dates in the object schedule are defined in working days, the Visual simulator maps the project start date entered to the first working day of the project. Each successive calendar date is mapped to its corresponding working day. It calculates five day work weeks and assumes the project starts on a monday. For example:



If a project start date = 2/5/90

then:

2/5/90 = working day 0	2/12/90 = working day 5
2/6/90 = working day 1	2/13/90 = working day 6
2/7/90 = working day 2	2/14/90 = working day 7
2/8/90 = working day 3	2/15/90 = working day 8
2/9/90 = working day 4	2/16/90 = working day 9

This date is also initially stored as both the simulation start date and simulation finish date (see Section 6.2.6).

### ***6.2.2.3 Object Name File***

After the user has entered the project start date, the system provides a prompt to enter the name of the Object Name File created for the WALKTHRU model file. The Object Name File defines names for each of the objects which exist in the WALKTHRU model file. The `get_file` function opens, reads, and stores the information in the Object Name File. This subsection describes the Object Name File and how it is used by the Visual Simulator.

When a 3D computer model is converted to a WALKTHRU model file, the WALKTHRU system assigns a unique number to each object. This is called the object's physical number. The first object converted is given the value of 1, the second object is given the value of 2, and so on. For example, if the user converts two identical models, each model containing 23 objects, and merges them into one WALKTHRU model file, WALKTHRU assigns physical numbers as follows:

## **6. VSS SYSTEM IMPLEMENTATION**

- When the primary model is converted its physical objects are defined with numbers starting with 1 and ending with 23.
- When the secondary model is converted its physical objects are defined with numbers starting with 24 and ending with 46. Physical object 1 is identical to physical object 24, 2 is identical to 25, and so on.

The term physical objects is used because these objects are physically defined within the WALKTHRU file. WALKTHRU also allows the user to create virtual objects. A virtual object is merely a copy of a physical object, using all of its physical attributes when displayed. Virtual objects are used because they take up no storage space. For example, if a model of a power station with 3 identical generators is converted into a WALKTHRU file, only one generator needs to be converted into an object. The other two generators can be created as virtual objects in the Object Name File (see the WALKTHRU User's Guide for more information). Only one physical object exists in the WALKTHRU model file, using one-third the memory necessary to store 3 identical physical objects.

Virtual Objects do not allow the Visual Simulator to change their colors independently. The Visual Simulator can only use physical objects. Therefore, even though it requires two identical 3D computer models of the construction project, each model must be defined as separate physical objects.

An Object Name File must be created for each WALKTHRU model file of the construction project before the Visual Simulator can be used. The Object Name File defines virtual objects numbers and object names associated with each of the physical objects in the

WALKTHRU model file by building an Object Name File. The user creates this file using a text editor.

The Object Name File must contain three fields of information for each object record. Field #1 defines the objects virtual number (this must be defined even though it is identical to the physical object number), field #2 defines the objects physical number, and field #3 defines the objects name. The virtual object and physical object numbers must be identical for each object. These numbers must be sequentially numbered, starting with 1 and there must be no gaps in the numbering sequence. The physical objects must exist in the WALKTHRU disk file or an error will result.

The object names are limited by the Visual Simulator to 15 characters and must begin with an alphabetic character (a-z). The names given to each object must be identical to the object names assigned in the model file containing the object schedule, or an error will result.

Figure 6.5 shows an example of an Object Name File used by the Visual Simulator.

The `get_file` function opens, reads and stores the information contained in the Object Name File defined for the construction project. The information for each line or record is stored in a "C" structure of type `obj_walk`. This structure is an array of elements capable of storing up to 200 records. The structure contains the following elements:

1. Virtual Object Number - This is an integer number defining the virtual number for each object.

VIRTUAL OBJECT NUMBER	PHYSICAL OBJECT NUMBER	OBJECT NAMES	
1	1	WEST_BARREL	Definition of the Primary model.
2	2	CENTER_BARREL	
3	3	EAST_BARREL	
4	4	W_TRANS_WALL1	
5	5	W_TRANS_SLAB	
6	6	W_TRANS_WALL2	
7	7	E_TRANS_WALL3	
8	8	E_TRANS_SLAB	
9	9	E_TRANS_WALL4	
10	10	W_WARP_SLAB	
11	11	W_WARP_WALL1	
12	12	W_WARP_WALL2	
13	13	E_WARP_SLAB	
14	14	E_WARP_WALL3	
15	15	E_WARP_WALL4	
16	16	W_HEAD_WALL	
17	17	W_STIFF1	
18	18	W_STIFF2	
19	19	W_STIFF3	
20	20	E_HEAD_WALL	
21	21	E_STIFF1	
22	22	E_STIFF2	
23	23	E_STIFF3	
24	24	WEST_BARREL	Definition of the Secondary model.
25	25	CENTER_BARREL	
26	26	EAST_BARREL	
27	27	W_TRANS_WALL1	
28	28	W_TRANS_SLAB	
29	29	W_TRANS_WALL2	
30	30	E_TRANS_WALL3	
31	31	E_TRANS_SLAB	
32	32	E_TRANS_WALL4	
33	33	W_WARP_SLAB	
34	34	W_WARP_WALL1	
34	34	W_WARP_WALL2	
36	36	E_WARP_SLAB	The object names must be identical to the Primary model.
37	37	E_WARP_WALL3	
38	38	E_WARP_WALL4	
39	39	W_HEAD_WALL	
40	40	W_STIFF1	
41	41	W_STIFF2	
42	42	W_STIFF3	
43	43	E_HEAD_WALL	
44	44	E_STIFF1	
45	45	E_STIFF2	
46	46	E_STIFF3	

Virtual and Physical  
object numbers must  
be identical

Object names must be  
identical to those defined  
in the model file

They must also be in  
the same order.

Figure 6.5: Example of a typical Object Name File

2. Physical Object Number - This is an integer number defining the physical number for each object.
3. Object Name - This is an array of 16 characters defining the name of each object.

Once this information has been read and stored, the Visual Simulator closes this file. The `get_file` function compares each of the object names in the `obj_walk` structure with each of the names in the schedule structure. When a match is found it assigns the physical object number defined in the `obj_walk` structure to the eighth element in the schedule structure (see Section 6.2.2.1.).

Once all physical objects have been correctly assigned to the appropriate object name in the schedule structure, the `get_file` function searches for the color of each object. The `get_file` function reads each of the physical object numbers in the schedule structure and extracts its corresponding color number from the `surf.color` structure from within WALKTHRU. This is an integer number corresponding to the color table entry defining the objects color. This information is assigned to the ninth element in the schedule structure. The `get_file` function is now finished processing.

The schedule structure contains the object name, early start date, early finish date, late start date, late finish date, actual start date, actual finish date, physical object number, and object color for each object in the WALKTHRU model file for a construction project. This information is used by the Visual Simulator to simulate the construction sequence. Figure 6.6 shows the information contained in a typical schedule structure for a construction project.

The Schedule structure is defined within the Visual Simulator's source code as follows:

```
struct schedule
{
    char obj_name[16]; /* Defines the name of the object */
    int e_start; /* Defines the objects early start date */
    int e_finish; /* Defines the objects early finish date */
    int l_start; /* Defines the objects late start date */
    int l_finish; /* Defines the objects late finish date */
    int a_start; /* Defines the objects actual start date */
    int a_finish; /* Defines the objects actual finish date */
    int walk_obj; /* Defines the objects physical object number */
    int color; /* Defines the objects color */
};
```

obj_name	e_start	e_finish	l_start	l_finish	a_start	a_finish	walk_obj	color
WEST_BARREL	39	52	39	59	39	54	1	3
W_WARP_WALL1	40	55	40	55	40	55	11	5
W_WARP_WALL2	40	63	40	63	40	62	12	5
EAST_BARREL	41	62	55	69	41	65	3	3
E_WARP_WALL3	42	71	56	71	50	70	14	5
E_WARP_WALL4	42	79	56	79	49	80	15	5
CENTER_BARREL	43	72	67	79	50	74	2	3
W_HEAD_WALL	53	56	72	75	66	71	16	7
W_STIFF1	53	56	72	75	64	70	17	10
W_STIFF2	53	56	72	75	65	69	18	10
W_STIFF3	53	56	72	75	68	71	19	10
E_HEAD_WALL	63	66	76	79	70	76	20	7
E_STIFF1	63	66	76	79	75	79	21	10
E_STIFF2	63	66	76	79	74	79	22	10
E_STIFF3	63	66	76	79	76	80	23	10
W_WARP_SLAB	64	66	93	95	72	77	10	12
E_TRANS_WALL3	80	83	92	95	89	94	7	4
E_TRANS_WALL4	80	83	92	95	91	95	9	4
E_WARP_SLAB	80	82	93	95	93	95	13	8
E_TRANS_SLAB	84	86	96	98	88	93	8	8
W_TRANS_WALL1	92	95	92	95	94	98	4	4
W_TRANS_WALL2	92	95	92	95	94	99	6	4
W_TRANS_SLAB	96	98	96	98	97	99	5	8

This information is retrieved from the model file containing the object schedule.

This information is retrieved from within the WALKTHRU system

This information is contained in a variable within the Visual Simulator. This variable is defined as follows:

```
struct schedule obj_dates[200];
```

This information represents the first 23 records in this array.

This information is retrieved from to Object Name File.

Figure 6.6: Example of information contained in the schedule structure

### ***6.2.3 The "Time -" Function***

The second item on the Visual Simulators main menu allows the user to define the simulation time between construction working days. To access the `get_time` function the user must place the cursor over this menu item. The default time value is 5. This corresponds to 1.25 seconds between working days (for a complete description of this feature see Section 5.3.2). In order to increase the time value, the user must press the right mouse button. The time value can be increased to a maximum value of 20. To decrease the time value, the user must press the left mouse button. The time value can be decreased to a minimum value of 0. If the time value is 0 then the simulation speed depends on the speed of the computer's processor.

This information is passed to the Visual Simulator's `simulate` function. The `simulate` function uses the time value to pause between processing information between simulation days. This causes the simulation process to proceed slowly or quickly depending on the time value given.

### ***6.2.4 The "Early Dates/Late Dates/Early & Late Dates" Function***

Defining the logic option is the third choice on the Visual Simulators main menu. To access the `get_logic` function, the user must place the cursor over this menu item. The default logic option is "Early Dates". To toggle through the available choices the user must press the right mouse button. This will change the logic option from "Early Dates" to "Late

Dates" to "Early/Late Dates".

The logic options are defined as a pointer to an array of three character strings called `logic_opts[3]`. This array is defined as follows:

1. `logic_opts[0]` = Early Dates
2. `logic_opts[1]` = Late Dates
3. `logic_opts[2]` = Early/Late Dates

Therefore, when the logic option choice is selected, the array value 0, 1, or 2 defines the logic option. This value is passed to the Visual Simulators simulation function for processing.

#### ***6.2.5 The "Planned/Actual/Planned & Actual" Function***

Defining the view option is the fourth choice on the Visual Simulators main menu. To access the `get_view` function the user must place the cursor over this menu item. The default view option is "Early Dates". To toggle through the available choices the user must press the right mouse button. This will change the view option from "Planned" to "Actual" to "Planned & Actual".

The view options are defined as a pointer to an array of three character strings called `view_opts[3]`. This array is defined as follows:



1. view\_opts[0] = Planned
2. view\_opts[1] = Actual
3. view\_opts[2] = Planned & Actual

Therefore, when the view option choice is selected, the array value 0, 1, or 2 defines the view option. This value is passed to the Visual Simulators simulation function for processing.

### ***6.2.6 The "Start Date -" and "Finish Date -" Functions***

The fifth and sixth items on the Visual Simulators main menu allows the user to define the simulation start date and simulation finish date. The Visual Simulator accesses the get\_start\_date and get\_finish\_date functions when the user places the cursor over the "Start Date -" and "Finish Date -" menu items. The start and finish dates are initially defined by the construction start date defined in the get\_file function (see Section 6.2.2.2).

To increase the simulation start and finish dates, the user positions the cursor over the appropriate menu item and presses the right mouse button. The system allows the user to increase the start and finish date up to 700 working days from the construction start date. To decrease the dates the user presses the left mouse button. To key in a date, the user presses the middle mouse button. The system prompts the user to enter either the start or finish date. This date must be entered using the mm/dd/yy format. The system checks to make sure the date is valid and within the limits between the construction start date and 700 working days ahead of this date. If the date entered is not in the correct format, then the system warns the user and prompts for entry of a correct date.

## **6. VSS SYSTEM IMPLEMENTATION**

The `get_start_date` and `get_finish_date` functions do not allow the user to specify a simulation start date later than the simulation finish date. The `get_start_date` function checks the simulation start date each time it is increased. If it is later than the simulation finish date, the simulation finish date is incremented to the same value as the simulation start date. The `get_finish_date` functions checks the simulation finish date each time it is decreased. If it is earlier than the simulation start date, then the simulation start date is decreased to the same value as the simulation finish date.

Each date assigned to the simulation start and finish date is assigned its equivalent working day value. For example:

if the user defines:

Construction Start Date = 2/5/90

Simulation Start Date = 2/15/90

Simulation Finish Date = 2/26/90

then,

Construction Start Date = working day 0

Simulation Start Date = working day 8

Simulation Finish Date = working day 16

(saturdays and sundays do not count as working days)

The working day values for the simulation start and finish dates are passed to the `simulate` function for processing. The simulation function will simulate the construction sequence between working day 8 and working day 16.

## 6. VSS SYSTEM IMPLEMENTATION

### ***6.2.7 The "Execute" Function***

The seventh item on the Visual Simulators main menu allows the user to execute or put into motion visual simulation of the construction sequence. The Visual Simulator accesses the simulate function when the user places the cursor over the "Execute" menu item. If the user presses the left mouse button all objects in the WALKTHRU model of the construction project are turned off. The system uses a function defined within the WALKTHRU system to turn each object off. If the user presses the right mouse button the Visual Simulator will simulate the construction sequence, day-by-day, starting at the simulation start date and ending at the simulation finish date. This section describes how the simulate function executes visual simulation.

The simulate function processes the user defined information passed in from the functions previously described. This information includes the simulation time, logic option, view option, simulation start date, simulation finish date, and scheduling structure. The scheduling structure holds information for the object name, early start date, early finish date, late start date, late finish date, actual start date, actual finish date, physical object number, and color of each object in the WALKTHRU model. This information must be defined before the user attempts to execute visual simulation or an error will result.

The simulate function first uses the view option defined by the user to determine if 1) the primary model needs to be used to show either planned or actual construction progress, or 2) both primary and secondary models need to be used to show planned and actual construction progress at the same time. The simulate function then uses the logic option defined by the user to determine which dates to use from the schedule structure. Figure 6.7

illustrates the information passed into the simulate function.

The simulate function then assigns the color for each object, defined in the schedule structure, to a variable called color\_obj. This is necessary to preserve the objects original color. The simulate function will change an object from its original color to pink in order to show that the object is still under construction. When construction of the object is finished the simulate function changes the object back to its original color. This color is retrieved from the color\_obj variable.

Simulation begins once all of the previously discussed data has been processed. The simulate function checks the status of each object for each simulation day. The simulate function processes information concerning the primary model, for either planned or actual construction progress. When the user wants to view both planned and actual construction, then the simulate function calls the sim\_act function.

The sim\_act function acts exactly the same as the simulate function. The only difference is the simulate function processes the information concerning the logic option, view option and color assignment. This information is passed to the sim\_act function. It also processes only the information for the physical objects in the primary model. The sim\_act function only processes information for the physical objects in the secondary model. The sim\_act function only uses the actual start and finish dates associated with each object because this function is only used to show actual construction progress.

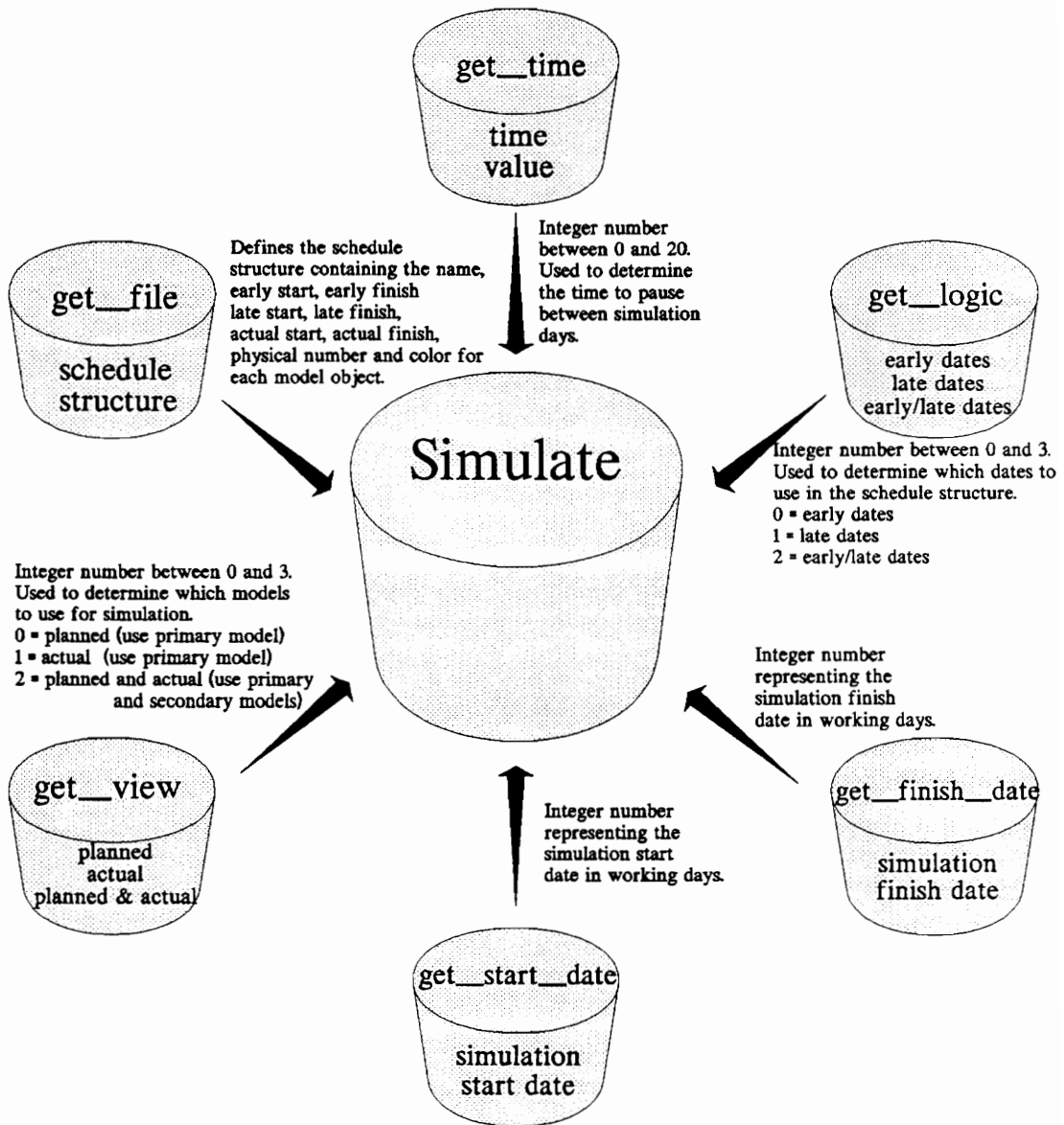


Figure 6.7: Information passed into the simulation function

For each simulation day, the system checks the status of all physical objects. The `simulate` and `sim_act` functions use the algorithm shown in Figure 6.8 to determine which objects to turn on and their color when displayed.

## Assign Dates

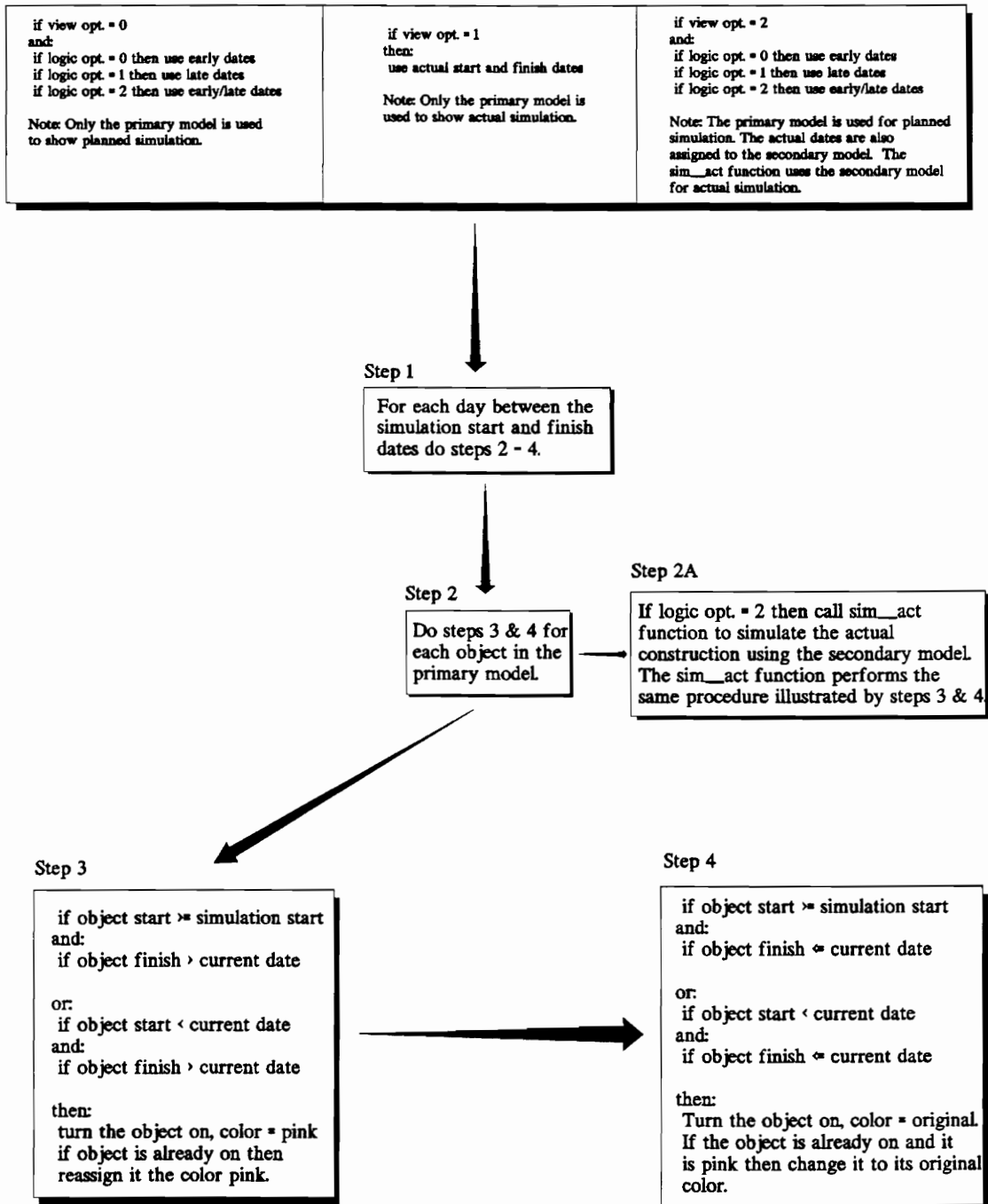


Figure 6.8: Simulation algorithm

## ***7. CONCLUSION***

This chapter reviews the VSS system and describes recommended uses and benefits in the construction industry. In addition, this chapter describes future extensions to the VSS system which can be implemented to make it a more comprehensive project management system. Finally, the last section provides conclusions of the work.

### ***7.1 Summary***

Use of computer based CPM technology has become a necessary means to help efficiently plan and control today's large and complex construction projects. Many problems exist however with how information, from these scheduling systems, is represented and interpreted. The power of this technology is, therefore, limited.

CAD systems are frequently used during the design phase of a construction project. After construction has begun, the 3D model generated by CAD systems has little use except for design changes. Construction personnel have little need for the 3D computer model other than to see how the final project will appear. Using CAD systems as only a design tool makes the CAD systems difficult to cost justify for many firms.

The Visual Schedule Simulation (VSS) system presented in this thesis was developed to



overcome the inadequacies with current planning, scheduling and controlling techniques. It also expands the use of CAD systems and the 3D computer models they produce. The VSS system links these two technologies by providing a tool which integrates traditional computer based scheduling techniques with 3D computer modeling technology. The result is a visual schedule simulation using a 3D computer model to represent every activity in the construction schedule.

The VSS system is made up of three distinct phases in order to produce visual simulation. Phase I is the Data Preprocessor. Phase II is the Database Manager. Phase III is the Visual simulator. The following paragraphs briefly describe these phases.

Phase I is the Data Preprocessor. The Data Preprocessor consists of preparing the CPM schedule and 3D computer model for a construction project. The CPM schedule is generated using traditional computer based techniques. The 3D computer model is created using a CAD system and traditional modeling techniques.

Phase II is the Database Manager. The Database Manager maps the activities in the CPM schedule to each object in the 3D computer model. The Database Manager uses a Relational Database Management System to manipulate data contained in a CPM scheduling file and 3D computer model information input by the user. The Database Manager allows the user to define logical relationships between the scheduled activities, and objects in the 3D computer model for a construction project. The result is an ASCII file containing the object schedule. This file defines start and finish dates for each 3D computer model object.

Phase III is the Visual Simulator. This is the heart of the VSS system. The Visual

## **7. CONCLUSION**

Simulator uses a computer based 3D simulation and visualization system. Using the 3D computer model of the construction project and the model file containing the object schedule created by the Database Manager, the Visual Simulator allows the user to view the construction sequence on a graphics display. The Visual Simulator also lets the user customize simulation. The user can choose to view planned or actual construction progress schedules. The VSS system can also show both planned and actual progress at the same time using two identical 3D computer models of the construction project. In each case of the planned schedule, the user can select to simulate early date scheduling logic, late date scheduling logic or early date and late date scheduling logic. The system also allows the user to visually simulate the construction schedule for any time period during the construction sequence.

The VSS system combines several different software and hardware technologies in order to make it a complete system. Sections 7.1.1. and 7.1.2. define the software and hardware used to create the VSS system.

### ***7.1.1 Computer Software***

The VSS system integrates the following software systems:

1. Primavera Project Planner was used as the project management system. This system was used to create the construction schedule.
2. CADAM was used as the 3D modeling system.
3. dBASE IV was used as the base system for the Database Manager.
4. The Database Manager was created using dBASE IV's programming

## ***7. CONCLUSION***

language.

5. WALKTHRU was used as the base system for the Visual Simulator.

WALKTHRU is a product of Bechtel Software Corporation.

6. "C" programming language was used to create the Visual Simulator.

### ***7.1.2 Computer Hardware***

Three different computer hardware platforms are necessary to support the VSS system's software requirements. The following is a list of these platforms.

1. An IBM PC or compatible with at least 512K bytes of memory and a 10M byte hard disk is necessary to run Primavera Project Planner, dBASE IV and the Database Manager. The PC must have a double-density, double-sided disk drive running PC or MS-DOS version 2.0 or greater.
2. An IBM 5080 graphics workstation runs the CADAM system.
3. A Silicon Graphics Iris 4D/80GT graphics superworkstation operates WALKTHRU and the Visual Simulator.

## ***7.2 Recommendations and Benefits***

The VSS system is a tool that can be used during all phases of a construction project. This section discusses these uses and describes how they are beneficial to total project management.

## ***7. CONCLUSION***

Traditional planning and scheduling techniques require that a construction project begin before planners can completely understand how it is impacted by the construction schedule. The VSS system alleviates this situation by allowing planners to quickly view the construction schedule using visual simulation. The planner can try many different scenarios and test them on a 3D computer model. This inevitably saves time and money by eliminating trial and error techniques once construction has begun.

Construction projects frequently suffer from delays and cost overruns. This is the result of inadequate or nonexistent controlling techniques. Current methods for controlling the construction process are laborious and often ineffective. The VSS system provides a convenient alternative giving project managers a superior method for managing construction activities. The VSS system allows for a visual comparison between scheduled and actual construction activities. Project managers can better communicate scheduling information to superintendents, foreman, and laborers. They will be more willing to "watch" a visual schedule simulation on a graphics display rather than try to understand lengthy textual and tabular reports, and symbolic networks.

The VSS system provides many other uses and benefits to the construction industry. The following list summarizes its advantages over traditional techniques for project management.

1. Construction personnel will be able to make better decisions without relying on instincts and opinions.
2. Planners will be able to better detect scheduling errors, and designers can detect design flaws and constructability problems before construction

## **7. CONCLUSION**

begins.

3. Project management personnel can view construction activities using simulation and 3D computer models, thus eliminating costly physical models.
4. Construction workers will be able to understand the construction schedule better allowing them to visualize how each activity relates to one another. This will significantly reduce communication and conflicting interpretation problems caused by traditional scheduling methods.
5. Project managers and planners can better determine equipment positioning and manpower congestion by being able to visualize problem areas.
6. Planners and designers can construct many "what-if" scenarios, thus eliminating costly trial and error techniques.
7. Project management can determine precisely the state of the construction project. They can better visualize and compare planned and actual construction progress allowing for quick revisions to the schedule.
8. Visual simulation provides an excellent learning and training tool for novice planners and schedulers. They can gain experience in the office, thus eliminating errors in the field.
9. Visual schedule simulation provides an improved method for documenting construction activities. This can prove to be invaluable in a construction claim situation.

The uses and benefits that the VSS system provides are many. The VSS System can be used during all phases of a construction project by designers, planners, schedulers, project

## **7. CONCLUSION**

managers, and other construction personnel. The VSS System reduces or eliminates many problems enabling a construction project to be completed efficiently and profitably.

### ***7.3 Future Extensions***

The VSS system is a collection of several different systems combined to create a visual simulation of the planned and actual construction progress. This system, however is envisioned as the basis for a more comprehensive and independent project management system.

The current state of the VSS system requires four different software systems operating on three separate hardware platforms. The VSS system can be enhanced such that only one software system running on one hardware platform is required. This will provide a self sufficient system relying on nothing other than a 3D computer modeling system.

The future of the VSS system, as seen by the author, would consist of an internal project management system capable of interacting with a 3D computer model. The system would operate on a graphics superworkstation such as the Silicon Graphics Iris series or IBM RISC series platforms. This would enable the user to use the 3D computer model and other graphics to enhance every phase of the visual simulation process. The future VSS system will be termed VSS+ throughout the rest of this chapter.

The VSS+ system would use the 3D computer model and the graphics environment to not only simulate construction activities, but give other relevant information as well. The planner would be able to select any object in the 3D computer model using a mouse, and

## ***7. CONCLUSION***

input all information associated with that object. This information would include the activities associated with the object, including their start and finish dates, the resources associated with the object, and the cost information.

This would eliminate the use of an outside CPM processor as the VSS system would not only schedule the project but calculate resource usage and leveling. It would aid the estimator by defining all of the costs associated with each object as well as the entire project. There would not be a need to map activities with model objects as this is done automatically by the planner when creating the CPM schedule.

Planners would be able to better determine resource usage. The planner could assign the resources necessary to complete each object in the 3D computer model of the construction project. The planner could check to see if there are any conflicting situations by simulating the construction sequence. The VSS+ system would provide graphical output showing the planner target resource consumption vs. actual resource consumption for the planned schedule. The planner would be able to determine conflicting situations and adjust the schedule so that resource usage could be optimized.

The VSS+ system would also provide an excellent tool for controlling the construction project. The simulation phase would not only allow the user to compare actual and planned construction by viewing a 3D computer model, it would display all other relevant data. For each day during the simulation process the VSS system would be able to display resource usage, construction costs, manpower reports, and other information. This can be accomplished by showing graphs which can be updated each day during the simulation process. These graphs can represent information for the total project or any object or

## **7. CONCLUSION**

activity the user selects. It will also include reports defining the state of the construction project. The user will be able to better determine which phases of the project are ahead or behind schedule and which phases are over or under budgeted costs.

The VSS+ system relies heavily on 3D computer models and graphical diagrams to communicate with the user. This is intended to augment lengthy textual and tabular reports currently used. It will also provide a more comprehensive understanding of the planned and actual construction project. The VSS+ system will enhance all phases of the construction project, from planning and estimating the project before it has begun to controlling the project during and after construction.

#### ***7.4 Conclusion***

Greater emphasis must be placed on planning, scheduling, and controlling the construction process due to the size and complexity of today's projects. Current methods are limited, because they fail to effectively present information to the user. The ineffective information presentation causes many firms to resist using computer based systems, forcing them to use simpler and less efficient methods.

The rapid advancement of computer technology makes it possible to present information to the user using new methods. The VSS system was developed to utilize new technologies to provide the construction industry with an alternative method for planning and controlling construction activities.

The VSS system provides construction personnel a tool that allows for greater

#### ***7. CONCLUSION***



understanding and awareness of the construction process. This system reduces the problems inherent with traditional techniques and allows the construction industry to move forward and keep up with current technology. It supplies the construction industry with a new instrument for planning, scheduling and controlling the construction process.

3D computer modeling and planning are both commonly used during the design and construction phases of a project. The VSS system integrates the technologies providing greater benefit than those which can be realized when applying CAD and construction planning applications separately.

The VSS system has many uses and benefits to the construction industry. It permits Architecture, Engineering, and Construction firms a means to cost justify the use of CAD systems and computer based planning systems. This results in optimizing the construction process. Construction personnel will have confidence that the work can be completed according to plan with a minimum amount of errors and rework. Projects should more likely be completed more efficiently, within time constraints, and within budget.

## ***A. USER'S GUIDE***

This guide provides step by step instructions on how to operate the Visual Schedule Simulation (VSS) system. The User's Guide is separated into three stages.

Stage 1 discusses Phase I of the VSS system, the Data Preprocessor. Stage 1 includes instructions for, 1) exporting the CPM schedule information from Primavera Project Planner, 2) naming each object in the 3D computer model of the construction project, and 3) converting the 3D computer model into a format compatible with the WALKTHRU system.

Stage 2 discusses Phase II of the VSS system, the Database Manager. Stage 2 gives the user instructions on how to operate the Database Manager from within the dBASE IV environment. This stage assumes the user has a working knowledge of how to operate the dBASE IV system. At a minimum the user must be able to access the "dot prompt" from the dBASE IV control panel.

Stage 3 discusses Phase III of the VSS system, the Visual Simulator. Stage 3 gives the user instructions on how to operate the Visual Simulator from within the WALKTHRU environment. This stage assumes the user has a working knowledge of how to operate the WALKTHRU system.

This User's Guide provides the user with complete instructions on how to operate the VSS system. The user may want to review the relevant sections in Chapter's 5 and 6 when reading this guide. These chapters provide a detailed discussion on how the VSS system operates.

The VSS system requires files to be transferred between different computer platforms. In order to transfer these files the user must either:

- Down load the file onto a tape or disk and then upload the file onto the correct file.
- Transfer the file via the ethernet link.

### ***A.1 Stage 1***

Stage 1 contains three sections. The first section of Stage 1 instructs the user on how to export schedule information from Primavera Project Planner. This section assumes the user knows how to operate and has already created a CPM schedule of the construction project using Primavera. The second section of Stage 1 instructs the user on how to name each object in the 3D computer model of the construction project. This section assumes the user has access to the 3D computer model created on a 3D computer modeling system. The third section of Stage 1 instructs the user on how to create a WALKTHRU model file capable of interacting with the WALKTHRU system (see Sections 5.2.2 and 6.2.1). The third section assumes the user has a working knowledge of both the 3D computer modeling system used to create the 3D computer model and the WALKTHRU system.

### ***A.1.1 Exporting the CPM Schedule***

The VSS system requires that the user creates or has access to a CPM schedule of the construction project. This CPM schedule must be created using Primavera Project Planner. The CPM schedule must contain, at a minimum, the following information for each activity:

1. Activity ID
2. Early Start Date
3. Early Finish Date
4. Late Start Date
5. Late Finish Date
6. Actual Start Date
7. Actual Finish Date

This activity information must be exported from Primavera Project Planner into a .dbf file format (see Section 5.2.1). This can be accomplished by using the following instructions:

1. Enter the Primavera Project Planner using normal procedures.

The main menu titled "P3 UTILITIES" should be displayed.

2. Select item #1, "SELECT an existing project".

At the bottom of the screen Primavera provides a prompt to "Enter project name:" A list of

all existing projects is displayed at the top of the "P3 UTILITIES" screen.

3. Key in the name of the project to use for visual simulation and press the "ENTER" key.

The "CONFIRM selection" screen should be displayed.

4. Select "Advance" from the commands line.

The "PROJECT DATA MENU" should now be displayed.

5. Select item #9, "Reports: Specification."

The "Types of Reports" menu should now be displayed.

6. Select item #6, "Export data files".

The "EXPORT DATA FILES" screen should now be displayed. This screen is illustrated in Figure A.1.

7. Select "Add" from the commands line.
8. Key in the reference number for the report and press the "Enter" key.
9. Key in the title for the report and press the "Enter" key.
10. Press the "End" key when the report's reference number and title have been correctly entered.

EXPORT DATA FILES				TEST	
Ref.No.:	EX-10	Title:	VSS Report		
CONTENT:	Current Schedule	Target 1 2	ASCII File Width 47	Current Schedule	Target 1 2
Activity ID	1	0	Resource	0	0
Title (No. of char 48)	0	0	Cost Account	0	0
Early Start	2	0	Res/Cost Percent	0	0
Early Finish	3	0	Res/Cost Lag & Dur.	0	0
Late Start	4	0	Units Per Day	0	0
Late Finish	5	0	Earned Value (Units)	0	0
Actual Start	6	0	Budget Quantity	0	0
Actual Finish	7	0	Actual Quantity/BCWS	0	0
Float Total/Free/Both T	0	0	Quantity to Complete	0	0
Calendar ID and Unit	0	0	Quantity at Completion	0	0
Original Duration	0	0	Earned Value (Cost)	0	0
Remaining Duration	0	0	Budget Cost	0	0
Percent Complete	0	0	Actual Cost/BCWS	0	0
ACT Code Field 1 - 1	0	0	Cost to Complete	0	0
Log Records 1 - 1	0		Cost at Completion	0	0
Commands: Add Delete Edit Help Level More Next Return Transfer Window eXecute					
Windows : Content Format List Order Selection					

Figure A1: The "EXPORT DATA FILES" Screen

11. Select "Window Contents" from the commands line.
12. Select "Edit" from the command line.

This allows the user to edit the information contained in the "EXPORT DATA FILES" screen. The user must edit the information under the "Current Schedule" field as follows:

<u>next to</u>	<u>enter</u>
Activity ID	1
Early Start	2
Early Finish	3
Late Start	4
Late Finish	5
Actual Start	6
Actual Finish	7

13. Press the "END" key when finished.

The "EXPORT DATA FILES" screen should now look similar to Figure A.1.

14. Select "Window Format" from the commands line.

The screen shown in Figure A.2 should now be displayed.

15. Select "Edit" from the commands line.

Ref.No.: EX-10 Title: VSS Report

FORMAT:

Enter name of output file [without extension]: VSSTEST

Type of output file to be created:

ASCII [.PRN] (A), dBASE III [.DBF] (D),  
 LOTUS 123 [.WK1] (L) or LOTUS 123 [.WKS] (S)? D

Title each column in .PRN, .WK1 and .WKS files (Y/N)? Y

Format dates as Calendar date or Work period or Both (C/W/B)? W  
 Summarize or Detail each activity's resource/cost data (S/D)? D  
 For actuals, export value for this Period, to Date, or Both (P/D/B)? D

For ASCII files, separate each field by 1 characters.  
 And insert 0 characters each time the content sequence number skips.

Relationships only, ignore Content window (Y/N)? N

Press Window eXecute

Figure A2: The "EXPORT DATA FILES" Screen (window/format)



Only the information concerning the output filename (field 1), the output file type (field 2) and the date format (field 4) needs to be edited. This information should be edited in the following manner:

- Key in the "filename" defining the output file and press the "Enter" key (this is field 1).
- Key in the letter "D" to define a .DBF file type and press the "Enter" key (this is field 2).
- Skip field 3.
- Key in the letter "W" to export date information in work period format (this is field 4).

The rest of the information contained in this screen is not relevant for this application.

16. Press the "END" key to terminate editing.

17. Select "eXecute" from the commands line.

A message will appear at the bottom of the screen prompting the user to either enter "P" to Print report criteria, "S" to Save report criteria, or "B" to Background print report criteria. Key in one of these three choices and press the "ENTER" key.

After step 17 has been completed execution begins. The Primavera Project Planner creates a .dbf file according to the parameters which have just been specified.

When execution is complete, the user may exit the Primavera system using normal

procedures. A file named "filename.dbf" will reside in the directory running the Primavera system. This file contains the schedule information necessary to operate Phase II of the VSS system, the Database Manager. This file must be transferred to the computer system and directory running the Database Manager system.

### ***A.1.2 Naming 3D Computer Model Objects.***

It is important to name each object in the 3D computer model of the construction project. The names of each object are necessary in order to operate both Phase II of the VSS system, the Database Manager, and Phase III of the VSS system, the Visual Simulator.

Phase II, the Database Manager, uses the object names to allow the user to link or map the activities in the CPM schedule with the 3D computer model (see Sections 5.2.2, 5.3.2, 6.1.1, and A.2). Phase III, the Visual Simulator, uses the object names to determine which objects to turn on and off to produce a visual simulation of the construction sequence (see Sections 5.4.2, 6.2.2, and A.3).

During Phase I, the Data Preprocessor, the user must give names to each of the objects defining the 3D computer model of the construction project. The user should display the 3D computer model using the 3D computer modeling system on which the model was created. The user then needs to record the names for each object in the 3D computer model. This entails creating a name which best describes the object. The user must either write this name on a sheet of paper or use some other method convenient for recording the object's name.

The object name must be limited to 15 characters and must begin with an alphabetic character (a-z). Names must be given to every object in the 3D computer model. It is important to record the exact names for each object as this information will be entered manually, by the user, into both Phase II, the Database Manager and Phase III, the Visual Simulator (see Sections A.2 and A.3 respectively). If the "exact" object names are not entered correctly into either the Database Manager or the Visual Simulator an error will result.

### ***A.1.3 Creating the WALKTHRU Model File***

Phase III of the VSS system, the Visual Simulator, uses the WALKTHRU system. Therefore, the 3D computer model must be transferred to the platform running the WALKTHRU system. The 3D computer model must be developed on a 3D CAD system that WALKTHRU supports. WALKTHRU currently supports CADAM, CATIA, Intergraph's Microstation32, Bechtel's 3DM, and system's which can produce an IGES data file. The 3D computer models can be transferred to the WALKTHRU system and converted to WALKTHRU model files in the following manner:

**CADAM and CATIA Models** - CADAM and CATIA models must first be translated into a WALKTHRU ASCII file format. This can be accomplished by using either the CADAM to WALKTHRU interface or the CATIA to WALKTHRU interface. The WALKTHRU ASCII file must then be transferred to the platform running the WALKTHRU system. This WALKTHRU ASCII file can then be converted into a WALKTHRU model file by using the WALKPRE program (see section 6.2.1.3).

**Microstation32 Models** - Microstation32 files must first be translated into an Intergraph design file. The Intergraph design file must then be transferred to the platform running the WALKTHRU system. The Intergraph design file can then be converted into a WALKTHRU model file by using the WALKIGDS program (see Section 6.2.1.1).

**3DM Models** - 3DM files must first be translated into a 3DM design file. The 3DM design file must then be transferred to the platform running the WALKTHRU system. The 3DM design file can then be converted into a WALKTHRU model file by using the WALK3DM program (see Section 6.2.1.1).

**Other Models** - Models created using a CAD system other than the ones previously discussed must be converted into an IGES data file. The IGES data file must then be transferred to the platform running the WALKTHRU system. The IGES data file can then be converted into a WALKTHRU model file by using the WALKIGES program (see Section 6.2.1.2).

The user should refer to CAD systems reference guide if they are unsure of how to convert the 3D computer model into one of the file formats compatible with the WALKTHRU system. Section 6.2.1 of this thesis and Chapter 2 of the WALKTHRU User's Guide describes converting CAD files into WALKTHRU model files in greater detail.

## ***A.2 Stage 2***

Stage 2 of the User's Guide provides step by step instructions on how to operate Phase II of the VSS system, the Database Manager. The Database manager allows the user to link or map the activities in the CPM schedule with the 3D computer model objects for a construction project. The Database Manager uses the relationships between activities and objects to compute start and finish dates for each object in the 3D computer model. The final result of the Database Manager is an object schedule in an ASCII file format which can be transferred to Phase III of the VSS system, the Visual Simulator. The Visual Simulator uses the object schedule to produce a simulation of the construction sequence.

The Database Manager produces the object schedule using four separate steps. These steps are defined as follows:

Step 1 - "Build Model File". This step allows the user to manually input the names for each object in the 3D computer model.

Step 2 - "Map Model Objects to Activities". This step allows the user to manually map the 3D computer model objects with the scheduled activities.

Step 3 - "Compute Object Dates". This step computes the object schedule for the construction project.

Step 4 - "Export dBASE File to ASCII File". This step creates an ASCII file from a user defined .dbf file.

To create a file containing the object schedule the user must follow these four steps in the order given.

The Database Manager is a user interactive system which allows the user to conveniently follow steps 1-4. In order to access the Database Manager system the user must do the following:

1. Enter the dBASE IV system.
2. Exit the "Control Panel" to the "Dot Prompt".
3. Key in the command "DO VSS".

The following screen will appear:

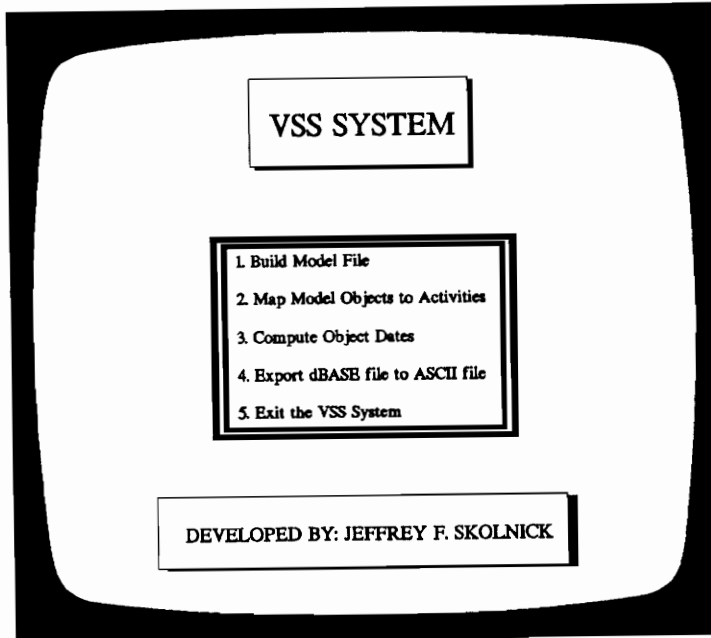


Figure A.3: The Database Manager's main menu

Menu items 1-4 allow the user to complete steps 1-4 respectively. Menu item 5 allows the user to exit the Database Manager. Select menu item 5 to leave the Database Manager system and return to the "Dot Prompt".

The following sections describe each of the steps and provide the user with instructions on how to operate the Database Manager.

### ***A.2.1 Step 1 - "Build Model File"***

Step 1 involves building the model file. This entails manually entering the names of each of the objects in the 3D computer model of the construction file. The model file will eventually be appended to include the start and finish dates (this is the object schedule) associated with each of the object names.

The user needs the list of object names previously recorded (see Section A.1.2). The following are step by step instructions for building the model file. Please note: When selecting menu items from the main menu the user can either, 1) use the up or down arrow keys to position the selection bar over the correct menu item, and press the "ENTER" key, or 2) press the number key defining the menu item.

1. Select the "Build Model File" menu item.

The "Build Model File" screen appears. This screen provides a prompt to "Enter the name of new model file".

2. Key in the name which defines the model file about to be created and press the "ENTER" key.

It is a good idea to define the model file name with text which will let the user know this file contains the object names. For example, if the project name is TEST, then give the model file the name "TEST\_O" to denote test objects. The file name is limited to 8 characters and the name must begin with a character. The file name is also limited to characters, numbers, and the underscore character "\_" or an error may result (press the escape (ESC) key to abort this process).

The Database Manager provides a prompt to enter the names of the 3D computer model objects after the model file name has been entered.

3. Key in the names of each object in the 3D computer model and press the "ENTER" key after each name has been correctly keyed in.

It is a good idea to use all capital letters when entering the object names). Object names must be limited to 15 characters and must begin with an alphabetic character (a-z).

4. Press the "ESC" key when all objects have been entered. This will terminate the "Build Model File" process and return the system to the main menu.

The model file now contains the names for each object in the 3D computer model. The



model file for a construction project must be built before the user can proceed to Step 2 - "Map Model Objects to Activities".

### ***A.2.2 Step 2 - "Map Model Objects to Activities"***

Step 2 involves mapping each object in the 3D computer model with its associated scheduled activity. The schedule file created using Primavera Project Planner must be transferred to the platform and directory running the Database Manager (see Section A.1.1).

The mapping process involves using the activity ID's defined in the schedule file and relating these activity ID's to each object name defined in the model file. The result is a mapping file defining the activity/object associations (see Sections 5.3.2 and 6.1.2).

The following provides step by step instructions for mapping 3D computer model objects with scheduled activities:

1. Select the "Map Model Objects to Activities" menu item.

The "MAP MODEL OBJECTS TO ACTIVITIES" screen should appear. This screen provides a prompt to "Enter name of schedule file to retrieve".

2. Key in the name of the schedule file and press the "ENTER" key.

This file should be the file exported from Primavera Project Planner (press the "ESC" key

to abort this process). The Database Manager provides a prompt to "Enter name of model file to retrieve" after the schedule file name has been entered.

3. Key in the name of the model file related to the construction project  
(press the "ESC" key to abort this process).

The Database Manager provides a prompt to "Enter the name to save mapping file" after the model file name has been entered.

4. Key in the name which defines the mapping file about to be created and  
press the "ENTER" key.

It is a good idea to define the mapping file name with text which will let the user know this file contains the mapping information. For example, if the project name is TEST, then give the mapping file the name "TEST\_M" to denote test mapping file. The file name is limited to 8 characters and the name must begin with a character. The file name is also limited to characters, numbers, and the underscore character "\_" or an error may result. (press the "ESC" key to abort this process).

The screen illustrated in Figure A.4 should now be displayed. The name of the first object in the model file appears near the top of this screen.

6. Key in the names of each activity ID associated with the object name  
displayed. Press the "ENTER" key after each activity ID has been keyed in.

**VSS SYSTEM**

OBJECT NAME:  
WEST\_BARREL

ACTIVITY ID:


MAP MODEL OBJECTS TO SCHEDULED ACTIVITIES

**CONTROLS**

Page Down Key -  
Moves to next object

Escape Key -  
Aborts this process

Figure A.4: The Map Model Objects to Activities Screen

The Database Manager limits the number of activities associated with each 3D computer model object to eight. When the eighth activity is entered the next object in the model file will be displayed. Step 6 should be repeated for the currently displayed object name.

If less than eight activities are associated with a 3D computer model object, the user can advance to the next object in the model file by:

7. Pressing the "Page Down" key.

#### 7A. Entering a blank space for an activity ID.

The next object name in the model file should now appear. The user should repeat steps 6 and 7 for each object name displayed. The mapping process is complete when the last activity ID for the last object name in the model file has been entered. The user can also abort the mapping process at any time by pressing the "ESC" key.

The main menu is displayed when the last activity ID has been entered for the last object name in the model file. The mapping file contains the object/activity associations defined by the user. If the user aborts the mapping process before the activities for the last object name have been entered, then the user must repeat the mapping process from step 1 in order to create a usable mapping file.

### ***A.2.3 Step 3 - "Compute Object Dates"***

Step 3 involves computing object dates. This entails using the mapping information between objects and activities to calculate the early start date, early finish date, late start date, late finish date, actual start date, and actual finish date for each object in the 3D computer model of the construction project.

The user must define the schedule file, model file, and mapping file related to the construction project. The compute object dates process defines the object start and finish dates by using the mapping information defined in the mapping file. The mapping information is used to determine which activity dates from the schedule file to assign to each object name in the model file. The result of this process is an appended model file defining the early, late, and actual start and finish dates for each object name (see Sections

5.3.3 and 6.1.3).

The following is a step by step procedure for computing the object schedule:

1. Select the "Compute Object Dates" item from the main menu.

The "COMPUTE OBJECT DATES" screen should appear. This screen provides a prompt to "Enter name of schedule file to retrieve".

2. Key in the name of the schedule file and press the "ENTER" key (press the "ESC" key to abort this process).

This file should be the file exported from Primavera Project Planner. The Database Manager provides a prompt to "Enter name of model file to retrieve" after the schedule file name has been entered.

3. Key in the name of the model file related to the construction project (press the "ESC" key to abort this process).

The Database Manager provides a prompt to "Enter the name of the mapping file to retrieve." after the model file name has been entered.

4. Key in the name of the mapping file related to the construction project. (press the "ESC" key to abort this process).

The Database Manager displays a message saying "Computing Object Schedule" after the mapping file name has been entered. This lets the user know that the object schedule is being created.

The main menu appears when the compute object dates process has been completed. The model file has been appended to include the objects early, late, and actual start and finish dates.

#### ***A.2.4 Step 4 - "Exporting dBASE File to ASCII File"***

Step 4 involves exporting the appended model file created in step 3 to an ASCII file format. The model file is currently in the .dbf file format. In order for Phase III of the VSS system, the Visual Simulator, to use the object schedule contained in the model file it must first be converted to an ASCII file format (see Sections 5.3.4 and 6.1.4).

The following is a step by step procedure for exporting the appended model file to an ASCII file format:

1. Select the "Export dBASE IV File to ASCII File" item from the main menu.

The Database Manager provides a prompt to "Enter name of file to export".

2. Key in the name of the appended model file to export and press the "ENTER" key.

An ASCII file is created from the appended model file after the file name has been entered. This file resides in the directory running the Database Manager system. This file has the same file name as the as the dBASE IV file from which it has been created. This file, however, has a .txt file extension. This file must be transferred to the platform and directory running Phase III of the VSS system, the Visual Simulator.

The user can exit the Database Manager by selecting menu item number 5 "Exit the VSS system" from the main menu. This returns the user to the dBASE IV system's "Dot Prompt". The user can exit the dBASE IV system by keying in "QUIT" and pressing the "ENTER" key.

### ***A.3 Stage 3***

Stage 3 provides step by step instructions on how to operate Phase III of the VSS system, the Visual Simulator. The Visual Simulator uses the WALKTHRU system to produce a visual simulation of the construction sequence.

The Visual Simulator uses the appended model file containing the object schedule developed in Stage 2. The Visual Simulator uses the object dates associated with each 3D computer model object to determine whether construction on that object has begun, is still in progress, or completed for each day during the simulation process.

Stage 3 is divided into 3 parts. Part 1 discusses preparing the 3D computer model so that it can interact with the WALKTHRU system and the Visual Simulator (the Visual Simulator

can be accessed from within the WALKTHRU system). Part 2 discusses creating the Object Name File. The Object Name File defines the names for each of the computer model object in the WALKTHRU model of the construction project. Part 3 describes how to access and use the Visual Simulator. Part 3 gives a step by step procedure to produce and customize a visual simulation of the construction sequence.

### ***A.3.1 Part 1 - Preparing the 3D Computer Model***

Part 1 involves preparing the 3D computer model so that it can interact with the WALKTHRU system and more specifically the Visual Simulator. The following provides step by step instructions for preparing the 3D computer model:

1. Transfer the model file from the CAD system on which the model was created to the platform and directory running the WALKTHRU system (see Section A.1.3).
2. Convert the model file to a WALKTHRU model file using the appropriate conversion program (see Sections 6.2.1 and A.1.3 of this thesis and Chapter 2 of the WALKTHRU User's Guide).
3. Using WALKTHRU's merge command, merge the same model file into the WALKTHRU model file created in the previous step (see Section 6.2.1.4 of this thesis and Chapter 2 of the WALKTHRU User's Guide).

Two identical copies of the 3D computer model for the construction project should now be included in the WALKTHRU model file. Both models, however, are positioned on top of each other. Therefore, in order to effectively use the Visual Simulator the second model



(secondary model) which has been merged into the WALKTHRU model file needs to be translated and positioned next to the first model (primary model).

4. Key the command "WALKTHRU 'filename'" and press the "ENTER" key.

The 'filename' is the name given to the WALKTHRU model file of the construction project. The primary and secondary 3D computer models of the construction project should now be displayed on the graphics display. It will appear, however, as if only one model is displayed. This is because both models have been converted into the same position within the WALKTHRU model file. The secondary model must therefore be translated and positioned next to the primary model.

5. Display WALKTHRU's main menu.
6. Position the cursor over the "Obj Dis" menu item and press the right mouse button.

The "Obj Dis" menu item accesses WALKTHRU's Object Display function. The Object Display menu will appear in the bottom left hand corner of the screen. The Object Display menu allows the user to turn objects off and on.

7. Turn off all objects in the primary model.

The secondary model should now be the only model displayed.

8. Using WALKTHRU's "Move Obj" (move object) function, translate each object in the secondary model.

The models should now be positioned next to each other as shown in Figure 6.4 of Chapter 6.

9. Record the object positions using WALKTHRU's record function (see Chapter 5 of the WALKTHRU User's Guide).

The objects positions should now be saved in a record and replay file. Each time the user displays the 3D computer model they can merely use WALKTHRU's replay function. This will automatically translate the secondary model to its desired position.

### ***A.3.2 Part 2 - Creating the Object Name File***

Part 2 involves creating an Object Name File. The Object Name File defines the names of each of the objects which exist in the WALKTHRU model file. The object names must be identical to those names which were used when building the model file (see Sections A.1.2 and A.2.1).

The Object Name File can be created using a text editor or any other method convenient to the user. The Object Name File must reside in the directory running the WALKTHRU system.

The Object Name File requires three fields of information for each object record in order to interact with the WALKTHRU system. Field #1 defines the objects virtual number, field

#2 defines the objects physical number, and field #3 defines the objects name.

The virtual and physical object numbers must be identical for each object. These numbers must be sequentially numbered, starting with 1 and there must be no gaps in the numbering sequence. The physical objects must exist in the WALKTHRU disk file or an error will result.

The object names must be limited to 15 characters and must begin with an alphabetic character (a-z). The object names must be identical to those in the model file created during Stage 2 or an error will result. Figure 6.6 in Chapter 6 shows an example of a typical Object Name File. See section 6.2.2.3 of this thesis and Chapter 2 of the WALKTHRU User's Guide for more details on Object Name Files.

### ***A.3.3 Part 3 - Operating the Visual Simulator***

Part 3 involves customizing and executing visual simulation of the construction sequence using the Visual Simulator. The Visual Simulator requires the following in order to operate:

1. The WALKTHRU model file created in Part 1 of Stage 3.
2. The Object Name File created in Part 2 of Stage 3.
3. The appended model file created in Part 4 of Stage 2.

The Visual Simulator can be accessed from within the WALKTHRU system. The user must follow the following step by step procedure in order to operate the Visual Simulator:

#### ***A. USER'S GUIDE***

1. Enter the WALKTHRU system and display the 3D computer model of the construction project using normal procedures.
2. Display WALKTHRU's main menu.
3. Position the primary and secondary models of the construction project next to each other using WALKTHRU's replay function.
4. Select the VSS item from WALKTHRU's main menu.

The Visual Simulator menu will appear in the bottom left hand corner of the screen. Figure A.5 illustrates the Visual Simulator's menu.

	VSS System
①	File -
②	Time - 5
③	Early Dates
④	Planned
⑤	Start Date - 1/1/90
⑥	Finish Date - 1/1/90
⑦	Execute
⑧	Exit

Figure A.5: The Visual Simulators Main Menu

The Visual Simulator's menu allows the user to interact with the Visual Simulator to customize and execute a visual simulation of the construction sequence. The user can interact with the menu using the mouse. The following sections provide instructions on how to interact with each menu item.

#### A. USER'S GUIDE

### ***A.3.3.1 Menu Item 1***

The first item on the Visual Simulators menu allows the user to define the Input File (this is the appended model file containing the object schedule), the project start date, and the Object Name File. This information must be associated with the 3D computer model of the construction project displayed or an error will result. The following provides step by step instructions for defining this information:

1. Position the cursor over the first menu item "File -" and press the RIGHT MOUSE button.

The Visual Simulator provides a prompt to "Enter the name of Input File:". This prompt appears on the command line.

2. Key in the complete name of the Input File and press the "ENTER" key.

The Input File is the appended model file containing the object schedule. The Visual Simulator provides a prompt to "Enter the project start date (mm/dd/yy):".

3. Key in the project start date using the mm/dd/yy format and press the "ENTER" key.

The Visual Simulator assumes the project start date is a monday and that the project has been scheduled using 5 day work weeks. The Visual Simulator provides a prompt to

"Enter the name of Object Name File:".

4. Key in the name of the Object Name File associated with the 3D computer model and press the "ENTER" key.

The user can now proceed to the next menu item after the input file, project start date, and Object Name File have been defined.

### ***A.3.3.2 Menu Item 2***

The second item on the Visual Simulator's menu allows the user to define the simulation time. The "Time -" menu item allows the user to adjust the time for visual simulation. The user may adjust the time from 0 to 20. These numbers represent the time, in quarter seconds, between each day during visual simulation. For example, if the user selects a time of 6, this represents  $6/4$  or 1.5 seconds between each day during visual simulation. If the user wants to simulate the construction sequence for 60 actual working days then simulation will last for 90 seconds.

The user must use the mouse to adjust the simulation time or they can accept the default value of 5. To adjust the simulation time the user must place the cursor over the "Time -" menu item and either:

- Press the "RIGHT MOUSE" button to increase the time value.
- Press the "LEFT MOUSE" button to decrease the time value.

### ***A.3.3.3 Menu Item 3***

The third item on the Visual Simulator's main menu allows the user to define the scheduling logic to use for visual simulation. The user can choose between:

- "Early Dates": The Visual Simulator will use the early start and early finish dates for each object during visual simulation of the planned construction sequence.
- "Late Dates": The Visual Simulator will use the late start and late finish dates for each object during visual simulation of the planned construction sequence.
- "Early/Late Dates": The Visual Simulator will use the early start and late finish dates for each object during visual simulation of the planned construction sequence.

The user can change the logic option by placing the cursor over the third menu item and pressing the "RIGHT MOUSE" button. This allows the user to toggle between the "Early Dates", "Late Dates" and "Early/Late Dates" choices. The default logic option is "Early Dates".

### ***A.3.3.4 Menu Item 4***

The fourth item on the Visual Simulator's main menu allows the user to define how to view

the schedule to use for visual simulation. The user can choose between the following view options:

- "Planned": The Visual Simulator will use the planned schedule during visual simulation. The primary model is used to show the planned schedule.
- "Actual": The Visual Simulator will use the actual schedule during visual simulation. The primary model is used to show the actual schedule.
- "Planned & Actual": The Visual Simulator will use both the planned and actual schedules during visual simulation. The primary model is used to show the planned schedule and the secondary model is used to show the actual schedule. Visual simulation occurs simultaneously.

The user can change the view option by placing the cursor over the fourth menu item and pressing the "RIGHT MOUSE" button. This allows the user to toggle between the "Planned", "Actual" and "Planned & Actual" choices. The default view option is "Planned".

#### ***A.3.3.5 Menu Item's 5 and 6***

The fifth and sixth items on the Visual Simulator's main menu allows the user to define the simulation start date and finish date respectively. The simulation start date and finish date define the time period during the construction project which the user wishes to visually simulate. The user may select any time period during the construction project. The Visual Simulator simulates only the construction activities which take place during this time



period.

The user can change the simulation start date and finish date by placing the cursor over either the "Start Date -" or "Finish Date -" menu item and pressing the following mouse button:

- Press the "RIGHT MOUSE" button to increase either the simulation start date or simulation finish date.
- Press the "LEFT MOUSE" button to decrease either the simulation start date or simulation finish date.
- Press the "MIDDLE MOUSE" button to key in either the simulation start date or simulation finish date.

If the "MIDDLE MOUSE" button has been pressed the Visual Simulator provides a prompt to "Enter the simulation start [or finish] date (mm/dd/yy):" on the command line.

- Key in the date in the mm/dd/yy format and press the "ENTER" key.

The simulation start date or simulation finish date will appear next to the appropriate menu item. The default simulation start and simulation finish date is the project start date.

### ***A.3.3.6 Menu Item 7***

The seventh menu item on the Visual Simulator's menu allows the user to execute visual simulation. Visual simulation proceeds day by day starting with the simulation start date

and ending at the simulation finish date.

The user can execute visual simulation by placing the cursor over the "Execute" menu item.

The user has the following mouse controls:

- Press the "LEFT MOUSE" button to turn off all objects shown on the graphics display.
- Press the "RIGHT MOUSE" button to simulate the construction sequence.

Visual simulation proceeds day by day when the "RIGHT MOUSE" button has been pressed. For each day during visual simulation the objects which begin on that day are turned on. The objects which are under construction are pink. The objects for which construction is complete are their original color. This lets the user determine which day construction on each object begins and ends.

### ***A.3.7 Menu Item 8***

The eighth item on the Visual Simulator's menu allows the user to exit the Visual Simulator. The user can place the cursor over the "EXIT" menu item and press the "RIGHT MOUSE" button. The Visual Simulator's menu will disappear.

### ***A.3.8 Quick Reference***

This section provides a "Quick Reference" for using the Visual Simulator.

#### **Menu Item 1**

Mouse controls:

RIGHT MOUSE - Activates this menu item. The Visual Simulator prompts the user to enter the Input File Name, the project start date, and the Object Name File.

#### **Menu Item 2**

Mouse controls:

RIGHT MOUSE - Increase the simulation time value.

LEFT MOUSE - Decrease the simulation time value.

#### **Menu Item 3**

Mouse controls:

RIGHT MOUSE - Toggle through the logic options "Early Dates", "Late Dates" and "Early/Late Dates".

**Menu Item 4**

Mouse controls:

RIGHT MOUSE - Toggle through the view options "Planned", "Actual", and "Planned & Actual".

**Menu Item 5**

Mouse controls:

RIGHT MOUSE - Increase the simulation start date.

LEFT MOUSE - Decrease the simulation start date.

MIDDLE MOUSE - Key in the simulation start date.

**Menu Item 6**

Mouse controls:

RIGHT MOUSE - Increase the simulation finish date.

LEFT MOUSE - Decrease the simulation finish date.

MIDDLE MOUSE - Key in the simulation finish date.

**Menu Item 7**

Mouse controls:

RIGHT MOUSE - Execute visual simulation.

**LEFT MOUSE** - Turn off all displayed objects.

### **Menu Item 8**

**Mouse controls:**

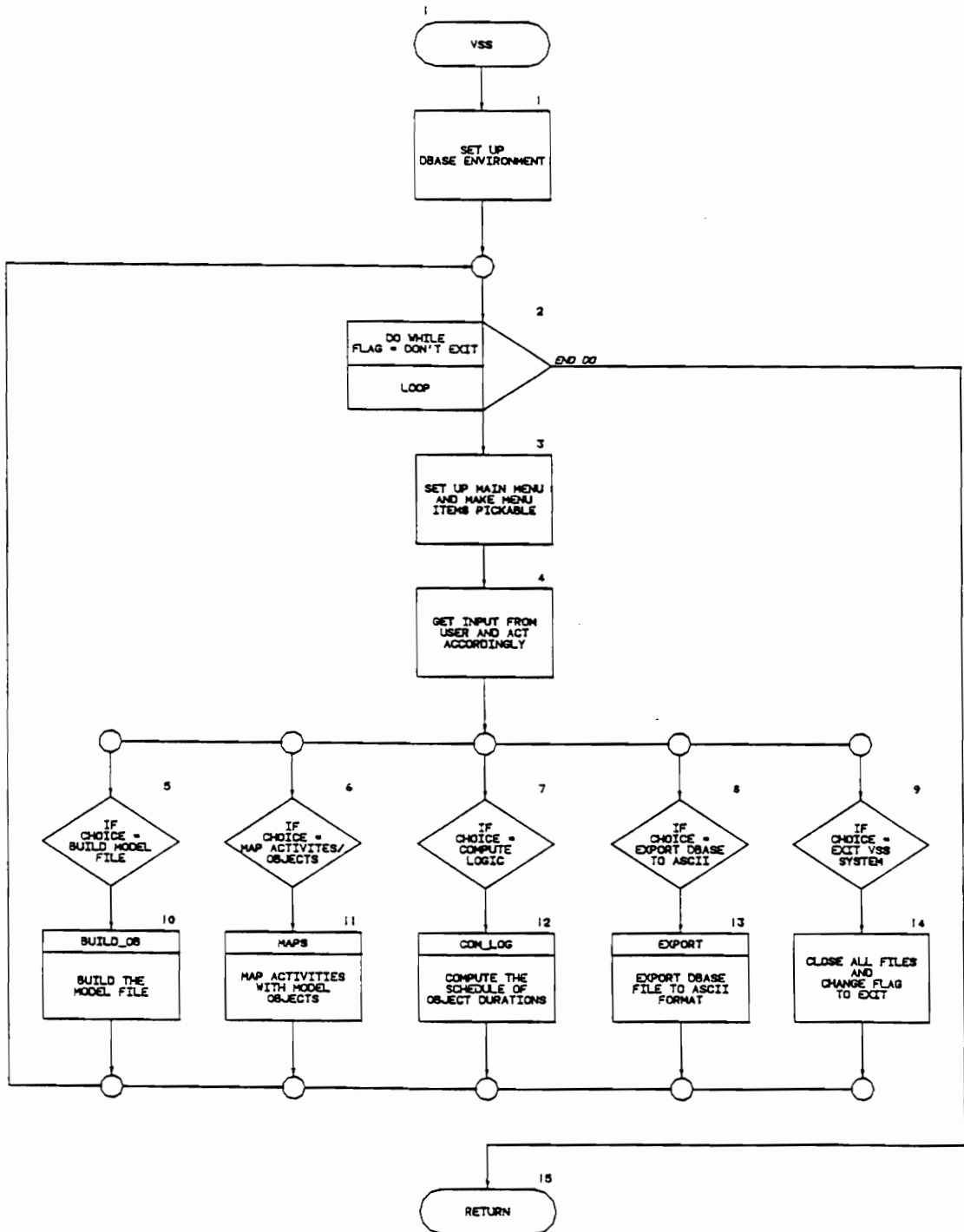
**RIGHT MOUSE** - Exit the Visual Simulator

## ***B. FLOWCHARTS***

The flowcharts for the VSS System have been provided for reference. Standard numbering techniques have been used to show proper flow of control for the design. Section B.1 contains the flow charts for Phase II of the VSS system, the Database Manager. Section B.2 contains the flow charts for Phase III of the VSS system, the Visual Simulator.

***B.1 The Database Manager - Phase II***

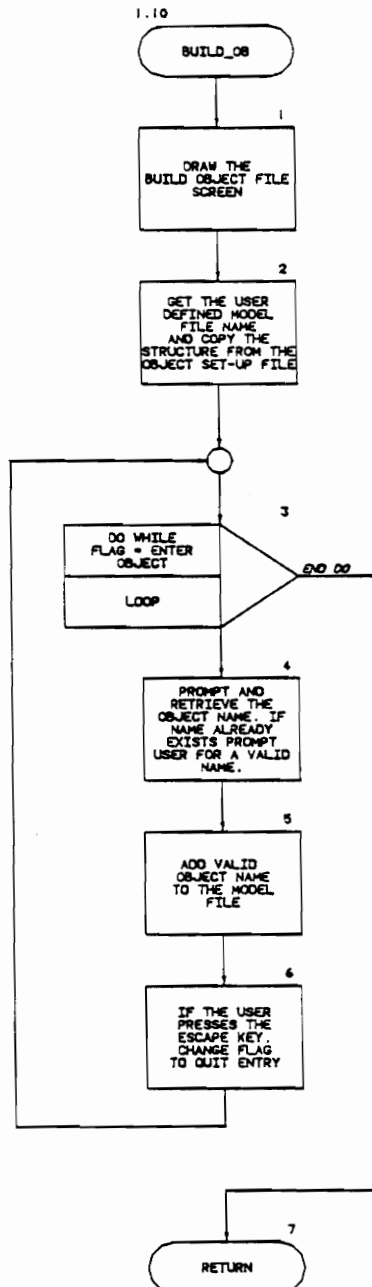
VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	VSS	MODULE: I
DESIGNED BY:	JEFF SKOLNICK	NOTE: PHASE II OF THE VSS SYSTEM - DATABASE MANAGER. ALLOWS USER TO INTERACT WITH THE SOFTWARE SYSTEM.
DATE:	7-22-90	



## B. FLOWCHARTS

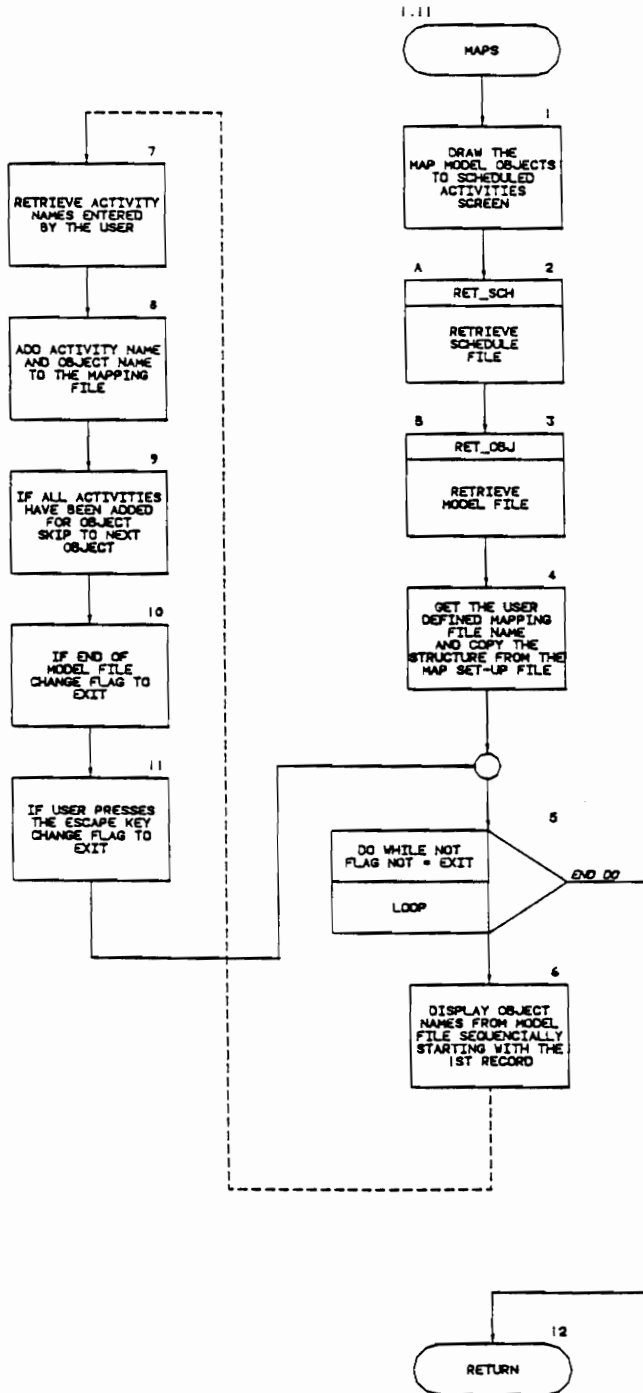


VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	BUILD_OB	MODULE: 1.10
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS PROCEDURE ALLOWS THE USER TO BUILD THE MODEL FILE.
DATE:	7-22-90	



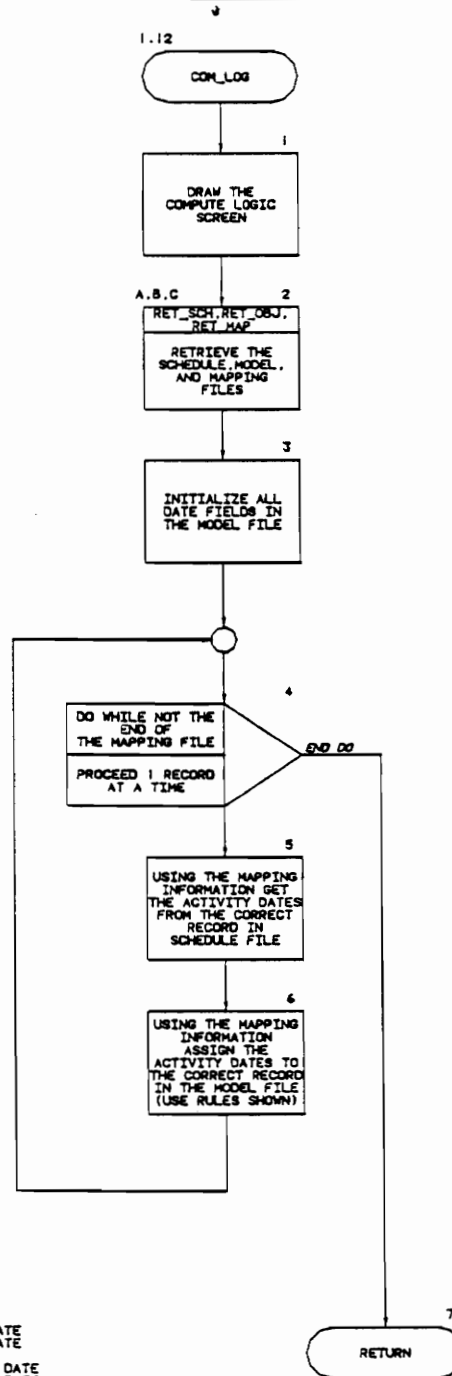
## B. FLOWCHARTS

VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	MAPS	MODULE: 1.11
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS PROCEDURE ALLOWS THE USER TO MAP SCHEDULED ACTIVITIES WITH 3D COMPUTER OBJECTS.
DATE:	7-22-90	



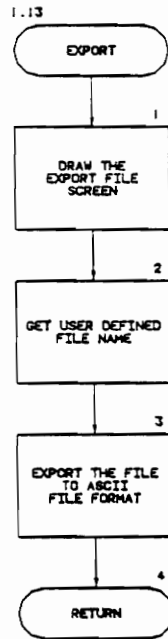
## B. FLOWCHARTS

VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	COM_LOG	MODULE: 1.12
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS PROCEDURE COMPUTES THE SCHEDULE OF OBJECT DURATIONS.
DATE:	7-22-90	

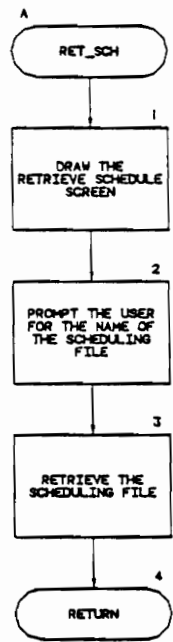


## B. FLOWCHARTS

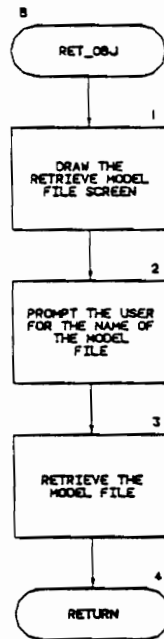
VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	EXPORT	MODULE: 1.13
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS PROCEDURE EXPORTS DBASE FILES TO ASCII FILE FORMAT.
DATE:	7-22-90	

**B. FLOWCHARTS**

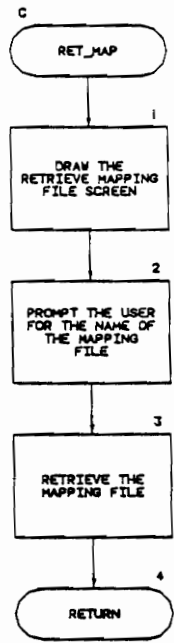
VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	RET_SCH	MODULE: A
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS PROCEDURE RETRIEVES A USER DEFINED SCHEDULING FILE.
DATE:	7-22-90	



VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME:	RET_OBJ	MODULE: B
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS PROCEDURE RETRIEVES A USER DEFINED MODEL FILE.
DATE:	7-22-90	



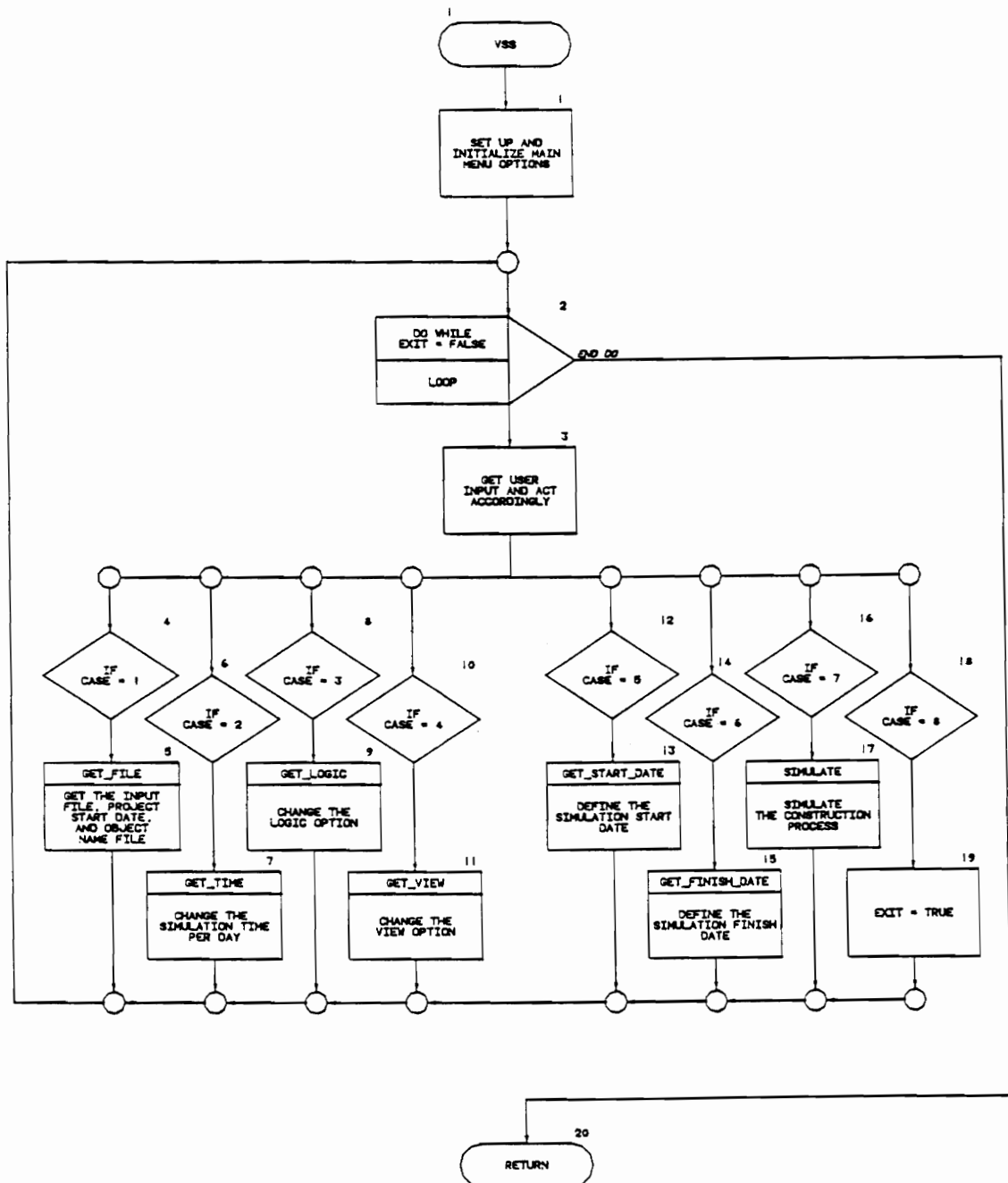
VT VSS SYSTEM - DATABASE MANAGER		
MODULE NAME :	RET_MAP	MODULE : C
DESIGNED BY :	JEFF SKOLNICK	NOTE : THIS PROCEDURE RETRIEVES A USER DEFINED MAPPING FILE.
DATE :	7-22-90	



## ***B.2 The Visual Simulator - Phase III***

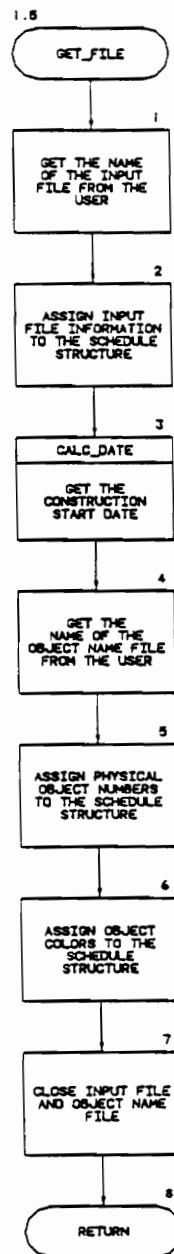


VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	VSS	MODULE: I
DESIGNED BY:	JEFF SKOLNICK	NOTE: PHASE III OF THE VSS SYSTEM - THE VISUAL SIMULATOR. THIS PROCEDURE SETS UP MAIN MENU AND ACTS ON USER INPUT.
DATE:	7-23-90	



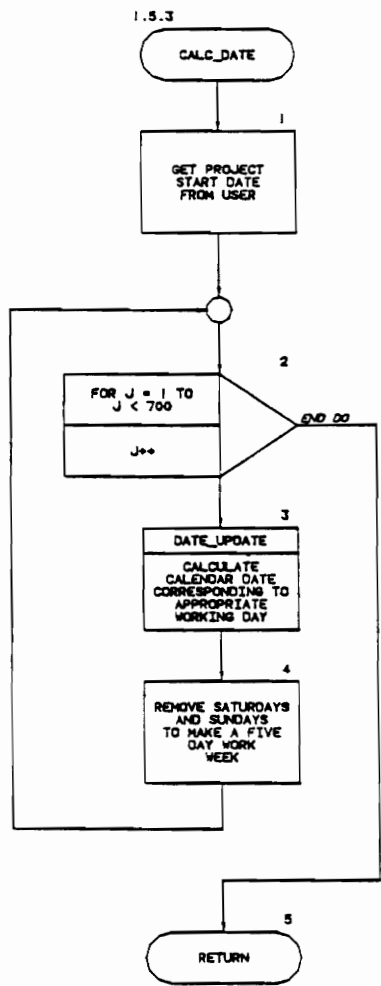
## B. FLOWCHARTS

VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	GET_FILE	MODULE: 1.5
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION PROMPTS THE USER TO DEFINE THE INPUT FILE, CONSTRUCTION START DATE, AND OBJECT NAME FILE.
DATE:	7-23-90	



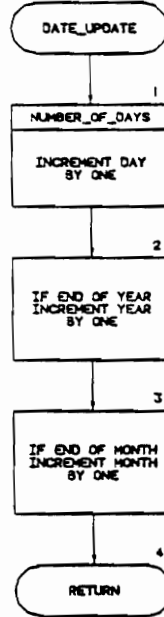
## B. FLOWCHARTS

VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	CALC_DATE	MODULE: 1.5.3
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION CALCULATES CALENDAR DATES CORRESPONDING TO WORKING DAYS.
DATE:	7-23-90	

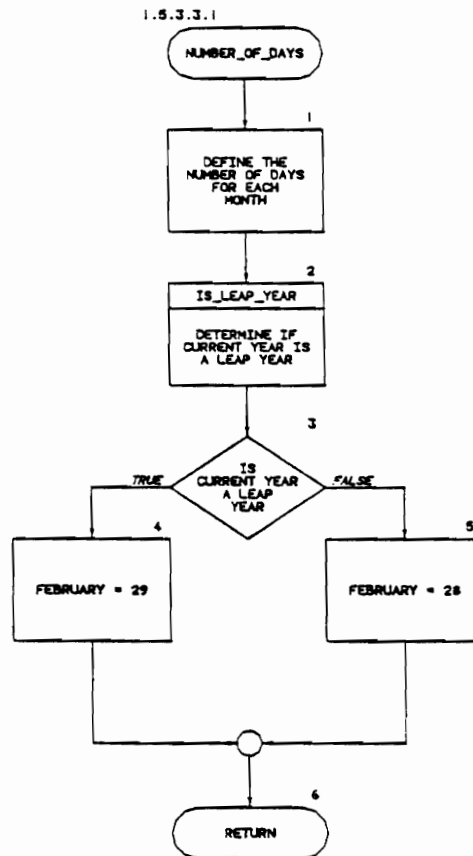


VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	DATE_UPDATE	MODULE: 1.5.3.3
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION CALCULATES THE NEXT CALENDAR DATE GIVEN THE CURRENT CALENDAR DATE.
DATE:	7-23-90	

1.5.3.3



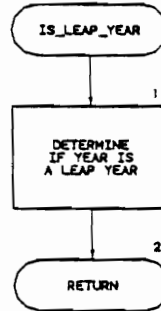
<b>VT</b> <i>VSS SYSTEM - VISUAL SIMULATOR</i>	
MODULE NAME: NUMBER_OF_DAYS	MODULE: 1.5.3.3.1
DESIGNED BY: JEFF SKOLNICK	NOTE: THIS FUNCTION DEFINES THE NUMBER OF DAYS FOR EACH MONTH.
DATE: 7-23-90	



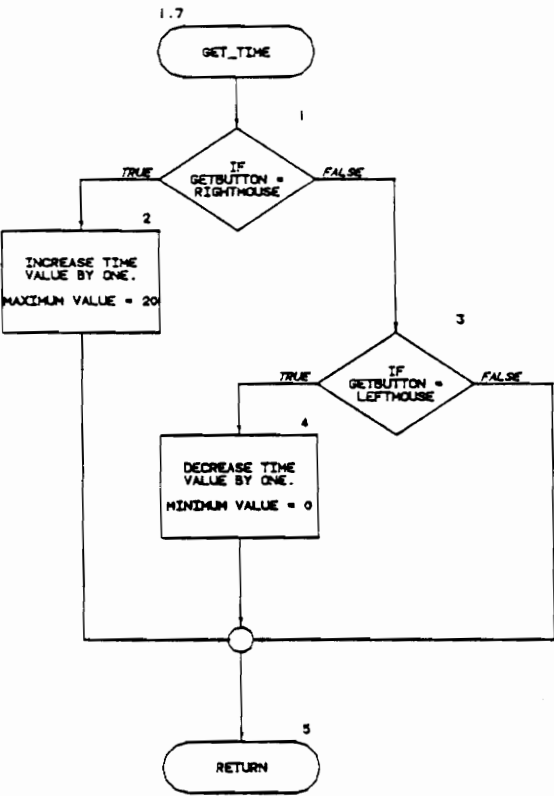
## B. FLOWCHARTS

VT VSS SYSTEM - VISUAL SIMULATOR	
MODULE NAME: IS_LEAP_YEAR	MODULE: 1.5.3.3.1.2
DESIGNED BY: JEFF SKOLNICK	NOTE: THIS FUNCTION DETERMINES IF A GIVEN YEAR IS A LEAP YEAR.
DATE: 7-23-90	

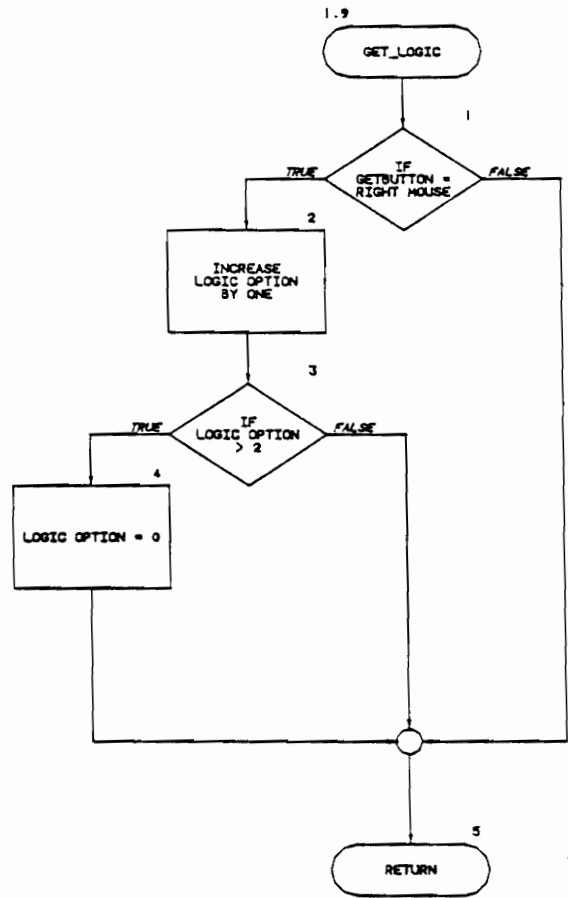
1.5.3.3.1.2

**B. FLOWCHARTS**

VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	GET_TIME	MODULE: 1.7
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION LETS THE USER INCREASE OR DECREASE THE SIMULATION TIME VALUE.
DATE:	7-23-90	

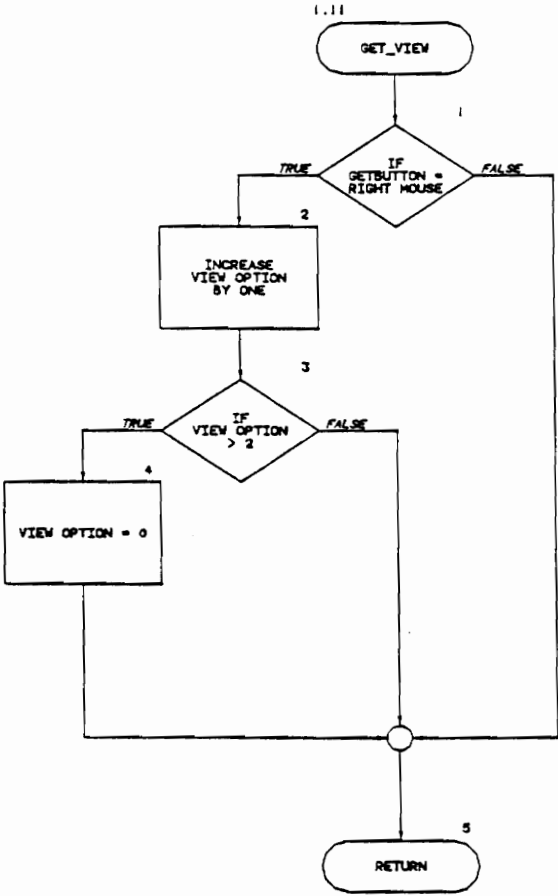


VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	GET_LOGIC	MODULE: 1.9
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION ALLOWS THE USER TO DEFINE THE LOGIC OPTION.
DATE:	7-23-90	

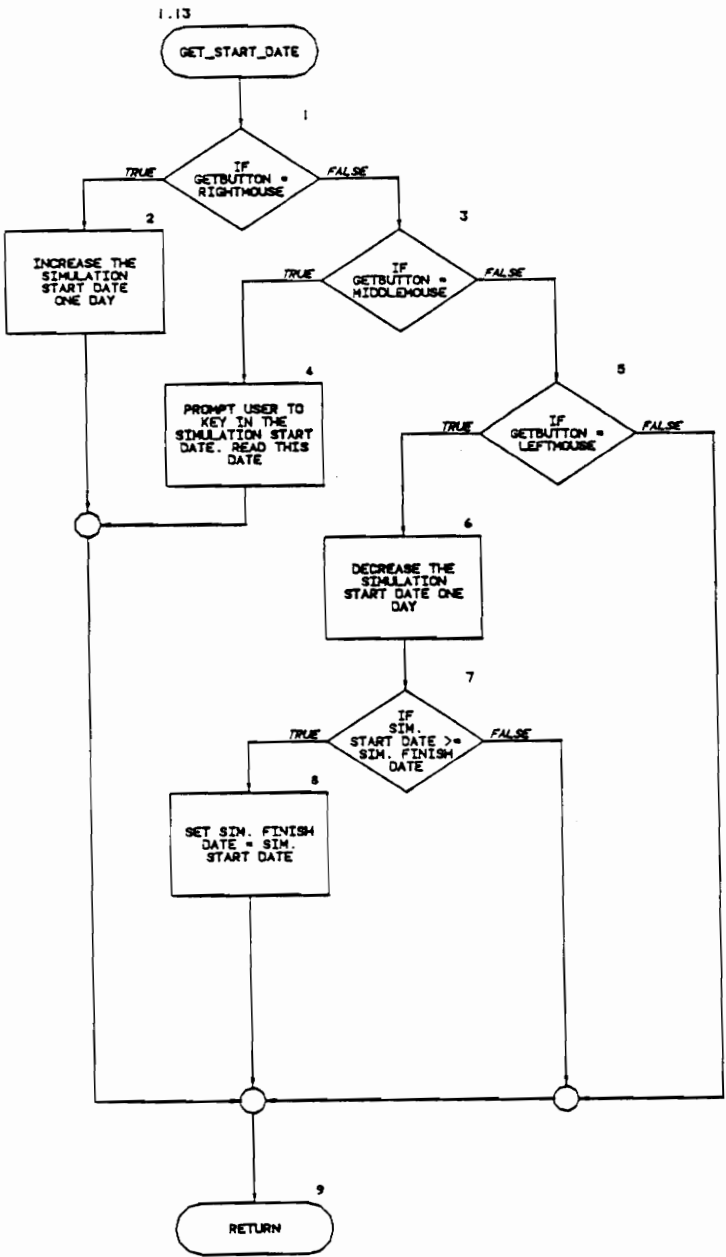




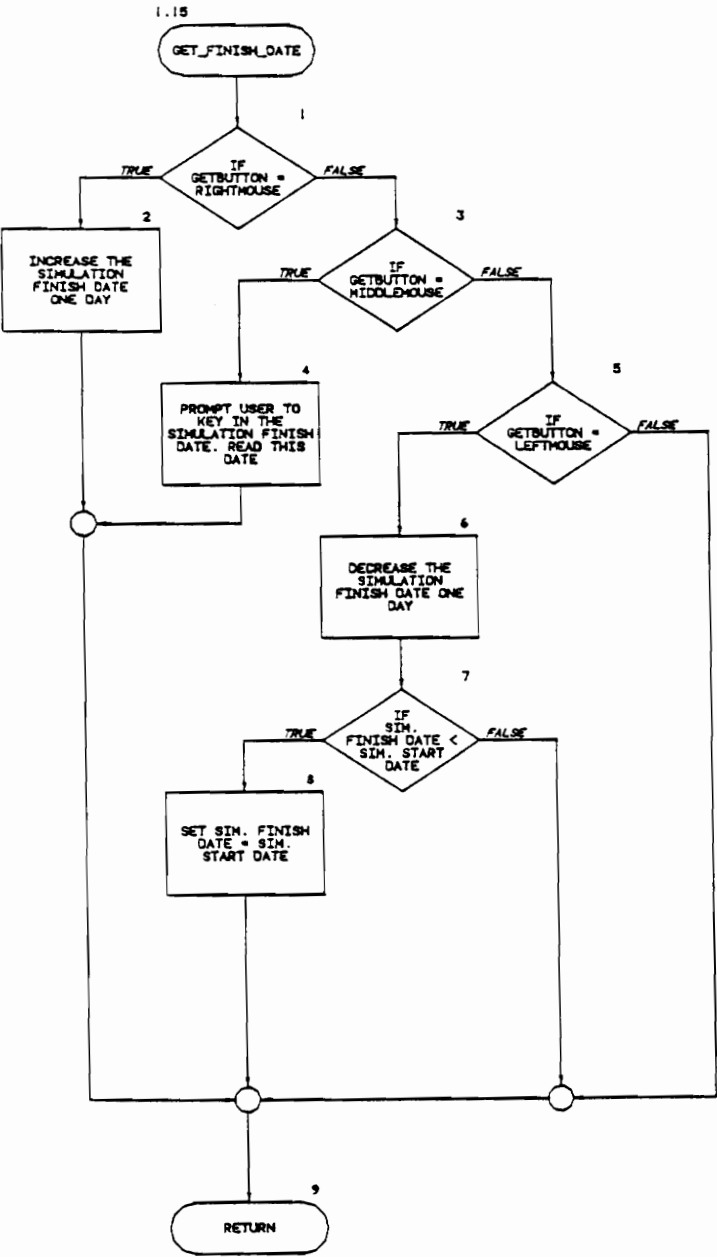
VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	GET_VIEW	MODULE: 1.11
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION ALLOWS THE USER TO DEFINE THE VIEW OPTION FOR SIMULATION.
DATE:	7-23-90	



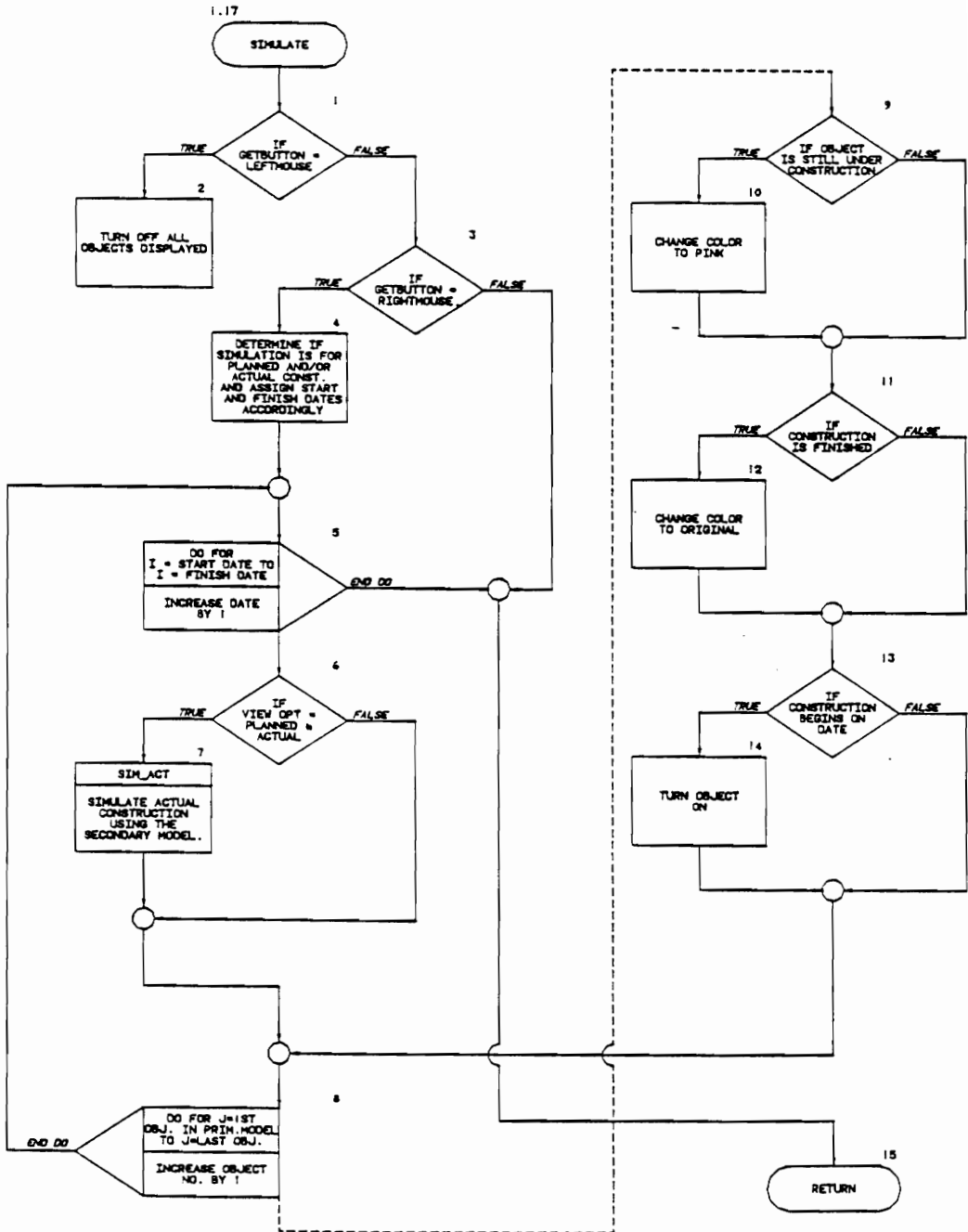
VT VSS SYSTEM - VISUAL SIMULATOR	
MODULE NAME: GET_START_DATE	MODULE: 1.13
DESIGNED BY: JEFF SKOLNICK	NOTE: THIS FUNCTION ALLOWS THE USER TO INCREASE OR DECREASE THE SIMULATION START DATE.
DATE: 7-23-90	



VT VSS SYSTEM - VISUAL SIMULATOR	
MODULE NAME: GET_FINISH_DATE	MODULE: 1.15
DESIGNED BY: JEFF SKOLNICK	NOTE: THIS FUNCTION ALLOWS THE USER TO INCREASE OR DECREASE THE SIMULATION FINISH DATE.
DATE: 7-23-90	

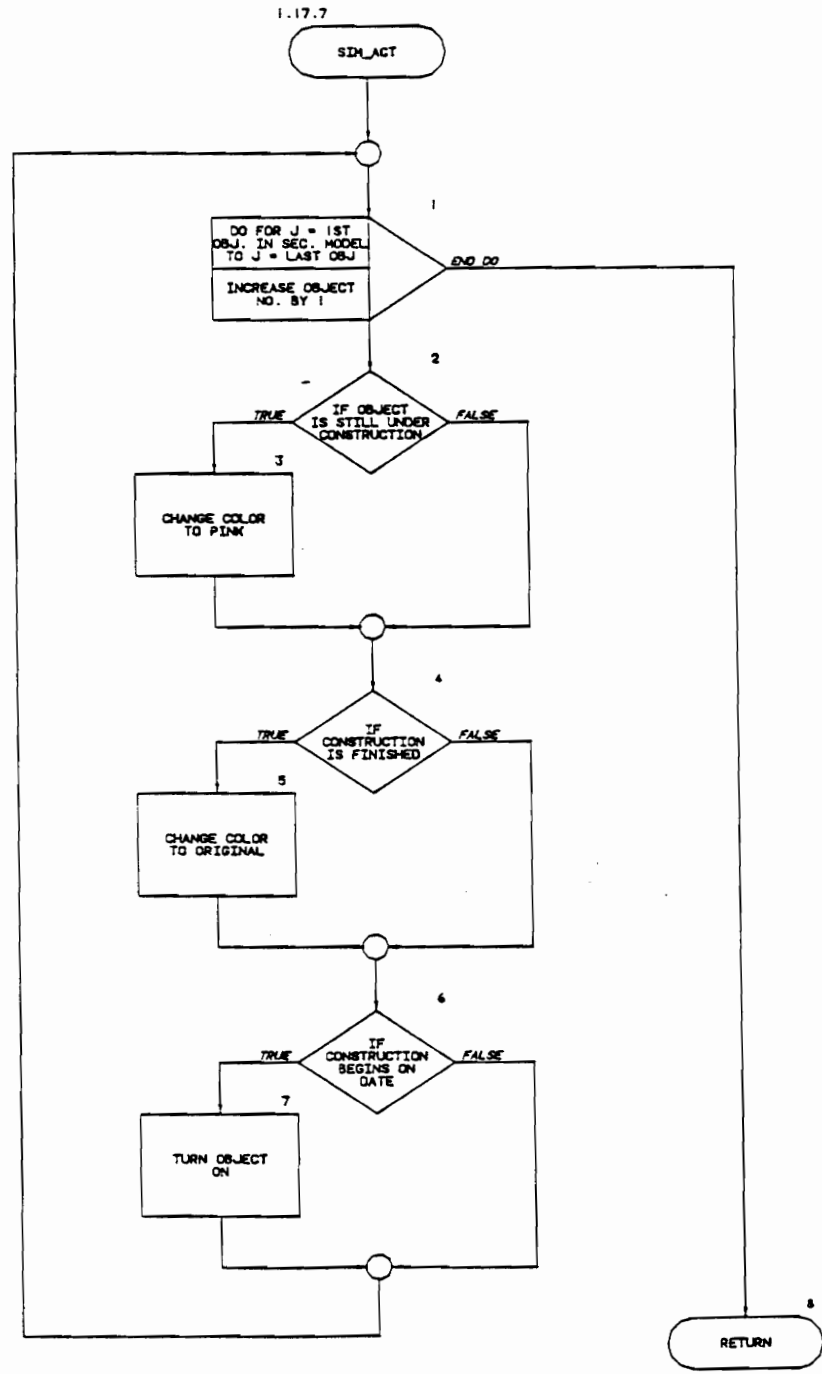


<b>VT</b> <i>VSS SYSTEM - VISUAL SIMULATOR</i>	
MODULE NAME: SIMULATE	MODULE: 1.17
DESIGNED BY: JEFF SKOLNICK	NOTE: THIS FUNCTION SIMULATES THE CONSTRUCTION PROCESS USING THE USER DEFINED PARAMETERS PASSED IN.
DATE: 7-23-90	



## B. FLOWCHARTS

VT VSS SYSTEM - VISUAL SIMULATOR		
MODULE NAME:	SIM_ACT	MODULE: 1.17.7
DESIGNED BY:	JEFF SKOLNICK	NOTE: THIS FUNCTION SIMULATES THE ACTUAL CONSTRUCTION PROCESS USING THE SECONDARY MODEL.
DATE:	7-23-90	



## ***C. Source Code***

The source code for the VSS system has been provided for reference. Section C.1 contains the source code for the Database Manager - Phase II. Section C.2 contains the source code for the Visual Simulator - Phase III.

### ***C.1 The Database Manager - Phase II***

```

*****
*****
**  Program:          VSS                               **
**  Developed By:     Jeffrey F. Skolnick                **
**  Date:             June 1, 1990                      **
**  For:              Partial fulfillment of the requirements for **
**                   the degree of Master of Science in   **
**                   Civil Engineering                   **
*****
**  Comments:         This program is Phase II - Database Manager of the **
**                   VSS system. The program operates as follows:  **
**                   **                                     **
**                   Step 1.  Import .DBF file generated from the CPM **
**                   scheduling processor containing activity      **
**                   id's, early, late and actual start and       **
**                   finish dates.                                **
**                   **                                     **
**                   Step 2.  Prompt the user to input the name of each **
**                   object in the 3D computer model of the       **
**                   construction project and record this         **
**                   information in the model register of         **
**                   this system.                                **
**                   **                                     **
**                   Step 3.  Prompt the user to map activities    **
**                   associated with each object of the 3D        **
**                   computer model.                              **
**                   **                                     **
**                   Step 4.  Process this information and generate **
**                   a schedule of object durations.              **
**                   **                                     **
**                   Step 5.  Export this information in ASCII format. **
**                   **                                     **
*****

```

```

*****
**  DECLARE VARIABLES **
*****

```

```

STORE SPACE(8) TO SCHED_FILE
STORE SPACE(8) TO OBJ_FILE
STORE SPACE(8) TO MAP_FILE
STORE SPACE(8) TO NAME_OBJ_FILE
STORE SPACE(8) TO EXPORT_FILE
STORE SPACE(15) TO O_NAME
STORE SPACE(10) TO ACTIVITY_DATA
STORE SPACE(1) TO RESPONSE

```

```

*****
**  SET UP dBASE ENVIRONMENT **
*****

```

```

SET SAFETY OFF
SET TALK OFF
CLOSE ALL
SET COLOR TO GR+/B,W+/R+
DEFINE WINDOW RET_WIN FROM 7,15 TO 13,62 DOUBLE COLOR GR+/N
SET PROCEDURE TO RET_SCH
SET PROCEDURE TO RET_OBJ
SET PROCEDURE TO RET_MAP
SET PROCEDURE TO HEADING
SET PROCEDURE TO BUILD_OB
SET PROCEDURE TO MAPS
SET PROCEDURE TO COM_LOG

```

### C. SOURCE CODE



```
*****
** BEGIN MAIN PROGRAM **
*****
```

```
AOK = .F.
DO WHILE .NOT. AOK
```

```
*****
** CLEAR THE SCREEN **
*****
CLEAR
```

```
*****
** DRAW THE MAIN MENU **
*****
DO HEADING
```

```
SET COLOR TO N/BG
@ 18,21 CLEAR TO 20,56
@ 19,22 SAY "DEVELOPED BY: JEFFREY F. SKOLNICK"
SET COLOR TO N/B
@ 21,22 SAY "_____ "
@ 20,57 SAY "["
@ 19,57 SAY "["
@ 18,57 SAY "\"
```

```
*****
** SET UP MAIN MENU OPTIONS **
*****
```

```
P1 = "1. Build Model File"
P2 = "2. Map Model Objects to Activities"
P3 = "3. Compute Object Dates"
P4 = "4. Export dBASE File to ASCII File"
P5 = "5. Exit the VSS System"
```

```
DEFINE POPUP MAIN FROM 8,20 MESSAGE
"Visual Scheduling Simulation System"
```

```
*****
** MAKE MENU ITEMS PICKABLE **
*****
```

```
DEFINE BAR 1 OF MAIN PROMPT P1
DEFINE BAR 2 OF MAIN PROMPT P2
DEFINE BAR 3 OF MAIN PROMPT P3
DEFINE BAR 4 OF MAIN PROMPT P4
DEFINE BAR 5 OF MAIN PROMPT P5
```

```
SET COLOR TO GR+/B,W+/R+
```

```
*****
** GET INPUT FROM USER **
*****
```

```
ON SELECTION POPUP MAIN DEACTIVATE POPUP
ACTIVATE POPUP MAIN
ANSWER = PROMPT()
CLEAR
```

```
*****
** PROCESS USER INPUT **
*****
```

```
DO CASE
CASE ANSWER = "1"
DO BUILD_OB
```

```
        CASE ANSWER = "2"  
        DO MAPS  
        CASE ANSWER = "3"  
        DO COM_LOG  
        CASE ANSWER = "4"  
        DO EXPORT  
        CASE ANSWER = "5"  
        CLOSE ALL  
        AOK = .T.  
        LOOP  
    ENDCASE  
ENDDO
```

```

*****
** Program:          BUILD_OB                      **
** Developed By:      Jeffrey F. Skolnick          **
** Date:              June 1, 1990                 **
*****
** Comments: This module prompts the user for the name of the **
**            model object file and the name for each object in **
**            the 3D computer model of the construction project. **
**            This information is stored using the OBJECT.DBF **
**            structure.                                     **
*****

```

```

*****
PROCEDURE BUILD_OB
*****

```

```

*****
** DECLARE VARIABLES **
*****

```

```
STORE SPACE(1) TO OVERWRITE
```

```

*****
** CLOSE ALL OPEN FILES **
*****

```

```
CLOSE ALL
```

```

*****
** DRAW THE MENU **
*****

```

```

DO HEADING
SET COLOR TO N/BG
@ 18,29 CLEAR TO 20,49
@ 19,29 SAY " BUILD MODEL FILE "
SET COLOR TO N/B
@ 21,30 SAY " _____ "
@ 20,50 SAY "[ "
@ 19,50 SAY "[ "
@ 18,50 SAY "\ "
SET COLOR TO GR+/B,W+/R+

```

```
ACTIVATE WINDOW RET_WIN
```

```

*****
** OPEN THE OBJECT SET-UP FILE AND PROMPT THE USER TO ENTER THE **
** THE NAME TO SAVE THE MODEL FILE. ALLOW THE USER TO ABORT **
** THIS PROCEDURE BY PRESSING THE ESCAPE KEY. WARN THE USER IF **
** THE FILE EXISTS AND GIVE HIM THE OPTION TO OVERWRITE THE FILE **
*****

```

```

OVER_FILE = .F.
DO WHILE .NOT. OVER_FILE
  @ 1,8 SAY "Enter name of new model file."
  @ 2,18 GET NAME_OBJ_FILE
  @ 4,2 SAY "Press the ESCAPE key to terminate entry."
  READ

  IF LASTKEY() = 27
    DEACTIVATE WINDOW RET_WIN
    RETURN
  ENDIF

```

```

OVERWRITE = "Y"
IF FILE("&NAME_OBJ_FILE" + ".DBF") = .T.
  CLEAR
  @ 1,5 SAY "This file already exists!"
  @ 2,5 SAY "Would you like to overwrite it?"
  @ 3,10 SAY "Enter Y or N."
  @ 3,25 GET OVERWRITE
  READ
ENDIF

IF UPPER(OVERWRITE) = "Y"
  OVER_FILE = .T.
  LOOP
ENDIF

IF UPPER(OVERWRITE) = "N"
  CLEAR
ELSE
  CLEAR
  @ 1,5 SAY "You did not enter a valid response"
  WAIT
  CLEAR
ENDIF

ENDDO
CLEAR

*****
** COPY THE STRUCTURE OF THE OBJECT SET-UP FILE TO THE **
** MODEL FILE NAME SPECIFIED BY THE USER.  OPEN THIS **
** FILE AND SORT IT BY OBJECT NAME.                  **
*****

USE OBJECT
COPY STRUCTURE TO &NAME_OBJ_FILE
USE &NAME_OBJ_FILE
INDEX ON OBJ_NAME TO &NAME_OBJ_FILE

*****
** PROMPT THE USER TO ENTER ALL OBJECT NAMES DEFINED IN THE **
** 3-D COMPUTER MODEL.  READ THE NAME THE USER INPUT AN **
** PROMPT HIM TO INPUT ANOTHER NAME.  ALLOW THE USER TO **
** TERMINATE ENTRY BY PRESSING THE ESCAPE KEY.  CHECK TO **
** MAKE SURE THE USER DOES NOT INPUT AN OBJECT NAME MORE **
** THAN ONCE.  IF ALL INPUT IS CORRECT ADD THE OBJECT NAME **
** TO THE MODEL FILE.                                     **
*****

OK_BUILD = .F.
DO WHILE .NOT. OK_BUILD
  FLAG = 0
  CLEAR
  @ 1,5 SAY "Enter object name." GET O_NAME
  @ 3,1 SAY "Press the ESCAPE key to terminate entry."
  READ

  IF LASTKEY() = 27
    OK_BUILD = .T.
    LOOP
  ENDIF

  GO TOP
  DO WHILE .NOT. EOF()
    IF OBJ_NAME = O_NAME
      CLEAR
      @ 1,1 SAY "THIS IS OBJECT NAME HAS ALREADY BEEN ENTERED!"

```

```

@ 2,1 SAY "PLEASE CHECK THE OBJECT NAME AND TRY AGAIN."
FLAG = 1
WAIT
EXIT
ELSE
SKIP
ENDIF
ENDDO

IF FLAG = 0
APPEND BLANK
REPLACE OBJ_NAME WITH UPPER(O_NAME)
ENDIF

ENDDO

*****
** POSITION THE POINTER TO THE FIRST RECORD IN THE MODEL FILE, **
** CLOSE ALL OPEN FILES AND DEACTIVATE THE OPEN WINDOW.      **
*****

GO TOP
CLOSE ALL
DEACTIVATE WINDOW RET_WIN
RETURN

```

```

*****
*****
** Program:          MAPS                      **
** Developed By:     Jeffrey F. Skolnick       **
** Date:             June 1, 1990             **
*****
** Comments: This module prompts the user for the scheduling **
**            and model file to use for activity/object mapping. **
**            The user is then prompted to enter the activity ids **
**            associated with each object. This information is **
**            stored sequentially in a mapping file specified by **
**            the user.                      **
*****

```

```

*****
PROCEDURE MAPS
*****

```

```

*****
** DECLARE VARIABLES **
*****

```

```

FLAG_MAP = 0
STATUS = 0

```

```

*****
** CLOSE ALL OPEN FILES **
*****

```

```

CLOSE ALL

```

```

*****
** DRAW THE MENU **
*****

```

```

DO HEADING
SET COLOR TO N/BG
@ 17,28 CLEAR TO 20,50
@ 18,29 SAY " MAP MODEL OBJECTS TO "
@ 19,29 SAY " SCHEDULED ACTIVITIES "
SET COLOR TO N/B
@ 21,29 SAY " _____ "
@ 20,51 SAY "["
@ 19,51 SAY "["
@ 18,51 SAY "["
@ 17,51 SAY "\"
SET COLOR TO GR+/B,W+/R+
STORE SPACE(1) TO OVERWRITE

```

```

*****
** RETRIEVE THE SCHEDULING FILE AND MODEL FILE AND CHECK TO SEE **
** IF THE USER WANTS TO ABORT THIS PROCEDURE.                  **
*****

```

```

DO RET_SCH
IF STATUS = 1
    RETURN
ENDIF

```

```

DO RET_OBJ
IF STATUS = 1
    RETURN
ENDIF

```

```

ACTIVATE WINDOW RET_WIN

```

### C. SOURCE CODE

```

*****
** PROMPT THE USER FOR THE NAME TO SAVE THE MAP FILE. **
** CHECK TO SEE IF THE USER WANTS TO ABORT THIS **
** PROCEDURE. WARN THE USER IF THE FILE EXISTS AND **
** GIVE HIM THE OPTION TO OVERWRITE THE FILE. **
*****

```

```

OVER_FILE = .F.
DO WHILE .NOT. OVER_FILE
  @ 1,9 SAY "Enter name to save map file."
  @ 2,18 GET MAP_FILE
  @ 4,2 SAY "Press ESCAPE to abort this process."
  READ

  IF LASTKEY() = 27
    DEACTIVATE WINDOW RET_WIN
    RETURN
  ENDIF

  OVERWRITE = "Y"
  IF FILE("&MAP_FILE" + ".DBF") = .T.
    CLEAR
    @ 1,5 SAY "This file already exists!"
    @ 2,5 SAY "Would you like to overwrite it?"
    @ 3,10 SAY "Enter Y or N."
    @ 3,25 GET OVERWRITE
    READ
  ENDIF

  IF UPPER(OVERWRITE) = "Y"
    CLEAR
    OVER_FILE = .T.
    LOOP
  ENDIF

  IF UPPER(OVERWRITE) = "N"
    CLEAR
  ELSE
    CLEAR
    @ 1,5 SAY "You did not enter a valid response!"
    WAIT
    CLEAR
  ENDIF

ENDDO
CLEAR
DEACTIVATE WINDOW RET_WIN

@ 0,0 CLEAR TO 21,13

```

```

*****
** OPEN AND USE THE SORTED VERSION OF THE SCHEDULE FILE, **
** MODEL FILE, AND MAPPING FILE. **
*****

```

```

SELECT 4
USE &SCHED_FILE INDEX &SCHED_FILE
REPLACE ALL ACT WITH LTRIM(ACT)
SELECT 1
USE &OBJ_FILE ALIAS OBJ_FILE
SELECT 2
USE MAP
COPY STRUCTURE TO &MAP_FILE
SELECT 3
USE &MAP_FILE

```

```

*****
** DISPLAY THE OBJECTS IN THE MODEL FILE ONE AT A TIME AND PROMPT **
** THE USER TO ENTER ALL ASSOCIATED ACTIVITIES. HE MAY ENTER UP **
** TO 8 ACTIVITIES PER OBJECT. ALLOW THE USER TO MOVE TO THE NEXT **
** OBJECT BY PRESSING THE PAGE DOWN KEY OR TO ABORT THIS **
** PROCEDURE BY PRESSING THE ESCAPE KEY. CHECK TO MAKE SURE EACH **
** ACTIVITY ENTERED IS IN THE ACTIVITY FILE AND WARN THE USER IF **
** IT IS NOT. ADD THE CORRECT ASSOCIATION BETWEEN THE OBJECT AND **
** ACTIVITY TO THE MAP FILE. WHEN THE END OF THE OBJECT FILE IS **
** REACHED QUIT THIS PROCEDURE. **
*****

```

```

SET COLOR TO B/W
@ 7,57 CLEAR TO 15,77
@ 8,63 SAY "CONTROLS"
@ 9,63 SAY "-----"
@ 10,58 SAY "Page Down Key -"
@ 11,58 SAY "Moves to next object"
@ 13,58 SAY "Escape Key -"
@ 14,58 SAY "Aborts this process"
SET COLOR TO N/B
@ 16,58 SAY "_____ "
@ 15,78 SAY "["
@ 14,78 SAY "["
@ 13,78 SAY "["
@ 12,78 SAY "["
@ 11,78 SAY "["
@ 10,78 SAY "["
@ 9,78 SAY "["
@ 8,78 SAY "["
@ 7,78 SAY "\"
SET COLOR TO GR+/B,W+/R+
OK_MAP1 = .F.
DO WHILE .NOT. OK_MAP1
  STORE " " TO ACTIVITY_DATA
  @ 5,33 SAY "OBJECT NAME:"
  @ 8,33 SAY "ACTIVITY ID:"
  J=9
  DO WHILE J < 17
    STATUS = .F.
    DO WHILE .NOT. STATUS
      SELECT 1
      SET COLOR TO BG/N
      @ 6,31 CLEAR TO 6,41
      @ 6,31 SAY OBJ_NAME
      SET COLOR TO GR+/B,W+/R+
      @ J,33 GET ACTIVITY_DATA
      READ
      IF LASTKEY() = 27
        ACTIVATE WINDOW RET_WIN
        @ 1,1 SAY "ARE YOU SURE YOU WANT TO ABORT THIS PROCESS?"
        @ 2,1 SAY "PLEASE ENTER Y OR N."
        @ 2,22 GET RESPONSE
        READ
        IF UPPER(RESPONSE) = "Y"
          DEACTIVATE WINDOW RET_WIN
          RETURN
        ENDIF
        IF UPPER(RESPONSE) = "N"
          DEACTIVATE WINDOW RET_WIN
        ELSE
          CLEAR
          @ 1,1 SAY "YOU DID NOT ENTER A VALID RESPONSE!"
          WAIT
          DEACTIVATE WINDOW RET_WIN
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  OK_MAP1 = .T.
ENDIF

```



```

ENDIF
ELSE
  STATUS = .T.
ENDIF
ENDDO

IF ACTIVITY_DATA # " "
SELECT 4
GO TOP
FIND &ACTIVITY_DATA

IF FOUND() = .F.
  FLAG_MAP = 1
  ACTIVATE WINDOW RET_WIN
  @ 1,1 SAY "THIS IS NOT AN ACTIVITY IN THE SCHEDULE!"
  @ 2,1 SAY "PLEASE CHECK THE ACTIVITY NAME AND TRY AGAIN"
  WAIT
  DEACTIVATE WINDOW RET_WIN
ENDIF

ENDIF

IF FLAG_MAP = 0
SELECT 3
APPEND BLANK
REPLACE OBJ_NAME WITH OBJ_FILE->OBJ_NAME
REPLACE ACT_WITH UPPER(ACTIVITY_DATA)
J = J + 1

IF LASTKEY() = 3 .OR. ACTIVITY_DATA = " "
  DELETE
  PACK
  J=17
  SKIP IN OBJ_FILE

  IF EOF([OBJ_FILE])
    OK_MAP1 = .T.
    LOOP
  ENDIF

  @ 5,33 CLEAR TO 16,45
  LOOP
ENDIF

ENDIF

FLAG_MAP = 0
ENDDO
ENDDO
RETURN

```

```

*****
*****
** Program:          COM_LOG                      **
** Developed By:     Jeffrey F. Skolnick          **
** Date:             June 1, 1990                 **
*****
** Comments: This module prompts the user for the names of the **
**            schedule file, model file, and mapping file. It  **
**            uses the information contained in these files to  **
**            generate an object schedule which can be         **
**            used by the Visual Simulator to produce visual   **
**            simulation of the construction process.          **
*****
*****

```

```

*****
PROCEDURE COM_LOG
*****

```

```

*****
** DECLARE VARIABLES **
*****

```

```
STATUS = 0
```

```

*****
** CLOSE ALL OPEN FILES **
*****
CLOSE ALL

```

```
CLEAR
```

```

*****
** DRAW THE MENU **
*****

```

```

DO HEADING
SET COLOR TO N/BG
@ 17,32 CLEAR TO 19,54
@ 18,32 SAY " COMPUTE OBJECT DATES "
SET COLOR TO N/B
@ 20,33 SAY " _____ "
@ 19,55 SAY "[ "
@ 18,55 SAY "[ "
@ 17,55 SAY "\ "
SET COLOR TO GR+/B,W+/R+

```

```

*****
** RETRIEVE THE SCHEDULING FILE, OBJECT FILE AND MAP FILE. CHECK **
** TO SEE IF THE USER WANTS TO ABORT THIS PROCEDURE.          **
*****

```

```

DO RET_SCH
IF STATUS = 1
RETURN
ENDIF

```

```

DO RET_OBJ
IF STATUS = 1
RETURN
ENDIF

```

```

DO RET_MAP
IF STATUS = 1
RETURN
ENDIF

```

### C. SOURCE CODE

```
*****
** LET USER KNOW THAT THE SYSTEM IS COMPUTING OBJECT DATES **
*****
```

```
ACTIVATE WINDOW RET_WIN
@ 1,1 SAY "COMPUTING OBJECT DATES"
@ 2,1 SAY "PLEASE STAND BY...."
```

```
*****
** OPEN AND INITIALIZE ALL VALUES IN THE MODEL FILE **
*****
```

```
USE &OBJ_FILE
GO TOP
DO WHILE .NOT. EOF()
    REPLACE ESO WITH 10000
    REPLACE EFO WITH 0
    REPLACE LSO WITH 10000
    REPLACE LFO WITH 0
    REPLACE ASO WITH 10000
    REPLACE AFO WITH 0
    SKIP
ENDDO
```

```
CLOSE ALL
```

```
*****
** OPEN THE MAPPING FILE AND GET THE RELATIONSHIP BETWEEN OBJECTS **
** AND ACTIVITIES FOR THE CURRENT POINTER POSITION. **
** OPEN AND GET THE START AND FINISH DATES FROM THE SCHEDULE FILE. **
** OPEN THE MODEL FILE, POSITION THE POINTER AT THE CORRECT OBJECT **
** NAME, AND APPEND THE DATES IF THEY MEET THE APPROPRIATE **
** CRITERIA. REOPEN THE MAP FILE, POSITION THE POINTER TO THE NEXT **
** RELATIONSHIP AND LOOP TO BEGIN THIS PROCESS AGAIN. **
*****
```

```
USE &MAP_FILE
GO TOP
I=0
DO WHILE .NOT. EOF()
    I=I+1
    GET_OBJ = OBJ_NAME
    GET_ACT=ACT
    USE &SCHED_FILE INDEX &SCHED_FILE
    FIND &GET_ACT
    E_START = ESW
    E_FINISH = EFW
    L_START = LSW
    L_FINISH = LFW
    A_START = ASW
    A_FINISH = AFW
    USE &OBJ_FILE INDEX &OBJ_FILE
    FIND &GET_OBJ

    IF ESO >= E_START
        REPLACE ESO WITH E_START
    ENDIF

    IF EFO <= E_FINISH
        REPLACE EFO WITH E_FINISH
    ENDIF

    IF LSO >= L_START
        REPLACE LSO WITH L_START
    ENDIF
```

```
IF LFO <= L_FINISH
  REPLACE LFO WITH L_FINISH
ENDIF

IF ASO >= A_START
  REPLACE ASO WITH A_START
ENDIF

IF AFO <= A_FINISH
  REPLACE AFO WITH A_FINISH
ENDIF

USE &MAP_FILE
GO TOP
SKIP I
ENDDO

DEACTIVATE WINDOW RET_WIN
CLOSE ALL
RETURN
```

```

*****
*****
** Program:          EXPORT                      **
** Developed By:     Jeffrey F. Skolnick        **
** Date:             June 1, 1990              **
*****
** Comments: This module prompts the user for the name of a .DBF **
**            file. This file transformed into ASCII format.    **
*****

```

```

*****
PROCEDURE EXPORT
*****

```

CLOSE ALL

```

*****
** DRAW THE MENU **
*****
DO HEADING
SET COLOR TO N/BG
@ 18,31 CLEAR TO 20,45
@ 19,31 SAY "EXPORT FILE "
SET COLOR TO N/B
@ 21,32 SAY "          "
@ 20,46 SAY "[ "
@ 19,46 SAY "[ "
@ 18,46 SAY "\ "
SET COLOR TO GR+/B,W+/R+

```

DEFINE POPUP FILES FROM 1,0 TO 20,13 PROMPT FILES LIKE \*.DBF  
SHOW POPUP FILES

```

*****
** ACTIVATE THE WINDOW AND PROMPT THE USER **
** FOR EXPORT FILE NAME **
*****
ACTIVATE WINDOW RET_WIN

```

```

RIGHT_FILE = .F.
DO WHILE .NOT. RIGHT_FILE
  @ 1,8 SAY "Enter name of file to export."
  @ 2,18 GET EXPORT_FILE
  READ

```

```

  IF LASTKEY() = 27
    DEACTIVATE WINDOW RET_WIN
    RETURN
  ENDIF

```

```

*****
** CHECK TO MAKE SURE FILE EXISTS **
*****

```

```

IF FILE("&EXPORT_FILE" + ".DBF") = .F.
  CLEAR
  @ 1,2 SAY "This file is not in the directory!"
  @ 2,2 SAY "Please check the file name and try again."
  WAIT
  CLEAR
ELSE
  RIGHT_FILE = .T.
ENDIF

```

ENDDO

```
*****  
**  USE USER DEFINED FILE, SORT IT ACCORDING TO EARLY **  
**  START AND CONVERT IT INTO ASCII FORMAT          **  
*****
```

```
USE &EXPORT_FILE  
INDEX ON ESO TO SORT_ESO  
COPY TO &EXPORT_FILE SDF
```

```
DEACTIVATE WINDOW RET_WIN  
CLOSE ALL  
CLEAR  
RETURN
```

```

*****
*****
** Program:          RET_SCH                      **
** Developed By:     Jeffrey F. Skolnick          **
** Date:             June 1, 1990                 **
*****
** Comments: This module prompts the user for the name of the **
**            scheduling file to be used and retrieves the file. **
*****
*****
PROCEDURE RET_SCH
*****

CLOSE ALL

*****
** SET UP SCREEN **
*****

SET COLOR TO N/W,GR+/N
@ 1,0 CLEAR TO 20,13
DEFINE POPUP FILES FROM 1,0 TO 20,13 PROMPT FILES LIKE *.DBF
SHOW POPUP FILES
SET COLOR TO GR+/B,W+/R+
ACTIVATE WINDOW RET_WIN

*****
** PROMPT THE USER FOR THE NAME OF THE **
** SCHEDULING FILE TO RETRIEVE **
*****

RIGHT_FILE = .F.
DO WHILE .NOT. RIGHT_FILE
  @ 1,2 SAY "Enter name of scheduling file to retrieve."
  @ 2,18 GET SCHED_FILE
  @ 4,2 SAY "Press the ESCAPE key to abort this process."
  READ

  IF LASTKEY() = 27
    DEACTIVATE WINDOW RET_WIN
    STATUS = 1
    RETURN
  ENDIF

*****
** SEARCH THE DIRECTORY AND WARN USER IF THE **
** FILE DOES NOT EXIST **
*****

IF FILE("&SCHED_FILE" + ".DBF") = .F.
  CLEAR
  @ 1,2 SAY "This file is not in the directory!"
  @ 2,2 SAY " Please check the file name and try again."
  WAIT
  CLEAR
  LOOP
ENDIF

*****
** USE THE USER DEFINED FILE AND CHECK TO SEE IF THE **
** FILE IS A SCHEDULING FILE. IF IT IS NOT, WARN THE **
** USER AND PROMPT HIM FOR ANOTHER CHOICE. **
*****

USE &SCHED_FILE

```

```

IF FIELD(2) = "ESW"
  RIGHT_FILE = .T.
ELSE
  CLEAR
  @ 1,2 SAY "This is not a scheduling file!"
  @ 2,2 SAY "Please check the file and try again"
  WAIT
  CLEAR
ENDIF

ENDDO

*****
**  SORT THE FILE ACCORDING TO ACTIVITY ID  **
*****
INDEX ON ACT TO &SCHED_FILE

DEACTIVATE WINDOW RET_WIN
CLOSE ALL
RETURN

```



```

*****
*****
** Program:          RET_OBJ                      **
** Developed By:     Jeffrey F. Skolnick          **
** Date:             June 1, 1990                 **
*****
** Comments: This module prompts the user for the name of the **
**            model file to be used and retrieves the file.    **
*****
*****
PROCEDURE RET_OBJ
*****
CLOSE ALL

*****
** SET UP SCREEN **
*****

SET COLOR TO N/W,GR+/N
@ 1,0 CLEAR TO 20,13
DEFINE POPUP FILES FROM 1,0 TO 20,13 PROMPT FILES LIKE *.DBF
SHOW POPUP FILES
SET COLOR TO GR+/B,W+/R+

ACTIVATE WINDOW RET_WIN

*****
** PROMPT THE USER FOR THE NAME OF THE **
** MODEL FILE TO RETRIEVE.             **
*****

RIGHT_FILE = .F.
DO WHILE .NOT. RIGHT_FILE
  @ 1,4 SAY "Enter name of model file to retrieve."
  @ 2,18 GET OBJ_FILE
  @ 4,2 SAY "Press the ESCAPE key to abort this process."
  READ

  IF LASTKEY() = 27
    DEACTIVATE WINDOW RET_WIN
    STATUS = 1
    RETURN
  ENDIF

  *****
  ** SEARCH THE DIRECTORY AND WARN THE USER IF **
  ** THE FILE DOES NOT EXIST.                  **
  *****

  IF FILE("&OBJ_FILE" + ".DBF") = .F.
    CLEAR
    @ 1,2 SAY "This file is not in the directory!"
    @ 2,1 SAY " Please check the file name and try again."
    WAIT
    CLEAR
    LOOP
  ENDIF

  *****
  ** USE THE USER DEFINED FILE AND CHECK TO SEE IF THE **
  ** FILE IS A MODEL FILE. IF IT IS NOT, WARN THE **
  ** USER AND PROMPT HIM FOR ANOTHER CHOICE.         **
  *****
  USE &OBJ_FILE ALIAS OBJ_FILE

```

```
IF FIELD(2) = "ESO"  
  RIGHT_FILE = ".T."  
ELSE  
  CLEAR  
  @ 1,2 SAY "This is not a model file!"  
  @ 2,2 SAY "Please check the file and try again."  
  WAIT  
  CLEAR  
ENDIF  
ENDDO  
  
DEACTIVATE WINDOW RET_WIN  
CLOSE ALL  
RETURN
```

```

*****
*****
** Program:          RET_MAP          **
** Developed By:     Jeffrey F. Skolnick  **
** Date:            June 1, 1990      **
*****
** Comments: This module prompts the user for the name of the  **
**            mapping file to be used and retrieves this file  **
*****
*****
PROCEDURE RET_MAP
*****
CLOSE ALL

*****
** SET UP SCREEN **
*****

SET COLOR TO N/W,GR+/N
@ 1,0 CLEAR TO 20,13
DEFINE POPUP FILES FROM 1,0 TO 20,13 PROMPT FILES LIKE *.DBF
SHOW POPUP FILES
SET COLOR TO GR+/B,W+/R+

ACTIVATE WINDOW RET_WIN

*****
** PROMPT THE USER FOR THE NAME OF THE  **
** MAPPING FILE TO RETRIEVE.          **
*****

RIGHT_FILE = .F.
DO WHILE .NOT. RIGHT_FILE
  @ 1,4 SAY "Enter name of mapping file to retrieve."
  @ 2,18 GET MAP_FILE
  @ 4,2 SAY "Press the ESCAPE key to abort this process."
  READ

  IF LASTKEY() = 27
    DEACTIVATE WINDOW RET_WIN
    STATUS = 1
    RETURN
  ENDIF

  *****
  ** SEARCH THE DIRECTORY AND WARN THE USER IF **
  ** THE FILE DOES NOT EXIST.                  **
  *****

  IF FILE("&MAP_FILE" + ".DBF") = .F.
    CLEAR
    @ 1,2 SAY "This file is not in the directory!"
    @ 2,1 SAY "Please check the file name and try again."
    WAIT
    CLEAR
    LOOP
  ENDIF

  *****
  ** USE THE USER DEFINED FILE AND CHECK TO SEE IF THE **
  ** FILE IS A MAPPING FILE. IF IT IS NOT, WARN THE **
  ** USER AND PROMPT HIM FOR ANOTHER CHOICE.          **
  *****

  USE &MAP_FILE

```

```
IF FIELD(2) = "ACT"  
    RIGHT_FILE = .T.  
ELSE  
    CLEAR  
    @ 1,2 SAY "This is not a map file!"  
    @ 2,2 SAY "Please check the file and try again."  
    WAIT  
    CLEAR  
ENDIF  
ENDDO  
  
DEACTIVATE WINDOW RET_WIN  
  
CLOSE ALL  
RETURN
```

```

*****
** Program:      HEADING                               **
** Developed By:  Jeffrey F. Skolnick                   **
** Date:         June 1, 1990                           **
*****
** Comments:     This module creates the main "VSS System" screen  **
**               heading.                                         **
*****

```

```

*****
PROCEDURE HEADING
*****

```

```

*****
** DRAW THE MAIN HEADING **
*****

```

```

SET COLOR TO GR+/R
@ 1,29 CLEAR TO 3,48
@ 2,29 SAY " V S S S Y S T E M "
@ 1,49 SAY "\" COLOR N/B
@ 2,49 SAY "[" COLOR N/B
@ 3,49 SAY "[" COLOR N/B
@ 4,49 SAY "[" COLOR N/B
@ 4,30 SAY " _____ " COLOR N/B

```

```

RETURN

```

## ***C.2 The Visual Simulator - Phase III***

```

/*****
/*****
/**
/** Function Name : vss.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This program is Phase III - The Visual Simulator for
/** the VSS system. This function does the following:
/**
/** 1. Sets up the Visual Simulator's environment.
/** 2. Sets up and initializes the main menu.
/** 3. Allows the user to select the menu items.
/** 4. Reacts to the user's selection by calling
/** the appropriate function.
/**
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"
#include "walkfiles.h"

```

```

/* ***** Define Structures ***** */

```

```

struct date
{
    int month;
    int day;
    int year;
};

```

```

struct schedule
{
    char obj_name[16];
    int e_start;
    int e_finish;
    int l_start;
    int l_finish;
    int a_start;
    int a_finish;
    int walk_obj;
    int color;
};

```

```

struct obj_walk
{
    int physical;
    int virtual;
    char name[16];
};

```

```

/* *****External Variable Declaration ***** */

```

```

/* Logic options */

```

### C. SOURCE CODE

```

static char *logic_opts[3] = { "Early Dates",
                                "Late Dates",
                                "Early/Late Dates" };

/* View options */
static char *view_opts[3] = { "Planned",
                               "Actual",
                               "Planned & Actual" };

static int start_date = 0; /* Project start date */
static int finish_date = 0; /* Project finish date */

int *s_date_opt; /* Pointer to project start date */
int *f_date_opt; /* Pointer to project finish date */

int user_function( WALK_ARGS )

/* ***** Argument Declaration ***** */
#include "walkargs.h" /* Definition of WALK_ARGS. (standard */
/* walkthru arguments. ) */

{

/* ***** Variable Declaration ***** */

    struct date begin_date;
    struct date end_date;
    struct date day_count;
    struct date this_date[700];
    struct schedule obj_dates[200];
    struct obj_walk objects[200];
    struct SURFACE_ELEMENT *curr_surf;

    int exit, /* Exit routine flag. */
        status, /* Status of the operation. */
        option, /* Menu choice. */
        time, /* Simulation time */
        logic_opt, /* Date option */
        view_opt, /* View option */
        count_obj, /* Number of objects counter */
        last_option, /* Last option selected by the user. */
        i;

    char options[8][24], /* Menu options. */
        title[24], /* Menu title. */
        file_name[150]; /* File name. */

    void simulate(); /* ***** */
    void get_file(); /* ***** */

    int get_time(); /*** Define ***/
    int get_logic(); /*** sub-functions ***/
    int get_start_date(); /*** ***/
    int get_finish_date(); /*** ***/

    struct date date_update(); /* ***** */

```



```

/* ***** Source Code ***** */

/* ***** Initialize Variables ***** */

    exit                = 0;
    status              = 0;
    last_option         = 0;
    time                = 5;
    logic_opt           = 0;
    view_opt            = 0;
    begin_date.month    = 1;
    begin_date.day      = 1;
    begin_date.year     = 90;
    end_date.month      = 1;
    end_date.day        = 1;   end_date.year      = 90;

    change_menu_box( USER_FUNCTION_BOX, 1 );
    set_int4_mask( opt->menu_mask, USER_FUNCTION_BOX );

    s_date_opt = &start_date;
    f_date_opt = &finish_date;

    this_date[0] = begin_date;

    for (i = 1; i < 700; i++)
    {
        day_count = this_date[i-1];
        this_date[i] = date_update(day_count);
    }

/* ***** Set up the menu options. ***** */

    sprintf( &options[0][0], " File - " );
    sprintf( &options[1][0], " Time - %d", time );
    sprintf( &options[2][0], " %s ", logic_opts[logic_opt]);
    sprintf( &options[3][0], " %s ", view_opts[view_opt]);
    sprintf( &options[4][0], " Start Date - %d/%d/%d ", begin_date.month,
        begin_date.day, begin_date.year);
    sprintf( &options[5][0], " Finish Date - %d/%d/%d ", end_date.month,
        end_date.day, end_date.year);
    sprintf( &options[6][0], "      Execute " );
    sprintf( &options[7][0], "      Exit " );
    sprintf( title, "      VSS System ");

/* ***** Initialize the menu. ***** */

    initialize_menu( title, MENU_X_ORIGIN,
        MENU_Y_ORIGIN, 8 , options, 23 );

/* ***** Let the user make the appropriate selections. ***** */

    while( !exit )
    {
        option = update_menu( title, MENU_X_ORIGIN,
            MENU_Y_ORIGIN, 8, options, 23);

        /* Based upon the selected option */
        /* perform the required function. */
    }

```

### C. SOURCE CODE

```

switch( abs( option ) )
{
    case 1 :    /* Get the file name.                                */
        (void) get_file(options, file_name, s_date_opt,
                        f_date_opt, this_date, obj_dates,
                        objects, &count_obj, WALK_ARGS);
        break;

    case 2 :    /* Get Time per day.                                */
        time = get_time(options);
        break;

    case 3 :    /* Change Logic option.                            */
        logic_opt = get_logic(options, logic_opts);
        break;

    case 4 :    /* Change View option.                            */
        view_opt = get_view(options, view_opts);
        break;

    case 5 :    /* Start Date option.                            */
        start_date = get_start_date(options, s_date_opt,
                                    f_date_opt, this_date);
        break;

    case 6 :    /* Finish Date option.                            */
        finish_date = get_finish_date(options, s_date_opt,
                                       f_date_opt, this_date);
        break;

    case 7 :    /* Execute Simulation                            */
        (void) simulate(options, &time, &logic_opt, &view_opt,
                        s_date_opt, f_date_opt, obj_dates, objects,
                        &count_obj, WALK_ARGS);
        break;

    case 8 :    /* ***** Move a tag. ***** */
        if( getbutton( RIGHTMOUSE ) ||
            getbutton( MIDDLEMOUSE ) ||
            getbutton( LEFTMOUSE ) )
        {
            remove_menu( );
            exit = 1;
        }
        break;

} /* ***** End the switch.***** */

/*      Check to see if the user want to preform any of the other      */
/*      functions. The only not available is read tag.                  */
if( get_parameters( WALK_ARGS ) )
{
    walk_functions( WALK_ARGS );
    if( opt->new_scene )

```

```

        {
            initialize_menu( title, MENU_X_ORIGIN,
                           MENU_Y_ORIGIN, 8 , options, 23 );
            status = 1;
        }

    }

    if( status )
    {
        update_screen( WALK_ARGS, W_SWAPCLEAR );
        status = 0;
    }

} /* End the while. */

change_menu_box( USER_FUNCTION_BOX, 0 );
clear_int4_mask( opt->menu_mask, USER_FUNCTION_BOX );
}

```

```

/*****
/*****
/**
/** Function Name : get_file.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function prompts the user for the name of:
/**
/**          1. input file
/**          2. project start date
/**          3. object name file
/**
/**          This function assigns these values to the appropriate
/**          structure variables.
/**
/**          Calling Function: vss.c
/**
/*****
/*****

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"
#include "walkfiles.h"

/* ***** Define Structures ***** */

struct date
{
    int month;
    int day;
    int year;
};

struct schedule
{
    char obj_name[16];
    int e_start;
    int e_finish;
    int l_start;
    int l_finish;
    int a_start;
    int a_finish;
    int walk_obj;
    int color;
};

struct obj_walk
{
    int physical;
    int virtual;
    char name[16];
};

int get_file(options, file_name, s_date_opt, f_date_opt, this_date,
              obj_dates, objects, count_obj, WALK_ARGS)

#include "walkargs.h"

```

```

/* **** Parameter Declaration **** */

struct date this_date[700];
struct schedule obj_dates[200];
struct obj_walk objects[200];

char options[][24],
      file_name[];

int *s_date_opt,
    *f_date_opt,
    *count_obj;

{
/* ***** Variable Declaration ***** */

FILE *input_file,           /* Input file name */
    *object_file;          /* Object file name */

void calc_date();           /* Function to calculate dates */

char object_name[16],       /* Object name from input file */
    o_walk[16];            /* Object name from object name file */

int early_start,            /* Activity early start date */
    early_finish,          /* Activity early finish date */
    late_start,            /* Activity late start date */
    late_finish,          /* Activity late finish date */
    actual_start,          /* Activity actual start date */
    actual_finish,        /* Activity actual finish date */
    not_found,            /* Search flag */
    phys_obj,             /* Physical object number */
    virt_obj,            /* Virtual object number */
    count,               /* Object counter */
    i,j;

static int color_flag = 1; /* Surface color flag */

/* ***** Source Code ***** */

if( getbutton( RIGHTMOUSE ) )
{
    /* Prompt the user for the name of the input file */

    not_found = 1;
    while (not_found)
    {
        walk_printf( "Enter the name of input file : " );
        file_name[0] = 0;
        walk_scanf( file_name );
        input_file = fopen( file_name,"r");

        /* Check to ensure file exists */
        /* and update menu if it does. */

        if ( input_file == (FILE *) NULL )
            walk_printf( "File could not be opened! Try again.\n" );
        else
        {
            not_found = 0;
            sprintf( options[0], " File - %s",

```

### C. SOURCE CODE

```

        file_name );

        draw_menu_option( 1, MENU_X_ORIGIN, MENU_Y_ORIGIN,
                        8, options[0],23,1 );
    }

}

/* Assign information for each record in the      */
/* input file to its appropriate structure value. */

i = 0;
while ( !feof( input_file ) )
{
    fscanf(input_file,"%s%d%d%d%d%d%d\n",object_name,
            &early_start, &early_finish, &late_start,
            &late_finish, &actual_start, &actual_finish);
    strcpy(obj_dates[i].obj_name,object_name);
    obj_dates[i].e_start = early_start;
    obj_dates[i].e_finish = early_finish;
    obj_dates[i].l_start = late_start;
    obj_dates[i].l_finish = late_finish;
    obj_dates[i].a_start = actual_start;
    obj_dates[i].a_finish = actual_finish;

    i++;
}

/* Get project start date      */
/* and update menu.            */

(void) calc_date(s_date_opt, f_date_opt, this_date);

sprintf( options[4], " Start Date - %d/%d/%d ",
        this_date[*s_date_opt].month, this_date[*s_date_opt].day,
        this_date[*s_date_opt].year);
draw_menu_option( 5, MENU_X_ORIGIN, MENU_Y_ORIGIN,
                8, options[4],23,0 );
sprintf( options[5], " Finish Date - %d/%d/%d ",
        this_date[*f_date_opt].month, this_date[*f_date_opt].day,
        this_date[*f_date_opt].year);
draw_menu_option( 6, MENU_X_ORIGIN, MENU_Y_ORIGIN,
                8, options[5],23,0 );

/* Prompt the user for the name of the object name file */

not_found = 1;
while (not_found)
{
    walk_printf( "Enter the name of Object Name File : " );
    file_name[0] = 0;
    walk_scanf( file_name );
    object_file = fopen( file_name, "r" );

    /* Check to ensure file exists */

    if ( object_file == (FILE *) NULL )
        walk_printf( "File could not be opened! Try again.\n " );
    else
        not_found = 0;
}

```

```

}

/* Assign information for each record in the          */
/* object name file to its appropriate structure value. */

count = 0;

while ( !feof( object_file ) )
{
    fscanf(object_file,"%d%d%s\n", &phys_obj, &virt_obj, o_walk);
    objects[count].physical = phys_obj;
    objects[count].virtual = virt_obj;
    strcpy(objects[count].name,o_walk);
    count++;
}

/* Assign the appropriate physical object number to */
/* each object name in the schedule structure.      */

for(i = 0; i < count/2; i++)
    for(j = 0; j < count/2; j++)
        if(strcmp(obj_dates[i].obj_name,objects[j].name) == 0)
        {
            obj_dates[i].walk_obj = objects[j].physical;
            obj_dates[i+count/2].walk_obj =
                objects[j+count/2].physical;
        }

/* Assign the appropriate color number to          */
/* each object name in the schedule structure.      */

if(color_flag)
    for(i = 0; i < count; i++)
    {
        obj_dates[i].color = surf[i].color;
        color_flag = 0;
    }

*count_obj = count;    /* Pass number of objects back to vss.c */

/* Close files */

fclose (input_file);
fclose (object_file);
}
return;
}

```

```

/*****
/*****
/****
/**** Function Name : get_time.c ****
/**** Developed By : Jeffrey F. Skolnick ****
/**** Date : July 17, 1990 ****
/**** For : Partial fulfillment of the requirement for the ****
/**** degree of Master of Science in Civil Engineering ****
/**** ****
/*****
/*****
/****
/**** Comments: This function lets the user increase or decrease ****
/**** the simulation time value, by pressing the right or ****
/**** left mouse button respectively. The time value can ****
/**** be increased from a maximum of 20 and decreased ****
/**** to a minimum of 0. This defines the simulation time ****
/**** between working days. This is the second item on the ****
/**** main menu. ****
/**** ****
/**** Calling function: vss.c ****
/**** ****
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"
#include "walkfiles.h"

```

```

int get_time(options)

/* **** Parameter Declaration **** */

char options[][24];

{

/* **** Variable Declaration **** */

static int time = 5;

/* **** Source Code **** */

/* Increase the time value */

if( getbutton ( RIGHTMOUSE ) )
{
time++;
if(time > 20) time = 20;
}

/* Decrease the time value */

if( getbutton( LEFTMOUSE ) )
{
time--;
if(time < 0) time = 0;
}

```

### C. SOURCE CODE



```
/* update the main menu */  
sprintf( options[1], " Time - %d", time );  
draw_menu_option( 2, MENU_X_ORIGIN, MENU_Y_ORIGIN, 8, options[1],  
                  23, 1);  
wait_for(200000);  
return(time);  
}
```

```

/*****
/*****
**
** Function Name : get_logic.c **
** Developed By : Jeffrey F. Skolnick **
** Date : July 17, 1990 **
** For : Partial fulfillment of the requirement for the **
** degree of Master of Science in Civil Engineering **
**
/*****
/*****
**
** Comments: This function lets the user choose the logic option **
** to use for visual simulation. By pressing the right **
** mouse button the user can choose between the **
** following logic options for the planned schedule: **
**
** 1. "Early Dates" - Early start and finish dates. **
** 2. "Late Dates" - Late start and finish dates. **
** 3. "Early/Late Dates" - Early start and late **
** finish dates. **
**
** Calling function: vss.c **
**
/*****
/*****

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"

int get_logic(options, logic_opts)

/* **** Parameter Declaration **** */

char options[][24],
    *logic_opts[];
{
/* ***** Variable Declaration ***** */

static int logic_opt = 0;

/* Toggle thru the logic options */

if( getbutton( RIGHTMOUSE ) )
{
    logic_opt += 1;

    if( logic_opt > 2 ) logic_opt = 0;

    /* update the main menu */

    sprintf( options[2], " %s ",
              logic_opts[ logic_opt ] );
    draw_menu_option( 3, MENU_X_ORIGIN,
                     MENU_Y_ORIGIN, 8, options[2],
                     23, 1);
    wait_for( 200000 );
}

/* return the selected logic option value */

```

```
return(logic_opt);  
}
```

```

/*****
/*****
**
** Function Name : get_view.c
** Developed By : Jeffrey F. Skolnick
** Date : July 17, 1990
** For : Partial fulfillment of the requirement for the
** degree of Master of Science in Civil Engineering
**
/*****
/*****
**
** Comments: This function lets the user choose the viewing
** option to use for visual simulation.
** By pressing the right mouse button the user can
** choose between the following view options
**
** 1. "Planned" - Only the planned schedule will be
** used for simulation.
** 2. "Actual" - Only the actual schedule will be
** used for simulation.
** 3. "Planned & Actual" - Both the planned and the
** actual schedules will be
** used for simulation.
**
** Calling function: vss.c
**
/*****
/*****

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"

int get_view(options, view_opts)

/* **** Parameter Declaration **** */

char options[][24],
    *view_opts[];

{

/* **** Variable Declaration **** */

static int view_opt = 0;

/* Toggle thru the viewing options */

if( getbutton( RIGHTMOUSE ) )
{
    view_opt += 1;

    if( view_opt > 2 ) view_opt = 0;

    sprintf( options[3], " %s ",
              view_opts[ view_opt ] );

    draw_menu_option( 4, MENU_X_ORIGIN,
                     MENU_Y_ORIGIN, 8, options[3],
                     23, 1);

    wait_for( 20000 );
}

```

```
/* Return the selected view option value */  
return(view_opt);  
}
```

```

/*****
/*****
/**
/** Function Name : get_start_date.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function lets the user select the simulation
/** start date. To increase the date press the right
/** mouse button. To decrease the date press the left
/** mouse button. To key in the date press the middle
/** mouse button.
/**
/** Calling function: vss.c
/**
/*****
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"

```

```

/* ***** Define Structures ***** */

```

```

struct date

```

```

{
    int month;
    int day;
    int year;
};

```

```

int get_start_date(options, s_date_opt, f_date_opt, this_date)

```

```

/* ***** Parameter Declaration ***** */

```

```

char options[][24];

```

```

int *s_date_opt,
    *f_date_opt;

```

```

struct date this_date[700];

```

```

{

```

```

void calc_date();

```

```

/* ***** Variable Declaration ***** */

```

```

int start_date,
    month,
    day,
    year,
    not_found,
    i;

```

```

char chardate[100];

```

### C. SOURCE CODE

```

/* ***** Source Code ***** */

/* Increase the simulation start date */
if( getbutton( RIGHTMOUSE ) ) *s_date_opt += 1;

/* Key in the simulation start date */
if( getbutton ( MIDDLEMOUSE ) )
{
    walk_printf ( "Enter the Start Date : ");
    chardate[0] = 0;
    walk_scanf ( chardate );
    sscanf(chardate, "%d/%d/%d", &month, &day, &year);

    /* Make sure date is valid */

    not_found = 1;
    i = 0;
    while(not_found)
    {
        if( (this_date[i].month == month ) &&
            (this_date[i].day == day) &&
            (this_date[i].year == year) )
        {
            *s_date_opt = i;
            not_found = 0;
        }
        if(i == 699) not_found = 0;
        i++;
    }

    /* Update the main menu */

    sprintf( options[4], " Start Date - %d/%d/%d ",
        this_date[*s_date_opt].month, this_date[*s_date_opt].day,
        this_date[*s_date_opt].year);
    draw_menu_option( 5, MENU_X_ORIGIN, MENU_Y_ORIGIN,
        8, options[4], 23, 0 );
}

/* Decrease the simulation start date */
if( getbutton( LEFTMOUSE ) )
{
    *s_date_opt -= 1;

    /* Check to ensure that simulation start date */
    /* is not before the construction start date. */
    /* If it is set it equal to construction start */
    /* date. */

    if( *s_date_opt < 0 ) *s_date_opt = 0;
}

/* Update the main menu */

sprintf( options[4], " Start Date - %d/%d/%d ",
    this_date[*s_date_opt].month, this_date[*s_date_opt].day,
    this_date[*s_date_opt].year);
draw_menu_option( 5, MENU_X_ORIGIN,

```

### C. SOURCE CODE

```

        MENU_Y_ORIGIN, 8, options[4],
        23, 1);

/* Check to ensure that simulation start date */
/* is not greater than simulation finish date */
/* If it is set simulation finish date equal */
/* to simulation start date. */
if( *s_date_opt >= *f_date_opt )
{
    *f_date_opt = *s_date_opt;

    /* Update the main menu */

    sprintf( options[5], " Finish Date - %d/%d/%d ",
              this_date[*f_date_opt].month, this_date[*f_date_opt].day,
              this_date[*f_date_opt].year);
    draw_menu_option( 6, MENU_X_ORIGIN,
                     MENU_Y_ORIGIN, 8, options[5],
                     23, 0);

}

wait_for( 150000 );

start_date = *s_date_opt;

/* Return the selected simulation start date value */
return(start_date);
}

```



```

/*****
/*****
/****
/**** Function Name : get_finish_date.c ****
/**** Developed By : Jeffrey F. Skolnick ****
/**** Date : July 17, 1990 ****
/**** For : Partial fulfillment of the requirement for the ****
/**** degree of Master of Science in Civil Engineering ****
/****
/*****
/*****
/**** Comments: This function lets the user select the simulation ****
/**** finish date. To increase the date press the right ****
/**** mouse button. To decrease the date press the left ****
/**** mouse button. To key in the date press the middle ****
/**** mouse button. ****
/****
/**** Calling function: vss.c ****
/**** ****
/*****
/*****

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"

/* ***** Define Structures ***** */

struct date
{
    int month;
    int day;
    int year;
};

int get_finish_date(options, s_date_opt, f_date_opt, this_date)

/* ***** Parameter Declaration ***** */

char options[][24];

int *s_date_opt,
    *f_date_opt;

struct date this_date[700];

{

/* ***** Variable Declaration ***** */

int finish_date,
    month,
    day,
    year,
    not_found,
    i;

char chardate[100];

/* ***** Source Code ***** */

```

### C. SOURCE CODE

```

/* Increase the simulation finish date */
if( getbutton( RIGHTMOUSE ) ) *f_date_opt += 1;

/* Key in the simulation finish date */
if( getbutton ( MIDDLEMOUSE ) )
{
    walk_printf ( "Enter the Finish Date : ");
    chardate[0] = 0;
    walk_scanf ( chardate );
    sscanf(chardate, "%d/%d/%d", &month, &day, &year);

    /* Make sure date is valid */

    not_found = 1;
    i = 0;
    while(not_found)
    {
        if( (this_date[i].month == month ) &&
            (this_date[i].day == day) &&
            (this_date[i].year == year) )
        {
            *f_date_opt = i;
            not_found = 0;
        }
        if(i == 699) not_found = 0;
        i++;
    }

    /* Update the main menu */

    sprintf( options[5], " Finish Date - %d/%d/%d ",
            this_date[*f_date_opt].month, this_date[*f_date_opt].day,
            this_date[*f_date_opt].year);
    draw_menu_option( 6, MENU_X_ORIGIN, MENU_Y_ORIGIN,
                    8, options[5], 23, 0 );
}

/* Decrease the simulation start date */
if( getbutton( LEFTMOUSE ) ) *f_date_opt -= 1;

/* Check to ensure that simulation finish date */
/* is not before the simulation start date. */
/* If it is set it equal to simulation start */
/* date. */

if( *f_date_opt < *s_date_opt ) f_date_opt = s_date_opt;

/* Update the main menu */

sprintf( options[5], " Finish Date - %d/%d/%d ",
        this_date[*f_date_opt].month, this_date[*f_date_opt].day,
        this_date[*f_date_opt].year);

draw_menu_option( 6, MENU_X_ORIGIN,
                MENU_Y_ORIGIN, 8, options[5],
                23, 1);

wait_for( 150000 );

finish_date = *f_date_opt;

```

### C. SOURCE CODE

```
/* Return the selected simulation finish date value */  
return (finish_date);  
}
```

```

/*****
/*****
/****
/**** Function Name : simulate.c
/**** Developed By : Jeffrey F. Skolnick
/**** Date : July 17, 1990
/**** For : Partial fulfillment of the requirement for the
/**** degree of Master of Science in Civil Engineering
/****
/*****
/*****
/****
/**** Comments: This function simulates the construction process
/**** using the user defined parameters passed in. These
/**** parameters include:
/****
/**** 1. time - simulation time
/**** 2. logic option
/**** 3. view option
/**** 4. simulation start date
/**** 5. simulation finish date
/****
/**** Calling function: vss.c
/****
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"
#include "walkfiles.h"

```

```

/* ***** Define Structures ***** */

```

```

struct schedule
{
    char obj_name[16];
    int e_start;
    int e_finish;
    int l_start;
    int l_finish;
    int a_start;
    int a_finish;
    int walk_obj;
    int color;
};

```

```

struct obj_walk
{
    int physical;
    int virtual;
    char name[16];
};

```

```

void simulate(options, time, logic_opt, view_opt, s_date_opt,
              f_date_opt, obj_dates, objects, end_count, WALK_ARGS)

```

```

#include "walkargs.h"

```

```

/* ***** Parameter Declaration ***** */

```

```

char options[8][24];

```

```

int *time, /* Simulation time */

```

### C. SOURCE CODE

```

    *logic_opt,           /* Logic option           */
    *view_opt,            /* View option          */
    *s_date_opt,          /* Simulation start date */
    *f_date_opt,          /* Simulation finish date */
    *end_count;           /* Number of objects    */

struct schedule obj_dates[200];
struct obj_walk objects[200];

{
    void sim_act();

/* ***** Variable Declaration ***** */

FILE *object_file;

int phys_obj,            /* Physical object number */
    virt_obj,            /* Virtual object number  */
    obj_posn[200],
    color_obj[],         /* Object color           */
    on_count,            /* No. of obj. to turn on per day */
    curr_obj,            /* Current object          */
    start[200],          /* Start date              */
    finish[200],         /* Finish date             */
    i,j,k,m,p;

    char object_name[16];

    struct schedule object_on[200];
    struct SURFACE_ELEMENT *curr_surf;
    struct SURFACE_ELEMENT *curr_surfl;

    on_count = 0;

/* ***** Source code ***** */

/* Clear the screen of all objects displayed */
if( getbutton ( LEFTMOUSE ) )
{
    for ( i = 0; i < *end_count; i++)
        clear_int4_mask( opt->disp_mask, i);
    update_screen( WALK_ARGS, W_SWAPCLEAR );
}

/* Simulate the construction process based on the */
/* user defined parameters.                        */

if( getbutton ( RIGHTMOUSE ) )
{
    /* Simulate only planned construction activities */
    if (*view_opt == 0)
    {
        /* Use Early start and finish dates */

```

### C. SOURCE CODE

```

if (*logic_opt == 0)
    for(i = 0; i < *end_count/2; i++)
    {
        start[i] = obj_dates[i].e_start;
        finish[i] = obj_dates[i].e_finish;

    }

/* Use late start and finish dates */
else if (*logic_opt == 1)
    for(i = 0; i < *end_count/2; i++)
    {
        start[i] = obj_dates[i].l_start;
        finish[i] = obj_dates[i].l_finish;
    }

/* Use early start and late finish dates */
else if (*logic_opt == 2)
    for(i = 0; i < *end_count/2; i++)
    {
        start[i] = obj_dates[i].e_start;
        finish[i] = obj_dates[i].l_finish;
    }
}

/* Simulate only actual construction activities */
if (*view_opt == 1)
    /* Use actual start and finish dates */
    for(i = 0; i < *end_count/2; i++)
    {
        start[i] = obj_dates[i].a_start;
        finish[i] = obj_dates[i].a_finish;
    }

/* Simulate both planned and actual construction activities */
if (*view_opt == 2)
{
    /* Use early start and finish dates for primary model */
    /* Use actual start and finish date for secondary model */

    if (*logic_opt == 0)
        for(i = 0; i < *end_count/2; i++)
        {
            start[i] = obj_dates[i].e_start;
            finish[i] = obj_dates[i].e_finish;
            start[i + *end_count/2] = obj_dates[i].a_start;
            finish[i + *end_count/2] = obj_dates[i].a_finish;
        }

    /* Use late start and finish dates for primary model */
    /* Use actual start and finish date for secondary model */

    else if (*logic_opt == 1)
        for(i = 0; i < *end_count/2; i++)
        {
            start[i] = obj_dates[i].l_start;
            finish[i] = obj_dates[i].l_finish;
        }
}

```

```

        start[i+ *end_count/2] = obj_dates[i].a_start;
        finish[i+ *end_count/2] = obj_dates[i].a_finish;
    }

    /* Use early start and late finish dates for primary model */
    /* Use actual start and finish date for secondary model */

    else if (*logic_opt == 2)
        for(i = 0; i < *end_count/2; i++)
        {
            start[i] = obj_dates[i].e_start;
            finish[i] = obj_dates[i].l_finish;
            start[i+ *end_count/2] = obj_dates[i].a_start;
            finish[i+ *end_count/2] = obj_dates[i].a_finish;
        }
    }

    /* Save the object's original color */
    for(i=0; i < *end_count; i++)
        color_obj[i] = obj_dates[i].color;

    /*XXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
    /* Begin Simulation */
    /*XXXXXXXXXXXXXXXXXXXXXXXXXXXX*/

    /* Proceed from simulation start date */
    /* to simulation finish date. */

    for(i = *s_date_opt; i <= *f_date_opt; i++)
    {
        printf("day = %d\n",i);

        /* If simulating both the planned and actual sequence */
        /* call sim_act to simulate the actual sequence. */

        if (*view_opt == 2)
            (void) sim_act(&i, s_date_opt, f_date_opt, start,
                          finish, color_obj, obj_dates, objects,
                          end_count, WALK_ARGS);

        /* Process each object for the primary model */

        for(j = 0; j < *end_count/2; j++)
        {
            curr_obj = obj_dates[j].walk_obj - 1;
            curr_surf = &surf[curr_obj];

            /* If construction on the object has started before */
            /* the simulation start date, but has not finished, */
            /* change the objects color to pink (no. 53). */

            if( (start[j] >= *s_date_opt && finish[j] > i) ||
                (start[j] < i && finish[j] > i) )
            {
                while (curr_surf != 0)
                {
                    curr_surf->color = 53;
                    curr_surf = curr_surf->next_surf;
                }
            }
        }
    }

```

```

/* If construction on the object has started after the */
/* simulation start date and finished before the */
/* current date, or if it has started before the */
/* current date and finished on the current date */
/* change the object back to its original color. */
if( (start[j] >= *s_date_opt && finish[j] <= i) ||
    (start[j] < i && finish[j] <= i) )
{
    while (curr_surf != 0)
    {
        curr_surf->color = color_obj[curr_obj];
        curr_surf = curr_surf->next_surf;
    }

/* If the object starts on the current date or if it */
/* starts before the current date and finishes on the */
/* current date, save the object name to the object_on */
/* list. */
if( (start[j] == i) || (start[j] < i && finish[j] >= i) )
{
    strcpy(object_on[on_count].obj_name,
           obj_dates[j].obj_name);
    on_count++;
}

/* Turn on all objects in the object_on list */
if(j == (*end_count/2 - 1) )
{
    for(k = 0; k < on_count; k++)
        for(m = 0; m < *end_count/2; m++)
            if (strcmp( object_on[k].obj_name,
                        objects[m].name ) == 0)
                set_int4_mask(opt->disp_mask,
                             objects[m].physical - 1);

/* Update the screen to show the objects which have */
/* just been turned on. */
update_screen(WALK_ARGS, W_SWAPCLEAR);

/* Pause simulation according to the time value. */
wait_for(*time * 250000);

on_count = 0;
}
}
}

return;
}

```



```

/*****
/*****
/**
/** Function Name : sim_act.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function simulates the actual construction
/** process for the simulate function.
/**
/** Calling function: simulate.c
/**
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"
#include "walkfiles.h"

```

```

/* ***** Define Structures ***** */

```

```

struct schedule
{
    char obj_name[16];
    int e_start;
    int e_finish;
    int l_start;
    int l_finish;
    int a_start;
    int a_finish;
    int walk_obj;
    int color;
};

```

```

struct obj_walk
{
    int physical;
    int virtual;
    char name[16];
};

```

```

void sim_act(i, s_date_opt, f_date_opt, start, finish, color_obj,
            obj_dates, objects, end_count, WALK_ARGS)

```

```

#include "walkargs.h"

```

```

/* ***** Parameter Declaration ***** */

```

```

int *i,
    *end_count,
    *s_date_opt,
    *f_date_opt,
    start[],
    finish[],
    /* Current day
    /* Number of objects
    /* Simulation start date
    /* Simulation finish date
    /* Object start date
    /* Object finish date

```

### C. SOURCE CODE

```

        color_obj[];                /* Object color                */
struct schedule obj_dates[200];
struct obj_walk objects[200];

{
/* ***** Variable Declaration ***** */
    FILE *object_file;

    int phys_obj;                    /* Physical object number      */
    int virt_obj;                    /* Virtual object number       */
    int on_count;                    /* No. of obj. to turn on per day */
    int curr_obj;                    /* Current object               */

    int j,k,m,p;

    char object_name[16];

    struct schedule object_on[200];
    struct SURFACE_ELEMENT *curr_surf;
    struct SURFACE_ELEMENT *curr_surfl;

    on_count = 0;

    /* Process each object for the primary model */
    for(j = *end_count/2; j < *end_count; j++)
    {
        curr_obj = obj_dates[j].walk_obj - 1;
        curr_surf = &surfl[curr_obj];

        /* If construction on the object has started before */
        /* the simulation start date, but has not finished, */
        /* change the objects color to pink (no. 53).        */
        if( (start[j] >= *s_date_opt && finish[j] > *xi) ||
            (start[j] < *xi && finish[j] > *xi) )
        {
            while (curr_surf != 0)
            {
                curr_surf->color = 53;
                curr_surf = curr_surf->next_surf;
            }

            /* If construction on the object has started after the */
            /* simulation start date and finished before the */
            /* current date, or if it has started before the */
            /* current date and finished on the current date */
            /* change the object back to its original color.    */
            if( (start[j] >= *s_date_opt && finish[j] <= *xi) ||
                (start[j] < *xi && finish[j] <= *xi) )
            {
                while (curr_surf != 0)
                {
                    curr_surf->color = color_obj[curr_obj];
                    curr_surf = curr_surf->next_surf;
                }
            }
        }
    }
}

```

```

/* If the object starts on the current date or if it      */
/* starts before the current date and finishes on the    */
/* current date, save the object name to the object_on   */
/* list.                                                  */
if( (start[j] == *i) || (start[j] < *i && finish[j] >= *i) )
{
    strcpy(object_on[on_count].obj_name,
           obj_dates[j - *end_count/2].obj_name);
    on_count++;
}

/* Turn on all objects in the object_on list */
if(j == (*end_count - 1) )
    for(k = 0; k < on_count; k++)
        for(m = *end_count/2; m < *end_count; m++)
            if (strcmp( object_on[k].obj_name,
                       objects[m].name ) == 0 )
                set_int4_mask(opt->disp_mask,objects[m].physical - 1);
}

return;
}

```

```

/*****
/*****
/**
/** Function Name : calc_date.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function prompts the user to enter the project
/** start date. It calculates dates from the entered date
/** to 700 days in the future. It calculates calendar
/** dates given working days by assuming the project will
/** always start on a monday and five day work weeks.
/**
/** Calling function: get_file.c
/**
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"
struct date
{

```

```

    int month;
    int day;
    int year;
};

```

```

void calc_date( s_date_opt, f_date_opt, this_date )

```

```

/* **** Parameter Declaration **** */

```

```

struct date this_date[700];

```

```

int *s_date_opt;
int *f_date_opt;

```

```

{

```

```

/* ***** Variable Declaration ***** */

```

```

struct date start_date,
            day_count,
            date_update();

```

```

char chardate[20];

```

```

int ok_date = 0;
int count;
int new_count;
int j;

```

```

*s_date_opt = 0;
*f_date_opt = 0;

```

```

/* Prompt the user to enter the project start date.  */
/* Read the date to ensure it is formatted properly. */

while (!ok_date)
{
    walk_printf("Enter the project start date (mm/dd/yy): ");
    chardate[0] = 0;
    walk_scanf ( chardate );
    sscanf(chardate, "%d/%d/%d", &start_date.month,
            &start_date.day, &start_date.year);
    if ( start_date.month < 1 || start_date.month > 12 ||
        start_date.day < 1 || start_date.day > 32 ||
        start_date.year < 0 || start_date.year > 99 )
        walk_printf("Incorrect Date Format!\n");
    else
        ok_date = 1;
}

/* Calculate the next 700 calendar days */

this_date[0] = start_date;
for (j = 1; j < 700; j++)
{
    day_count = this_date[j - 1];
    this_date[j] = date_update(day_count);
}

/* Take out all saturday and sunday calendar days */

count = 0;
new_count = 0;
while (count < 700)
{
    if ( count % 5 == 0 && count != 0)
    {
        new_count = count;
        while( new_count < 700)
        {
            this_date[new_count] = this_date[new_count + 2];
            new_count++;
        }
    }
    count++;
}

return;
}

```

```

/*****
/*****
/**
/** Function Name : date_update.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function calculates the next calendar date given
/** the current calendar date.
/**
/** Calling function: calc_date.c
/**
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"

```

```

/* ***** Define Structures ***** */

```

```

struct date
{
    int month;
    int day;
    int year;
};

```

```

struct date date_update(today)

```

```

/* ***** Parameter Declaration ***** */

```

```

struct date today;
{

```

```

/* ***** Variable Declaration ***** */

```

```

    struct date tomorrow;
    /* Increment date by one day */
    if (today.day != number_of_days (today) )
    {
        tomorrow.day = today.day + 1;
        tomorrow.month = today.month;
        tomorrow.year = today.year;
    }

    else if ( today.month == 12)                /* end of year */
    {
        tomorrow.day = 1;
        tomorrow.month = 1;
        tomorrow.year = today.year + 1;
    }

```

C. SOURCE CODE

```
else
{
    tomorrow.day = 1;                               /* end of month */
    tomorrow.month = today.month + 1;
    tomorrow.year = today.year;
}
return (tomorrow);
}
```

```

/*****
/*****
/**
/** Function Name : number_of_days.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function defines the number of days per month.
/**
/** Calling function: date_update.c
/**
/*****
/*****

/* ***** Define Structures ***** */

struct date
{
    int month;
    int day;
    int year;
};

int number_of_days (d)

struct date d;
{
    int answer;

    /* Define number of days per month */

    static int days_per_month[12] =
        { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    /* Check to see if the year is a leap year to determine
    /* the number of days in february.

    if ( is_leap_year (d) && d.month == 2)
        answer = 29;
    else
        answer = days_per_month[d.month - 1];

    return (answer);
}

```



```

/*****
/*****
/**
/** Function Name : is_leap_year.c
/** Developed By : Jeffrey F. Skolnick
/** Date : July 17, 1990
/** For : Partial fulfillment of the requirement for the
/** degree of Master of Science in Civil Engineering
/**
/*****
/*****
/**
/** Comments: This function determines if a year is a leap year.
/**
/** Calling function: number_of_days.c
/**
/*****
/*****

```

```

#include "gl.h"
#include "standard.h"
#include "device.h"
#include "walk.h"
#include "stdio.h"

```

```

struct date
{
    int month;
    int day;
    int year;
};

```

```

int is_leap_year (d)
struct date d;
{
    int leap_year_flag;

    if ( (d.year % 4 == 0 && d.year % 100 != 0 ) ||
         d.year % 400 == 0 )

        leap_year_flag = 1;          /* It's a leap year */

    else
        leap_year_flag = 0;          /* Not a leap year */

    return (leap_year_flag);
}

```

## REFERENCES

- Bent, J. A., and Thumann, A. (1989) Project Management for Engineering and Construction. The Fairmont Press, Inc., Litburn, GA.
- Cleveland, A. B. (1989) "Manipulation of CAD Databases In A Distributed Environment.", Computing in Civil Engineering, Sixth conference., September.
- Cleveland, A. B., and Francisco V. (1988) "The use of real-time animation in the construction process.", Paper submitted to the Transportation Research Board for national meeting for computers in construction., January.
- Groover, M. P., and Zimmers, E. W. (1984) CAD/CAM: Computer-Aided Design and Manufacturing, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Hordeski, M. F. (1986) CAD/CAM Techniques, Reston Publishing Company, Inc., Reston, VA.
- Jackson, M. J. (1986) Computers in Construction Planning and Control, Allen & Unwin Ltd., London, UK.
- Learning dBASE IV. (1988) Ashston-Tate Corp., Torrance, CA.
- Moder, J. J., Phillips, C. R., Davis, E. W. (1983) Project Management with CPM, PERT, and Precedence Diagramming, Van Norstrand Reinhold., New York, New York.
- Morad, A. A. (1990) "KNOW-PLAN: Knowledge-Based Planning System using CAD Technology", a dissertation submitted to the faculty of Virginia Tech in partial fulfillment of the requirements for the Ph.D. degree in Civil Engineering, Blacksburg, Va., May.
- Morad, A. A., and Beliveau, Y. J. (1991) "A Knowledge-Based Planning System", Paper submitted to the ASCE Journal of Construction Management, to be published, March.
- Morad, A. A., and Beliveau, Y. J. (1989) "Visual Scheduling", ASCE Construction Congress, San Francisco, CA.
- Norona, G. F. (1988) "CADD: A New Tool in Construction Management.", Paper submitted to the Transportation Research Board 67th Annual Meeting, Washington, D.C., January.

- Primavera Project Planner. (1987) Volume 1: User Guide., Primavera Systems, Inc, Bala Cynwyd, PA., April.
- Robbins, J. (1989) Understanding dBASE IV Programming., Howard W. Sams & Co., Indianapolis, IN.
- Simons, K. L., Thornberry, H. L., Wickard, D. A. (1988) "Simulation System for Construction Planning and Scheduling." Paper presented at the Jt. ASME/IEEE Power Generation Conference., Philadelphia, PA, September 25-29.
- WALKTHRU Bechtel 3-D Simulation System User's Guide. (1988) Version 3.0., Bechtel Corporation, Bechtel Software, Inc.
- Zabilski, J. R. (1988) "Three-Dimensional Solid Modeling and Database Integration for Construction Project Management.", Paper presented to American Society of Civil Engineers Annual Convention., St. Louis, MO., October 27.

## ***VITA***

Jeffrey F. Skolnick was born on January 27, 1963 in Minneapolis, Minnesota. He graduated from Kempsville High School in 1981. He completed his undergraduate degree in Civil Engineering at Virginia Tech in July, 1985. In January of 1989 he returned to Virginia Tech to pursue a Master's degree in the Construction Engineering and Management Division of Civil Engineering.

The author's work experience includes work as a Structural Engineer for Newport News Shipbuilding. He has also worked as a Computer Aided Design (CAD) Software Test Engineer for the Jonathan Corporation in Norfolk, Virginia. The author also worked as a Graduate Research Assistant for the Construction Engineering and Management Division of Civil Engineering while pursuing his Master's Degree. His work included developing a translator capable of converting CADAM and CATIA computer models to Bechtel WALKTHRU models. The author wishes to pursue a career in the development of CAD systems for construction applications.