Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/TX-92+1139-1F		

4. Title and Subtitle	5. Report Date
INTERACTIVE GRAPHICS INTERSECTION DESIGN SYSTEM: FIRST-STAGE DEVELOPMENT	November 1991
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Thomas W. Rioux, Robert F. Inman, Charles H. Berry, Jr., Clyde E. Lee, and Randy B. Machemehl	Research Report 1139-1F

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Center for Transportation Research The University of Texas at Austin Austin, Texas 78712-1075	11. Contract or Grant No. Rsch. Study 3-8/18-89/1-1139

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Texas Department of Transportation Transportation Planning Division P. O. Box 5051 Austin, Texas 78763-5051	Final
	14. Sponsoring Agency Code

15. Supplementary Notes

Study conducted in cooperation with the U. S. Department of Transportation, Federal Highway Administration

Research Study Title: "Interactive Graphics Intersection Design System"

16. Abstract

The reported research establishes a versatile foundation for initial and future development of an Interactive Graphics Intersection System (IGIDS). Functional requirements, basic capabilities, and computer program structure for the system are established. Candidate hardware and software system components were identified and evaluated for selection of suitable examples used in the first-stage development. An extensive amount of programming was accomplished interfacing the example graphics engine (MicroStation), database engine (Informix), and intersection analysis program (TEXAS Model for Intersection Traffic). Provisions were incorporated to allow implementation of several other computer applications as the need arises. The research demonstrates the feasibility of integrating computer hardware and software components into a user-friendly interactive graphics system supporting the intersection designer.

17. Key Words	18. Distribution Statement
Interactive Graphics Intersection Design System (IGIDS), computer program, hardware, software, components, interface, development, database, analysis	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	52	

INTERACTIVE GRAPHICS INTERSECTION DESIGN SYSTEM: FIRST-STAGE DEVELOPMENT

by

Thomas W. Rioux
Robert F. Inman
Charles H. Berry, Jr.
Clyde E. Lee
Randy B. Machemehl

Research Report 1139-1F

Interactive Graphics Intersection Design System
Research Project 3-8/18-89/1-1139

conducted for

Texas Department of Transportation

in cooperation with the

U.S. Department of Transportation
Federal Highway Administration

by the

CENTER FOR TRANSPORTATION RESEARCH

Bureau of Engineering Research

THE UNIVERSITY OF TEXAS AT AUSTIN

November 1991

NOT INTENDED FOR CONSTRUCTION,
PERMIT, OR BIDDING PURPOSES

Clyde E. Lee (Texas No. 20512)
Randy B. Machemehl (Texas No. 41921)

Research Supervisors

PREFACE

Research Study No. 3-18-88-1139, "Interactive Graphics Intersection Design System (IGIDS)," was initiated in 1988 as a three-year study to define an operational framework for the initial development of a computer-aided intersection analysis and design system. The primary objective of the first-stage development was to identify, evaluate, select, and develop appropriate computer hardware and software that could support engineers in analyzing, designing, and modifying intersections. Emphasis would be placed on using existing interactive computer-graphics and simulation techniques as much as feasible and on developing computer programs to interface selected basic components of the system. When necessary, special computer programs would be written, or adapted, to support activities in the intersection design and operational-analysis process as identified by the study advisory contact individuals.

Throughout the study, communications with these individuals provided valuable guidance to the direction of the research. Beginning in the fall of 1990, an experienced design engineer, Charles H. Berry, Jr., joined the research study team as a participant in the new graduate education program of the State Department of Highways and Public Transportation (now TxDOT). As a full-time graduate student, Mr. Berry was able to participate actively in all aspects of the study and apply his personal experience to IGIDS development. He made many significant suggestions, and, as part of his contribution to the study, conducted a survey of intersection designers in the Department to determine the useful and needed features for the system. These features were prioritized and are being considered strongly in subsequent development phases of the system.

The objectives of Research Study No. 1139 have been realized and are reported herein. Further development and refinement of IGIDS is being accomplished in Research Study No. 3-18-92-1308, "Interactive Graphics Intersection Design System." A versatile foundation for a computer-aided design system to support engineers in designing and analyzing new intersections and in modifying existing ones has been established.

LIST OF REPORTS

Research Report 1139-1F, "Interactive Graphics Intersection Design System: First-Stage Development," by Thomas W. Rioux, Robert F. Inman, Charles H. Berry, Jr., Clyde E. Lee, and Randy B. Machemehl, reports on the three-year study to define an operational framework for the initial development of a computer-aided intersection analysis and design system.

ABSTRACT

The reported research establishes a versatile foundation for initial and future development of an Interactive Graphics Intersection System (IGIDS). Functional requirements, basic capabilities, and computer program structure for the system are established. Candidate hardware and software system components were identified and evaluated for selection of suitable examples used in the first-stage development. An extensive amount of programming was accomplished interfacing the example graphics engine (MicroStation), database engine (Informix), and intersection analysis program (TEXAS Model for Intersection Traffic). Provisions were incorporated to allow implementation of several other computer applications as the need arises. The research demonstrates the feasibility of integrating computer hardware and software components into a user-friendly interactive graphics system supporting the intersection designer.

SUMMARY

Intersection design is a complex process involving a number of different disciplines in transportation engineering. Elements of transportation planning, traffic engineering, and geometric design contribute to the process of designing the best practicable facility for handling traffic at an intersection. An integrated, interactive-graphics-based computer system capable of coordinating selected intersection design applications into a single package will provide a useful tool for the intersection designer.

This report describes first-stage development of the Interactive Graphics Intersection Design System (IGIDS). The requirements, capabilities and structure of IGIDS were determined in the research in addition to defining the type of computer programming necessary for computer graphics and database software management. Use of existing computer applications is emphasized throughout. System development establishes a sound basis for the continued expansion of IGIDS to incorporate additional computer-aided intersection design tools. Modular implementation of design and analysis applications allows staged system development.

IGIDS controls associated graphics and database applications through a higher-level programming language and allows the designer to describe intersection components using either IGIDS or graphics-application commands. Intersection design and analysis applications are accessed through IGIDS to create multiple design alternatives. IGIDS will handle at least five intersection design alternatives. Analysis of the traffic operational aspects of each alternative is achieved by accessing one or more of the analysis packages supported with convenient commands in the system.

MicroStation is the graphics application, Informix is the database engine, and the TEXAS Model for Intersection Traffic is the intersection analysis program selected for initial development of IGIDS. Provisions have been made so that additional applications can be implemented as the need arises. The research demonstrates the feasibility of integrating computer hardware and software together into a user-friendly, interactive-graphics system supporting the intersection designer. Further development of the first-stage IGIDS incorporating user-identified enhancements and emerging technology will provide practical, productive, and efficient support for designing new intersections and modifying existing ones.

IMPLEMENTATION STATEMENT

The first-stage development of an Interactive Graphics Intersection Design System (IGIDS) has linked together selected computer hardware and software components into a system that supports the engineer in analyzing and designing new intersections or in modifying existing ones. A solid and versatile framework for a comprehensive computer-aided intersection design system has been structured, and one example of each of the major system components has been implemented at this time to demonstrate system workability. A recent survey of Department design engineers indicated a number of additional features which are desired for IGIDS. These have been prioritized, and further development to incorporate them into the system is being accomplished under Research Study No. 3-18-92-1308, "Interactive Graphics Intersection Design System," a two-year study which began in September 1991. As this study progresses, it will be desirable to test IGIDS in a few sample situations to guide the system development to a fully-operational status. At this time, the objectives of Research Study No. 1139 have been realized and the research results have been documented in this report, but IGIDS awaits further development before it is ready for implementation into intersection design practice.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

BACKGROUND

Intersection design is a complex process that involves a number of different disciplines in transportation engineering. Elements of transportation planning, traffic engineering, and geometric design contribute to the process of designing the best practicable facility for handling traffic at an intersection. Traditionally, the design engineer has relied upon the application of manual, or sometimes computer-aided, procedures to determine the most appropriate alternative that satisfies the objectives. When computer-assisted tools that support intersection design have been used, they have usually required individual application with awkward transfer of data between programs as the design progresses. An integrated, interactive graphics-based system that is capable of coordinating several selected computer applications into a single package would provide a useful tool for the intersection designer. An overall system that facilitates the use of different applications while also minimizing the cumbersome transfer of data from program to program would significantly improve the process.

In September 1988, the Center for Transportation Research at The University of Texas at Austin, in cooperation with the Texas Department of Transportation and the Federal Highway Administration, entered into an agreement under the Cooperative Highway Research Program to study the development of an Interactive Graphics Intersection Design System (IGIDS). The objective of the study was to define the requirements, capabilities, and structure of the system. Current and emerging technology would be implemented to utilize features such as interactive graphics. Data necessary for execution of different computer analysis applications would be stored and retrieved automatically from a common database. The system would be versatile and user-friendly. Its development would support the use of different types of computer hardware. Utilization of existing computer software applications would avoid duplication of development efforts. The basis for this system is the subject of this report.

OBJECTIVES

The ultimate objective of this study is the development of an Interactive Graphics Intersection Design System (IGIDS) which assists engineers in the analysis and design of isolated at-grade intersections, including diamond interchanges. The different functions which IGIDS would perform were identified early in the study. That identification evolved into definition of the requirements and capabilities of selected functions which the automated system would support. Modular implementation of computer applications was structured to permit staged system development. Candidate computer hardware, operating systems, and software applications were selected after a thorough investigation was made to determine their capabilities, their limitations, and their compatibility with other system components. Conceptual development then proceeded to writing the computer code necessary for managing the various IGIDS system components so that the desired functions could be realized. The objective of IGIDS first-stage development was to establish a solid foundation for continual evolution and expansion of a useful and practical computer-aided intersection design system.

OVERVIEW

Determination of functional performance was achieved through examination of the conventional intersection design procedure. Analysis of the different activities involved in intersection design, together with the general order in which they are accomplished, identified functions which should be incorporated into the IGIDS system.

The selection of system hardware and software components was based on how each component relates to the intersection design process in addition to ensuring access to IGIDS through the type of hardware that is currently used by many public

and private design professionals. IGIDS requires an operating system which allows multi-tasking (the concurrent operation of multiple programs). The selection of the operating system which satisfies that requirement is described in this report.

Generic requirements for the system hardware components were designed to permit use of equipment from different manufacturers. Development of the first-stage version of IGIDS has proceeded on Intergraph Corporation's workstation hardware. Intergraph equipment was selected for its versatility and widespread use among public and private engineering professionals. Additional hardware will be supported in subsequent development of the system.

The software needed to perform the graphics and database operations of IGIDS was selected for its availability on different types of equipment on which the system would likely be used. The first-stage version of IGIDS supports the MicroStation graphics application from Bentley Systems, Incorporated. The database currently utilized by IGIDS will be upgraded to increase its versatility in subsequent stages of system development.

Recently, a survey that was conducted to guide the future development of IGIDS identified several important features for possible implementation. Those design and analysis features which support the intersection design process were evaluated and prioritized for future development [39]. Compatibility with other computer applications which complement the use of IGIDS can be accomplished through continued coordination and planning.

The combination of the hardware components, operating system, interactive graphics and database applications, together with design and analysis software that support intersection design, will provide a useful tool for the engineer. The first-stage development of IGIDS has established a basis for the continued expansion of the system to incorporate additional computer-aided intersection design tools.

CONCEPTUAL DESIGN

The Interactive Graphics Intersection Design System (IGIDS) is intended to assist the engineer in analyzing and designing individual, at-grade, vehicular-traffic intersections, including diamond interchanges. This involves the development of computer-aided tools for defining the geometry of the intersection, the location and type of traffic control devices, and the traffic flow conditions. When implemented, IGIDS will provide a convenient and user-friendly interface for storing and accessing the data needed to execute several analysis and design software packages, and provisions will be made for adding or developing related expert systems.

Figure 1.1 illustrates the various software components which comprise IGIDS as it is perceived in the first stage of development. The functional requirements for each component, and the interrelationship with other components currently included in the design of the system are discussed briefly here; more detail about the system design is presented in Chapters 3 and 4.

IGIDS will use a *graphics engine* (software) to perform all interactive graphics operations and to maintain the *graphics-engine database*. IGIDS software will operate above and drive the graphics engine through a higher-level language interface. IGIDS will allow the user to switch easily between executing IGIDS commands and graphics-engine commands. The commands available within the graphics engine will be used for this purpose as much as possible. IGIDS will not provide any plotting capabilities, but will rely upon the graphics engine to perform these operations.

IGIDS will use an SQL-compliant, relational *database engine* to perform all database storage, retrieval, and reporting operations and to maintain the *IGIDS relational database*. IGIDS software will operate above and drive the database engine through a higher-level language interface. IGIDS will not provide any specialized reporting functions but will rely upon the database engine for this. IGIDS will save all modifications to the IGIDS relational database immediately after each IGIDS command has been executed, or at user-specified times.

IGIDS will handle at least five alternative designs for an intersection. Existing intersection conditions will normally constitute one alternative. Each alternative, and its major graphical component groupings, will be placed on a separate graphical level (or plane), so that it can be displayed independently — or not displayed in a particular view — by the graphics engine. IGIDS will allocate a user graphical level (or plane), and a scratch graphical level (or plane). All, or part, of an intersection alternative can be copied to another alternative, and all, or part, of an intersection alternative can be modified by IGIDS commands.

In its first-stage development, IGIDS graphics will be two-dimensional in plan view and will use a state-plane-coordinate system with no programmed impediments to a future three-dimensional system. Coordinates, distance, and other real numeric data will be stored as 16-significant-digit, 64-bit, double-precision, floating-point variables in

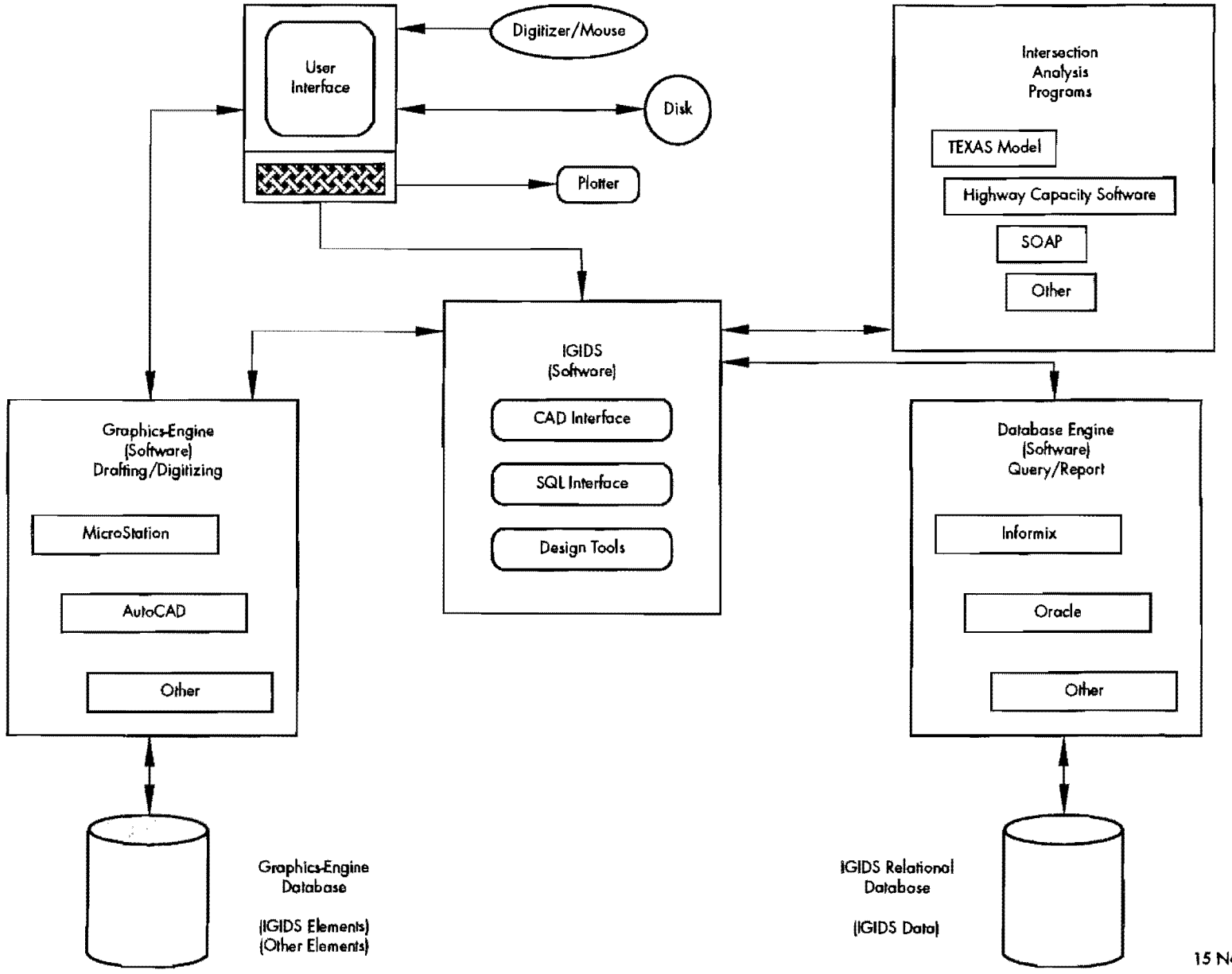


Figure 1.1 Components of the Interactive Graphics Intersection Design System (IGIDS)

feet. All angular data will be stored as the same type variables, but in degrees. All counter, or indexing-type, numbers will be stored as 10-significant-digit, 32-bit, integer-variables. All other integer numbers with no perceived possibility of exceeding several hundred will be stored as 5-significant-digit, 16-bit, integer-variables.

IGIDS will use hierarchical geometry. This means that an item will comprise only one parent item and may have zero or more children items. An item can have more than one parent item type, with the type of parent being associated with the attributes of an item. An item can have more than one category of children. The number of children items accommodated in IGIDS must be virtually infinite. Each parent item will maintain the current number of children items and have a pointer to the beginning and ending child items for each category of children. Each item will have a pointer to the previous, and to the next, item on the list. Most of the higher-level items will serve to group the children items, and only the lowest-level items will have a graphical representation. Any procedure applied to an item will be applied automatically to all of the children of the item.

IGIDS will use relational geometry. Both the absolute and the relative definition of an item will be calculated and stored. The user can enter an item using a relative definition, and the absolute definition will be calculated; or, the user can enter an item using the absolute definition, and the relative definition will be calculated. IGIDS will define the relative item for each type of item. IGIDS will calculate the station and the offset of a coordinate from the leg centerline for all items that are children of the leg. Only IGIDS commands can be used to manipulate the geometry because of the need to update the data in the IGIDS relational database. IGIDS will minimize forcing the user to enter data in a defined order or sequence. To accomplish this objective, IGIDS will automatically sort each list of children items as new children items to be added to the list so that the user can enter geometry data items in any order. IGIDS will automatically set the direction of any entered graphical item so that it will be in conformance with the sorted direction of the list of which it is a part.

Each IGIDS graphical item in the graphics-engine database will contain the ID of the corresponding item in the appropriate structure and IGIDS relational database table where the attribute data will be stored. The type of the graphics-engine element (arc, line, or text) will be used to determine the item type (segment or text) and, therefore, relate it to the appropriate structure or IGIDS relational database table. The ID will be a unique number defined by IGIDS and will be the entry number, the instance number, or the row number in the appropriate structure and IGIDS relational database table. Given an ID, IGIDS can search the graphics-engine database or access the appropriate structure or IGIDS relational database table for the specified item. The higher-level (grouping) items may not have a graphical representation.

The IGIDS relational database will be the master database. All graphics and attribute data items will be contained in the IGIDS relational database, and the value stored there will have precedence over any other value. Thus, the graphics-engine database can be deleted, and IGIDS will be able to re-create the graphics previously entered into IGIDS. Coordinate, distance, angular, and other data in the IGIDS relational database will be considered the definitive values. IGIDS will always use the values in the IGIDS relational database for all calculations. IGIDS will keep the entire IGIDS relational database in memory within the IGIDS software so that no disk I/O will be involved in reading a data item; this will allow the software to operate as fast as possible.

Intersection analysis and design software packages will be executed when the user selects from a menu the software package to be run. IGIDS will check the IGIDS relational database for the appropriate data and prompt the user for any missing data. IGIDS will then extract data from the IGIDS relational database and build the required input files for the software package that was selected. The software package will be executed by the IGIDS operating system as an external or background process, and the user may then use graphics-engine commands to present the output from the executed software package for review. When appropriate, the output from the software package will be displayed by IGIDS.

ABBREVIATIONS

Several abbreviations are used throughout this document. The following list is included here for the convenience of the reader.

(1) AASHTO – American Association of State Highway and Transportation Officials
(2) AT&T – American Telephone and Telegraph
(3) CAD – Computer Aided Design
(4) CPU – Central Processor Unit
(5) DEC – Digital Equipment Corporation
(6) DOT – Department of Transportation
(7) ID – identification; the ID is a unique number defined by IGIDS and is the entry number or instance number or row number in the appropriate structure and IGIDS relational database table

(8) ID_NULL is a "#define" constant which stands for an invalid ID and has a value of -1
(9) IGIDS – Interactive Graphics Intersection Design System
(10) IGrds – AASHTO's Interactive Graphics Roadway Design System
(11) ISAM – Indexed Sequential Access Method
(12) MS-DOS – Microsoft's Disk Operating System
(13) NULL – a pointer to void with a value of zero which is an invalid address
(14) OSF – Open Software Foundation
(15) OS1 – Operating System 1
(16) SQL – Structured Query Language

TERMINOLOGY

Several terms have been adopted and used throughout this document. The following list comprises those with specific application to the first-stage development of IGIDS.

(1) "Graphics engine" refers to the commercial CAD software package which performs all interactive graphics operations and maintains the graphics-engine database; example graphics engines include Intergraph's MicroStation and Autodesk's Autocad.
(2) "Graphics-engine database" refers to the database or external file maintained by the graphics engine; the graphics-engine database contains graphics-engine elements.
(3) "Graphics-engine element" refers to an entry in the graphics-engine database which defines displayable graphics such as an arc, a line, and text.
(4) "Database engine" refers to the commercial SQL relational database software package which performs database storage, retrieval, and reporting operations and maintains the IGIDS relational database; example database engines include Informix and Oracle.
(5) "IGIDS relational database" refers to the SQL relational database maintained by IGIDS through calls to the database engine; it contains the IGIDS relational database tables.
(6) "IGIDS relational database table" refers to a table or entity within the IGIDS relational database; the IGIDS relational database table contains attributes or columns and instances or rows; the IGIDS relational database table is stored and referenced first by instance or row and finally by attribute or column.
(7) "IGIDS relational database attribute or column" refers to a single piece of information stored for each IGIDS relational database instance or row; the IGIDS relational database attribute or column may also be thought of as a column heading where every instance or row has the same information type; the information type may be an integer value, a real numeric value, or a character string; each entry in the IGIDS relational database table will have to be read to get the value of an IGIDS relational database attribute or column for each instance or row.
(8) "IGIDS relational database instance or row" refers to a single entry in the IGIDS relational database table which contains the value for each IGIDS relational database attribute or column; a single entry in the IGIDS relational database table will have to be read to get the value of an IGIDS relational database instance or row.
(9) "Declaration" refers to places where the nature of the variable is stated but no storage is allocated.
(10) "Definition" refers to the place where the variable is created or assigned storage.

SUMMARY

The reported research has established a versatile foundation for initial and future development of an Interactive Graphics Intersection Design System (IGIDS). Functional requirements, basic capabilities, and computer program structure for the system have been defined. Candidate hardware and software components for support by the selected system have been identified and evaluated, and suitable examples have been chosen for use in the first-stage development. An extensive amount of computer code has been written for IGIDS to interface an example graphics engine (MicroStation), database engine (Informix), and intersection-analysis program (TEXAS Model for Intersection Traffic). Provisions have been made for supporting several other existing and potential products as the need arises. The research demonstrates the feasibility of integrating computer hardware and software into a user-friendly interactive graphics system supporting the intersection designer. Further development of the first-stage IGIDS system to incorporate user-identified enhancements and emerging technology will provide a practical, productive, and efficient aid for designing new intersections and modifying existing ones.

CHAPTER 2. INTERSECTION DESIGN

OVERVIEW

The fact that conflicting traffic movements share the same area of pavement in an intersection has long been a concern of the highway designer. Various methods of maximizing capacity and assigning the right-of-way for traffic in at-grade intersections have been used, but the key element to every intersection design is controlling traffic conflicts. The various stages through which intersection design normally progresses in order to control these conflicts are discussed here as they provided a basis for defining the functional requirements for an interactive graphics intersection design system.

In the literature, different authors present the levels of intersection design with varying emphasis. Leisch [22] covers planning and general design techniques in considerable depth, NCHRP Report 279 [11] concentrates on conceptual and operational design of channelization and AASHTO [1] provides the most detail and up-to-date information on operations, concentrating on final design. This chapter reviews conventional intersection design procedures and provides information on analysis methods. It analyzes the procedures involved in intersection design and describes a framework upon which an automated design support system can be developed. Subsequent chapters directly address the intersection analysis procedures and how they may be applied to IGIDS.

FUNCTIONAL DESIGN PRINCIPLES

The functional design principles of intersection design [16] describe the *overall objectives* of an effective design. The principles are stated as follows:

1. Minimize the number of points of conflict.
2. Simplify the geometry of conflict areas.
3. Limit the frequency (or duration) of conflicts.
4. Minimize the severity of conflicts.

The principles essentially present intersection design as the management of traffic path conflicts. They are expanded upon in the following sections with descriptive examples where appropriate.

Minimize the Number of Conflict Points

The nature of an intersection inherently creates conflicts in traffic paths. Conflict has been defined as the demand for a common space on the roadway by two or more users [11]. Conflict points are therefore the locations of those common spaces on the roadway created by various traffic paths. The general types of conflict are crossing, diverging, and merging. Leisch [22] also includes the weaving conflict in his description. All are commonly found at intersection locations as illustrated in Figure 2.1.

The number of conflicts at an intersection should be limited to only those necessary for its efficient operation. Their number increase with the number of legs and allowed movements at an intersection. Reduction in the number of legs or allowed movements will decrease the number of conflict points.

Simplify the Geometry of Conflict Areas

Roadway geometrics should simplify the conflict areas by avoiding unnecessary changes in alignment, profile or use of more than four intersection legs. Simplicity is the key to intuitive driver understanding of desired paths and interpretation of traffic control. Geometry and channelization should compliment the traffic control scheme to increase driver understanding and reduce the chance of a collision.

Limit the Frequency of Conflicts

Conflict frequency can be reduced by decreasing the number of conflicts or minimizing the amount of time a vehicle path conflicts with another. An example of decreasing the number of conflicts is provision of a left-turn lane so turning traffic no longer slows or stops on a through-traffic lane. Roadway geometry that produces a near-right-angle intersection limits the frequency of conflict by minimizing the time the vehicles are subjected to conflicting paths. Figure 2.2 illustrates the concept of conflicting paths at two-way intersections.

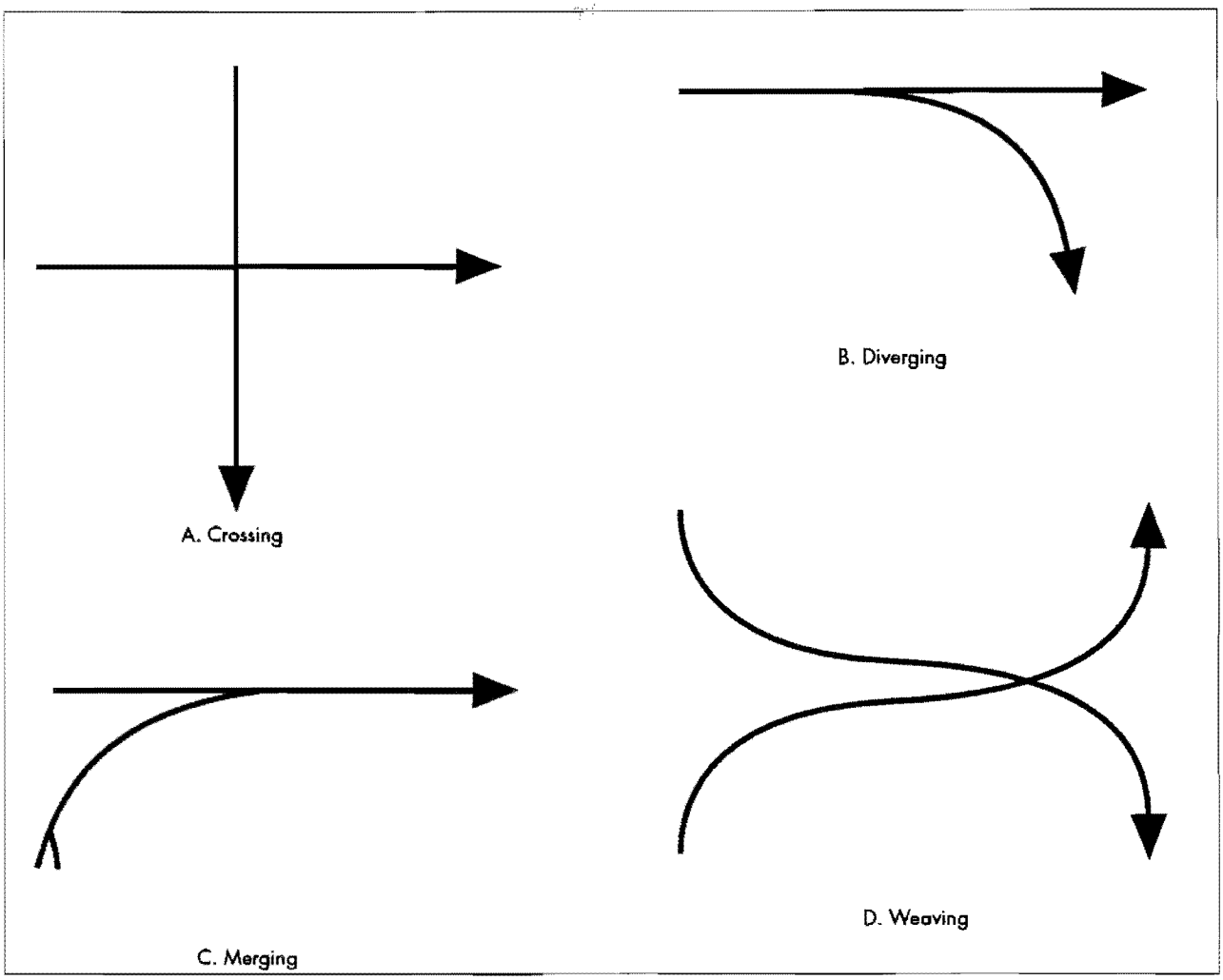


Figure 2.1 Traffic paths showing traffic movement types

Minimize the Severity of Conflicts

The geometry of an intersection affects the severity of possible collisions. Through traffic should cross an intersection at near right angles to reduce the time the vehicle is exposed to the conflict area. Merging or diverging traffic should enter or exit at flat angles to prevent excessive speed reductions on the through lanes. The flat angles also permit merging traffic additional time to select an appropriate gap which can be of a shorter length since a merging vehicle is able to adjust to the speed of the through traffic stream. Similarly, diverging traffic movements made at flat angles can be achieved at speeds near those of through traffic preventing conflicts due to speed differences.

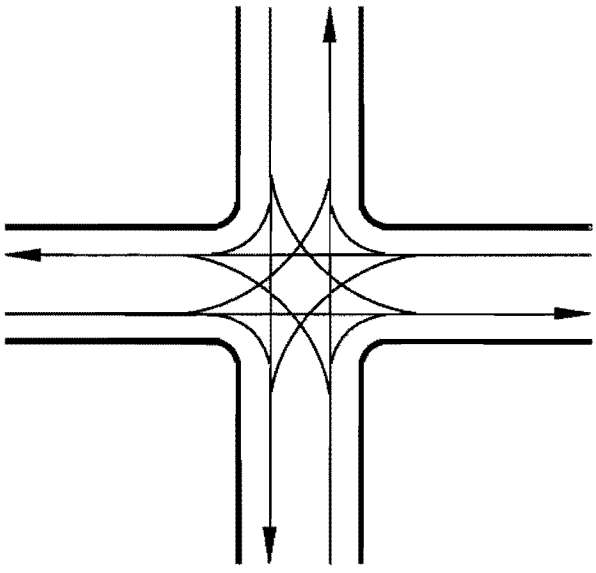


Figure 2.2 Conflicting travel path at four-way two-leg intersection [11]

Levels of Design

Intersection design can be separated into two basic levels, planning and operational, as explained by Leisch [22]. The operational level can be further divided into preliminary design and final design.

As a project progresses through these levels, the degree of detail and the refinement of results increase. The information required to complete each level also changes. When approached in a systematic manner, design can avoid costly rework by managing inputs and analyses within appropriate limits for the particular level of design performed. The designer must also consider the depth of investigation warranted in each of the levels of design as determined by the project's complexity, cost and value of possible benefits.

The objective of intersection construction or reconstruction is to solve a problem identified and defined early in the planning stage. Various alternatives are developed and analyzed during the planning level in order to select two or three that meet the desired objectives so they can subsequently be analyzed at the preliminary design level. The selected alternatives are expanded and analyzed to determine the best alternative for subsequent final design-level development. A more detailed description of information requirements and design detail is explained in the following sections. The levels are diagramed to show their relationships in Figure 2.3.

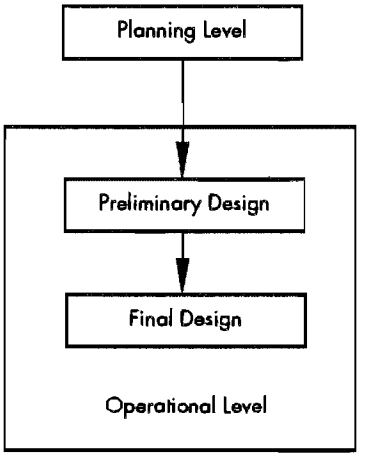


Figure 2.3 Levels of design (showing the chronological relationship of the levels)

Planning Level

The overall objective of design at the planning level is to select two or three design alternatives from a number of possible alternatives for more detailed analysis in the preliminary design. Analysis in the planning level must recognize that a more detailed review will take place in succeeding levels. Preliminary design will be performed only on the cases selected from the planning level that meet the established criteria.

The planning level must consider all significant factors in order to produce satisfactory results. Planning should provide for expected traffic volumes, service quality, staged construction, drainage requirements, construction traffic control plans, constructability, maintainability, special circumstances, and coordination with organizations outside the designer's own organization. Additionally, planning should consider construction specification requirements especially their methods of construction and payment.

All the alternatives developed in the planning level are analyzed with respect to established objectives in order to select two or three for preliminary plan development. The degree of refinement of these analyses must recognize that further analysis is pending. For example, a commonly used rule-of-thumb for establishing preliminary cost is that 20% of the construction items (the major items) constitute 80% of the construction cost. Caution must be exercised not to exclude any major cost items since this would significantly affect the outcome of the analysis.

Planning should determine such things as basic design criteria, physical site data, basic horizontal alignment, traffic volumes, lane use, and consideration of drainage, environment, construction traffic control, operations and special circumstances. Details of the procedure are discussed in the section entitled *Design Procedure*.

Operational Level

The operational level of design is intended to develop more refined details of the intersection design alternatives selected in the planning process and produce the construction plans, specifications and estimate. In the context of this investigation both preliminary and final design are included under the operational level. An important consideration is to adapt the refinement of information required to its particular level of design to avoid rework or redundancy, thereby optimizing the design process without sacrificing the quality of the design. The following sections describe both preliminary and final design while outlining typical levels of refinement used in each case.

Preliminary Design. The objective of preliminary design is to determine the optimum alternative from those selected in the planning level for continued development in final design. The procedure is more detailed than the planning level and requires additional information with a higher degree of refinement. More numerical analysis is

required as compared to the conceptual analysis of details required in the planning level. Preliminary design extends the degree of refinement sufficiently to determine a single selection for final design development. Care should be exercised to prevent rework by analyzing only enough detail to determine the optimum alternative, without excluding or neglecting significant considerations. The alternative selected from the preliminary design level is further developed in the final design stage.

The designer has a significant amount of information already available from the planning level analysis. The planning level analysis determines basic design criteria, physical site data, basic horizontal alignment, channelization, design hourly volumes, movement volumes, right-of-way requirements, and basic consideration of drainage, environmental concerns, construction traffic control, constructability, maintainability, staged construction, specifications, and any special circumstances. The designer should expand on some of these factors and develop others as required within the preliminary design level. Details of the procedure are addressed in the section *Design Procedure*.

Final Design. The objective of final design is to produce the construction plans, specifications, and estimate from the alternative selected as a result of the preliminary design. A significant amount of information is available from previous work. If managed efficiently, that information can be used directly in producing the final design and construction plans without a significant amount of rework or redundancy.

Final design includes complete detailed development of geometrics, profile, channelization, drainage, traffic control (both permanent and for construction), and payment quantities for development of the construction cost estimate. The amount of detail required varies with the complexity of the intersection design. Sufficient accuracy and refinement is necessary, however, to satisfactorily develop the details necessary for construction.

DESIGN PROCEDURE

This section details the various steps required to achieve an intersection design with references to the levels of design described previously. The procedure can be used to determine a rational approach to design which can later be applied to an automated system. The information requirements are deliberately managed to provide a quality design, streamline the process, and prevent rework or redundancy. Some of the procedures may seem intuitive, but a detailed description of the process is necessary to commit to purposeful planning of the actions required in design to prevent unnecessary procedures and emphasize those that create a quality design.

A variety of elements are used to meet the objectives of intersection design including traffic control, traffic islands, street closures, roadway alignment, auxiliary turn lanes, traffic markings, signs, and signals. The elements are concurrently considered with vehicle operation characteristics, driver behavior, and safety considerations. Because all these elements interact to affect the capacity of an intersection and the quality of its service, design is achieved through analysis of possible alternatives. A procedure is suggested in the *Traffic Engineering Handbook* [9] for the design of intersections and repeated here:

1. Obtainment and analysis of traffic data to determine design-hour volumes for all through and turning movements, including future expansion.
2. Obtainment of physical data for the site, including maps showing topography and culture, and plates showing existing buildings and those likely to exist in the future.
3. Determination of the location, type and general design features of all highways, and other development, both existing and planned, in the area which may have a bearing on the design.
4. Preparation of study sketches for several likely intersection schemes that are suitable to meet traffic needs and are practical for the site and design controls.
5. Analysis of alternate schemes and selection of the better two or more for further study and for preparation of preliminary plans and profiles.
6. Preparation of preliminary plans and profiles for the alternates selected under step 5.
7. Evaluation of each alternate preliminary plan with respect to design features, capacity vs. volume, operational characteristics (including suitability for effective signing), overall adaptability, maintenance of traffic during construction, and suitability to stage construction.
8. Calculation of preliminary cost estimates for each alternate preliminary plan, including land acquisition, clearing the site, construction, maintenance, utility changes, maintenance of traffic during construction, etc.
9. Calculation of road-user costs and road-user benefit ratios for each alternate preliminary plan.
10. Joint analysis of values from steps 7, 8, and 9 to reach conclusions as to the preferred plan.

Table 2.1 Design procedure (showing the relationship of design levels to the design procedure)

Level	Activity
Planning Level	1. Obtain and Analyze Traffic Data
	2. Obtain Physical Site Data
	3. Determine Location, Type, and Features of Area and Highways
	4. Prepare Study Sketches
	5. Analyze Alternates to Select Best 2 or 3
Operational Level	
Preliminary Design	6. Prepare Preliminary Plans for Selected Alternatives
	7. Evaluate Each Alternative
	8. Calculate Preliminary Cost Estimate
	9. Calculate User Cost and Benefits
	10. Analyze Alternatives and Select Best One
Final Design	11. Prepare Construction Plans, Specifications and Estimate

11. Final design, including preparation of construction plans, specifications, and estimates.

The procedure consists of a multitude of tasks including traffic volume determination, traffic control, capacity, level of service, construction traffic control, construction, maintenance and user cost estimates. All the tasks must be considered concurrently during design and many of the tasks are inter-related requiring that the design procedure be achieved through an iterative process. The process assumes a design which is analyzed and compared to other alternatives and the desired service characteristics. An optimum design is selected from the alternatives investigated. The procedure outlined above can be classified into the design levels discussed earlier and illustrated in Table 2.1.

Intersection design can be supported by a battery of computer software analysis packages and manual methods specifically developed to provide a designer with information from which to make comparisons and decisions. The analysis procedures are combined with manual or computer-aided-drafting techniques to produce the numerical analyses and drawings necessary for the design of an intersection. The design steps are individually analyzed with references made to the degree of refinement required within each step in order to describe the intersection design process. The procedure will later be used to develop a plan for an automated design system.

Before proceeding into additional design work the primary objective is to identify and define the problem to be solved by the improved intersection. Is the problem one of capacity, safety, environmental concern, a combination of these or something totally different? This step is intuitive yet critical in determining the objectives of the proposed design work. Information at this level should be specific enough to permit analysis of the alternatives and comparison to the objective(s).

The next step is to establish the basic design criteria. Table 2.2 lists typical criteria used to develop intersection design. These criteria can be established by the individual designer, or by committee consensus, to help prevent critical changes to basic values once design has begun. The ensuing design is based on the basic design criteria at a fundamental level. Subsequent revision to established values can create a significant amount of rework. To help prevent unnecessary rework some organizations use a Preliminary Planning Conference to establish basic design criteria in addition to collecting other information [35]. The conference attendees include representatives from various departments who can contribute to developing an informed decision.

Table 2.2 Basic design criteria. Typical list of values to establish before proceeding to design

Functional Class Intersecting Roadways
Design Year
Design Speed
Present Average Daily Traffic (ADT)
Design Year ADT
Desired Level of Service (LOS)
Minimum Acceptable LOS
Design Vehicle(s)

The description of the design process is supplemented with information to identify important points and information requirements. The following sections describe each step and relate them to the level of design in which each exists. The relationship will show the order in which information is processed and the degree of refinement necessary as the design develops. The first three

sections describe procedures whose order can be changed or accomplished concurrently. The primary objective is to make the information available before continuing into detailed design. The procedure can be automated to create an effective tool for the intersection designer.

Obtain and Analyze Traffic Data to Determine Design Hour Volume (DHV) and Movement Volumes

The designer needs to acquire the traffic and turning movement volumes from one of a number of sources which usually depends on the practice of the design organization. Traffic information may be provided from another office within or outside the organization or the designer may be required to develop the information personally. With either of these two methods, or something in between, volumes for each leg of the intersection will be used in design. If current traffic counts are available, an acceptable means of determining the design-year volumes will be necessary. Most frequently the design horizon is twenty years. Growth factor or exponential growth are two common means of determining the design-year value for traffic volumes. If volumes are provided to the designer, they are commonly in the form of future traffic volumes.

Movement volumes from network planning studies should not be used in the preliminary design level of intersections. Movements determined from the various traffic assignment techniques are not accurate enough for microscopic analysis of intersection operations. The values derived from planning studies are more suitable for analysis on the system level for an entire network rather than an individual intersection.

Data should be presented in a manner that specifically defines whether the volumes are current-year or design-year values. The presentation style should clearly signify the total values, movements breakdowns and legs they are associated with to prevent misunderstanding or use of improper values on a specific leg. A sketch should be made with the movement volumes represented schematically in a clearly-understood manner to prevent errors. Frequently North is chosen as facing the top of a sheet or the center line of the major roadway is chosen to run from left to right on the width of a sheet. The roadway names should be annotated along with the cardinal directions or a north arrow. This procedure will help to prevent confusing volumes and legs at a later stage which could cause rework of the design.

Obtain Physical Site Data

There are many sources from which to obtain site data. The range can be described by an example of having a topographical map of the area at the desired scale provided to the designer at their request to requiring an on-site visit personally by the designer before developing design alternatives. Indeed the range varies, and different organizational structures support methods somewhere within the range. This section will not describe where to obtain the information rather it describes the type of information necessary in order to develop design alternatives.

Physical site data should include all elements within the vicinity of the proposed intersection that will significantly affect its design. The degree to which the location of the intersection is already known determines the definition of *vicinity*. If the location is not well defined a preliminary topographic survey will serve temporarily until more detailed information is required. Physical elements such as buildings, roadways, pavements, utilities, drainage structures, fence lines and other structures should be included in the topographic survey. Physical survey control points such as permanent monuments should also be included to provide a permanent reference for the survey.

Once the information is collected, it should be mapped to scale so a two dimensional representation can be established. The scale selected should match that to be used for development of preliminary design sketches of design alternatives. Transparent overlays of the design alternatives can be used to show the relationship of the proposed design to the existing physical elements.

The Preliminary Planning Conference can be helpful in determining physical site data. Terrain, foundation materials, existing right-of-way, utilities, and existing structures are located and their effects are determined. Multiple inputs at this level of investigation are important to prevent subsequent problems or rework. Inappropriate designs are ruled out and as many appropriate designs as possible are proposed for further analysis. The contributions of the group at an early stage are invaluable since departmental specialization can provide important information not easily obtained across department, geographic or specialty lines. The information can be extremely valuable to project development plans by preventing rework or delays and encouraging an informed approach to the solution of the problem.

Determine Location, Functional Class and General Design Features of Nearby Roadways and Development That May Affect Design

Determination of roadway functional classification (or functional usage) along with the surrounding land use allows the designer to produce an informed decision on the probability of specific growth rates in determining future traffic projections and increased capacity. Examination of each factor allows the designer to analyze the *big picture* in relating the intersection to its environment. The analysis will help indicate what degree of staged construction should be recommended or whether ultimate development is required at the time of initial construction.

Usual sources for determining this information include state or local network maps, and master plan studies adopted by the area governments. Official documents can be supplemented with private development plans, however, caution should be exercised to assure the reliability of the information with respect to whether improvements will be implemented.

Prepare Preliminary Sketches of Alternatives

Preliminary sketches begin as single-line schematics similar to signal phase drawings that address the needs determined by traffic volumes and site investigation. The single-line schematics evolve into required numbers of lanes and their usage. A preliminary capacity analysis is appropriate at this stage where average service capacities per lane are used to determine the approximate number of lanes that will be required on each leg of the intersection. Through-traffic volumes are assumed to reverse for the opposite direction of peak flow; so, the number of through lanes on the same roadway are equal in both directions. The number of auxiliary lanes, however, can be different on each approach due to turning demand. A staged development plan can be investigated at this level although it will be directly addressed later.

Preparation of the preliminary sketches requires sufficient information to compare each alternative to the stated objectives for selection of two or three alternatives for more detailed development. Until reaching the more refined development in the preliminary design, preliminary sketches should be kept to a minimum with only enough detail to define the intent. Sketches should be to scale, but "free hand" sketch quality is recommended by Leisch [22] to minimize development time. Typical design elements include channelization, traffic control, pavement width, lane designation, signing, drainage, construction traffic control plan, and cost estimates.

Intersection angles are analyzed for compliance with functional design principles. Crossing paths should intersect at near right angles and merging paths should intersect at flat angles. Consideration of the effects of vertical alignment combined with the horizontal alignment should also be made at this level for critical or complex designs.

Once one or more basic alignments are established the design hour volume (DHV) and movement volumes (left, through, right) are superimposed on the diagrams. If the number of lanes are known service capacity can be determined or if service capacity per lane is known the number of required lanes can be determined. Either of these approaches will establish the plan width of the intersection to determine approximate right-of-way requirements.

Channelization is a major consideration and deserves additional explanation. The "Intersection Channelization Design Guide" [11] provides nine recommended principles of channelization that are repeated below:

1. Undesirable or wrong-way movement should be discouraged or prohibited through channelization.
2. Desirable paths for vehicles should be clearly defined by all elements of the intersection.
3. Desirable and safe vehicle speeds should be encouraged by the design of the intersection.
4. The design of the intersection should wherever possible separate points of conflict.
5. Traffic streams should cross at near-right angles and merge at flat angles.
6. The design of the intersection should facilitate the movement of high priority traffic flows.
7. The intersection should facilitate its scheme of traffic control.
8. The intersection should accommodate decelerating, slow, or stopped vehicles outside higher speed through traffic lanes.
9. Safe refuge from motor vehicles for pedestrians, handicapped, and others should be provided where appropriate.

The report's repeated use of the word "should" implies recommended usage and not a requirement. This author believes its use suggests the desire to provide information guidelines without unduly increasing the liability of a designer. This is caused, it is believed, by the current litigious environment of our society.

The guide [11] continues by providing a list of various elements of channelization design used to achieve the previously stated principles as follows:

1. Designation and arrangement of traffic lanes
2. Traffic islands
3. Median dividers
4. Corner radii
5. Approach geometry
6. Pavement tapers and transitions
7. Traffic control devices (signs, signals, etc)

A good channelization plan reinforces the selected traffic control devices. The geometry, channelization and traffic control are all closely related. Changes in one usually affect the others. Table 2.3 [11] illustrates the relationship between the design elements and channelization principles.

Analyze and Evaluate Alternatives and Select Two or More of the Better Ones

Analysis of the alternatives developed in the previous steps should be limited to determining two or three alternatives that best meet the desired objective to solve the problem at the intersection.

The Highway Capacity Manual (HCM) [2] describes a planning-level capacity analysis for signalized intersections. The procedure provides basic information related to the suitability of a design with respect to capacity. The alternative either exceeds capacity, is near capacity or is under capacity. Stop sign controlled roadways can be analyzed by procedures given in HCM Chapter 10, or all-way-stop control can be analyzed by the method proposed by Kyte [17]. These procedures are supported by computer software analysis programs.

The selected planning-level alternatives should be developed for analysis in the preliminary design. The designer should resist the temptation to stop developing planning-level alternatives once an "obviously" optimum design is reached since this will suppress achievement of better designs which may require innovative solutions. General consideration should be given to financial constraints at this level, but they should not be too detailed since cost will be addressed in the next level of design development.

Environmental considerations such as air, noise, water, and wetlands will also have an effect on the design selected so these items must be considered in the planning level. The 1990 Air Quality Bill could have a significant effect on plan development, especially in areas where air quality standards have not been met. Likewise additional analysis procedures will be required to address surface runoff water quality as proposed by the Environmental Protection Agency (EPA).

Leisch [22] recommends only overall aspects be considered at the planning stage, with visual or mental checks of the more specific aspects. AASHTO states capacity is one of the most important considerations in intersection design [1]. The priorities and weights of the desired attributes for use in ranking alternatives can be determined either objectively or subjectively. Which ever method is selected for comparison, the alternatives developed in the planning level will be compared to each other and the best two or three will be selected for further investigation.

The selection of the best two or three alternatives is the last activity included in the planning level. A flow chart describing the procedure to this stage is provided in Figure 2.4.

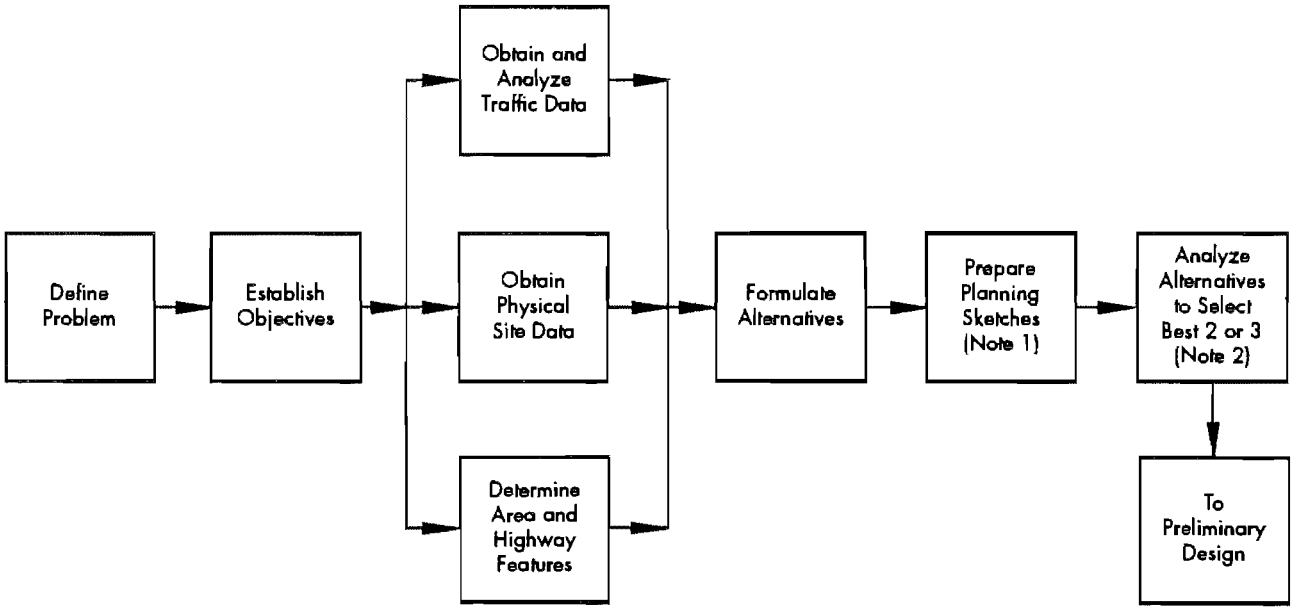
Prepare Preliminary Plans and Profiles for Alternatives Selected In Step 5

Initially the physical site conditions should be reverified and subsequently expanded upon, especially for urban projects. Utilities or other physical elements not previously considered must be analyzed for their possible impact on the design. Drainage requirements should be reviewed to continue development of the basic drainage plan for each alternative. The profile considered in the planning level is assigned approximate elevations that complement the horizontal alignment and drainage requirements. The drainage plan is considered in the development of the profile to minimize cost yet function efficiently. Intersections usually have flatter grades than the open highway so additional drainage structures are frequently required to prevent undesirable ponding on the roadway, especially when curbs are provided. Detailed location of structures, however, is temporarily deferred until precise roadway widths, cross section elements and profiles are determined. For reconstruction projects, the drainage plan is usually controlled by existing requirements to match existing profile control points such as driveways, utilities or adjacent drainage facilities.

Lanes, shoulders, curb, transverse slope and other cross section elements not established in the planning level are now determined. Usually the same cross section information is used in the analysis of all the preliminary design alternatives although not necessarily always. The cross section

Table 2.3 Relationship between design elements and channelization principles from "Intersection Channelization Design Guide" [11]

Principles	Traffic Lanes	Traffic Islands	Median Dividers	Corner Radius	Approach Geometry	Pavement Tapers/ Transition	Traffic Control Devices
Prohibit Movements		X	X	X	X		X
Define Vehicle Paths		X	X	X	X	X	
Promote Safe Speeds				X	X	X	
Separate Conflicts		X	X		X		X
Cross and Merge Angle		X	X		X	X	
Facilitate Priority Movements	X			X	X		X
Facilitate Traffic Control	X	X	X				X
Accommodate Slowing Vehicles	X					X	
Safe Pedestrian Refuge		X	X	X			X



Note 1: Sketches Should Include:
- Curvature
- Channelization
- Lanes, Tapers, and Transitions
- Curb Returns
- Approx Vertical Profile
- Approx Drainage Str Locations
- General Traffic Control Scheme
- Signs and Pavement Markings

Note 2: General Analysis Includes:
- Satisfaction of Objectives
- Estimated Cost
- Lane Requirements and Usage
- Capacity and Level of Service
- Right-of-Way Requirements
- Land Use Impacts
- Lane Continuity
- Construction Requirements
- Environmental Concerns
- Consider Special Circumstances

Figure 2.4 Planning level—design procedure flow chart

information will establish the usual roadway section which may be supplemented by auxiliary lanes, channelization or special transit facilities.

Consideration is also given to use of the intersection by traffic other than motor vehicles such as pedestrians, handicapped persons (for wheelchair access) and bicycles in continued development of the design. Common factors include roadway crossing width for pedestrians, wheelchair ramps and sidewalk locations for handicapped persons and special lane designations for bicycles.

The channelization developed in the preliminary sketch should be reviewed and described sufficiently to provide a clear representation of the intersection. The principles of channelization should be followed. Channelization design should meet the objectives of optimizing the operational quality of the intersection, increasing its safety by minimizing accidents and decreasing accident severity.

Evaluate Each Alternative with Respect to Desired Features

Operational characteristics are analyzed to determine speed requirements of the turning movements. High-volume turning movements should be given specific consideration. High speed turns require large radii and should be designed for operational speeds which approximate 0.7 of design speed. Low-speed turns (speed less than 10 mph) require pavement edge designs to accommodate vehicles operating at their minimum turning radii and maximum offtracking paths. The decision should be made on which type of turn to accommodate. High-speed turns typically require more right-of-way and channelization to define the desired vehicle path. Low-speed turns, however, provide lower capacity and cause vehicles to slow before turning, thus affecting safety by causing differences in operating speeds. Once the decision has been made which type of turn to provide, the intersection radii, turning roadway widths and superelevation rates can be established.

The geometric design of intersection radii, turning roadway widths and superelevation rates can be established once the speed-curvature information is determined. The preliminary plan level should determine the magnitude and geometric location of the radii to analyze affects to the site but survey quality locations are usually not required until the final design stage. Vehicle turning templates and AASHTO policy [1] are used to determine intersection radii for the design vehicle. The results should be accurate enough to represent the intersection design alternative in a scaled drawing so effects of the topography, roadway alignments and right-of-way can be determined.

An initial review of right-of-way requirements should be performed before proceeding further. The review is intended to use roadway alignments, lane configurations, widths, channelization and border requirements for establishing the approximate right-of-way lines so the effect on physical site elements can be analyzed. At this level the right-of-way analysis should be used to identify significant elements requiring adjustment. Recommended clearances are used in approximations since the precise locations of right-of-way lines will be determined upon identification of the alternative selection for final design. At this level the right-of-way analysis is primarily used to identify problems that affect further plan development. Inspection should include ground level, underground and overhead obstructions, any of which can affect design. The designer may typically want to be within five feet of the final right-of-way location.

Construction traffic control should be evaluated next especially if the intersection involves reconstruction of an existing facility. Provisions must be made to provide for existing traffic either through or around the construction area. The construction traffic control plan (TCP) accounts for traffic patterns and construction phases to provide for concurrent activities in the travel lanes and the construction area. The simplest TCP is to provide an alternate route, detour traffic and close the intersection during construction. The analysis should determine if the additional traffic on the detour route will degrade the quality of service or safety considerations beyond an acceptable limit. Detours, however, are frequently not possible and it may be necessary to allow traffic through the intersection area during construction. A traffic capacity analysis is recommended to assure acceptable quality of service is achieved. Many methods are used to accomplish a TCP including staged construction, realignment of traffic lanes, narrowing lanes, over-constructing to provide additional room for lanes, modifying signal timing, night construction, and others. Different geographic areas may have preferences for methods that work in their particular applications. Various methods may be justified to accommodate the TCP. If necessary, each procedure can be included in the economic analysis of the particular design alternative. Since the TCP itself can have alternatives it is possible to get multiple TCP alternatives for each intersection design alternative, for example, alternative Design 1 with TCP alternatives a, b, and c, and alternative Design 2 with TCP alternatives d, e, and f.

Once the geometric layout of the intersection is decided, the designer can proceed to a preliminary design of the permanent traffic control and perform an operational analysis to determine the level of service (LOS). This step applies to all traffic control schemes to some degree but is usually emphasized for signalized intersections. The traffic signal timing plan is designed using traffic volumes, lane designations and movement volumes, which have already been established. Manual methods, charts, and computer software are frequently used to support signal timing design. With the data, the ratio of green time to cycle length can be calculated, so the capacity and level of service (LOS) of each approach can be determined.

A more detailed investigation of right-of-way requirements is now in order. The designer should be able to determine whether the design alternative will fit within proposed right-of-way limits or determine the amount of right-of-way required to construct the alternative. A second review of the construction traffic control requirements is also recommended especially if construction will be accomplished under traffic conditions. The second review is intended to identify and solve any problems with the geometric features developed since the earlier review. The second review should be more in depth than before.

Although safety is always a consideration of design, a deliberate review for safety aspects is warranted and should be undertaken at this stage. The channelization principles discussed earlier include some safety criteria and now that the design alternative is nearly complete, an overall safety analysis can be performed. Since many of the channelization principles apply only to specific areas of interest, the overall configuration of the intersection should be investigated and rated once the complete geometry is determined. This step becomes even more critical when complex intersections, higher speeds, or a high degree of channelization are proposed. A three-dimensional analysis supported by manual drafting techniques, computer-aided drafting or mental images can be utilized to help discover sight obstructions, complicated channelization or other detrimental features occasionally designed into a complex intersection.

Additional intersection elements should be designed before continuing such as basic signing, pavement markings, and lighting. The amount of detail required to establish the designs should support the preliminary nature of development to this point. Sufficient information is necessary to determine a reasonable cost of the work.

Prepare Preliminary Cost Estimates for Each Alternative

Preliminary cost estimates should include construction and operational costs to effectively compare preliminary design alternatives. Construction costs should include right-of-way, right-of-way preparation, and construction specification items. Operational costs should include maintenance items such as overlays, crack sealing, and miscellaneous items such as power requirements for lighting and traffic control systems. The time period for the operational costs may also be needed to amortize dollar values in the selected economic analysis procedure.

Determine User Cost

User costs are usually associated with intersection delay determined in capacity analysis and a unit cost or value of travel time related to fuel consumption, vehicle depreciation and maintenance. User cost can also include collisions or more conceptual costs such as traffic fatalities, aesthetics, or community values. A value can be determined per unit of measurement so the total user cost can be calculated for each design alternative. The total values are then associated to their respective design alternative for analysis in the next step.

Perform Joint Analysis of Values from Steps 7, 8, and 9 and Determine Best Plan

The preliminary design for each intersection design alternative is now complete enough for economic analysis. Design has determined quantities of construction items. Capacity analysis determined user benefits in reduced delay, expected reductions in accidents, or other measurements and an estimate of the operation costs can be established. The benefit-to-cost ratio (b/c), equivalent uniform annual cost (EUAC), or present worth (PW) economic analysis procedures can be used to assign an objective value to each intersection design alternative for economic comparison. Any number of analysis procedures can be selected. Some organizations have requirements regarding which method should be used. The objective, however, is to use a common measurement device to compare multiple alternatives in order to select the most effective solution. The desired objective may influence the selection of an economic

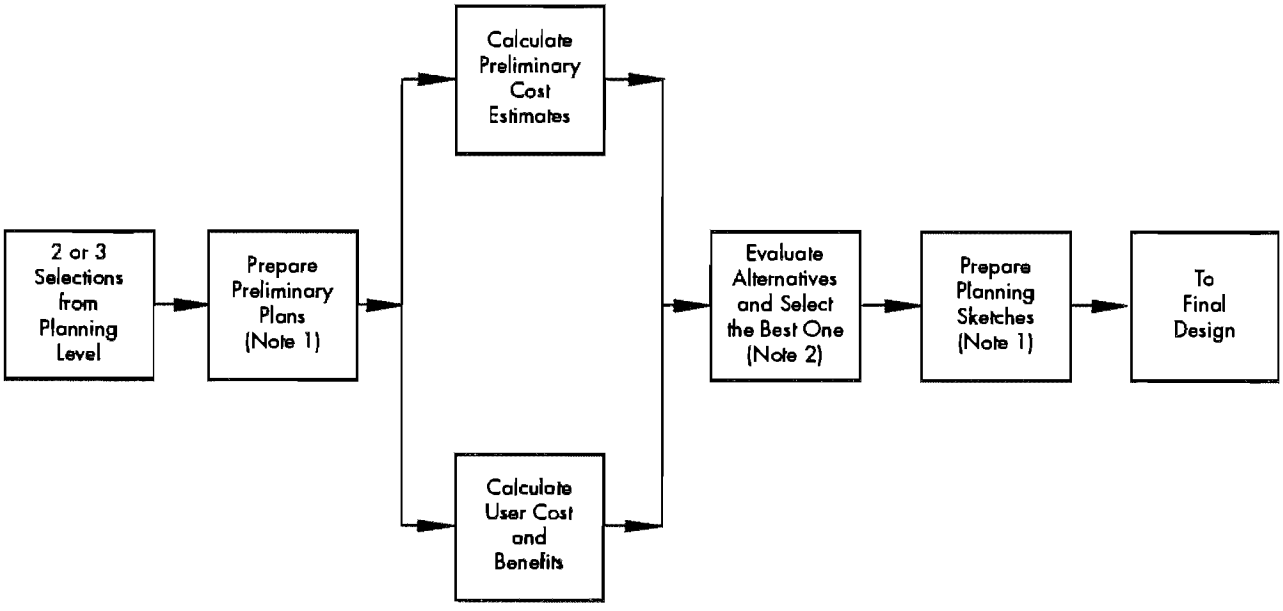
analysis procedure. The alternative selected by the comparison is the design that will continue into the final design level to develop the construction plans specifications and estimate for the intersection.

The results of the economic analysis will yield a single selection for development in the final design level. It is recommended that once this information is available any additional right-of-way required for construction should be described sufficiently for acquisition procedures to begin. Since right-of-way requires a significant time period to acquire, the procedure should begin as soon as possible. Once a preliminary design alternative is selected for final design development all the necessary information is available to proceed. It is also possible that the appropriate information such as intersection location and right-of-way requirements is determined at an earlier stage. Right-of-way acquisition, which includes plans of required right-of-way, legal descriptions, and appropriate surveys, should begin at the earliest possible opportunity. Immediately following selection of the optimum preliminary design alternative is probably the latest time project development would like to begin the process. Caution should be exercised in proceeding too early before substantial information is available to prevent prematurely acquiring too little right-of-way. This is especially important for urban construction in densely populated areas where right-of-way costs can be very high. For rural projects where right-of-way is usually less costly, a more conservative view can be taken in acquiring sufficient area for future development.

The selection of the best alternative is the last activity included in the operational level for preliminary design. A flow chart describing the procedure to this stage is provided in Figure 2.5.

Final Design of Selected Alternative

The objective of the final design level is to develop the plans, specifications and estimate necessary to construct the proposed improvements. The information developed in the preceding levels is expanded upon to develop the design without rework or redundancy. The 1990 AASHTO Policy on Geometric Design of Streets and Highways [1] provides a very good description of requirements for geometric design and serves as a good



Note 1: Evaluation Includes:
- Construction Methods
- Construction Control Plan
- Capacity and Level of Service
- Coordination with Nearby Intersections
- Benefit and Cost Analysis
- Combined Horizontal and Vertical Alignments

Note 2: Preliminary Plans Include:
- Right-of-Way Requirements
- Land Use Considerations
- Graphically Sketch Horizontal Alignment to Scale
- Graphically Sketch Vertical Alignment
- Use Approximate Elevations
- Cross Section and Related Information
- Drainage Plan
- Pedestrian, Wheelchair, and Bicycle Items
- General Construction Traffic Control Plan

Figure 2.5 Operational level—design procedure flow chart for preliminary design

reference for recommended practice. The principal aspects are discussed in the following paragraphs.

The preliminary design of the intersection has established the intersection configuration. The information will be used to develop details necessary for ultimate development. A review of the existing physical features at the site is appropriate at this stage to assure that the design conforms to expected results. A review of significant details in the design is also recommended because it is much easier to make corrections here than at any later stage. A conceptual yet deliberate overall review will be time well spent.

Before proceeding to more details the designer should consider special or unusual conditions and how they may affect the final design of the facility. Special notes can be made for later reference once design of affected elements is reached. A general perspective of the intersection is helpful in determining effects from unusual conditions.

The alignment established in the preliminary design level was essentially determined by graphical procedures. It is necessary to define the alignment with coordinates so additional features and elements of the intersection can be related to the survey center line(s). The center line should be referenced to permanent monuments with either a State Plane Coordinate System or relative project coordinates depending on the complexity of the construction and the needs of the designer. Bearings or azimuths are calculated for individual alignments. Horizontal curvature is also described once overall tangent alignments are established. Horizontal points of intersection (PI's), points of tangency (PT's), points of curvature (PC's) and if necessary points on tangent (POT's) are located by coordinates and referenced to the center lines by station. The horizontal control of the intersection is an important component of the design because many design elements are referenced to it. Extreme care should be exercised in assuring proper descriptions are made or a substantial amount of work may have to be redone later.

Construction materials affect some of the design elements therefore it will be necessary to describe some materials in sufficient detail for design to continue. The paving material and type of curb, for example, affect the location of the pavement edges. Concrete pavement is typically placed to the back of the curb while asphaltic concrete pavement can be placed to the inside or outside edge of the curb or somewhere beyond the back of the curb. Precast elements such as inlet boxes may require additional pavement area or special pavement considerations. Consideration and deliberate review will help to prevent rework of affected areas at a later stage.

The cross section developed in the preliminary design level is used to develop the horizontal and vertical (profile) control of the intersection. Once the designer has established the cross section elements they can be used to define the locations of pavement edges. Those edges not parallel to the center line should be defined with sufficient detail to accurately describe their location. Frequently pavement edges are not parallel to the center line near intersections due to pavement tapers, auxiliary lanes, channelization, drainage structures, or transit facilities. Since it will be necessary to later investigate pavement edge profiles, the location of the edges should be defined with respect to the alignment center line or profile grade line. Although pavement edge elevations are frequently shown in intersection plans, their profiles should be reviewed especially if not specifically provided in the construction plans to assure smooth lines and grades that drain. The pavement edge alignments will also serve in construction staking for curb and gutter placement or to define pavement limits.

Median and turn-island edges are located for reasons similar to those for pavement edges. For curbed islands, it is necessary to designate curb locations, while for unpaved or flush islands it is necessary to define the pavement edges for construction staking. Various methods can be used to locate the islands, but whichever are chosen, the plan presentation style should provide a complete yet concise representation of the layout for construction. Station and offsets to the center line are frequently provided since alignments have been previously established with survey monuments. Tabular presentation can also be used to help prevent cluttering the design drawings with additional text. Standard designs from a detail sheet are frequently not practical since alignment of channelization is specific to a location and does not lend itself to standardization.

The design then proceeds to the vertical control elements of the geometric plan. The center line vertical profile is designed to provide sufficient sight distance, drainage, driving comfort, and to match existing conditions when necessary. The vertical control elements should complement the horizontal control elements, as described in the AASHTO Policy [1], to prevent undesirable lines of sight or hazardous conditions. Design of the vertical profile should also account for drainage, existing physical requirements, and the interaction of the roadway cross slopes on the various intersection legs. The intersecting roadways' profiles and cross slopes frequently form warped planar sections different from their typical roadway sections. A constant cross section through the intersection is

not required, but a smooth plane is necessary for satisfactory operation, rideability and safety. When all the intersection legs cannot be accommodated with their usual cross sections the major roadway controls. The center line profile grade, cross slope, superelevation, and pavement width are used to determine the profile of the pavement edge. Pavement edge profiles are established from calculated values and smoothed to provide a desirable profile for drainage and aesthetics. They should conform to minimum or desirable drainage requirements to minimize runoff ponding within the vicinity of the intersection.

Once pavement elevations are determined the drainage structures roughly located in the preliminary design stage are permanently located to prevent undesirable encroachment of flow along the pavement edge onto the driving lanes. The inlet structures should be spaced as needed along profile tangents and at pavement low points (sumps). The connecting conduits are subsequently located to minimize material, labor and effects to existing underground facilities such as utilities. When locating these elements consideration should be made for other ground penetrations not yet designed such as other storm drain networks, relocated utility services, or foundations for signals, lighting, and large signs.

The size of the drainage inlets and conduits are determined next. Since intersection design usually involves smaller drainage areas (less than 300 acres) the Rational Method is usually used with appropriate runoff coefficients and times of concentration. When the conduit sizes and top of inlet and outlet elevations are determined, the design of the underground system can proceed to determine invert and soffet elevations for inlets, junctions, and conduits. Clear cover over pipe conduits is frequently a critical consideration. Standard pipe sizes are recommended for economic reasons but odd sizes, arch or elliptical pipes can be used for critical situations. Different bedding methods can be used to strengthen conduits that are located near the surface to strengthen them but these methods should be avoided when possible for economic reasons. Excavations in excess of five feet of depth require a positive means of trench protection from cave in during construction. Various methods can be employed such as shoring, movable box protection, or laying back the excavation slopes where soil conditions and space permit. An economic analysis of the alternatives may be necessary for large drainage systems or complex situations with substantial savings potential.

The traffic control scheme determined in the preliminary design level should be finalized. Traffic signal control is the most complicated method and it will be addressed here since stop or yield control methods principally involve sign placement and pavement marking. The major elements of the signal design after the signal timing plan has been decided are the determination of mast arm lengths, and locations of supports, controller cabinet, conduit, junction boxes, pull boxes, and traffic detectors where applicable. Many of the installations are underground and must be coordinated with locations of other underground components. Since mast arms increase in cost with increases in length, if a signal support foundation conflicts with another underground element it may be best to relocate the other element depending on the cost differences. Conduit runs are relatively inexpensive to construct *before* pavement materials are placed so conservative estimates on the amounts or sizes needed are usually cost-effective over the life of a facility. The conduits are relatively fragile, however, when confronted with the types of equipment used in highway construction. When constructed early in the project they should be well referenced and marked. Conduits destroyed by construction operations and not replaced before paving will require a great deal more effort and cost to be replaced. Recommended maximum run lengths between junction boxes or pull boxes should be adhered to so problems do not develop when placing electrical conductors.

Intersection appurtenances include such items as fixed source lighting, bus stops, sidewalks, and wheelchair ramps. Lighting is the most important of these features because it involves safety. Isofootcandle templates are available from lighting manufacturers for use in determining the locations of lighting supports. A print of the plan view and appropriate illumination templates are used to assure sufficient coverage by the light sources. Power cables are designed for specific loads and run lengths. They can be underground or overhead. Care should be exercised in locating the light poles to minimize the probability of being struck by an errant vehicle. The 1990 AASHTO Policy [1] recommends that break-away light foundations not be used in busy commercial areas especially when significant pedestrian traffic is expected since the secondary impact caused by a falling light support could create more damage than the initial impact with the vehicle. Bus stops, sidewalks and wheelchair ramps should be closely coordinated since they all affect pedestrian traffic. Their layouts should complement each other.

Interfaces and interactions between design elements should be investigated to help prevent problems especially during the construction or operation stages. Since typical conditions are frequently used in design, interfaces of dissimilar

components or materials are sometimes overlooked. A deliberate effort should be made to assure elements or components connect, react, or abut correctly. A typical example may be the specification of a joint seal material that is incompatible with the pavement. The two should have been considered simultaneously for compatibility.

Miscellaneous items typically involve project specific requirements that enhance the intersection yet may not significantly affect its operations or safety. A regional or municipal requirement may create additional construction items not typically considered in intersection design. Municipal franchise agreements with utilities may require additional empty conduits be placed with any utility modifications. Significant increases in underground placements could affect clearances between facilities.

Once the design is complete, the payment quantities should be calculated for the purpose of inclusion in the construction cost estimate. Quantities determined in the preliminary design level are helpful to assure that major items are not omitted from the final design quantity estimate. Additionally, approximate quantities measured by scaling or similar means can be prepared as the different segments in final design are completed. The approximate quantities are useful for updating the preliminary design estimate on a regular basis and as a check of final calculated quantities at the conclusion of the plan work. A methodical approach is used to prevent duplication or omission of quantities for payment. Estimates tabulated by plan sheets, sections, work divisions or functions can be used. An automated computer-based procedure is helpful in preventing math errors and transposing data. The presentation format for the estimated quantities varies, but a summary of the quantities by location is useful in determining material requirements and identifying discrepancies between field measured quantities and office estimates.

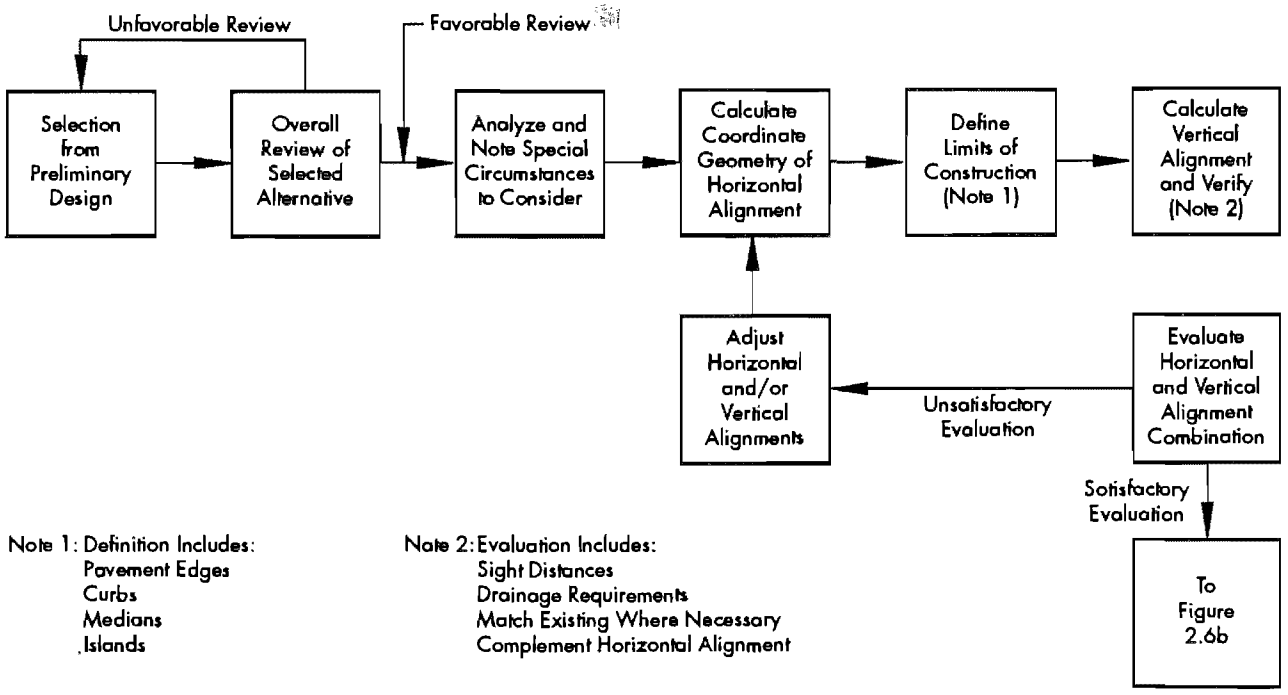
Once all the information is compiled, it can be transferred to the contract documents. The Texas Department of Transportation utilizes an automated system for transferring information from the estimate prepared in the design stage to a plan sheet format of estimated unit quantities.

The completion of the contract documents is the last activity included in the operational level for final design. A flow chart describing the procedure to this stage is provided in Figure 2.6.

SUMMARY OF INTERSECTION DESIGN

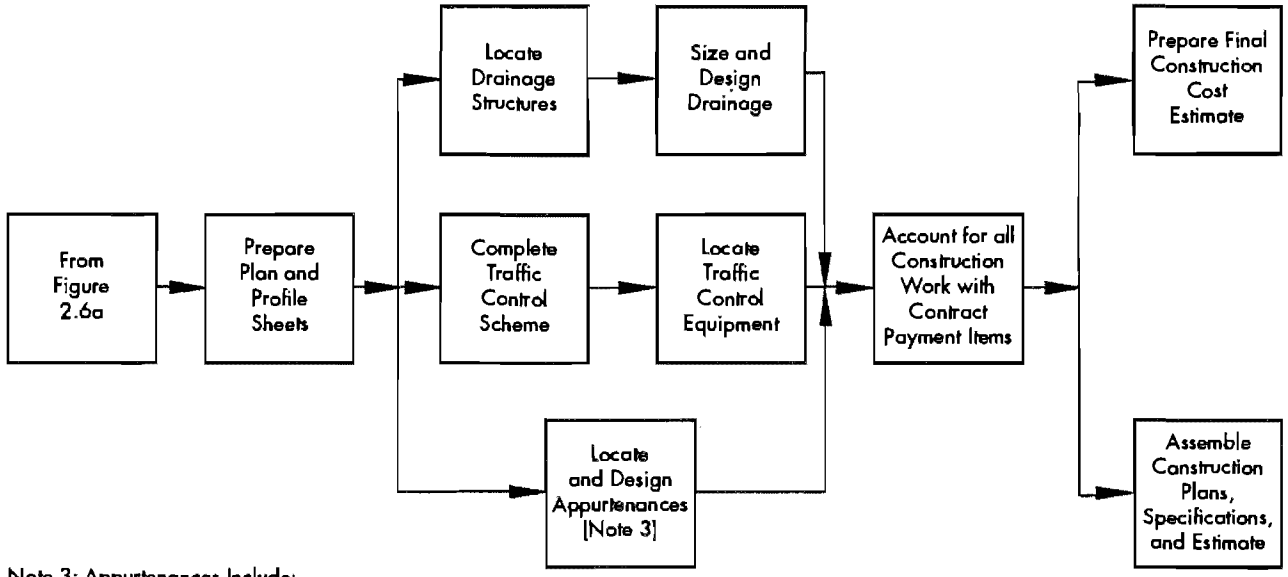
Intersection design basically is the management of traffic conflicts in the intersection area by providing appropriate geometric and traffic control features. It progresses through different levels of design where, at the planning level, the number of considerations is initially very high and attention is given mostly to generalized details. As design proceeds to the operational level, the number of considerations is somewhat reduced, but the level of detail required for each consideration increases. The conclusion of operational level design is final design where individual considerations are accounted for in great detail.

If managed in a deliberate manner, design input and output progress toward the end result efficiently while minimizing rework or redundancy in the process. To support the engineer in various phases of the intersection design and decision-making process, the development of a computer-aided interactive-graphics system has been initiated. Details of the components of this system are addressed in subsequent sections of this report.



Note 1: Definition Includes:
- Pavement Edges
- Curbs
- Medians
- Islands

Note 2: Evaluation Includes:
- Sight Distances
- Drainage Requirements
- Match Existing Where Necessary
- Complement Horizontal Alignment



Note 3: Appurtenances Include:
- Lighting
- Sidewalks
- Wheelchair Ramps
- Bus Stops
- Others as Required

Figure 2.6 Operational level—design procedure flow chart for final design

CHAPTER 3. IGIDS SYSTEM DESIGN

GRAPHICS ENGINE

Several good graphics engines (CAD software packages) which operate on popular workstations are available from different manufacturers. Each provides an interface with selected workstation hardware and has numerous commands to create, display, modify, manipulate, delete, save, and plot graphical data. The CAD software packages generally maintain an external graphics-engine database containing graphics files which may be saved from session to session. The CAD software packages also provide functions for reading and writing various standard graphics-engine database exchange formats. These CAD software packages are being constantly enhanced and corrected by a permanent staff of programmers in response to user requests. IGIDS software has been developed, or will be developed, to interface with chosen graphics engines.

Several criteria were established for choosing graphics engines with which to interface IGIDS. The graphics engine would have to (1) offer a higher-level language, real-time interface so that the IGIDS software could operate above and drive the graphics engine software, (2) allow IGIDS to take control of the interactive graphics workstation, (3) allow the user to switch easily between executing IGIDS commands and graphics-engine commands, (4) provide plotting capabilities to numerous plotter devices, (5) be used by many state DOTs and by many design professionals, (6) operate on several moderately-priced workstations from different manufacturers, and (7) have a graphics-engine reference database capability. IGIDS makes calls to defined generic graphics functions. A library of generic graphics functions has been, or will be, developed for each selected graphics engine. The graphics engines chosen for IGIDS are Intergraph's MicroStation and Autodesk's Autocad. First-stage development has been on Intergraph's MicroStation.

DATABASE ENGINE

A number of database engines (SQL, relational database software packages) which can operate on popular workstations are available on the market. Each provides an interface with the workstation operating system and the on-disk storage. The SQL database software packages maintain an external database engine containing the data which may be saved from session to session. In developing IGIDS, interfacing software has been written to access selected database engines.

The criteria that were established for choosing candidate database engines required that such an engine would (1) offer a higher-level language, real-time interface so that the IGIDS software could operate above and drive the database engine software, (2) be compatible with the graphics engines interfaced to IGIDS, (3) allow IGIDS to take control of the interactive graphics workstation, (4) allow the user to easily switch between executing IGIDS commands and database engine commands, (5) provide reporting capabilities to numerous output devices, (6) be used by many state DOTs and by many design professionals, and (7) operate on several moderately-priced workstations from different manufacturers. For IGIDS a list of generic database functions which perform the requested database operation has been, or will be, developed for each selected database engine. The database engines chosen are Informix and Oracle. Initial development has been on Informix.

OPERATING SYSTEM

Many operating system alternatives are currently available. Most large computer companies offer proprietary operating systems developed by the computer company to operate only on their own machines, and they often offer operating systems that adhere to international standards, government standards, or standards developed by a

consortium of competing computer companies. The operating systems may be designed for batch operation, on-line transaction processing, real-time process control, and/or interactive operation. The operating systems may support a single user or multiple users. They may allow only one process to be memory-resident and active at one time (example: MS-DOS), allow many processes to be memory-resident and allow the user to designate which process is active (example: Apple's Macintosh MultiFinder), or allow many processes to be memory-resident and allow the operating system to designate which process is active through a scheduling algorithm so that the processes share the CPU through multi-tasking (example: DEC's VMS and AT&T's Unix). The operating system may support virtual memory addressing or direct memory addressing.

The criteria for choosing an operating system for use by IGIDS included the fact that it would have to (1) be designed for batch operation and interactive operation, (2) support multiple users, (3) allow many processes to be memory resident and allow the operating system to designate which process is active through a scheduling algorithm so that the processes share the CPU through multi-tasking, (4) support virtual memory addressing with large real memory, (5) support all graphics engines interfaced to IGIDS, (6) support all database engines interfaced to IGIDS, (7) support higher-level language interfaces used by the graphics and database engines interfaced to IGIDS, (8) support computer software languages used by analysis programs interfaced to IGIDS, (9) support the computer software language chosen for IGIDS, (10) be used by many state DOTs and by many design professionals, and (11) operate on several moderately priced workstations from different manufacturers. The operating system chosen for initial implementation was AT&T's Unix, or a Unix clone. OSF's OS1 has been selected for support at a later time.

COMPUTER SOFTWARE LANGUAGE FOR IGIDS

A computer software language may be (1) an assembly language where one statement in the language is translated into one machine instruction, takes the most language statements to accomplish a given task, is designed to operate on one computer's machine instruction set, and is generally defined by the computer company manufacturing the computer; (2) a macro language where one statement in the computer software language is translated into one or more computer machine instructions, takes the second most computer software language statements to accomplish a given task, is designed to operate on one computer's machine instruction set, and is generally defined by the computer company manufacturing the computer; or (3) a higher-level language where one statement in the computer software language is translated into many computer machine instructions, takes the least computer software language statements to accomplish a given task, is designed to operate on many computer machine instruction sets, and is generally defined by an international standard.

A higher-level computer software language called C was chosen for IGIDS. This language can (1) be supported by all graphics engines interfaced to IGIDS, (2) be supported by all database engines interfaced to IGIDS, (3) be supported by the operating system, (4) be applicable to engineering and geometric calculations, (5) be a state-of-the-art computer software language, (6) be defined by an international standard, (7) allow the allocation and use of dynamic memory at execution time through the use of pointers, (8) support 16 significant digits, 64 bit, double precision floating point arithmetic, (9) support 10 significant digits, 32 bit, integer arithmetic, (10) support 5 significant digits, 16 bit, integer arithmetic, (11) allow the inclusion of global variable declarations and definitions from an external file, (12) allow upper and lower case characters in input and output records, and (13) be used by many state DOTs and by many computer software development professionals.

SOFTWARE CODING STANDARDS

Computer software coding standards are strongly recommended, as they result in computer code that is consistent, easily understood by another programmer, and easier to maintain. Once a coding standard becomes familiar, the programmer develops an expectation of the manner in which the computer code should look, and non-conforming items are easily detected.

The standard adopted for IGIDS specifies that all items in a list should appear in alphabetical order. This standard applies to "#include" statements, in-line macro definitions, "#define" statements, function prototypes, "typedef" statements, variable declarations and definitions, structure declarations and definitions, union declarations and definitions, and most other list items. The exception to this standard is to list items in hierarchical order where appropriate. This standard has the advantage that items can be found more easily and logically.

The standard also specifies that each function should be contained in its own source file and that the source file name should be a unique sub-

set of the function name. This standard has the advantage of minimizing the size of each source file, thus making backup and editing of the function easier. A programmer may modify a single function without disturbing others; the function can be individually compiled; there is a corresponding object file for each source file; a single function can easily be tested; and searching for a string within all functions will display the source file name (function name) for each string match, thus indicating the string usage. This standard has the disadvantages that the function names are limited in size by the largest unique file name size (currently 14 characters on AT&T's Unix), and that the programmer must potentially edit multiple source files to change a variable used in multiple functions.

All IGIDS core software global declarations and definitions should be contained in a single include source file. A declaration refers to places where the nature of the variable is stated but no storage is allocated, while a definition refers to the place where the variable is created or assigned storage. This standard has the advantages that there is only one include source file to maintain, it eliminates errors caused by differences in declarations, and searching for a string within a single function or all functions will display only where the string is referenced or used, as opposed to where the string is also declared or defined. Additionally, making a change in the single include source file will cause all functions which include that source file to be compiled using the make command. The authors also decided that all graphics engine software global declarations and definitions should be contained in a single include source file and that all database engine software global declarations and definitions should be contained in a single include source file.

All global declarations, definitions, and variables should be contained in the include source files. The include source file may contain: "#include" statements for other include source files needed by the functions, in-line macro definitions, "#define" statements, function prototypes, "typedef" statements, variable declarations and definitions, structure declarations and definitions, and union declarations and definitions. Techniques would be used so that once included in a compile, the include source file would not be included again even though it was requested additional times.

All compiled IGIDS core functions are contained in a single object library. This standard has the advantages that a single function can be easily modified and tested without changing the function in the library, and the linked image contains only the functions called. This standard has the disadvantages that the link process may have to include the object library several times to accommodate the unsatisfied externals, and the archive command must be used to replace object files in the object library. The authors also decided that all compiled graphics engine functions and compiled database engine functions should each be contained in a single object library.

Each function name should begin with "igids_" and contain a maximum of 18 upper and lower case characters (6 characters for "igids_" and 12 characters for the rest of the function name). In addition, the file name for the function's source file is 12 characters for the rest of the function name followed by ".c" and the file name for the function's object file is 12 characters for the rest of the function name followed by ".o". Finally, there should be two consecutive blank lines at the end of the source file and no other occurrence of two consecutive blank lines in the source file. This standard has the advantages that the function name can be descriptive of the operations performed, there should be minimal conflicts with function names from other software, one can readily distinguish the IGIDS functions from others, the source files may be concatenated into one large file and later broken into individual files, and the end of each function is standard and thus can be easily located while editing. Example function names are shown in Table 3.1.

Table 3.1 Example function names

igids_addLegClKey
igids_bitset
igids_dbload
igids_delSegCntrln
igids_drwarc
igids_errmsg
igids_lansrt
igids_loadAltLib
igids_moveLegLatrl
igids_openInterNew
igids_recreateInt
igids_search
igids_segswp
igids_selLanOutNxt
igids_set_keyFun
igids_tumTemplate
igids_tx_mdl
igids_vnumdf

Each variable name should begin with a five character prefix, followed by the name, optionally end with a four character suffix, and contain a maximum of 31 upper and lower case characters. The five character prefix would be a single

character indicating the scope of the variable, followed by two characters indicating the variable type, followed by a single character indicating the dimension of the variable, and terminated with an "_" character. Each variable that is a pointer would have the mandatory four character suffix "_ptr" for pointer to a variable or "_pfn" for pointer to a function. The single character scope would be: "g" for global static, "l" for local dynamic, "p" for parameter dynamic (parameters to the function), "s" for local static, "s" for structure member, "u" for union member, or "z" for zone. The two character type would be: "ch" for a single character, "cz" for character string zero (null) byte terminated, "df" for double floating point, "fs" for file stream, "si" for signed int, "sl" for signed long, "ss" for signed short, "td" for typedef variables, "uc" for unsigned char, "ui" for unsigned int, "ul" for unsigned long, "us" for unsigned short, or "vo" for void. The single character dimension would be "a" for array or "v" for single variable. This standard has the advantages that the variable name can be descriptive of the data storage; a programmer can readily distinguish the scope, type, and dimension of the variable and whether the variable is a pointer; the parameters to a function are treated in a special manner; and errors in formal parameter types not matching actual parameter types is reduced. Example variable names are presented in Table 3.2.

Table 3.2 Example variable names

gslv_dest_seg_id_num
gstv_inter_entry_ptr->sslv_num_alters
gstv_alter_entry_ptr->scza_desc
gstv_leg_ent_ptr->ssla_num_lanes[LANE_INB]
gstv_lane_ent_ptr->sdfv_width
gstv_seg_ent_ptr->sslv_iosc_flag
gstv_text_ent_ptr->sslv_font
ldfv_initial_angle
lfsv_file_ptr
lsiv_list_modified
lslv_return_code
lssv_i
lstv_previous_leg_ptr
ltda_cards[TX_MAX_CARDS]
lucv_byte
pczv_error_message_ptr
pdfv_sweep_angle
psiv_med_width
pslv_user_id_num
pucv_bit_mask_ptr
scza_file_name[FILE_NAME_NC+1]
ssiv_number_of_outbound_lanes
sslv_resetfunc_pfn
sssv_save_command_stage
suna_element[2]
ztdv_statedata

Each function that could directly detect an error or call a function that could return an error must return a signed long error code indicating success or failure. All other functions could return nothing (void) or could return a value. A function that properly completed would return an error code of "RETURN_SUCCESS", while a function that detected an error would return an error code of "RETURN_NON_FATAL_ERROR" for non-fatal (user recoverable) errors or return an error code of "RETURN_FATAL_ERROR" for fatal (programming) errors. A function that called a function that returned an error would deal with the error or return the error code. It would be the responsibility of the function detecting the error to issue an appropriate error message and the detecting function name. It would be the responsibility of a function that called a function that returned an error to issue the calling function name (i.e. provide a trace back for the error).

Virtually all constants should be coded using the symbolic name capability within C. The syntax of the symbolic name statement is "#define", followed by the name, and followed by the replacement string. Subsequent occurrences of the name in the source code (not in quotes and not part of another name) are substituted with the replacement string before compilation of the source code is initiated. Additionally, the name used in the "#define" statement should be upper case characters only, all constants used in more than one function for the same purpose should be defined in the global include source file, and local "#define" statement names should be prefixed with "LOCAL_". Usages of this feature include dimensions for arrays, index values for arrays, switch case statement values, function return values, inline macro definitions, and state or stage values.

This standard has the advantages that errors caused by different values being used for the same purpose are greatly reduced or eliminated, the meaning of constants is more readily conveyed, and the source code becomes more self-documenting. Additionally, the symbolic name capability may also be used for in-line macros. The syntax of the symbolic macro statement is "#define"; followed by the macro name, "(", the macro parameter(s), and ")"; and followed by the replacement string. The authors decided to allow both upper and lower case characters in the name for the symbolic macro statement. Example "#define" statements are illustrated in Table 3.3.

The following statement order and syntax was chosen for all functions:

(1) "#include" statement(s) starting in column 1 and followed by 1 blank line

Table 3.3 Example "#Define" Statements

#define	ALTER_DESC_NC	(size_t)	80
#define	ALTER_PROC_ALT	(long)	1
#define	ALTER_PROC_LEG	(long)	2
#define	ALTER_PROC_TXT	(long)	4
#define	ALTER_PROC_ALL	(long)	7
#define	CONCENTRIC_RADIUS_MINIMUM		50.0
#define	FILE_NAME_NC	(size_t)	127
#define	ID_NULL	(long)	-1
#define	LANE_INB	(long)	0
#define	LANE_OUT	(long)	1
#define	LANE_LENGTH_MINIMUM		10.0
#define	LANE_WIDTH_MINIMUM		8.0
#define	LANE_WIDTH_MAXIMUM		16.0
#define	LOCAL_FILE_NAME_NC	(size_t)	127
#define	RETURN_SUCCESS	(long)	0
#define	RETURN_NON_FATAL_ERROR	(long)	1
#define	RETURN_FATAL_ERROR	(long)	2
#define	max(a,b)	(((a)>(b))?(a):(b))	

(2) "type function_name (function_parameter(s))" statement starting in column 1
(3) "type function_parameter; /* comment */" statement(s) starting in column 1
(4) "{" in column 1
(5) comment statement(s) starting in column 3 describing the purpose of the function and followed by 1 blank line
(6) comment statement(s) starting in column 3 describing the global input requirements and followed by 1 blank line, if required
(7) comment statement(s) starting in column 3 listing the function(s) which call the function and followed by 1 blank line
(8) comment statement(s) starting in column 3 listing the function(s) called by the function and followed by 1 blank line, if required
(9) local "#define" statement(s) starting in column 3 and followed by 1 blank line, if required
(10) local structure(s), union(s), and variable(s) in alphabetical order starting in column 3 and followed by 1 blank line, if required
(11) function code block(s) starting in column 3; a function code block includes (a) comment statement(s) starting in column 3 describing the purpose of the following code, (b) computer code starting in column 3, and (c) 1 blank line
(12) "return;" or "return (RETURN_SUCCESS);" statement starting in column 3 if not conditionally returned in the preceding code
(13) 1 blank line, "return_fatal_error:" starting in column 3, 1 blank line, "igids_detmsg (" function_name ");" starting in column 3, and "return (RETURN_FATAL_ERROR);" starting in column 3, if required
(14) 1 blank line, "return_non_fatal_error:" starting in column 3, 1 blank line, "igids_detmsg (" function_name ");" starting in column 3, and "return (RETURN_NON_FATAL_ERROR);" starting in column 3, if required
(15) 1 blank line, "return_returned_error:" starting in column 3, 1 blank line, "igids_trcmsg (" function_name ");" starting in column 3, and "return (lslv_return_code);" starting in column 3, if required
(16) "}" in column 1
(17) 2 blank lines

Within the function code, 2 spaces would be used for indention. No tab characters would be in the code. The standard maximum line length would be 80 characters for functions and include source files. The comments for lines within include source files may be extended to 132 columns. The opening "{" and closing "}" for the "do", "for", "if", "switch", and "while" statements would start in the same column directly below the first character of the "do", "for", "if","switch", or "while" statement and all code between the opening "{" and the closing "}" would be indented 2 spaces. The opening "{" and the closing "}" would be used even when there is only one statement between them except an "if" statement without an "else" that does not exceed the standard maximum line length. The "else" for the "if" statement would start in the same column directly below the first character of the "if". The following is an example of a nested "if" and "else":

```
if ( aaa == bbb )
{
    if ( ccc == ddd )
    {
```

```
            www = xxx;
        }
        else
        {
            www = yyy;
        }
        zzz = (3.0*www) + 0.5;
    }
```

The "switch" and "case" statement is preferred over the "if" and "else" statement when there are more than 2 choices. The "case" and "default" for the "switch" statement would be indented 2 columns below the first character of the "switch" statement. The code and any "break;" statement(s) would be indented 2 columns below the first character of the "case" or "default" statement. The following is an example of a "switch" and "case":

```
switch (pslv_function)
{
    case FUNCT_ADD_DBE:
      lslv_return_code = igids_alter_add_dbe ();
      if (lslv_return_code != RETURN_SUCCESS)
      {
        goto return_returned_error;
      }
      break;
    case FUNCT_CPY_DBE:
      lslv_return_code = igids_alter_cpy_dbe ();
      if (lslv_return_code != RETURN_SUCCESS)
      {
        goto return_returned_error;
      }
      break;
    default:
      sprintf (gcza_err_msg,
               "Undefined function = %ld",
               pslv_function);
      igids_error ( gcza_err_msg);
      goto return_fatal_error;
    }
```

For statements within a block of code, the "="s, ";"s, and any comments at the end of a line should be aligned if reasonable. There should be no blank space before the ";" statement terminator unless needed for alignment. For a statement continued onto another line, the continued line(s) would be indented a minimum of 2 columns and preferably lined up with the first parameter following a "(", "=", or "+". The following are examples of these computer software coding standards:

```
lslv_seg_id_num    = gslv_seg_id_num ;
/* save global id num        */
lslv_id_num_blink  = lslv_seg_id_num  ;
/* set .local blink   */
sprintf ( gcza_err_msg,"Undefined function = %ld",
                pslv_function );
ldfv_dx = gstv_seg_ptr[lslv_seg_id_num].sdfv_beg_x
                -ldfa_ref_end_x[lsiv_end]    ;
```

There should be no blank space(s) at the end of any line. For comma separated lists of arguments, there should be no blank spaces before or after the comma unless needed for alignment purposes, whereas for comma separated statements, there should be 1 blank space before and after the comma. For the separating ";" within the "for" statement, there should be 1 blank space before and after the ";". Finally, there should be 1 blank space before and after an opening "(" and the closing ")" while there should be no blank space(s) before or after the opening "[" or the closing "]" unless needed for alignment. Function references which do not have parameters would not have any space between the "(" and the ")" as in "()". The following are examples :

```
lslv_return_code=igids_intchk (0l,gslv_dim_inters-1);
for (lsiv_i=0, lsiv_j=0; lsiv_i<lsiv_n; lsiv_i++,lsiv_j—)
lsla_ref_id_num[LOCAL_SEG_BEG]=gslv_seg_id_num;
lslv_return_code = igids_segsta ();
```

IGIDS keeps the entire IGIDS relational database in memory within the IGIDS software so that no disk I/O will be involved in reading a data item, thus making the software operate as fast as possible. Each IGIDS relational database has a corresponding structure within IGIDS with an instance of the structure for each entry or row in the IGIDS relational database table. Each IGIDS relational database attribute or column within an IGIDS relational database table would have a corresponding variable within the structure. Because of this last requirement, variables of type "int" should not be used because the number of bytes for an "int" can be different from workstation to workstation. Each structure and IGIDS relational database table would contain an attribute or column which indicated whether the entry or row had been modified since the last time that the internal copy of the IGIDS relational database was written to the external copy. IGIDS would allow the user to select between updating the IGIDS relational database after each command or at user requested times.

Many computer software languages, including C, support the concept of a structure. A structure is a collection of one or more variables, possibly of different types, grouped together under a single name for convenient handling. Structures are called "records" in some computer software languages. Structures help to organize complicated data because they permit a group of related variables to be treated as a unit instead of as separate

entities. The variables named in a structure are called members. The liberal use of structures is strongly supported by the authors.

Some higher-level computer software languages require the declaration of all variables at compile time and thus have a fixed maximum problem size, whereas others, including C, allow the allocation and use of dynamic memory at execution time through the use of pointers and thus the maximum problem size is limited only by the virtual address space available from the operating system. This execution-time allocation and use of memory was established as one of the criteria for choosing a higher-level computer software language for use by IGIDS. The liberal use of execution-time allocation and use of memory is strongly supported to generate what might be called dimensionless programming.

At compile time, IGIDS declares a pointer to a structure, a variable for the current allocation of instances of the structure, and a variable for the current use of instances of the structure. Since all reference to the structure is through the pointer, IGIDS can change the value of the pointer at execution time to reference different areas of memory. During the initialization phase at execution time, IGIDS requests the operating system to allocate memory for an initial number of structure instances, say 20, set the pointer to the address of this initial memory, set the current allocation to 20, and set the current use to 0. During normal execution, IGIDS uses these instances until there is a need for the 21st instance. IGIDS then requests the operating system to allocate memory for a new number of structure instances, say 30, copy the data from the 20 instances into the first 20 instances of the 30 instances, sets the pointer to the address of this new memory, returns the memory occupied by the original 20 instances to the operating system for further allocation, sets the current allocation to 30, and allows the program to continue. This process can be repeated until the virtual address space available to IGIDS is exhausted. The amount of virtual address space available to a process can normally be set within the operating system and the maximum is generally 1 to 4 billion bytes. The advantage of this technique is having virtually no limits within the program. The disadvantages of this technique are the increased execution time because of pointer addressing as opposed to direct addressing, the increased execution time because of the need to check the allocation each time another instance is needed, the increased execution time because of allocation and reallocation of dynamic memory, the need to initialize the allocated memory at execution time, and the increased risk of program failure by referencing memory outside the allocation range of the structures.

To overcome the increased risk of program failure by referencing memory outside the allocation range of the structures, a function (named "igids_xxxchk" where "xxx" is the 3 character structure name and "chk" stands for "check") was developed for each structure. It checks to ensure that an index number is in the range from zero through the current number of structure instances allocated minus one and in the range from a minimum value (normally zero) through a maximum value (normally the current number of structure instances in use minus one). Additionally, when an instance of a structure is no longer needed and can normally be deleted, that instance is added to a linked list of deleted instances for the structure (last added is first used and first added is last used) and the instance is marked as deleted. When a new instance of a structure is needed, first the deleted instances are used, then the allocated but unused instances are used, and finally additional instances are allocated from the operating system.

If possible, IGIDS was to be limited only by the graphical engines or the database engines. The graphical engines have a finite number of graphical levels, or planes, for separating the graphics and the database engines have no known limitations. The number of alternatives is limited by the minimum number of graphical levels, or planes, available on all graphics engines divided by the number of graphical levels, or planes, allocated per alternative. Currently, MicroStation has a limit of 63 graphical levels or planes and IGIDS has allocated 4 graphical levels or planes per alternative; therefore, IGIDS is currently limited to 15 alternatives.

COMPUTER HARDWARE

Many workstations are available on today's market and the choice of a particular vendor's workstation should be one of the least important considerations in influencing the IGIDS system design. The IGIDS system design attempts to avoid anything that might exclude any vendor's workstation.

Several criteria were established for assisting the user in choosing an IGIDS workstation, based on experience with software packages similar to IGIDS, experience with MicroStation and Autocad, and experience with Informix and Oracle. The workstation should (1) support the user's choice for a graphics engine, (2) support the user's choice for a database engine, (3) support the operating system, (4) have a minimum performance of an Intel 386SX based computer system with math

coprocessor, (5) have a minimum of 1 screen with a preference for 2 screens, (6) have a minimum screen resolution of 72 dots-per-inch with a preference for 100 dots-per-inch, (7) have a minimum screen size of 15 inches diagonal with a preference for 19 inches diagonal, (8) have a minimum of 32 colors for the largest screen with a preference for 256 colors for the largest screen, (9) have a pointing device (either the absolute positioning cursor or the relative movement mouse), (10) have a minimum of 2 megabytes of physical memory with a preference for 8 megabytes of physical memory (the user should acquire as much physical memory as can be afforded and justified to keep the graphics engine database and the IGIDS relational database memory resident), (11) have a minimum of 40 megabytes of hard disk with a preference for 300+ megabytes of hard disk (this will depend upon the size and quantity of the user's files), (12) have a 1.2 megabyte or 1.44 megabyte floppy disk drive, (13) optionally have networking hardware and software, (14) be moderately priced, and (15) be used by many state DOTs and by many design professionals.

ANALYSIS PROGRAMS

Engineers sometimes use computer software packages in the analysis and design of individual, at-grade, vehicular-traffic intersections and diamond interchanges. Intersection geometry, location and type of traffic control devices, and traffic flow conditions must be defined in order to use these computer aids. Criteria for selecting the intersection analysis and design software packages to be interfaced by IGIDS include the following considerations. Packages chosen to run external to IGIDS should (1) be supported by the operating system, (2) accept all necessary input from input file(s) without user interaction, (3) generate results into output file(s), (4) be moderately priced, (5) be readily available, and (6) be used by many state DOTs and design professionals. Packages chosen to run internally within IGIDS should (1) benefit from a closer relationship with IGIDS, (2) be available in source code without copyright infringement, (3) be supported by the operating system, (4) be supported by the computer software language, and (5) be used by many state DOTs and by many design professionals. External programs chosen are the TEXAS Model for Intersection Traffic and the Highway Capacity Software. The only internal program that has been implemented in IGIDS is the Texas Truck Off-tracking Model.

IGIDS MAIN STRUCTURES

IGIDS uses hierarchical, relational geometry and keeps the entire IGIDS relational database in memory. Additionally, a criterion established for the operating system was that it support virtual memory addressing with large real memory. A criterion established for the computer software language for IGIDS was that it allow the allocation and use of dynamic memory at execution time through the use of pointers. Finally, computer software coding standards were established which require that each IGIDS relational database table have a corresponding structure within IGIDS with an instance of the structure for each entry or row in the IGIDS relational database table. Each IGIDS relational database attribute or column within an IGIDS relational database table has a corresponding variable within the structure. Each IGIDS relational database table contains an attribute or column which indicates whether the entry or row had been modified since the last time that the internal copy of the IGIDS relational database was written to the external copy.

The primary structures (IGIDS relational database tables) are *intersection* (inter or int), *alternative* (alter or alt), *leg*, *lane* (lan), *segment* (seg), and *text* (txt). The ID is the number that is used to index the particular entry of the array of structures of the same type. The IDs start at 0 and are positive integers. The "#define" constant ID_NULL stands for an invalid ID and has a value of -1. The hierarchical relationships developed for these structures are shown in Table 3.4.

The intersection, alternative, leg, lane, segment, and text structures each contain a flag to indicate whether the instance has been modified since the last time that the internal copy of the IGIDS relational database was written to the external copy. Each list of instances has the ID of the beginning instance or ID_NULL if there are no instances on the list, the ID of the ending instance or ID_NULL if there are no instances on the list, and the number of instances on the list. This doubly-linked list allows IGIDS to traverse a list from beginning to end or from end to beginning, allows a list to be traversed from any instance forward to the end or backward to the beginning, and allows for easy insertion/deletion of an instance to/from a list. For each structure, a global variable is defined for a single temporary instance or entry of a structure and a single input instance or entry of a structure. Additionally for each structure, a global pointer is defined for the array of

Table 3.4 Hierarchical relationship of primary structures

```
Intersection:
    list of alternative IDs
    other intersection attributes

        Alternative:
            parent intersection ID
            list of leg IDs
            list of text IDs
            other alternative attributes

                Leg:
                    parent alternative ID
                    list of centerline segment IDs
                    list of inbound lane IDs
                    list of outbound lane IDs
                    other leg attributes

                        Lane:
                            parent leg ID
                            list of inner edge segment IDs
                            list of outer edge segment IDs
                            list of stop line segment IDs
                            other lane attributes

                                Segment:
                                    parent leg/lane ID
                                    list of text IDs
                                    other segment attributes

                                    Text:
                                        parent alternative/segment ID
                                        other text attributes
```

structures (this pointer is only modified by the allocation and reallocation functions), for a single instance or entry of a structure (usually the instance or entry to work with), for a single temporary instance (initialized to point to the single temporary instance), and for a single input instance (initialized to point to the single input instance or entry of a structure). The input instance or entry of a structure is used by IGIDS to store data until all necessary data are defined by the user, and then it is added to the main group. Finally, a "#define" constant is defined for the size, in bytes, of a single instance or entry of a structure.

The alternative, leg, lane, segment, and text structures each contain a flag to indicate whether the instance or entry is in-use or deleted, as well as the ID of the previous entry on the list (backward link or blink), the ID of the next entry on the list (forward link or flink) or ID_NULL if the entry is last or first on the list, and the ID of the parent. This doubly-linked list allows IGIDS to traverse a list from any instance or entry forward to the end or backward to the beginning and allows for easy insertion/deletion of an instance or entry.

There is only one instance of the intersection structure, and its ID is 0. In addition to the attributes listed above, the intersection structure contains many one-of-a-kind attributes such as:

(1) the maximum gap between segments; the selected alternative ID;
(2) the global next ID (available for use) for alternative, leg, lane, segment, and text;
(3) the global maximum ID (memory allocated but unused) for alternative, leg, lane, segment, and text; the global number of instances (currently in use) for alternative, leg, lane, segment, and text;
(4) the graphics level or plane for the user and scratch;
(5) the beginning level or plane for alternatives;
(6) the number of levels or planes for alternatives; the relative level or plane for centerlines;
(7) the relative level or plane for lanes;
(8) the relative level or plane for traffic control;
(8) the relative level or plane for texts; and
(9) the intersection description provided by the user.

The intersection structure itself has no displayable graphics.

The number of alternatives is limited by the minimum number of graphical levels, or planes, available on all graphics engines divided by the number of graphical levels, or planes, allocated per alternative. Currently, MicroStation has a limit of 63 graphical levels, or planes, and IGIDS has allocated 4 graphical levels or planes per alternative; therefore, IGIDS is limited to 15 alternatives. Thus, there are zero to 15 instances of the alternative structure. In addition to the attributes listed above, the alternative structure contains:

(1) the center x and y coordinate,
(2) the alternative number (1->2 billion) provided by the user,
(3) the beginning level or plane for the alternative; the selected leg ID; the selected lane ID,
(4) the selected segment ID,
(5) the selected text ID,
(6) the alternative description provided by the user, and
(7) the alternative application data.

The alternative structure itself has no displayable graphics. IGIDS automatically sorts the list of legs of an alternative using the centerline absolute angle of the leg converted to an azimuth (north is zero degrees and clockwise is positive), starting at north, and traversing clockwise.

There are zero to any number of instances of the leg structure. In addition to the attributes listed above, the leg structure contains:

(1) the centerline absolute angle, distance and offset from the intersection center, station number at intersection center, and direction for station numbers (increasing or decreasing);
(2) the tie point x and y coordinate;
(3) the leg number (1->2 billion) provided by the user;
(4) a flag indicating whether the centerline segments are completed; the selected centerline segment ID;
(5) the selected lane ID;
(6) the leg description provided by the user; and
(7) the leg application data.

The leg structure itself has no displayable graphics. The centerline is a connected list of segments. IGIDS automatically sorts the list of centerline leg segments using the distance from the intersection center, starting at the segment end nearest the intersection center, and traversing away from the intersection center. IGIDS automatically sorts the lists of inbound and outbound lanes of a leg using the beginning offset from the centerline of the first segment of the inner edge of the lane, starting at the lane nearest the median and farthest from the curb (lane 1), and traversing away from the median toward the curb (lane N) (left to right in the direction of travel). IGIDS also assigns the lane number (1->N) to each lane after sorting the lists of inbound and outbound lanes.

There are zero to any number of instances of the lane structure. In addition to the attributes listed above, the lane structure contains:

(1) the lane width at the intersection;
(2) the lane length;
(3) the centerline station for the intersection end of the lane, the beginning and ending of any blockage, and the end of the lane;
(4) the relative distance and direction from the tie point;
(5) the tie point x and y coordinate and absolute angle;
(6) the lane number (1->N) calculated by IGIDS from median lane (lane 1) to the curb lane (lane N);
(7) a flag indicating whether the lane is inbound or outbound;
(8) 3 flags indicating whether the inner edge, outer edge, and stop line edge are completed;
(9) the selected segment ID; and
(10) the lane application data.

The lane structure itself has no displayable graphics.

The inner edge, outer edge, and stop line are connected segment lists. IGIDS automatically sorts the lists of inner and outer edge segments of a lane using the distance from the intersection center, starting at the segment end nearest the intersection center, and traversing away from the intersection center. IGIDS automatically sorts the list of stop line segments using the distance from the stop line segment end with the minimum offset from the leg centerline (the stop line segment end nearest the median and farthest from the curb), starting at the stop line segment end with the minimum centerline offset and traversing away from the median toward the curb (left to right in the direction of travel).

There are zero to any number of instances of the segment structure. A segment is an arc or a line. In addition to the attributes listed above, the segment structure contains the following attributes:

(1) relative distance and direction from the tie point;

(2) for the beginning end of the segment, the relative angle, the centerline station and offset, the absolute angle, and the x and y coordinate;
(3) for the ending end of the segment, the centerline station and offset, the absolute angle, and the x and y coordinate;
(4) the 4 parameters defining the segment geometry;
(5) the segment number (1->N) calculated by IGIDS from the beginning end (1) to the ending end (N) depending on the type of segment list (inner edge, outer edge, stop line, and centerline);
(6) a flag indicating whether the segment is an arc or a line;
(7) a flag indicating whether the segment is an inner edge, outer edge, stop line, or centerline (which defines whether the segment parent is a leg or a lane); and
(8) the selected text ID.

For an arc segment, the 4 parameters are the radius, the sweep angle, the center x coordinate, and the center y coordinate. For a line, the first parameter is the length of the line, and the remaining 3 parameters are not used. The segment structure itself has displayable graphics.

There are zero to any number of instances of the text structure. In addition to the attributes listed above, the text structure contains:

(1) the text angle,
(2) the text height,
(3) the text width,
(4) the lower left x and y coordinate,
(5) the four parameters for the text,
(6) a flag indicating whether the text parent is an alternative or a segment, and
(7) the number of characters, the text font, and the text string.

For alternative text, the 4 parameters are the absolute angle for the lower left x/y coordinate, the distance for the lower left x/y coordinate, the absolute text angle, and parameter 4 is not used. For segment text, the 4 parameters are the relative angle for the lower left x/y coordinate, the distance for the lower left x/y coordinate, the absolute/relative text angle, and a flag indicating whether the text is at an absolute angle or at an angle relative to the segment. The text structure itself has displayable graphics.

OBJECT-ORIENTED PROGRAMMING TECHNIQUES

Object-oriented programming techniques have been used throughout IGIDS development. Although C++ is available, it was not used because it is not defined by an international standard.

IGIDS uses hierarchical, relational geometry. The primary structures or IGIDS relational database tables are *intersection* (int), *alternative* (alt), *leg* (leg), *lane* (lan), *segment* (seg), and *text* (txt). There is a global variable for each structure indicating the current instance or row. Each structure has a list of IDs of its children and its parent. This feature allows traversal of the structure hierarchy.

IGIDS has complementary functions which set the parent and child for a structure instance, and, through function calls, automatically propagate up or down the hierarchy. Every structure except *intersection* has a parent and a flag indicates the parent structure.

Every structure except *text* has a child structure although list of children may be empty. There may be more than one child for a structure thus the function checks each list of children until it finds a particular child instance. These functions are respectively named "igids_xxxspr" and "igids_xxxsch" where xxx is a character structure name and "spr" and "sch" indicate set parent or set child, respectively.

There is a corresponding function which sets the database entry called "igids_xxxsdb."

The appropriate "igids_xxxspr" or "igids_xxxsdb" functions yield the attributes (or columns) of the parent structures but the programmer must know the structure that contains the attribute or column that is needed. As an example, suppose that "gslv_seg_id_num" is set to the ID of a segment and that the programmer called the "igids_segspr" function, then the x coordinate of the center of the intersection for the segment's parent alternative would be referenced by "gstv_alter_ent_ptr->sdfv_center_x".

The primary structures or IGIDS relational database tables are *intersection*, *alternative*, *leg*, *lane*, *segment*, and *text*. The primary operations to be performed on those structures are:

(1) add a database entry (FUNCT_ADD_DBE),
(2) copy a database entry (FUNCT_CPY_DBE), delete a database entry (FUNCT_DEL_DBE),
(3) initialize the database (FUNCT_INI_DBE) (only valid for the intersection structure),

(4) select a database entry (FUNCT_SEL_DBE),
(5) calculate the graphics for a database entry (FUNCT_CAL_GRA),
(6) draw the graphics for a database entry (FUNCT_DRW_GRA),
(7) erase the graphics for a database entry (FUNCT_ERS_GRA), highlight the graphics for a database entry (FUNCT_HIL_GRA), and
(8) calculate the station and offset for a database entry (FUNCT_CAL_STA) (only valid for a leg, a lane, and a segment structure).

The only operations that can be performed on the intersection structure are initialize the database (FUNCT_INI_DBE), draw the graphics for a database entry (FUNCT_DRW_GRA), and erase the graphics for a database entry (FUNCT_ERS_GRA). All operations can be used with all structures except as noted. Table 3.5 summarizes the relation between primary operations and the structures.

Table 3.5 Relationship between primary operations and structures

Primary Operation	int	alt	leg	lan	seg	txt
FUNCT_ADD_DBE	no	yes	yes	yes	yes	yes
FUNCT_CPY_DBE	no	yes	yes	yes	yes	yes
FUNCT_DEL_DBE	no	yes	yes	yes	yes	yes
FUNCT_INI_DBE	yes	no	no	no	no	no
FUNCT_SEL_DBE	no	yes	yes	yes	yes	yes
FUNCT_CAL_GRA	no	yes	yes	yes	yes	yes
FUNCT_DRW_GRA	yes	yes	yes	yes	yes	yes
FUNCT_ERS_GRA	yes	yes	yes	yes	yes	yes
FUNCT_HIL_GRA	no	yes	yes	yes	yes	yes
FUNCT_CAL_STA	no	no	yes	yes	yes	no

An object is a single instance of a structure and all its children (such as a leg or an inbound lane) or a list of children and all of their children (such as the centerline segments of a leg). To allow the user to select an object by pointing at it:

(1) the user chooses the appropriate command which defines the operation to perform and the object type (such as rotate leg),
(2) the user points at some displayable graphics (arc, line, or text) for the object (such as the center line segments of the leg; the inner edge, outer edge, or stop line segments of an inbound or outbound lane for the leg; or text attached to a segment of the leg),
(3) the graphics engine searches the graphics-engine database and gives IGIDS the ID from the graphics-engine element that the user selected,
(4) based upon the ID and the type (arc, line, or text) of the graphics-engine element, the appropriate "igids_xxxspr" function is called,
(5) IGIDS checks to confirm that the selected structure(s) refer to the requested object (automatically rejecting the graphics-engine element if it does not refer to the requested object and repeating the process starting at (3)), and
(6) IGIDS highlights the selected object for acceptance or rejection by the user.

The following is a list of the IGIDS objects and the corresponding "#define" object type:

IGIDS Objects				"#DEFINE" Object Types
one	alternative			OBJECT_ALT_ALL
all	legs	of an alternative		OBJECT_ALT_LEGS
all	texts	of an alternative		OBJECT_ALT_TXTS
one	leg			OBJECT_LEG_ALL
all	centerline segments		of a leg	OBJECT_LEG_CNTRLN_SEGS
all	inbound	lanes	of a leg	OBJECT_LEG_INBLAN_LANS
all	outbound	lanes	of a leg	OBJECT_LEG_OUTLAN_LANS
one	lane			OBJECT_LAN_ALL
one	inbound	lane		OBJECT_LAN_INBLAN_ALL
one	outbound	lane		OBJECT_LAN_OUTLAN_ALL
all	inner edge	segments of a lane		OBJECT_LAN_INNEDG_SEGS
all	outer edge	segments of a lane		OBJECT_LAN_OUTEDG_SEGS
all	stop line	segments of a lane		OBJECT_LAN_STOPLN_SEGS
one	segment			OBJECT_SEG_ALL
one	inner edge	segment		OBJECT_SEG_INNEDG_ALL
one	outer edge	segment		OBJECT_SEG_OUTEDG_ALL
one	stop line	segment		OBJECT_SEG_STOPLN_ALL
one	centerline	segment		OBJECT_SEG_CNTRLN_ALL
all	texts of a segment			OBJECT_SEG_TXTS
one	text			OBJECT_TXT_ALL
one	alternative	text		OBJECT_TXT_ALT_ALL
one	segment	text		OBJECT_TXT_SEG_ALL

In order to allow IGIDS to perform an operation on a structure or one or more selected children, a global processing mask (a bit mask; true = process, false = do not process) was created (gslv_proc_mask). Functions, such as delete database entry for the leg, conditionally process the operation on each list of children or conditionally process the operation on the entry itself based upon the value of a bit in the global processing mask. The first processing mask is for the structure instance itself and uses bit 0 (value = 1). The "_ALL" processing mask is a bitwise "or" of all the possible values for a particular structure. The following is a list of the global processing masks and the corresponding "#define" name:

Processing Mask	Name
process intersection only	INT_PROC_INT
process intersection alternatives	INT_PROC_ALT
process intersection and all children	INT_PROC_ALL
process alternative only	ALTER_PROC_ALT
process alternative legs	ALTER_PROC_LEG
process alternative texts	ALTER_PROC_TXT
process alternative and all children	ALTER_PROC_ALL
process leg only	LEG_PROC_LEG
process leg centerline segments	LEG_PROC_CNTRLN
process leg inbound lanes	LEG_PROC_INBLAN
process leg outbound lanes	LEG_PROC_OUTLAN
process leg and all children	LEG_PROC_ALL
process lane only	LANE_PROC_LANE
process lane inner edge segments	LANE_PROC_INNEDG
process lane outer edge segments	LANE_PROC_OUTEDG
process lane stop line segments	LANE_PROC_STOPLN
process lane and all children	LANE_PROC_ALL
process segment only	SEG_PROC_SEG
process segment texts	SEG_PROC_TXT
process segment and all children	SEG_PROC_ALL
process text only	TEXT_PROC_TXT
process text and all children	TEXT_PROC_ALL

The traditional computer programming approach is to write a function wherein the operation is primary and the object is secondary. Each function knows how to perform the operation on every object. When the programmer wants to add another object, each function that performs an operation has to be modified to perform the operation on the new object. When the programmer wants to add another operation, a new function has to be developed which knows how to perform the operation on all objects.

The object-oriented programming approach is to write a function wherein the object is primary and the operation is secondary. Each function knows how to perform every operation on the object. When the programmer wants to add another object, a new function has to be developed which would know how to perform all operations on the object. When the programmer wanted to add another operation, each function would have to be modified to perform the new operation on the object.

The approach adopted in IGIDS is to develop a function for each object which dispatched the operation to the appropriate function(s) and to develop a function for each object-operation combination where there was actually something to be done other than perform the operation on all children. IGIDS has a function (named "igids_xxx" where "xxx" is the three-character structure name) which takes as a parameter the "FUNCT_" operation to be performed and dispatches the specified operation to the appropriate function(s). Additionally, IGIDS has a function (named "igids_xxxyyy" where "xxx" is the three-character structure name and "yyy" is "adb" for FUNCT_ADD_DBE, "cdb" for FUNCT_CPY_DBE, "cgr" for FUNCT_CAL_GRA, "ddb" for FUNCT_DEL_DBE, "dgr" for FUNCT_DRW_GRA, "egr" for FUNCT_ERS_GRA, "hil" for FUNCT_HIL_GRA, "sdb" for FUNCT_SEL_DBE, and "sta" for FUNCT_CAL_STA) which performs operation "yyy" on structure "xxx". Finally, IGIDS has a function (named "igids_xxxpgf" where "xxx" is the three-character structure name and "pgf" stands for "process generic function") which performs the specified "FUNCT_" operation on the selected "_PROC_" structure instance or entry. These "igids_xxxpgf" functions are used to perform an operation on all children. Currently, there is an "igids_xxxpgf" function for the *intersection*, *alternative*, *leg*, *lane*, and *segment* structures or IGIDS relational database tables.

CHAPTER 4. IGIDS FUNCTIONAL DESIGN

MULTIPLE INTERSECTION ALTERNATIVES

IGIDS allows for analysis and design of a minimum of five intersection alternatives. This capability was accomplished by ordering the hierarchical parent-child relationship as intersection-alternative-leg. Currently, MicroStation has a limit of 63 graphical levels, or planes, and IGIDS has allocated 4 graphical levels or planes per alternative; therefore, IGIDS is currently limited to 15 alternatives. Commands were developed to copy or modify all or part of an alternative. Additionally, commands were developed to display or not display an entire alternative.

LEVEL ASSIGNMENTS

Each alternative and the major graphical component groupings of the alternative were to be placed on separate graphical levels or planes so that they could be independently displayed or not displayed in a particular view. IGIDS allocates a user graphical level or plane and a scratch graphical level or plane and allows the user to display or not display graphics by alternative and items. These capabilities were accomplished by defining the following attributes in the IGIDS relational data base table and by assigning them the values shown in Table 4.1.

Additionally, commands were developed which allowed the user to display or not display an entire alternative and to individually display or not display the centerline, lanes, traffic control, and text for an alternative.

USER INTERACTION WITH COMMANDS

IGIDS uses an interactive event-driven user interface. The generic inputs required to be available from the graphics engine are (1) command selection, (2) coordinate entry, (3) reset entry, and (4) keyboard entry.

Command selection could be by user keyin, function key activation, screen menu selection, or digitizer menu selection. The net result of a command selection is that IGIDS can detect that a command has been selected, that IGIDS can determine whether the command is a graphics engine command or an IGIDS command, and that IGIDS can get the command name or command number for an IGIDS command. The actual event causing the command selection is not important to IGIDS.

Coordinate entry could be by user keyin, function key activation, mouse button activation, or cursor button activation. The net result of a coordinate entry is that IGIDS detects that a coordinate entry has been performed and that IGIDS can get the coordinate value in real world coordinates. The actual event causing the coordinate entry is not important to IGIDS.

Reset entry could be by user keyin, function key activation, screen menu selection, digitizer menu selection, mouse button activation, or cursor button activation. The net result of a reset entry is that IGIDS can detect that a reset entry has been performed, but the actual event causing the reset entry is not important to IGIDS. Keyboard entry could be by user keyin or function key activation.

Table 4.1 Assigned graphical attributes and values

Attribute Name	Description	Value
sslv_lev_user	Level for user graphics	1
sslv_lev_scratch	Level for scratch graphics	2
sslv_beg_lev_alter	Beginning level for alternatives	3
sslv_num_lev_alter	Number of levels for alternatives	4
sslv_rel_lev_center	Relative level for centerlines	0
sslv_rel_lev_lanes	Relative level for lanes	1
sslv_rel_lev_tc	Relative level for traffic control	2
sslv_rel_lev_text	Relative level for texts	3

The net result of a keyboard entry is that IGIDS can detect that a keyboard entry has been performed and that IGIDS can get the character string that was entered. The actual event causing the keyboard entry is not important to IGIDS.

Each command is processed by a single function. To accomplish this, the concept of a processing stage was developed. The processing of a command would proceed from stage to stage until the command was completed or the user selected another command. The global variables were defined to implement processing of commands by stages as shown in Table 4.2.

IGIDS knows the function to call (a large "switch" and "case" statement in function "igids_prccmd") and the stage number to set (STAGE_1) when the user selects a command which is an IGIDS command. IGIDS maintains a pointer to the function to be called and the stage number to be set when the user performs a coordinate entry, reset entry, or keyboard entry. The pointers to functions are paired with the previously discussed global stage variables. The following zone variables were defined to maintain a pointer to the function to be called:

Response	Event	Event Description
(*sslv_datafunc_pfn)	data button	coordinate entry
(*gslv_home_pfn)	high-level return	
(*sslv_keyinfunc_pfn)	keyin	keyboard entry
(*gslv_reenter_pfn)	re-enter last data	
(*sslv_resetfunc_pfn)	reset button	reset entry
(*gslv_return_pfn)	low-level return	

IGIDS basically is in an event loop waiting for the user to perform a (1) command selection, (2) coordinate entry, (3) reset entry, or (4) keyboard entry. Upon the graphics engine notifying IGIDS that the user has performed one of these events, IGIDS sets the global stage number to the appropriate value and calls the designated function. Upon return from the designated function, IGIDS waits for the user to perform another entry and the process begins again. The generalized event loop processing is illustrated in Figure 4.1.

It is thus the responsibility of a command function to designate at each stage of processing the appropriate function and stage for a coordinate entry, reset entry, and keyboard entry. Most of the time in IGIDS, the reset entry is used to reject a selection. Normally, the coordinate entry and keyboard entry events are mutually exclusive at a given stage of processing for a command so that a function and stage are designated to output a message to the user that invalid input has been received and dismissed. The general processing of a command is illustrated in Figure 4.2.

IGIDS implements both noun/verb and verb/noun command processing. In noun/verb command processing, the user selects the object to be operated upon (uses the selected object or selects a new object), selects the operation to be performed, and finally accepts or rejects the highlighted object. An example of noun/verb command processing is "leg rotate". In verb/noun command processing, the user selects the operation to be performed by initiating the command, selects the object to be operated upon by pointing, and finally accepts or rejects the highlighted object. An example of verb/noun command processing is "rotate leg". IGIDS provides a toggle between noun/verb and verb/noun command processing. Since the noun/verb command processing uses the selected object, more methods of choosing the object are available with noun/verb command processing. The generalized code to implement both noun/verb and verb/noun command processing is illustrated in Figure 4.3.

IGIDS was to allow the user to back up to previous input entry. This was accomplished by defining the following global variables (as previously discussed) and by developing a command to set the global stage to the re-enter stage and executing the re-enter function:

```
gsiv_stage_reenter          re-enter last data stage
(*gslv_reenter_pfn)  ()     re-enter last data funct
```

It is thus the responsibility of a command function to designate at each stage of processing the appropriate function and stage for an input re-entry operation.

IGIDS allows the user to cancel an IGIDS command and to choose another IGIDS command at any point. These features were accomplished by making all commands collect input and store the data in global or static variables until all necessary input was received. Additionally, upon receiving a request for a graphics engine immediate command or an IGIDS immediate command, IGIDS can save the state of IGIDS command processing, execute the immediate command, restore the state of IGIDS, and continue processing the interrupted IGIDS command. An immediate command is one that may interrupt another command. The state of IGIDS command processing is the current values for the stage and function variables as discussed earlier.

IGIDS allows the user to switch between IGIDS commands and graphics engine commands. This was accomplished by IGIDS filtering all event loop data and sending the graphics engine data to the graphics engine for final processing.

Table 4.2 Global variables and stage definitions

Variable Name	Description			
gsiv_stage	current command	stage		
gsiv_stage_datapt	data button	stage	coordinate	entry
gsiv_stage_home	high-level return	stage		
gsiv_stage_keyin	keyin	stage	keyboard	entry
gsiv_stage_re-enter	re-enter last data	stage		
gsiv_stage_reset	reset button	stage	reset	entry
gsiv_stage_return	low-level return	stage		

The following stages were defined:

Variable Name	Description
STAGE_0	Command Processing Stage Number 0 (special)
STAGE_1	Command Processing Stage Number 1
STAGE_2	Command Processing Stage Number 2
STAGE_3	Command Processing Stage Number 3
STAGE_4	Command Processing Stage Number 4
STAGE_5	Command Processing Stage Number 5
STAGE_6	Command Processing Stage Number 6
STAGE_7	Command Processing Stage Number 7
STAGE_8	Command Processing Stage Number 8
STAGE_9	Command Processing Stage Number 9
STAGE_10	Command Processing Stage Number 10
STAGE_11	Command Processing Stage Number 11
STAGE_12	Command Processing Stage Number 12
STAGE_13	Command Processing Stage Number 13
STAGE_14	Command Processing Stage Number 14
STAGE_15	Command Processing Stage Number 15
STAGE_16	Command Processing Stage Number 16
STAGE_17	Command Processing Stage Number 17
STAGE_18	Command Processing Stage Number 18
STAGE_19	Command Processing Stage Number 19
STAGE_20	Command Processing Stage Number 20
STAGE_END_COMMAND	Command Processing Stage Number 99

```
lsiv_dispatchtype  = ! MS_TERMINATE;
gsiv_exit_session  = FALSE;

while (   ( lsiv_dispatchtype != MS_TERMINATE     ) &&
              ( ! gsiv_exit_session                                    ) )
{
     Wait_for_events ( USER_EVENT,&lsiv_event );
     if ( lsiv_event & USER_EVENT )
     {
          Get_user_event_data ( &lsiv_i );
          dispatch_asynch_request ( &lsiv_dispatchtype );
          switch ( lsiv_dispatchtype )
          {
               case MS_INPUT:
                    if ( input_for_igids )
                    {
                         switch ( ztdv_statedata.stdv_iqel.hdr.cmdtype )
                         {
                              case DATAPNT:                    /* coordinate entry */
                                   gsiv_stage = gsiv_stage_datapt;
                                   (*ztdv_statedata.sslv_datafunc_pfn) ();
                                   break;

                              case KEYIN:                      /* keyboard entry */
                                   gsiv_stage = gsiv_stage_keyin;
                                   (*ztdv_statedata.sslv_keyinfunc_pfn) ();
                                   break;

                              case RESET:                      /* reset entry */
                                   gsiv_stage = gsiv_stage_reset;
                                   (*ztdv_statedata.sslv_resetfunc_pfn) ();
                                   break;

                              default:
                                   igids_wrnmsg ( "Undefined cmdtype" );
                                   break;
                         } /* end switch ( cmdtype ) */
                    }
                    else /* ! input_for_igids */
                    {
                         /* NOT OURS - forward it */
                    }
                    break;    /* end case MS_INPUT */

               case EXECUTE:                                   /* command selection */
                    igids_prccmd ();
                    break;    /* end case EXECUTE */

               default:
                    igids_wrnmsg ( "Undefined dispatchtype" );
                    break;
          }    /* end switch ( lsiv_dispatchtype ) */
     }    /* end if ( lsiv_event & USER_EVENT ) */
}    /* end while (     ( lsiv_dispatchtype != MS_TERMINATE     ) &&      */
     /*                      ( ! gsiv_exit_session

                            ) )   */
```

Figure 4.1 Generalized event loop

```
long igids_command ( )
{
    top:
        igids_cmdmsg      ( "command message" );

    switch ( gsiv_stage )
    {
        case STAGE_1:
            igids_set_datFun        ( igids_badEnt,STAGE_0 );
            igids_set_keyFun        ( igids_keyFun,STAGE_1 );
            igids_set_resFun        ( igids_command,STAGE_1 );
            igids_set_retFun        ( igids_command,STAGE_2 );
            igids_prmmsg            ( "keyboard entry - input 1:" );
            break; /* wait for user input */

        case STAGE_2:
            if ( data_not_ok )
            {
                igids_wrnmsg ( "data not ok" );
                gsiv_stage = STAGE_1;
                goto top;
            }
            /* save data in static or global variable for later use */
            igids_set_renFun        ( igids_command,STAGE_1 );

        case STAGE_3:
            igids_set_datFun        ( igids_datFun,STAGE_1 );
            igids_set_keyFun        ( igids_badEnt,STAGE_0 );
            igids_set_resFun        ( igids_command,STAGE_1 );
            igids_set_retFun        ( igids_command,STAGE_4 );
            igids_prmmsg            ( "coordinate entry - input 2" );
            break; /* wait for user input */

        case STAGE_4:
            if ( data_not_ok )
            {
                igids_wrnmsg ( "data not ok" );
                gsiv_stage = STAGE_3;
                goto top;
            }
            /* save data in static or global variable for later use */
            igids_set_renFun        ( igids_command,STAGE_3 );

        case STAGE_5:
            /* process command with saved input */
            igids_infmsg ( "command completed" );
            gsiv_stage = STAGE_3; /* to re-cycle on last input */
            goto top;

        default:
            igids_errmsg ( "programming error in command" );
            goto return_fatal_error;
    }

    return ( RETURN_SUCCESS );

    return_fatal_error:

    igids_detmsg ( "igids_command" );
    return ( RETURN_FATAL_ERROR );
}
```

Figure 4.2 Command processing

```
long igids_command ( )
{
    if ( gsiv_noun_verb && ( gsiv_stage == STAGE_1 ) )
    {
        gsiv_stage = STAGE_2;
    }

    top:
        igids_cmdmsg     ( "command message" );

    switch ( gsiv_stage )
    {
        case STAGE_1:
            /* select object */

        case STAGE_2:
            /* hilite and accept selected object */
            if ( selected_object_not_accepted )
            {
                gsiv_stage = STAGE_1;
                goto top;
            }

        case STAGE_3:
            /* process remainder of command */
            break;

        default:
            igids_errmsg ( "programming error in command" );
            goto return_fatal_error;
    }

    return ( RETURN_SUCCESS );

    return_fatal_error:

    igids_detmsg ( "igids_command" );
    return ( RETURN_FATAL_ERROR );
}
```

Figure 4.3 Generalized code for noun/verb, verb/noun processing

DATA BASE "SAVE" OPTIONS

IGIDS saves all modifications to the IGIDS relational data base after each IGIDS command is executed or at user-specified times. This will be accomplished by allowing the user to specify automatic (after each command) and manual (upon user request) updating. Additionally, each structure and IGIDS relational data base table would contain an attribute or column which indicated whether the entry or row had been modified since the last time that the internal copy of the IGIDS relational data base was written to the external copy.

IGIDS COMMANDS

The following is an initial list of major commands to be incorporated in IGIDS:

- add
- copy
- delete
- end
- hilite
- key-in
- load
- modify
- move
- noun-verb
- recreate
- re-enter data
- rotate
- save to
- select
- show information
- show station/offset
- tools
- verb-noun
- view

The following is an initial list of commands to be incorporated in IGIDS:

- add alternative
- add leg centerline by keyin
- add leg centerline from scratch level
- add inbound lane by keyin
- add outbound lane by keyin
- add inbound lane header from scratch level
- add inbound lane inner edge segments from scratch level
- add inbound lane outer edge segments from scratch level
- add inbound lane stop line segments from scratch level
- add outbound lane header from scratch level
- add outbound lane inner edge segments from scratch level
- add outbound lane outer edge segments from scratch level
- add outbound lane stop line segments from scratch level
- add text on alternative from scratch level
- add text on alternative by keyin
- add text on segment from scratch level
- add text on segment by keyin
- copy alternative
- copy leg
- copy text on alternative
- copy text on segment
- delete alternative
- delete leg
- delete inbound lane
- delete outbound lane
- delete inner edge segment
- delete outer edge segment
- delete stop line segment
- delete text on alternative
- delete text on segment
- end IGIDS
- hilite current alternative
- hilite current leg
- hilite current leg centerline
- hilite current leg inbound lanes
- hilite current leg outbound lanes
- hilite current lane
- hilite current lane inner edge segments
- hilite current lane outer edge segments
- hilite current lane stop line segments
- hilite current segment
- hilite current text
- key-in "yes"
- key-in "no"
- key-in default
- load from TEXAS Model file
- load from data base
- load standard 3X2 intersection
- load standard 3X3 intersection
- load standard 4X2 intersection
- load standard 4X3 intersection
- load standard 4X4 intersection
- load standard 5X4 intersection
- load standard 5X5 intersection
- load standard 6X4 intersection
- load standard 6X5 intersection
- load standard 6X6 intersection
- load standard 7X4 intersection
- load standard 7X5 intersection
- load standard 7X6 intersection
- load standard 7X7 intersection
- load standard 4T2 intersection
- load standard 4T3 intersection
- load standard 4T4 intersection

modify intersection
modify alternative
modify leg
modify lane
modify text on alternative
modify text on segment
move alternative
move leg longitudinal
move leg lateral
move lane longitudinal
move lane lateral
move text on alternative
move text on segment
noun-verb
recreate intersection
recreate alternative
re-enter data
rotate alternative
rotate leg
rotate text on alternative
rotate text on segment
save to TEXAS Model file
save to data base
select alternative current
select alternative by ID
select alternative by data point
select alternative next
select alternative previous
select leg current
select leg by ID
select leg by data point
select leg next
select leg previous
select lane current
select lane by data point
select lane next
select lane previous
select inbound lane by ID
select inbound lane next
select inbound lane previous
select outbound lane by ID
select outbound lane next
select outbound lane previous
select segment current
select segment by ID
select segment by data point
select segment next
select segment previous
select text current
select text by ID
select text by data point
select text next
select text previous
show information full
show information short
show station/offset
tools TEXAS Model graphics
tools turn template for P
tools turn template for BUS
tools turn template for ABUS
tools turn template for SU
tools turn template for WB40
tools turn template for WB50
tools turn template for WB60
tools turn template for WB62
tools turn template for RMD
tools turn template for WB114
tools turn template for WB96
tools turn template for PT
tools turn template for PB
tools turn template for MH
verb-noun
view all alternatives on
view all alternatives off
view current alternative on
view current alternative off
view current leg centerline on
view current leg centerline off
view current lane on
view current lane off
view current text on

SUMMARY

First stage implementation of the functional design described within this chapter has been completed. The IGIDS System which has evolved is a very useful planning tool but will require second-stage implementation before it becomes the planned computer-aided design tool.

The first-stage system operates on an Intergraph workstation using MicroStation and Informix as graphics and database engines. The system permits interactive computer assisted sketch planning of major intersection components. A typical line drawing of leg, lane, and other geometric elements as produced by the first stage system is provided as Figure 4.4. The system provides easy access to the TEXAS Model and TXTOM analysis packages.

As indicated earlier, the complete IGIDS System will provide significantly more design and analysis capability than the first stage system.

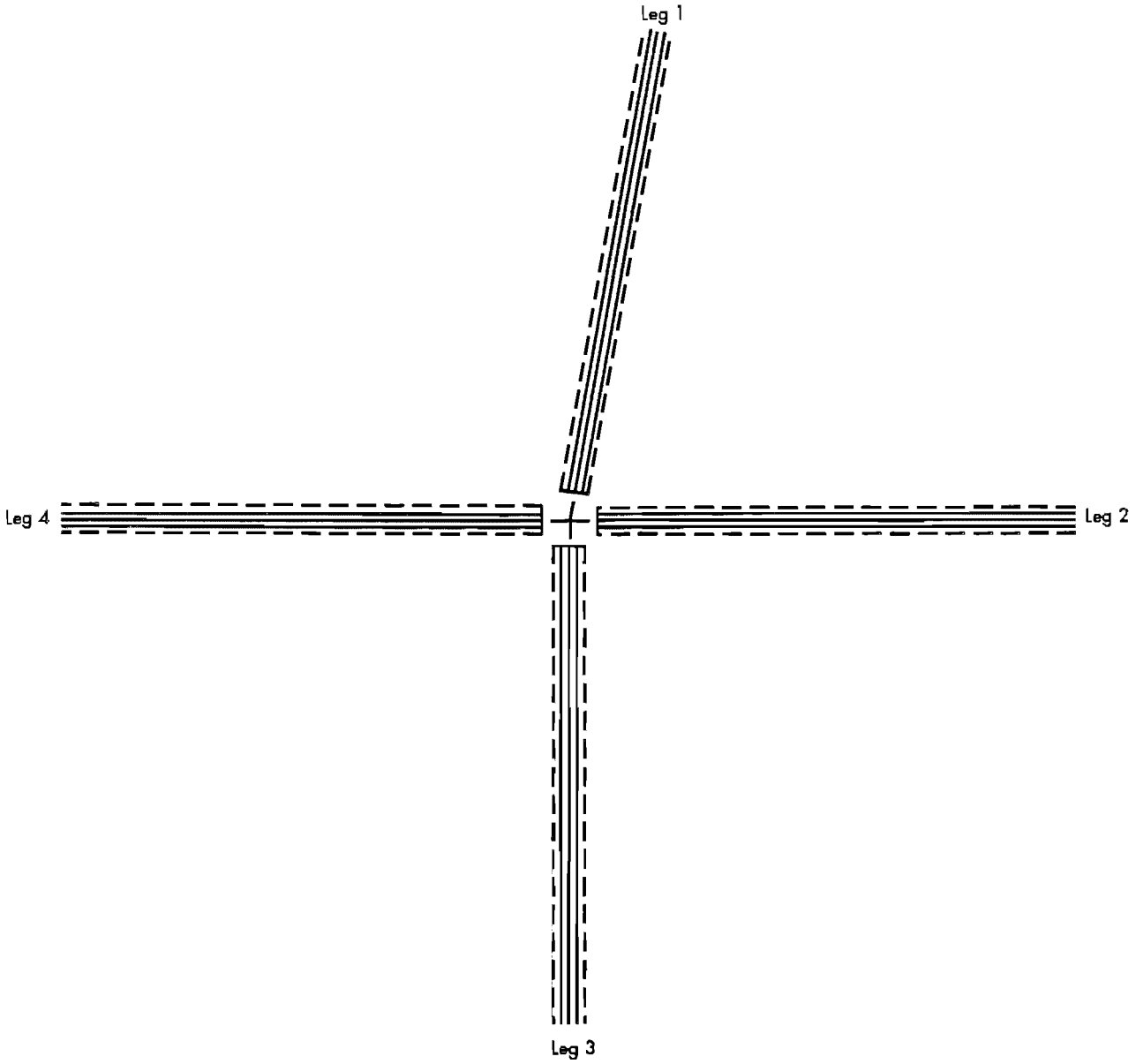


Figure 4.4 Major intersection elements as depicted by Phase 1 IGIDS screen graphics

REFERENCES

1. *A Policy on Geometric Design of Highways and Streets 1990*, American Association of State Highway and Transportation Officials, Washington, D.C.

2. *Highway Capacity Manual 1985*, Transportation Research Board Report 209.

3. SOAP84 (Review Manual), Implementation Package, US Government Printing Office, 1985-501-737-20.130, January 1985.

4. Chang, Edmond Chin-Ping, "Interactive Intersection Design Using an Expert Systems Approach," Transportation Research Board, *Transportation Research Record* 1239, 1986.

5. Fambro, Daniel B., J. M. Mason, Jr., and N. S. Cline, "Intersection Channelization Guidelines for Longer and Wider Trucks," *Transportation Research Record* 1195, pp 48-63, 1988.

6. *Highway Design Division Operations and Procedures Manual* (Rev. 1-81), Texas State Department of Highways and Public Transportation, Appendix D, "Capacity and Level of Service."

7. *Highway Design Division Operations and Procedures Manual* (Rev. 1-81), Texas State Department of Highways and Public Transportation, Part IV, Rev. Feb 20, 1987.

8. *Highway Capacity Manual 1965*, Highway Research Board Special Report 87, Chapter 6.

9. *Traffic Engineering Handbook*, 3rd. ed., Institute of Traffic Engineers, Washington, D.C., 1965, p 662.

10. *Transportation and Traffic Engineering Handbook*, 2nd ed., Institute of Transportation Engineers, Washington, D.C., pp 605-609.

11. *Intersection Channelization Design Guide*, National Cooperative Highway Research Program Report No. 279, Transportation Research Board, National Research Council, 1985.

12. Stover, Vergil G., and F. J. Koepke, *Transportation and Land Development*, Institute of Transportation Engineers, Prentice Hall, 1988.

13. Homburger, Wolfgang S. and J. H. Kell, "Fundamentals of Traffic Engineering," 12th Ed., University of California at Berkeley, Institute of Transportation Studies, 1988. Report No. UCB-ITS-CN-88-1, Chapters 20 and 21.

14. "Design Criteria for Left Turn Channelization," *ITE Journal*, Institute of Transportation Engineers, Feb 1981.

15. Simkowitz, Howard J., "Integrating Geographic Information System Technology and Transportation Models," Transportation Research Board, National Research Council, *Transportation Research Record* 1271, 1990, pp 44-47.

16. *Traffic Engineering Handbook*, 4th Ed., Institute of Transportation Engineers, Chapter 6 "Geometric Design," pp 113-128, 1991.

17. Kyte, Michael, "Estimating Capacity of an All-Way-Stop-Controlled Intersection," *Transportation Research Record 1287*, 1990.

18. Messer, C. J., and D. B. Fambro, *A Guide for Designing and Operating Signalized Intersections in Texas*, Texas Transportation Institute Research Report 203-1, 1975.

19. Hugo, F., J. T. O'Connor, and W. V. Ward, *Highway Constructability Guide*, Center for Transportation Research, Bureau of Engi-

neering Research, The University of Texas at Austin, 1990.

20. *Evaluation of Design Effectiveness*, Construction Industry Institute, The University of Texas at Austin, 1986.

21. *Vehicle Turning Characteristics for Use in Geometric Design*, Texas State Department of Highways and Public Transportation, 1987.

22. Jack E. Leisch & Associates, *Planning and Design Guide At-Grade Intersections*, 1981, Rev. 4/15/88, Evanston, Illinois.

23. Riggs, James L., "What's the Score?," *The Military Engineer*, Sept-Oct 1985, Vol 77, No. 503, pp 496-499.

24. *CEAL User's Manual*, Civil Engineering Automation Library, CLM/Systems Inc., p 2.376 ff, 1991.

25. *Highway Capacity Software User's Manual*, U.S. Department of Transportation, Federal Highway Administration, Jan 1987, Revised Nov 1987, Chapters 1, 2, 9, and 10.

26. Highway *Capacity Software — Signalized Intersections*, Release 2, McTrans Center, University of Florida, 1990.

27. *Arterial Timing Optimization Using PASSER II-90 — Program User's Manual*, Revised Draft, June 30, 1991, Research Report 467-2F, Texas Transportation Institute, Texas A&M University.

28. *A Report on the User's Manual for the Microcomputer Version of PASSER III-88*, Research Report 478-1, Texas Transportation Institute, Texas A&M University, September 1988.

29. *TRANSYT-7F User's Manual*, Transportation Research Center, University of Florida, Oct 1988.

30. Lee, Clyde E., et al, *The TEXAS Model for Intersection Traffic — User's Guide*, Research Report 184-3, Center for Transportation Research, Bureau of Engineering Research, The University of Texas at Austin, July 1977.

31. Lee, Clyde E., et al, *User-Friendly TEXAS Model — Guide to Data Entry*, Research Report 361-1F, Center for Transportation Research, Bureau of Engineering Research, The University of Texas at Austin, Nov 1985.

32. Lee, Clyde E., et al, *The TEXAS Model Version 3.0 (Diamond Interchanges)*, Research Report 443-1F, Center for Transportation Research, Bureau of Engineering Research, The University of Texas at Austin, January 1989.

33. Wallace, C. E., and Kenneth G. Courage, *Arterial Analysis Package (AAP) User's Guide*, Vol 2 of a Series Prepared for FHWA by Courage and Wallace, Gainsville, Florida, Dec 1990.

34. *TRAF-NETSIM User's Manual*, U.S. Department of Transportation, Federal Highway Administration, Nov 1989.

35. *Level III Design Training — Preliminary Design*, Texas State Department of Highways and Public Transportation, Austin, Texas.

36. *MicroStation 32 User's Guide*, Bentley Systems, Inc., and Intergraph Corporation, Huntsville, Alabama, 1991.

37. Personal Interview with Ms. Judy Skeen, Manager of Engineering Devlopment, Texas Department of Transportation.

38. Catalog, McTrans Center for Microcomputers in Transportation, University of Florida, Gainesville, Florida, June 1991.

39. Berry, Charles H., *A Development Plan for the Interactive Graphics Intersection Design System (IGIDS),"* Master's Thesis, The University of Texas at Austin, Dec 1991.