The Grid Sketcher: An AutoCad-based tool for conceptual design processes

Brian Martin Gardner
University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/3365

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
THE GRID SKETCHER - AN AutoCAD BASED

TOOL FOR CONCEPTUAL DESIGN

PROCESSES

by

Brian M. Gardner

Bachelor of Science
San Jose State University
1971

Master of Business Administration
San Jose State University
1981

Master of Science
San Jose State University
1984

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Architecture

in

Architecture

School of Architecture
University of Nevada, Las Vegas
December 1997

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Thesis prepared by

Brian M. Gardner

Entitled

The Grid Sketcher - AutoCAD Based Tool for Conceptual Design Processes

is approved in partial fulfillment of the requirements for the degree of

Master of Architecture

Examination Committee Chair

Dean of the Graduate College

Graduate College Faculty Representative
ABSTRACT

The Grid Sketcher - An AutoCAD Based Tool for Conceptual Design Processes

by

Brian M. Gardner

Dr. Hugh Burgess, Examination Committee Chair
Professor of Architecture
University of Nevada, Las Vegas

Sketching with pencil and paper is reminiscent of the varied, rich, and loosely defined formal processes associated with conceptual design. Architects actively engage such creative paradigms in their exploration and development of conceptual design solutions. The Grid Sketcher, as a conceptual sketching tool, presents one possible computer implementation for enhancing and supporting these processes. It effectively demonstrates the facility with which current technology and the computing environment can enhance and simulate sketching intents and expectations.

One pervasive, and troubling, undercurrent however is the conceptual barrier between the variable processes of human thought and those indigenous to computing. Typically with respect to design, the position taken is that the two are virtually void of any fundamental commonality. A designer’s thoughts are intuitive, at times irrational, and rarely follow consistently identifiable patterns. Conversely, computing requires predictability in just these endeavors. Computing is strictly an algorithmic process while thought is not always so predictable. Given these dichotomous relationships, the
computing environment, as commonly defined, can not reasonably expect to mimic the typically human domain of creative design. In this context, this thesis accentuates the computer’s role as a form generator as opposed to a form evaluator. The computer, under the influence of certain contextual parameters can, however, provide the designer with a rich and elegant set of forms that respond through algorithmics to the designer’s creative intents.

The software presented in this thesis is written in AutoLISP and exploits AutoCAD’s capacious 3D environment. Designs and productions respond to a bounded framework where user selected parametric variables of size, scale, proportion, and proximity, all which reflect contextual issues, determine the characteristics of a unit form. Designer selected growth algorithms then arbitrate the spatial relationships between the unit forms and their propagation through the developing design.

While the Sketcher implements only the GRID as an organizational discipline, many other paradigms are possible. Within this grid structure a robust set of editing features, supported by the computer’s inherent speed, allows the designer to analyze successive productions while refining ever more complex solutions. Through creative manipulation of these algorithmic structures ideas eventually coalesce to formalize images that represent a given design problem’s solution set.
TABLE OF CONTENTS

ABSTRACT ..........................................................................................................................iii
LIST OF FIGURES .............................................................................................................viii
ACKNOWLEDGMENTS ....................................................................................................xi

CHAPTER 1 INTRODUCTION .........................................................................................1
  Of Form and Knowledge ...........................................................................................2
  Computability ............................................................................................................3
  Problem Statement .....................................................................................................6
  Propositions and Intentions ........................................................................................7
  Implementation ..........................................................................................................9
  Methodology .............................................................................................................10
  Organization, Mode, and Outcome .........................................................................12

CHAPTER 2 DESIGN INTENTIONS ........................................................................... 14
  Creative Design ........................................................................................................15
  A Further Refinement ...............................................................................................18
  Image and the Computed Pencil ..............................................................................19
  The Inviolability of Dimension .............................................................................21
  Metaphorical Reference ............................................................................................24
  Shapes and Emergent Form .....................................................................................27
  Emergent Value and Design Validity .....................................................................31
  Abstraction and Concept .........................................................................................33
  Ambiguity and Context ............................................................................................36
  The Requirement for Generality ............................................................................38
  Intentions ..................................................................................................................41

CHAPTER 3 REPRESENTATION, ANALYSIS, AND SUPPOSITION ....................43
  Representation and Analysis ...................................................................................44
  Supposition ...............................................................................................................45
  Computational Foundations ....................................................................................46
  Influences ..................................................................................................................49
  Formal Composition ................................................................................................52
  The Formally Composed Metaphor .........................................................................54
  Emergent Lines .........................................................................................................56
  Relational Modeling ................................................................................................57
  Dynamic Form Generation ......................................................................................59
  The Physics of Form ................................................................................................61
LIST OF FIGURES

Figure 1  Attributes of Dimension ................................................................. 21
Figure 2  A Suggestion of Metaphor ................................................................. 25
Figure 3  An Original Shape ........................................................................... 28
Figure 4  Possible Derived Subshapes .............................................................. 29
Figure 5  A Shape Grammar ............................................................................ 79
Figure 6  Four Possible Limits on Growth Space ............................................. 85
Figure 7  The Grid Organizing Schema ............................................................ 86
Figure 8a  Block Replacement Option ............................................................... 88
Figure 8b  EDGES - With Block Substitution .................................................. 89
Figure 9  The Effects of Random Distribution ................................................ 91
Figure 10a  Two Combinations of Three Layers .............................................. 94
Figure 10b  A combination of Four Layers ....................................................... 95
Figure 11  A Set of Production Rules ............................................................... 97
Figure 12  The CORNERS Algorithm .............................................................. 98
Figure 13  The EDGES Algorithm .................................................................. 100
Figure 14  The FACES Algorithm ................................................................... 102
Figure 15a  The STACKS Algorithm ............................................................... 104
Figure 15b  The STACKS Algorithm ............................................................... 105
Figure 16a  The SLOPES Algorithm ............................................................... 108
Figure 16b  The SLOPES Algorithm ............................................................... 109
Figure 16c  The SLOPES Algorithm ............................................................... 110
Figure 16d  The SLOPES Algorithm ............................................................... 111
Figure 17  Overlay Composite - STACKS, SLOPES, FACES ......................... 113
Figure 18  Overlay Composite - STACKS, CORNERS, FACES ....................... 114
Figure 19  Overlay Composite - STACKS, FACES .......................................... 115
Figure 20  Overlay Composite - STACKS, FACES, SLOPES ......................... 116
Figure 21a  A composition in CORNERS .......................................................... 129
Figure 21b  A composition in CORNERS .......................................................... 130
Figure 22a  A composition in EDGES ............................................................... 131
Figure 22b  A composition in EDGES ............................................................... 132
Figure 23a  A composition in FACES ............................................................... 133
Figure 23b  A composition in FACES ............................................................... 134
Figure 23c  A composition in FACES ............................................................... 135
Figure 24a  Height Variations in STACKS ....................................................... 138
Figure 24b  Height Variations in STACKS ....................................................... 139
Figure 24c  Height Variations in STACKS ....................................................... 140
Figure 25a  Eccentricity Variations in STACKS .............................................. 141
Figure 25b  Eccentricity Variations in STACKS .............................................. 142
Figure 26a  Spacing Factor in STACKS ............................................................ 144
Figure 26b  Spacing Factor in STACKS ................................................................. 145
Figure 26c  Spacing Factor in STACKS ................................................................. 146
Figure 26d  Spacing Factor in STACKS ................................................................. 147
Figure 27a  STACKS .............................................................................................. 148
Figure 27b  STACKS .............................................................................................. 149
Figure 27c  STACKS .............................................................................................. 150
Figure 27d  STACKS .............................................................................................. 151
Figure 27e  STACKS .............................................................................................. 152
Figure 28a  Slope Density in SLOPES ................................................................... 155
Figure 28b  Slope Density in SLOPES ................................................................... 156
Figure 28c  Slope Contour in SURFACE SLOPES .............................................. 157
Figure 28d  Slope Character in SLOPES ............................................................... 158
Figure 29  Slope contour & character in SLOPES ................................................. 159
Figure 30  Project Site ........................................................................................... 168
Figure 31  Resort - Initial Derivation ..................................................................... 169
Figure 32  Introducing Circular Form .................................................................... 171
Figure 33  Modified Circular Form ........................................................................ 173
Figure 34a  Complex Central Form ....................................................................... 175
Figure 34b  Complex Central Form ....................................................................... 176
Figure 34c  Complex Central Form ....................................................................... 177
Figure 34d  Complex Central Form ....................................................................... 178
Figure 34e  Complex Central Form ....................................................................... 179
Figure 34f  Complex Central Form ....................................................................... 180
Figure 35a  Central Form with Entry Node ............................................................ 184
Figure 35b  Central Form with Modified Entry Node ............................................ 185
Figure 35c  Form with Multi-Mode Entry Node ..................................................... 186
Figure 35d  Form with Multi-Mode Entry Node ..................................................... 187
Figure 35e  Form with Multi-Mode Entry Node ..................................................... 188
Figure 35f  Form with Multi-Mode Entry Node ..................................................... 189
Figure 36a  Form Extension to Lake ...................................................................... 193
Figure 36b  Form Extension to Lake ...................................................................... 194
Figure 36c  Modified Distribution at East Parcel ................................................... 195
Figure 36d  Modified Distribution at East Parcel ................................................... 196
Figure 36e  Modified Distribution at East Parcel ................................................... 197
Figure 36f  Modified Distribution at East Parcel ................................................... 198
Figure 37a  Manually Modified Distribution ......................................................... 201
Figure 37b  Manually Modified Distribution ......................................................... 202
Figure 37c  Manually Modified Distribution ......................................................... 203
Figure 37d  Manually Modified Distribution ......................................................... 204
Figure 37e  Manually Modified Distribution ......................................................... 205
Figure 37f  Manually Modified Distribution ......................................................... 206
Figure 37g  Manually Modified Distribution ......................................................... 207
Figure 37h  Manually Modified Distribution ......................................................... 208
Figure 37i  Manually Modified Distribution ......................................................... 209
ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my Graduate Thesis Committee for their commitment, encouragement and assistance in reviewing this project in preparation for the final draft. In particular I would like to thank my Thesis Chair, and good friend, Dr. Hugh Burgess, for his inspiration, design expertise, and ever present critical comments throughout this thesis’s evolutionary process. Dr. Burgess posed exceptional standards, and challenges, that demanded both an inquisitive search for solutions and a commitment to excellence.

As well, I feel it imperative to recognize the invaluable assistance rendered by Dr. Filiz Ozel, who was my first advisor and Thesis Committee Chair while she was on the faculty of the School of Architecture. Without her enthusiasm, knowledge of computing, and persistence, I would not have pursued the rich, and productive subject matter of this thesis.

I am also very pleased to say thanks to my friends and classmates, Gary Eddington, Brian Holm, and Mary Venable for their spirited interest, support and thoughtful contributions to my endeavors. The many hours we spent together in the pursuit of architecture were both personally rewarding and invaluable to the work presented here.
CHAPTER 1

INTRODUCTION

"To terms of magnitude, and of direction, must we refer all our conceptions of Form. For the Form of an object is defined when we know its magnitude, actual or relative, in various directions; and Growth involves the same concepts of magnitude and direction, related to the further concept, or 'dimension', of Time."

From On Growth and Form, by D'Arcy Wentworth Thompson

Computer based systems for architecture blossomed along many avenues after their introduction in the early 1970s. The most prevalent related, quite pragmatically, to resolving technical issues such as drafting systems, presentation drawing, technical resolution, isometrics, two and three dimensional object modeling and animation of the finished artifact (Eastman 1989; Mitchell 1992). Now however researchers are beginning to show a heightened sensitivity to computing’s explosive growth in computational power, algorithmic diversity and user accessibility. Such explorations embrace ideas and concepts directed towards demonstrating software designed as a strictly formalistic tool to develop conceptual abstractions of form in design (Volker 1992; Bonn 1989; Terzidis 1992).
1989). These implementations specifically address issues of spatial and formal relationships, and the rather ambiguous interplay within the processes designers manipulate while sketching at the beginning of a design problem.

Of Form and Knowledge

Most of these solutions to computational form generation vigorously pursued a diverse set of theories (and algorithms) which, for the most part, were either knowledge based, or worked within the context of the more generalized but formally explicit shape grammars. Inherent in knowledge based systems is the implication that simulations of value reasoning may reveal the suitability of designs (Coyne et al. 1990). This is a challenging issue fraught with contentious and subjective questions of social perceptions, emergent value (McLaughlin 1993), creative content, and the simulation process' own self-awareness.

Formal processes, those generators explicitly restricted to emanations of architectural form, on the other hand make little pretense to such volatile questions, leaving their content, interpretation, and resolution explicitly to the designer.

Avoiding system specific "expert" decisions simplifies both the definitions and structures developed in a computational support system. Energies can concentrate on form definition in the context of one or more design theories that begin to approximate the same model-space forms a designer's thought processes might produce. The specific task is to use the computational diversity inherent in computer hardware and software to bring conceptual FORM to the computer screen. Computational programs should be
agile explorers, rather than definers, of architectural composition (Novak 1988). The lexicon for such comprehensive, diverse, and suppositional search paradigms finds its definition in various prototypical visual design precepts, for example, scale, proportion, order, adjacency and rhythm (Rasmussen 1959). All of these serve quite eloquently to describe formal design domains.

Knowing that the "sketch" is an intimate expression of evolutionary self-communication, the designer must also perceive the computational process as a legitimate, self-fulfilling, and ideally, a superior analog to available manual drawing alternatives.

Computability

Independent of particular software implementations, the computer stands by itself as a significant design tool. Inherently, computational speed immediately suggests to the designer that concerns for cumbersome manual drawing operations are no longer valid. They now are replaceable by concerns for more productive and efficacious design processes while the computer mechanizes routine drawing tasks. Introducing a modeling program adds a second layer of expediency that allows the computer to realize its drawing potential in the generation of deterministic formal processes. As emphasized by Gianni (1991) the computer and a 3-D modeler now entice the designer to explore forms relatively free of the usual requirement to redraw subsequent transformations. Once formal composition exists in the computer's descriptive environment the usual CAD operations can quickly and expressively invoke the designer's transformational intentions.
Nowhere is there greater proof of the computer's power than in the resolution of 3-D perspective (or even axonometric) projections. There are similarities between a 3-D model existing in the computer's virtual world and a model in the tactile world of reality. For example, in both cases 3-D viewing from various vantage points is virtually unlimited requiring no modification of the model. However, the two descriptions serve an even greater purpose in their contrasts. While the real model is static the virtual one is dynamic. In the virtual model, transformations of form are easily consummated and may even assert themselves in real sequential time.

Uniquely, the computer also presents its images as projections on a two dimensional surface that is always available for manipulation by the designer. There is tremendous creative potential in the controversy over just how to interpret an assumed 3-dimensional object in terms of 2-dimensional perceptions. One appropriate conclusion is that computing enhances the enticement of visual ambiguity and speculative conjecture. These are invaluable exponents of conceptual design inherent to the computed virtual image but not the real model.

At this point, computing unarguably delimits production as a constraint on design and in exchange returns to the designer a greater freedom to search for creative processes. Once the designer views the computer in this way, computational processes begin to assume a greater significance. Logically what follows are expectations that question the possibilities of design oriented algorithms. For example, are there particular algorithmic schemata that will appropriately generate form in response to a specified context? Also,
if such algorithms exist, are they sufficiently capacious to act as facilitators to the exigencies of conceptual design?

Such questions are central to creative design, and in this restricted arena, a lot of work revealed that computer sensitive algorithms are eloquently capable of representing form development in rational ways (Mitchell & McCullough 1991; Stiny 1980; Stiny & Gips 1978). As well, several software implementations (Bonn 1989; Mitchell, Liggett, and Tan 1990; Mitchell et al. 1991; Knight 1991) actually proved algorithmic computability. However almost all of these projects polarized around either theoretical demonstrations of the possibilities of the algorithms, either knowledge based or formal, (Gianni 1991) or specific performance bound examples driven by the explicit requirements of solutions (Gross et al. 1987; Mitchell 1991).

Somewhere in between theory and solution lay possible design tools that begin to coalesce the theoretical implications of form computation while transitioning to the next level of architectural design - concept exploration. The designer cannot adequately recognize, utilize, or react to the full impact of computing systems as design tools unless the system’s computational activity includes all (or almost all) the components of the design process.

Many proposals yielded adequate conceptual forms that the designer can evaluate in terms of possible formal solutions (Muller 1992; Novak 1989; Woodbury et al. 1992). But their generation remained relatively free of any particular design or problem related parameters. Many of the results displayed casual, non-contextual form - a relatively
unrestricted emanation of the form oriented shape grammars and algorithms that defined their production.

These examples clearly demonstrate, however, that pursuing this particular formal development as a conceptual design tool for the architect has worthy potential and, more importantly, that the computing environment represents a rich and diverse design medium. Yet while avoiding specific results oriented knowledge, much of the prior work tends to universally disregard any significant contextual reference at all. However, context is integral and necessary to design, so much that the seed of specific context, for example proportion, scale, and dimension, requires definition as a precursor to more purposeful design oriented computer tools.

Problem Statement

What this discussion, and the evidence, implies so far is that computing serves very well as a pragmatic, goal oriented production tool. But rarely is computing held as a conceptually distinct design medium with unique characteristics and the potential to interact with and stimulate theoretical design.

This thesis' objective then is to explore the proposition that today's advanced computing mechanisms are capable of supporting conceptual, intuitive, and computable algorithms that emulate both the design intentions and the design characteristics imbedded in the conceptual sketch.

One successful outcome of this investigation is a useful and intuitively believable demonstration of computing as an exploratory design environment. While this is the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
intent it is also possible that alternate outcomes may show either a weak relation between computing and conceptual design, or even a negative correspondence.

Propositions and Intentions

Emphasis targets precisely the region of conceptual design for two reasons. First a relative void exists between the computer and the design as concept, and second, and perhaps most important, the conceptual sketch's creative design environment is precisely where designers first encounter the uninhibited challenges presented by the design and its attendant context. It is the arena of formal transformations, conceptual shifts, and concept definition. It also provides the designer the crucible for blending purpose, style, method, and interpretation to distill and congeal the two or three structural concepts that channel the design problem towards an appropriate solution.

The functional implications of successful conceptual design are both pervasive and indispensable. Creative conceptual design subsists in a profoundly elegant space rich and bountiful in opportunities for the imagination. Algorithmic computing, considered in the usual way, is today arguably just as capacious and thus implies a degree of mutual compatibility and interplay between the two. The assumption here is that through such an affinity computing algorithms can replicate the intent of certain specified creative design processes and that such replications are capable of verification.

At the outset two questions arise. First, what are the identifiable indicators, or perhaps perceptual processes, associated with the conceptual environment that designers first embrace in a design, and second, if such processes reveal themselves are they suitable as a basis for comparative evaluation? As an exploratory mechanism, this thesis
presents an implemented computer software package, the Grid Sketcher, capable of generating drawings that exhibit a conceptual nature. The value and utility of the Grid Sketcher resides in a capability to form effective judgments about how well these drawings, and their underlying algorithms parallel conceptual design processes.

The investigations that follow recognize that a certain knowledge is inherent in and necessary to any computerized process. Knowledge defined for these purposes considers just the numerical information that will tie form generation to the contextual issues described by dimension, proportion, scale, proximity, and organization. As widely recognized, these design parameters are basic to any description of architectural form. This thesis considers them either the seminal or essential knowledge required to imbue the design with a contextual nature. More specifically, implicitly underlying the software’s expression is the imposition of GRID as organization. This disciplined assumption derives from seminal knowledge, while the implied dimensions reflect essential knowledge.

Specifically avoided, again, are attempts to further refine the broader cognitive knowledge base as either a discriminator of value, or a description of expectations of a generated form’s final performance. Even though the designer must eventually resolve these questions, this thesis takes the position that the underlying processes are too complex to address here. However central they are to solving design problems, their investigation requires an exceedingly comprehensive analysis to describe their design intentions.
Implementation

As strictly a form generator, the Grid Sketcher’s affluence resides most significantly in the particular algorithms that beget its forms. Any effective implementation must utilize the computer’s strengths of speed, computing agility, graphic interface and software sensitivity. The Grid Sketcher software demonstrates just this potential by exploring several issues. The first is that, for example, in the context of shape grammars, formal processes can form the basis of rather complex computational processes. Second, that these formal production algorithms experience significant enhancement as form generators by forcing their output to conform to one or more contextual parameters while avoiding strictly results based solutions. Third, that the algorithm’s ability to rapidly develop a robust array of alternative conditioned solutions, many surprisingly unexpected, will expand the designer’s field of perception beyond that normally expected using either traditional sketching techniques or typical CAAD detailing and modeling tools.

In this context the Grid Sketcher attempts to develop the computer’s capabilities as a conceptual sketching tool. Similar to the pencil and paper as design metaphor, the software’s algorithms tend to simulate the loosely defined exploratory processes where formal design solutions bubble to the surface under the progressively more informed and refined decisions of the designer. The rules and definitions associated with formal shape grammars set the conceptual foundation for constructing the software’s form generating algorithms. The designer, by interactively defining the dimensions of a unit form, creates the grammar’s initial shape. Growth algorithms then implement parametric production
rules that control the replacement operations that generate intermediate and final forms in
the grammar's language. A completed production evaluates either as a final design, or
more often, as an intermediate template for successive overlays of additional production
algorithms. Productions may be iterative as well where each repetition responds to the
same set of designer specified parameters yet is subject to the software's randomizing
influences. The designer may intervene in the process at any point to evaluate solutions,
modify initial shapes, or select alternative productions.

The system is capable of producing an infinite variety of solutions quickly in the
typical three-dimensional computer environment. Many will be unexpected, and some
typically not thought of by the designer. However the responsibility of evaluating the
efficacy of any solution still remains with the designer since the program's intent is to act
as a design tool rather than a qualitative decision maker.

Methodology

Perceptions of the character of design methods, and just which indicators are most
relevant to the creative design environment, vary among designers. Yet for analysis
certain concepts are sufficiently robust and composed to form an identifiable and
evaluative foundation. While such proposals may project an arbitrary nature, it is still
necessary to recognize them as evidence of process.

For the purposes of defining and exploring the functional implementation of
creative design the following perceptual concepts, frequently referred to in the literature
as components of creative process, seem particularly germane:

  metaphor
  emergent form
Having once identified these core indicators as a conceptual criteria set, they then form one possible characterization of a unified conceptual design regime. A descriptive analysis in Chapter 2 will further define and elaborate each indicator sufficiently to demonstrate the concept and show how it represents and relates to the intent of creativity in conceptual design. As representative design criteria for evaluation purposes, the set represents a basis for inquiry where it is possible to pose questions about how a computer process creates and works within a similar, parallel, and conjunctive design environment. Adequate responses effectively describe the methodology pursued to evaluate the Grid Sketcher’s value as a computerized conceptual design tool.

This thesis offers its own, focused, and in depth analytical assessment of the criteria set and the software. Evaluation of solutions, demonstrated in Chapter 7, will compare the Grid Sketcher’s implemented processes, and drawings, to the set of six conceptual design criteria. Each indicator infers a distinct design associated concept. Drawings derived by the Grid Sketcher present, illustrate, and discuss for each indicator the drawing’s forms as either comparable or unsuitable to the intent of the indicator. The results of each comparison must assume some sort of qualitative rating, for example a good, fair, or poor affinity for design application.

The drawings used for evaluation and exploration are examples derived from two sources: 1) software generated drawings included as illustrations in the body of the thesis and, 2) examples explicitly related to the studio design project. Specifically, the
design project, briefly presented in Chapter 6, describes a rather reclusive resort set between a desert mountain and a lake shore. All the drawings for the design project respond to real time dimensional parameters derived from the project’s program.

The Grid Sketcher’s initial motivation reflects its early purpose as an AutoCAD based tool to expand and accelerate the design project’s solution space. As software development proceeded it became convenient to exploit the Grid Sketcher as the central topic and focus of this written thesis.

Organization, Mode, and Outcome

This thesis first explores several significant issues of formal design as a precursor to the foundation of a software design paradigm. This analysis, presented in Chapter 2, attempts to identify and demonstrate the viability of a compatible set of conceptual concepts as one possible methodology for deriving and implementing a computational form generating system. Next, Chapter 3 develops a rather in-depth look at similar computerized methods and sets the background for the software introduced later. The quest here is to identify a particular void where conceptual design issues can successfully interleave with capable user oriented software to yield an efficacious design tool.

Following this Chapter 4 introduces the Grid Sketcher’s form generation algorithms, including the influence of shape grammars on their development. Chapter 5 then presents the software implementation in AutoLISP/AutoCAD and reveals the parametric variations that allow a system user to create and control the various emanations generated through iterative explorations. An initial investigation of the LISP
programming language and its interface with AutoCAD was integral to developing the software's algorithms.

As indicated earlier, Chapter 6 details a corporeal design project, one of significant formal content. Discussion and illustration merge to develop a rather robust example of design resolution. The process is logically sequential and effectively demonstrates many of the Grid Sketcher's algorithmic attributes and drawing tools.

The last Chapter implements a series of evaluations demonstrating the practical value and worth of the system as an enhancement to the intuitive design processes. In particular, this analysis will strive to demonstrate the Grid Sketcher's capabilities as a formal analogue to the designer's pencil sketches. As a summary note, the Grid Sketcher's purpose is not to replicate exactly the pencil's strokes but rather to exist in a mode that mimics the intent and expressive strengths inherent in the conceptual design environment's modal processes. A successful solution will further solidify computing's claim to a place at the table of architectural design. Finally, the chapter offers the conclusions drawn from using and evaluating the system, and will detail further enhancements that if added could improve the systems capabilities.
CHAPTER 2

DESIGN INTENTIONS

This chapter considers several fundamental issues central to creative design. The investigation describes both a functional context for the Grid Sketcher and elaborates on the set of perceptive concepts introduced earlier in Chapter 1. While design is a universal striving, and achievement, for numerous human disciplines the following discussion considers creativity from the particular point of view of the Architect. Both the work of architecture and the internalized processes the architect manipulates to produce the work of architecture are fairly treated as a unified design discipline of unique character and purpose.

Archea (1987) suggested that the design process is fundamentally different from the typical "problem-solving" process in which "desired effects are stated as explicit criteria and the known limitations to achieving those effects are stated as explicit constraints before a course of action is initiated". Rather the architect's unique sense of design is an attempt at creating a combination of effects that is unique to specific time, place and context. The solution process follows a development path derived from a
unique combination of rules, precedent, metaphor, image, and architectural detail that form an appropriately coherent design.

At the outset the architect knows neither the particular rules, the collection of parts, nor the specific relations among them that will reflect in a particular end. The process is an iterative search through ever more resolved combinations to find a result that is acceptable and complete. Woodbury (1991) similarly expresses the process as an exploration through "spaces of designs" where design transformations successively derive other designs. His description presents the "reality" of the designer's working environment in terms of a design space metaphor.

Recognizing this particular interpretation of architectural design provides a loosely defined regime that is sufficiently intuitive and flexible to allow recognition of both the concept and concretization of the concept simultaneously. The concept is that seed of thought that is the beginning of the process. Concretization derives from the rules, parts, images, and tools available to the process. Certainly the architect advances "process" by manipulating design tools (pencil and paper) in successive repetitions towards an end. Computer tools should act and appear just as pliable.

Creative Design

Designers work in this ill-structured creative arena with information and knowledge gained through experience and research. Creative designs, according to Richard Norman (1987) converge towards a solution following an intuitive leap. Although intuition is not a computable talent, the computing environment can substantially facilitate the intuitive database through generation and suggestion of
alternatives. The implicit statement in this position is that the computer makes no pretense to possessing inherent decision making processes or knowledge based manipulations. Its processes are strictly formal.

The Grid Sketcher intends to embrace this intuitive, conceptual, and iterative realm of design. Since the analogies between the computed line and the pencil reflect architectural issues, a more detailed exploration of creativity in architectural design seems appropriate.

Synthesis of form follows from exploratory processes that explicitly pursue formal solutions. These processes evolve around varying concepts derived from design methods typically thought of as creative (Coyne & Subrahmanian 1993; Logan & Smithers 1993, Novak 1988). Koberg and Bagnall (1991) offer a thorough and illuminating dissection of one such interpretation of creative process. Theirs is particularly interesting for its comprehensive sensitivity to lifestyle and personal philosophy. Throughout their book, The Universal Traveler, they stress a completely cognitive view of design creativity that leads essentially to a lifestyle of creative behavior as a problem solving paradigm. While not specifically related to the computing environment, such a description of creativity emphasizes a constructive attitude different from most based on awareness, active involvement in a defined goal oriented process, and thinking clearly.

The undeniable strength of their position derives from the implied discipline that subsumes all the process’ functional components. Discipline of purpose, point of view,
and expectations is crucial to creativity. This is particularly noteworthy as an extremely important formal concept to balance the more philosophical analysis that follows.

The concept of creative design is essential to understanding the intention of the Grid Sketcher, for it is in just this particular domain that the software's forms belong. To help clarify the meaning of the "seed of thought" in creative design, it is instructive to consider the interpretive ideas of fantasy, imagination, and reality (Antoniades 1990). The three concepts are interdependent and actively interrelated by their mutual influence on creativeness at different levels of endeavor. Fantasy suggests a rather boundless realm where ideas, unfettered by reality, metamorphose across conceptual states of unknown derivation. Such ideas could perhaps never exist in the physical world, but yet provide a collage of images that form a perceptual background for the imagination.

Imagination on the other hand can see objects in the mind that exist explicitly in the real world, yet are not immediately observable. A fanatical thought or idea may tangentially graze a more concrete and familiar image residing just at that moment in the designer's mind. The designer's pencil strokes on paper then may represent just this fantasy and serve to excite and inform the dialectic interests of the imagination.

Effectively filtering fantasy through the imagination leads to interpretations of fanciful thoughts that can assume a physical interpretation and existence in reality. To do so though really requires the designer to assume two mutually supportive attitudes. At one extreme creative design solutions require professional intents and outcomes (reality) while at the other they must exploit the less prosaic humanitarian, spiritual, philosophical,
and visual needs of human existence. Without these two adjacent influences design solutions will invariably fail the test of creativity.

A Further Refinement

Creative designs reside in a region of possible solutions that is by definition fundamentally different from what is presently in existence. Even though a problem’s contextual and program requirements may follow those of an existing design, recasting these requirements as a conceptually different set of parametric design variables invariably leads to an alternate design region replete with a distinct sense of creativity. In a further refinement, Rosenman and Gero (1993) suggest three distinct design regimes: routine design, innovative design, and creative design. Using their descriptions as a point of departure, routine designs follow from essentially predetermined solutions that respond very pragmatically to new values for pre-defined variables. Routine design iteratively generates similar instances of the same type.

The common architectural problem of house design provides an example. A program requiring 1200 square feet and five rooms solved by conceiving the solution set as just the totality of the all the possible ways of partitioning a rectangle into five spaces, using standard components, is routine design. Innovative design adds the possibility of transforming existing design solutions by introducing conceptual ideas about transformation processes. Extending the house problem to include, for example, wall as window or window as wall yields a more innovative interpretation of design variables. Yet the house remains still, fundamentally, a house. The glass surfaces of Gropius’ Bauhaus aptly demonstrate this sense of transformation and are, among their other
exceptional architectural expressions, an extremely erudite and expressive definition of
wall.

While the Bauhaus is unquestionably creative design, a more conceptually
straightforward example is Philip Johnson's Glass House. Creative design requires more
than just extending existing program variables. What is necessary is an obvious, virtual
recasting of design intentions to either modify existing variables, or establish an alternate
set of variables, that define a not yet existing solution domain. The Glass House recasts
the "house" as metaphorical layered space where the interior layer subtly separates from
the exterior layer by nothing more than the most minimal of structural elements. The
structure is just sufficient to define that edge, otherwise the two spaces are continuous.
Interior and exterior become extensions of each other.

This particular definition of design, creative design, expresses precisely the realm,
the spirit, and intention underlying the Grid Sketcher's development.

Image and The Computed Pencil

Taking the position that a CAAD system's responsibility is to supplant in some
discernible way the designer's creative sketching intentions requires a considered
statement about just the character of those intentions. First of all, as noted earlier, the
designer works within a transformational continuum that somehow begins to codify,
perhaps quite often abstractly, a contextual environment for the design. The particular
context follows various callings, for example, context of the external environment,
context of the designer's particular interests and expectations, context of visual
perception of forms, or context of physical representations.
These cognitive contextual issues begin to form the knowledge base the designer uses as an interpretive envelope for the developing sketch. In response, the internalized images the designer embraces take substantive meaning from the sketch's qualitative content. This requires then that the computer maintain capacious and unusually rich computed images that show a strong affinity for the mental images to which the designer responds.

Rudolf Arnheim (1969) in his book *Visual Thinking* advances the tenet that we think just particularly in this “realm of images." These images of thought, derived from imagination, move along a continuum polarized at one extent by the almost perfect analog of reality, to the opposite where the mind attains highly abstracted and often subliminal images. As the mind works, the abstractive world tends to grow at the expense of reality, on balance, because of the abstraction’s capacious ability to represent a single reality over a range of differing images.

Architecture's visually oriented design disciplines entice, and require, that thought manipulate its processes in the language of images. Where mathematical relationships communicate for the mathematician, visual, drawn images communicate for the architect. Perhaps the metaphysical determinant of architecture is the image, the particular image of fantasy and imagination.

For the designer, thought's compendium of mental images, immersed in abstraction, must begin to take meaning from the coexisting contextual information associated with the selected design. In particular it is the context conditioned abstractive thought and image manipulations that the designer lays out before himself on sketch
paper. Even the first line, representing perhaps the ultimate abstraction, has some formal, architectural meaning. The computational exercise presented on the CRT is no more or no less than the sketched line, even considering its precise definition in Euclidean space. Consistency is important as well, and the level of abstraction in the computer, no matter how complex the image, should parallel at all times that of the designer's drawn sketch.

There is no need to mimic exactly the pencil's strokes, but rather just the intent that motivates the strokes. As long as the designer's intentions, and response, exist equally in both the pencil's shapes and the computed shapes, the precision and discipline of computation are useful.

The Inviolability of Dimension

Some consideration of dimension and its cohesive role as a unifying determinate requires discussion as a precursor to further elaboration and descriptive analysis of the conceptual design criteria set. By extension, dimension commonly distributes over all physical emanations of reality, and is an essential attribute of all of them (Thompson 1961). Perhaps dimension becomes the initializing conduit between image and the first ties to context and reality, for no matter what physical attribute an image attaches to it is somehow dimensional. Scale, proportion, and size all refer to dimension, and serve to expand the contextual expression that furthers progress towards reality. Area, volume, and mass modulate by change in dimension and even proximity and mobility assume significant meaning in the presence of dimensional variance. Geometry's particular dimensional orientation takes on special meaning in the realm of design. As shown by
Antoniades (1990) there is clear and compelling support of architectural creativity by the clarity, appeal, and topological consistency of geometric form.

Given the pervasiveness then of dimension the computer must recognize and pay close attention to this fundamental determinant. Fortunately, the computer does so at its most seminal foundations since by definition computing is a numeric system defined, for design purposes, over Euclidean dimension.

As mentioned earlier a sketched shape of geometric topology, no matter how abstract and primordial, reflects from the start some contextual expression and sense of dimension. The computing software must show an adequate capacity to satisfy the designer’s conceptual need for this expression. One way to meet this requirement is through designer stipulation of parametric variables, in particular the shape’s dimensions along the three coordinate axes. Any shape, regardless of its degree of complexity, is through an additive process reducible to its unique set of maximal lines (Stiny 1990). All lines inescapably recognize at least one inviolable physical characteristic - their Euclidean length, or dimension.

While a single line is necessarily any meaningful sense the first expression of form, it has difficulty conveying of relative dimension. On the other hand a pair of lines, in any topological relation to each other, clearly defines at least the dimensional attribute of proportion. In figure 1-a the single line, while completely known to the computer’s dimensionally oriented database, holds no meaning except a division of space, i.e., either one side or the other. Figure 1-b expresses a distance relation since additional line is some proportion (perhaps the Golden Mean) of the first. As well, the two lines begin to say
something about the space between them. Figure 1-c extends the dimensional expression out along all three axes.

The shapes of figure 1 represent possibly the most abstract level at which designer’s draw images. Clearly, the computer can sketch these shapes as well as the pencil provided the designer has access to the variables that control dimension. By repeatedly adjusting such parameters the designer iteratively adds lines, and shapes, in combinations of ever more complex forms. At this juncture the software has at least replicated the pencil.

Figure 1 Attributes of Dimension

Clearly demonstrated is the computer’s seminal relation to conceptual design by the temerity with which it manipulates dimension. The investigations that follow proceed
in this particular context of computer sketching. All the Grid Sketcher's forms, which must interactively stimulate the designer's perceptions and imagination within the arena of creative design, respond essentially to the designer's manipulation of dimension.

Metaphorical Reference

Metaphors are pervasive, reflecting a universal truth in thought and communication, and compelling arguments exist supporting the inclusive nature of their metaphorical reference (Coyne 1992; Fargas and Papazian 1992). At its most descriptive, a metaphorical event happens when something understandable, either a concept or an object, is "seen" in terms of another or "looks like" something else. The grid appears as a molecular lattice, the sky as a protective blanket, communication as self-fulfillment, or self-determination as power, for example. These four examples represent two instances each of both tangible (object oriented) and intangible (concept oriented) metaphor (Antoniades 1990). It is easier to assimilate a tangible metaphor particularly if the object's visual defining characteristics are obvious, but the intangible metaphor may be more useful to a design's interpretations. Although the typical metaphor moves in a singular mode, the metaphorical transfer yields greater power and meaning when both types of reference work together. It is informative to note also that the mechanism of the metaphor is, of its own right, a creative process and suggestive of other creative processes.

The designer first embracing a conceptual design problem looks for tangible, object oriented visual images imbedded in sketches while mental images translate between both the concept and the sketch. Concepts associated with the problem's context
and the temporal emanations of the concept in physical terms assume an ever more important role as the metaphorical transfer moves into the intangible. The sketch progressively develops into a metaphorical stimulus for other adjacent ideas about which the designer has additional and relevant contextual information. Eventually the metaphor may suggest perhaps the first vestiges of a defining idea's concretization. A metaphorical reference can serve as the initial stimulus in the progression towards reality.

Expressive metaphorical sketches exhibit certain characteristics that stimulate and enhance metaphorical interplay, interpretation, and response. For example, sketched forms, in their holistic structure and visual presentation, should freely suggest other forms, ideally associated with a reference conceptually or visually detached from the original. The form in Figure 2, a literal compendium of circles, might suggest a biological process, an arrangement for a physical barrier, or an organizing theme that implies broken process. From the designer's point of view the sketch's form should repeatedly elicit the rhetorical (or perhaps logical) question, "What is that?".

Sketched forms should reveal as well some basal affinity for the context in which the designer considers them. Contextual issues are just those about which the computer is relatively uninformed, yet without them metaphorical reference is virtually meaningless. Context reflects the power inherent in the knowledgeable background of the designer and the computing environment should react appropriately by maintaining a structure that reflects as much contextual information as possible.

Since architectural design is at issue, the forms should exhibit a typically architectural image supported by familiar and recognizable architectural attributes. As
examples of architectural character and image, the circles in Figure 2 are all, topologically, circles, a basic form considered an architectural centerpiece. As well, their radii varys, while the displacement between them displays a pleasantly rhythmic architectural character. Maintaining a visual sense of architectural structure enlivens the repertoire of tangible metaphor at the designer’s disposal.

Figure 2 A Suggestion of Metaphor

The sketched forms must display, or at least suggest, a strength of metaphorical purpose; a sense of undeniable virtuosity suggesting a certainty of fundamental principles. For example, principles that bespeak of strength and purity of form, discipline, or perhaps undeniable beauty. Metaphorical transfer from the familiar to the
unfamiliar is rarely void of complexity so by expressing an unmitigable clarity of purpose
the form solidifies its statement while enhancing its ability for interpretation.

Shapes and Emergent Form

The forms of architecture’s design processes often appear uniquely inspiring. They represent a continuum of precedent ideas refined by a persuasive history, architectural movements, and at times blatant iconoclastic departures. A regularized geometry underlies virtually every formal composition that possesses architectural character no matter how remote that geometry may first seem. Compare the temples of classical Greece with Cubism’s expressively multidimensional forms. The two virtually deny most of each other’s generative determinants yet both clearly portray an allegiance to highly articulated and controlled rectilinear form. Again by comparison, a building of deconstructivist orientation falls apart along apparent random axes yet unless its derivation is completely stochastic it is possible to discern a supportive systematic geometry, an internal logic, that bows to both the hand of rational determination and regularity. Typically a formal geometry stands as a point of departure (Tschumi 1989).

Designers, particularly in architecture, tend to speak of form as either the idealized realization of Platonic shapes or more often as a refined form representing the comprehensive description of an artifact. Even this sense of completeness is not at the outset a completely valid image, however. It is rather a distillation of structural components, the “structural skeleton” representative of the form’s most notable components, that the designer first perceives (Arnheim 1974). A new form’s initial perceptions project from the form’s generalities and with increasing designer familiarity
explicit definition of the form's constituent shapes improve as descriptions of the form's visual character. So the repetitive process of selectively filtering the generalities of form through the details of shape oscillates between the general and the particular. A generalized form finally reveals itself followed by a discrimination of its details in subshapes which then coalesce to reveal a different generalized form.

Shapes then, while still easily representing basic geometries, assume a pivotal role in the composition of form. At first thought, shapes might easily appear as simply the building blocks of a larger and more progressive form. While it is acceptable to assume this particular shape utility, they carry, as Milton Tan (1990a) notes, a far greater responsibility as facilitators of design transformation. Gero and Yan (1993), Tan (1990b), and Muller (1992) also acknowledge shape's dominating influence as a component of the visual. They consider shapes sufficiently important to justify real time computer implementations directed at defining and illuminating emergent component shapes. Stiny (1993) likewise recognizes the essential nature of emergence by fully integrating the functionalism of his work with shape grammars. It is through the simplicity and expressiveness of the constituent shape that form evolves to assume, at any particular instant its uniquely defined character.

As design progresses, the architect reconceptualizes forms through their suggested images towards a greater meaning. A particular form contains in its bindings a plethora of subforms, or more directly subshapes, that can conceptually reveal emerging forms of different intent and composition. For example, the three overlapping squares in Figure 3 taken together appear very regular and perhaps non-controversial. But another
interpretation reveals an emerging schema of subforms available for use as a generator of other forms. Figures 4a - 4e illustrate subshapes that, although still of a rectilinear topology, can assume varied interpretations. The form’s original definition suggests many possible restatements of its subshapes. In particular, if the form of Figure 3 defines a closed space, then Figure 4c might assume the position of open space juxtaposed over enclosed space.

Figure 3 An Original Shape

The active search for emergent forms embedded in other forms represents implicitly what the architect does through layers of tracing paper. Lines traced on the top layer represent an unforeseen and unique combination of those on the layer, or layers, below. This recursive process of identifying emergent shapes and recombining them into
ever more expressive form is a very powerful method of imaginative perceptual exploration.

Figure 4 Possible Derived Subshapes

Two computed paradigms (Tan 1990a; Tan 1990b) explicitly recognize the creative impact of emergent form. Each implement algorithms that maintain a data structure in which emergent shapes are both recognized and topologically defined. These systems are noteworthy and influential for their elucidation of two issues: first that emergent shapes require the designer's recognition before they become useful, and second, that they must yield to manipulation. Such computational exercises emphasize the point that any computing environment that approaches conceptualization and creativity must accommodate the pervasive complexity of emergent shape.
Emergent Value and Design Validity

As previously mentioned, the designer’s personal perceptions of a design’s context are multidimensional and extremely varied. The developing design must recognize at least a cross-section of these contextual issues to document its validity as a potential design solution. McLaughlin (1993) explicitly emphasizes knowledge of the contextual determinant as a fundamental and necessary influence on creative design. She then proposes that the uniquely creative value of an artifact emanates almost exclusively from a particular set of just such determinants. In essence, the proof of creative process resides in the creative products it produces. In turn then a creative product attains its definition and meaning from a unique combination of "existing values, attitudes, and knowledge" of society. While some interpretations emphasize process as the definitive ground for creative design, McLaughlin recognizes the conceptually stronger influences derived from the boundless world of contextual reference.

To be thought of as creative a product a design must also express originality. Such originality assumes that the set of contextual interpretations and relationships that define the product’s value must also somehow be unique. This requirement implies that the value of a product is unknown at the beginning of the design process and evolves through the designer’s branching decisions in response to the design’s emerging value.

This view of creative design raises the important and necessary question of how well a computational process performs in supporting creative designs. A computing process may by some standard adequately define a form but recognizing the form as a creative solution requires a synthesis of contextual intents and values. The salient point is
that perhaps typically internalized computational processes are simply incapable of
defining algorithms that even tangentially represent human value systems. If such is the
case, or even if computerization can not do so with reasonable effort, then a valid position
exists to functionally separate form generation from form evaluation.

Such a division sets two distinct frames of reference in creative process and
effectively allows the pursuit of computational issues separately for each. This is
particularly useful and convenient, because it recognizes the inherent attributes and
strengths of both the human designer and the mechanized computer. Even with
sophisticated attempts by expert systems to model human knowledge, the supposition that
computer based processes can replicate the structure of human thought in any meaningful
way is still very weak. The issue then is finding just those implementations that will
apply the computer's expansive computational abilities in ways that enhance and entice
the designer's proclivity for manipulating value judgments.

Among the existing paradigms describing a computational view of creative
design, three information sets seem necessary for an understanding and summary of the
underlying process. In the realm of contextual value and computation, the first is
sufficient insight into ideas that describe the designer's perceptual schema as background
for understanding how to effectively generate visual form. Second, a certain definitive
knowledge of algorithmic structure, sequential procedure, and the injunctive layering of
functions is necessary to assimilating the relationships of computational process. Third,
basic knowledge that is most typical of the architectural domain, compared to engineering
for example, must exert itself as the subject matter of computation.
For the purposes of investigation, the specific issues presented so far in this chapter serve to describe the designer's perceptual environment. (Note that the particular implementation oriented nature of computation is the subject of chapter 4). Selected architectural knowledge derives its potential for expression from exactly the knowledge of perceptions. These expressions are maximal when the character of the forms chosen to convey such expression yield to and facilitate interpretation within the knowledge base's perceptual criteria. Any particular architectural parameters associated with algorithmic definitions must then maximize the generation of these particular forms.

The issue of formal architectural character finds its definition in the algorithms implemented by the software. These algorithms reflect the structure of certain combinatorial processes within the architectural domain and the dimensional parameters that describe scale, size, proportion, and proximity. Form then becomes the mediator between perception and generation where the algorithms specifically project a rich composition of architectural detail, complexity and design versatility.

In this way, computation of form receives significant emphasis over computation of value as the most effective and compassionate use of the computer. The continuously developing contextual meanings that represent emergent value express the aura of the designer rather than the computer. A solution's candidacy as a creative artifact then depends upon how well the designer interprets contextual influences and how well the computer responds to the designer's intentions.

Abstraction and Concept
Returning to the realm of images, and thought's interpretations of images, it is plausible to assume that images represent abstractions since much of thought is abstract. A suitable description of abstract thought and its role in creative design might begin by referring again to Arnheim (1969). He proposes that for an abstractive idea to effectively represent productive thought the abstraction must hold the "structural essence", or structural properties of the object or idea the abstraction represents. This perception recognizes that the most useful abstractions characterize their referents through not just a particular set of attributes or characteristics but rather by eliciting the image of what is most meaningful or important in a particular referential context. For example assume that a nicely grouped set of three small tables and chairs, all of superior material and craftsmanship, sits close to three rudimentary card tables and their chairs. Clearly both sets share at least the commonality that they are furniture and the image of furniture as a particular abstraction of the two groupings is quite effective. The abstract concept of furniture carries with it certain connotations, or generalities, about furniture but in any given context such an abstraction might be meaningless. The two sets of three tables and chairs are capable of portraying other abstract images, for example the noteworthy difference in quality.

As a more meaningful abstraction, quality has a stronger impact if the contextual setting intended an emotional response. Even if this was not the case the abstract image of quality, independent of context, carries a greater value than the abstraction of furniture. Quality as an abstraction also suggests the possibility of metaphorical reference. For example, the three tables may loose their functional utility in the metaphorical
interpretation of Beauty for aesthetic purpose. Perceived in this way, the tables become an art form’s expressive artifacts, devoid of any meaningful reference to furniture.

Abstractive images generalize the most influential issues common to a set of similar ideas or objects (Tan 1990b). But the concept of generalizing apparently shares an equally seminal influence with abstraction since the thought that created the furniture, and its context, must have considered the abstraction of Beauty prior to expressing it. Abstraction and generalization maintain cohesive and supporting roles, an issue of importance for its usefulness in considering the implications of creative form. A form’s salient character may suggest a new abstraction while at the same time being itself an intermediate product of a prior abstraction.

Intrinsically bound within abstraction is the useful connotation of concept. An abstract image that begins to hold for many instances, or iterations, of a form begins to suggest a concept of greater import. Using the set of tables and chairs again, if the image of Beauty achieves further concretization by redefining the dimensions, or perhaps the finish materials, or even the structural composition, then the developing image becomes a more tenacious creative concept.

The furniture’s contextual setting provides again another transformational opportunity to test the abstraction’s progression towards concept. Assume that adding exceptional natural lighting renders the furniture in a patina of emotionally evocative shades and shadows. Lighting then serves as an additional object over which the abstraction of Beauty extends, but not an object of the abstraction “furniture”. Obviously this contrived sense of beauty can endure a continual stream of transformation which
carries essential meaning for the abstraction. Eventually then the abstractive image comes to hold an impeccable conceptual position in its particular creative context.

The dynamic expansion of an abstracted image presents for the designer an extremely expressive process for arbitrating between generative forms. A given abstraction may ultimately metamorphose into an undeniable design concept. The concept then is the search product extended by the designer pursuing a process of finding and solidifying abstractions.

Ambiguity and Context

An architect's design environment spans, at least initially, a continuum that enjoins obscurity and works its way towards complete equivocation. Perhaps this description is too expressive, but then again perhaps not. Compared to the dogmatic design schema typically followed by Hellenistic architecture, contemporary architects find very little inspiration from any particular unified, clearly elucidated design intention. Not only do design styles, techniques and implementations show almost complete individuality but core design philosophies vary almost linearly over the range of architects (Lawson 1990).

Ambiguity is rampant in design, creative expression, and particularly architecture. Architectural design programs, no matter how refined and constrained they first seem, are deficient in all descriptive attempts except the proliferous enumeration of numbers. All else in the program is flagrantly ambiguous, but fortunately for the designer, this overwhelming ambiguity (just exactly what is a reclusive resort?) at once transcends impossibility to reveal opportunity (Mitchell 1989). Creativeness in design can harness
ambiguity and exploit its abstruse content to exceptional advantage. Specifically, the
designer wants to develop and maintain a forceful presence of ambiguity to promote the
possibilities of contextual shift. The context underlying the designer’s thoughts should
allow stimulus from the sketching process sufficient latitude to suggest alternate
contextual interpretations for the design.

Contextual shifting, or the variation between two or more perceptual viewpoints,
empowers the possibility of alternative configurations. This conceptual vehicle institutes
a dichotomous balancing against the dominating tendency towards concretization. Even
though conceptual issues must eventually converge towards a dominating concept, any
particular set of defining details may not serve well without having fought for its stature
against enigmatic conjecture.

Ambiguity by definition obscures the obvious and subsequently elicits heuristic
exploration of forms by establishing suppositional variance. Certain formal arrangements
may presuppose a designer’s contextual intents, but by remaining recondite the forms
effectively entice the designer’s natural inquisitiveness and quest for definition. To be
more explicit, this process requires the form’s character to assume certain expressive
properties that encourage and enhance creative insight (Finke, Ward & Smith 1992).
Here the authors define a “preinventive structure” that essentially describes the formal
constructs of the conceptual sketch. Such structures, or forms, become particularly adept
at forcing alternate constructs when their sense of ambiguity resides in novelty, emergent
features (forms that project other unexpected forms), and highly conspicuous
incongruities among their features at all visual levels.
In a more computational view Stiny (1989) finds useful ambiguity residing in the many descriptive interpretations of a line, or formal composition of lines. By parsing and reparsing the fundamental component of line the designer may realign basic structure to manipulate ambiguity in the search for formal definition.

Achieving such a formal character implies a finely grained complexity abundant in the capacity for detail. While such detail might possibly increase the form's ambiguous image, only in the detail reside the discrete articulations that eventually converge to integrate context and form. An evolving patina of detail begins to articulate the form's purpose within the designer's interpretations of contextual reference while concurrently readjusting the same referential motif to acknowledge a unique detail's emerging presence.

A fruitful relationship exists between ambiguity and metaphor. While metaphor is a more influential concept manipulator, ambiguity (as well as generality and transformation) is a concept facilitator. The process of refining ambiguous form provides the robust detail that rearticulates emerging forms in their expression of metaphorical transfer.

The Requirement for Generality

Conceptual design harbors numerous perceptual processes as indicated in the previous discussions. These perceptions, taken together, form a substantive capability for exploring the intentions of the designer. Design problems often appear weakly stated, and weakly structured, at the outset even though the design may exist within a well structured and definitive external design context.
It is now necessary to consider the requirement for generality in this loosely coherent design environment. Coyne et al. (1990) proposed that design processes which assimilate general concepts and cope well with generalities demonstrate a greater capacity to enhance creative design. This position is important to conceptual design given the assumption of ill-defined problems. Initial investigations of a design problem must necessarily proceed in generalities rather than specifics, for if the specifics exist the problem solution exists as well.

Generalities are not ambiguities nor necessarily abstractions (although there is a certain generalizing intent in abstraction), but rather take the form of a concept. The designer's creative environment must conspicuously embrace general concepts. For example, repetition, as a design concept, or organization as an architectural concept, or perhaps contrast as a visual concept, and groupings or cohesiveness as a social concept. The designer must think in generalities while simultaneously engaging other perceptual processes. Generalities should inform the designer facilitated by the methods and tools of the process. The process should, moreover, encourage a range of possibilities suggested by contextual implications.

According to Arnheim (1969) generalization is an event where a concept is restructured "through the discovery of a more comprehensive whole." By this he means that several artifacts somehow meld together under the auspices of a common concept. Such commonalty does not associate particularly with the number of artifacts, their traits, similarities, or even perceived likenesses. Instead the generality is a reflection of a
conceptual thought process that finds a structural affinity in a group of artifacts for the same unifying concept. The artifacts become specific instances of a more general case.

The line is an example, particularly the architectural version. As formal transformations progress the vehicle of generalization assumes a somewhat contrarian role by suggesting derivative sub-problems that dissect a larger problem. This partitioning in turn provides the designer simplifications that are easier to work with. An organizational schema expressed as linear is, in general, a line. However, such a generalization suggests specifics. Refining the line to line segments implies a partial problem, and a partial solution, in terms of a general description of the segment. The segment might represent, in general, a module, depending upon how the designer responds to thoughts and perceptions. Such modularity might then collectively redefine the line. In any case, the mechanism of generality necessarily informs the designer, and the design.

The particular line segments become the components, or instances, of the conceptual line. The line itself may serve as a conceptual organizational device while the modules in themselves are free to follow other organizing schemata. Yet the uniform generality among modules extended to line is an irrefutable concept of linear organization.

While the mind naturally pursues generalizations in the acquisition of knowledge, their effects are not always clear to the designer. Selectively and actively engaging in the formation of generalities as cognitive descriptions of visual form will add continuity and a sense of predictability to the formal transformation process.
Intentions

As Paul-Alan Johnson (1994) points out, it is very contemporary for designers, and architects in particular, to disavow any allegiance to the ego-centric position of a design solution being imbedded in a singular, all-encompassing central idea. More preferable is the integrated societal view that architecture and its consequent design solutions must be subservient to the greater callings of societal context and human demands. While such a position is both laudable and emotionally credible, it presents numerous difficulties in the quest for knowledge about design process.

For example, by what sense of human insight does a building particularly represent any social commentary at all except for the act of containment for functional use? And further, how does a particular solution's form or visual image reflect societal values or mores? These are questions of process since present-day architectural thought requires design to somehow pursue such issues in its quest for realization. The most demanding, and troubling, question is simply how are these decisions particularly arrived at in the due course of a design process that produces static objects in a dynamic environment?

Answers are frequently elusive but it is clear that a "process" is the vehicle for contriving architectural solutions to perceived architectural challenges. Further, the idea of Concept is one that must not loose out in favor of social imperative as a means of diluting design rigor and discipline.

In an attempt to maintain a deterministic and generally rational context for process, this chapter seeks to define an accessible, useable set of design concepts. The
ideas presented form a crucible for considering design a process that is simultaneously both capable of analysis and contingently reactive to continuously varying conceptual perceptions. Process is not a vague seeking of solutions but rather a more useful, and understandable human proposition if considered in the realm of a set of definable parameters. The Concept, or Terminal Idea, is far more integral to design than commonly suspected and Process tends toward that limit.
CHAPTER 3

REPRESENTATION, ANALYSIS, AND SUPPOSITION

Previous work done of interest to this thesis includes several experiments using different methods to generate conceptual form relationships, an extensive computer software. While not all projects appeal explicitly to the perceptual concepts forwarded by this thesis, all hold within their descriptive content certain implicitly useful references. Each project illustrates one or more affine principles reflective development of shape grammars, and several examples that describe the practical implementation of creative design.

The survey found the examples assuming one of three prevalent computational attitudes, either Representational, Analytical, or Suppositional. Of the three, Supposition as active, conjecture and speculative, is paramount and holds the greatest influence for this thesis.

These three particular morphological distinctions conveniently form a tripartite structure that defines a context of constraints for the Grid Sketcher. Although each of the studies cited express to some degree all three, they uniformly and explicitly tend to
emphasize one at the expense of the other two. The subordinate characteristics however remain supportive of the dominate characteristic. Similarly, the Grid Sketcher, while expressing a certain loose affinity for both Representation and Analysis, is foremost Suppositional; a form generator designed specifically to create forms speculative in nature.

Representation and Analysis

The first set of studies, those that develop representational issues, emphasize a particular technical issue, for example the topological replacements of the Bonn (1989) study or the three dimensional layer slicing of 3D-Sketch (Marshall 1992). Essentially such algorithms address the rather expressive content of computerized representation while also demonstrating a certain proclivity for formal expression.

A second group exhibits a clear impetus for and pragmatic knowledge based solutions to explicitly defined design problems. The Topdown model (Mitchell et al. 1990), and relational modeling (Gross 1990), are examples. These implementations are fundamentally analytical and tend to solve problems through design knowledge. They test the functionality of solutions against knowledgeable value judgments and criteria imbedded in the software itself. In essence computing not only generates the form, but also invokes quality decisions as well. The computer dedicates tremendous assets to interpretation of cognitive knowledge since it is essentially attempting to emulate an expert system. Even beyond the asset issue though it is worth questioning the approach's appropriateness because of the fundamental multi-faceted metaphysical complexity of conceptual design. Computing, as it presently exists is a strict enforcer of algorithmic
process. The question is whether the fundamental precepts of conceptualization, which avoid any pretense to an ordered algorithmic existence, will ever remain intact and functional under the imposition of computerization?

Computational systems that are primarily representational or analytical add to the database and are influential for their contribution of specific computational issues. But they do not embrace the Grid Sketcher's contextual intent - Supposition.

Supposition

There is a demonstrated affinity among certain researchers for systems that, within sumptuously speculative contexts, act as prolific form generators. Several of these projects also explicitly recognize the suggestive status of external limiting parameters. While the idea of physical constraints on formal shape might appear as another application of knowledge, in these particular examples the constraints are free of the subjective values associated with human intervention. They simply define objective dimensional limits between formal objects and their physical environment. The Barnes (1990), Novak (1989), and Terzidis (1989) studies are examples. These demonstrations are broadly suppositional, conveying the idea that form generating systems can respond to a set of dimensional propositions independent of analytical knowledge.

As mentioned earlier, this thesis looks to Supposition as the mediator between Representation and analysis, and the fertile media for the Grid Sketcher. While representation is a powerful tool, it does not accommodate conceptual design very well. At the other extreme, Analysis, at least in its present state, simply fails to comprehend the
infinitely complex and often irrational world of contextual influences. Yet a computerized design system cannot deny context altogether.

One way to embrace this contextual requirement is to exploit inductive perceptions. Assume that there exists certain contextual parameters that are both closely bound to form and loosely bound to knowledge. The supposition then is that such a set holds the capacity to compute form in a creative and uninhibited exploratory environment. By induction, size, scale, proportion, and proximity represent four dimensional parameters that will satisfy the supposition. Establishing such a parameter set frees the designer then to pursue design in a space that is strictly formal yet responds to contextual reference.

Computational Foundations

Suggestions and thoughts of computation in the design arena are not new. An understanding of some of these early investigations is useful, and necessary, to establish a complete appreciation for the complexity of integrating computation and design.

Stiny and Gips (1978) proposed a structure for design algorithms that has served as a general model for the more formal and comprehensive systems that followed. Essentially their paradigm begins with a "perceptor" that senses a set of initial conditions. An algorithm then responds by producing a set of specifications describing the initial conditions. Next, an algorithmic subset of aesthetics and synthesis, produces a description of the "object" that meets the initial "input" requirements. An "effector" follows by physically realizing the object described by the synthesis algorithm. What is most noteworthy though are the references to "input", "symbols", "output", and
“encoding.” These are explicit indicators of an algorithmic (computable) process capable of computer implementation. The authors devote considerable effort to developing further detail within a computing context.

Their structural description functions as a generalized paradigm that will accommodate unlimited algorithmic definitions. It is essentially a structuring mechanism for other more specific algorithmic design interpretations. The paradigm endures very well and now finds a unitary correspondence with computing. For most purposes the receptor is the computer input device, the keyboard and mouse devices are the most common examples. The initial conditions represent the particular variables that apply to an invoked computation. Aesthetic systems and synthesis algorithms find definition in the encoded algorithmic processes that instantiate form, or objects, within the computer's representational system. Presently the designer seems satisfied to consider a particular computerized visual image as an effective creation of the object, at least in the context of continuing design processes.

Such an analogy seems rather straightforward thus acknowledging the clarity of perception in the original description. However, the "receptor - effector" function is gradually declining towards triviality in the shadow of design aesthetics and design synthesis. Computing systems that comprehend implications of these two issues now hold the power of the paradigm and consequently capture most of the designer's attention.

Computational schemata representing algorithmic design expression followed and now extend in many directions. Mitchell (1990) coalesces many previous ideas and concepts within the schema of shape grammars. In a description analogous to constructs
in the languages, computation becomes as a combinatorial process where shapes or "shape tokens" represent discrete design elements in a graphic vocabulary. The next schematic level introduces a very powerful transform where design operators manipulate elements in a series of transformations that move from one formal state to the next. The emergent shape, or form, grows more complex under higher level operators - scale, rotation, translation and reflection.

Transforms may follow reformations other than these. For example, a very interesting and conceptually diverse set of transformations is suggested by the biologist D'Arcy Thompson (1961). The set of deformations, shear and displaced coordinates, produce an endless stream of stretched regular and irregular forms. Thompson's examples illustrate nature's organic influence by suggesting that these diverse distortions all share the common source of natural evolution.

The transformations so far are uniarly, where a single form experiences unilateral reshaping. Mitchell continues by extending transformations to binary operations. Two objects combine to form a (usually) more complex shape. The binary shape operations are the Boolean union, intersection, subtraction, and negation.

While transformations reshape objects, replacement events can occur to manipulate forms by topologically swapping one form for another, or, through addition, redefining shapes to yield combinations of greater complexity.

All these processes are iterative and demonstrably computable. Mitchell further gathers these concepts in a generalized "design world" by defining a formal design algebra. Expressed as a triple (V, T, C), the algebra defines V as a vocabulary of shapes...
available for instantiation in the design world, T as a repertoire of shape transformation operators, and C as a repertoire of shape combination operators. The algebra's carrier set, denoted $V^*$, consists of all those shapes producible by instantiating vocabulary elements, transforming shapes, and combining shapes.

Complex forms, perhaps compete solutions such as buildings are constructed through "production" rules that incrementally manipulate and add defined elements (shapes) to an initial state (perhaps null) until the desired construct is complete. Although the examples work towards a known final state, clearly the process can take different directions towards an almost infinite number of alternative results.

Influences

The exploratory examples of design paradigms, schemata, transformations and computational grammars discussed so far demonstrate the widely held, but perhaps intuitive belief in the implied "process" of design. Consequently, these propositions also significantly influenced the Grid Sketcher's quest for "process" in design.

Additional investigations by several authors develop specific applications and examples which further illustrate the computational process. The projects chosen for review and comment in this thesis represent a selected set of determinants considered influential to the Grid Sketcher's perceptual interests.

Summarized below for each computational system is its most important feature as a particular influence on or application to the Grid Sketcher. This feature is the system's primary application however other conclusions exist in the system's implications.
Reported first are those systems that advance Suppositional tenets, the substance of the Grid Sketcher. Other computational systems and examples more peripherally associated but still influential to the Grid Sketcher follow under the sub-headings of Representation and Analysis. This brief but considered summary sufficiently expresses the many system's features and their impact on the Grid Sketcher's derivation.

For a more in depth reading a complete investigation of each system follows in the sections after the summary. Analysis of each project follows in three parts; first, a brief description of the project, second, its contribution as a design enhancement and its particular interplay within the field of perceptual concepts, and third, where appropriate, its specific influence on the structure of the Grid Sketcher.

SUPPOSITION -

Formal Composition -
A project that presumes visual information as one of the most important influences in creative design. Computational implementation manipulates dimensional variables as prolific form generators.

EstheR -
Loosely defined metaphorical rules invoke instances of formal combinations derived from a given set of forms. New compositions evolve out of old guided by metaphorical concept.

Reint-Ops -
Decomposes 3-D objects to essentials for recombination. Algorithms search for varying set of shapes residing within the decomposition and presents them for designer evaluation. A prolific form generator.

Co-
Explores the relational model of parametric computation. The process maintains a dynamic spatial relationship between designer specified variables. Illustrates an internal geometric consistency principle while responding to formal manipulation.
Dynamic Form Generation -

Presents a transformational paradigm that experiments with the formal variations as one predetermined shape evolves towards, and possibly beyond, another. Emphasizes the influence and importance of evolutionary process.

DICE -

This project is notable for its variety. By using properties of Physics as formal determinants it handily demonstrates the speculative, pliant and capacious nature of computing.

Tartan Worlds -

Inserts the computational determinance of shape grammars into a grid metaphor. Demonstrates the architectural clarity derived from purposeful formal organization and the facility of the computational implementation.

MARCOS -

Demonstrates the recursive power of shape grammars as algorithmic processes for computation and transformation. Replacement and attachment operations recursively produce a formal. Includes the concept of randomness.

REPRESENTATION -

Sketch 3-D -

Introduces the extremely important concept of layering in transformations of compositions. This particular CAD project combines selected objects from layers into a composite drawing.

Grid Manager -

As a representational tool the organized Grid becomes both a constraint and regularized geometry in form production. The Grid provides a medium for transferring pragmatic dimensional context to the computed design environment.

C.Mod -

Recognizes the fundamental relationships between spatial parameters and their effect on form generation. Spatial parameters become contextual constraints that limit, but yet help define, formal expression.
Shape Grammar Shell (SGS) -
This project expressively illustrates the computational strength of algorithmic process. Implemented is a formal shape grammar that demonstrates a capacious and elegant ability to generate form.

A Representational Panoply -
A dozen or so computable attributes, for example, “slicing” and “transformations”, serve to clearly establish the panoramic graphic content of computing. The designer must inherently respond to this uniquely presented and derived graphical environment.

ANALYSIS -

Topdown -
Topdown demonstrates the restrictive environment of knowledge based systems. It assumes at the outset a generalized but abstract solution to a formal problem. Then by following a comprehensive rule set (in this instance a shape grammar) Topdown finds one of the bounded set of refined solutions. Essentially a predetermined solution set precludes much of the conceptual search process essential to creative design.

Formal Composition

One of the most focused descriptions of pure form generation appears supportively in two articles by Novak (1988 & 1989). The system's conceptual foundation rests on two premises. First, that the visual information content residing in a form is fundamentally one of the most influential determinants of creative design, and second that the role of the computer in creative design is to generate, actively and insistently through "computational composition", increasingly informative and expressive forms.

Designers record their explorations in visual images of varying structural interest and complexity. An image, either on trace paper or the computer screen, represents just

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
one moment in the transformation between other images of differing complexity and visual content. The proposal then is that this transformational process should yield an ever increasing information content in its forms. Such enhanced information will then improve visual interest and better serve to inform the designer's explorations.

The system's implementation centers on a parametric algorithm that assigns dimensional values to coordinate partitions of either 2-D or 3-D space. The number of partitions and the relative distance between them are parametric variables. A set of partitions aligned with each of the spatial axes serves to enclose a subspace that delineates separate and distinct objects within the larger compositional space. Transformations in the composition space then follow a dynamic change in the parametric variables. This system represents a very straightforward example dedicated to the singular purpose of generating form.

Within the algorithmic structure resides the most progressive part of the process, the algorithm that manipulates the partitioning parameters. The algorithm consistently seeks to increase the composition's visual interest by pursuing two objectives. First is to maximize the displacement variance between objects, and second to maximize the geometrical differences in the shapes of individual objects. Their combined effect is to increase dimensional diversity, and perhaps complexity, in the composition. The subsequent transformational activity is most interesting for its attention to process. Iteratively it is analogous to biological mutation. It randomly selects an object's set of partitioning variables, referred to as a "gene strip", applies a random dimensional modification, and then effects the change if it tests positive for an increase in
informational content. This "mutation" process continues, presumably, under the
designer's control until a meaningful form or composition emerges.

Here is an extremely precise example of how dimensional variables function as
the sole and unique generator of form. The algorithm's randomness mimics a natural
process while intentional partitioning represents an enforced but malleable organizational
schema. The system is particularly suppositional, void of cognitive analysis or value
reasoning. Without the power of computing this process is surely inaccessible to the
designer. It is a computation intensive model that requires the computer's unique and
unprecedented ability to expeditiously consummate transformations.

The Formally Composed Metaphor

Fargas and Papazian (1992) explore the design imperatives resident in the
metaphor. Their software project, EstheR (Esthetic Replicant) has its roots in earlier
experiments of metaphorical meaning. Certain features of EstheR illustrate quite clearly
the dynamic range of computational activities inherent in the computer. The software
accepts an arbitrary formal composition, the "document", and a set of organizing
principles. In the authors' example, the particular principles are alignment of blocks,
compactness of massing, a constant footprint/volume ratio, and visibility in 3-D
projection.

The composition progresses through a transformation schema where one or more
of the five metaphor modules sequentially modifies, or transforms, the composition. The
five modules, overlap, number, align, corners, and environ function as constraints, or a
set of design knowledge, perhaps variables, that adjust compositional form to their
metaphorical requirements. For example, the align metaphor identifies possible alignments between individual forms and then realigns the composition along those imposed axes.

EstheR also includes a solution finder that compares a composition with an arbitrary standard. The formal arrangement is a possible solution if it meets the solution criteria. However, this analytical function is not of particular concern to the software's primary purpose of metaphorical exploration.

While the influence of the metaphor modules appears manipulative rather than generative, it is important to realize that conceptually the modules actively generate new compositions. Through their metaphorical transformations, using a rather constant set of derivative forms, new compositions evolve out of the old. This is a distinctly interactive and dynamic process where each invocation of a metaphorical rule may yield a different transformation depending upon the influences and focus of the other modules. The schema is different in this respect from those that are generally more rigid and deterministic in their constraints. The metaphorical rules are loosely constraining while the compositional forms are loosely compliant, allowing the metaphorical reference to translate loosely between modules.

What results is a possibly never ending stream of formal compositions all anchored in a common set of forms. Any particular composition's image reflects a unique combination of one or more of the computed metaphors. The designer controls the full range of metaphorical activity and has a compliment of at least 120 (5 factorial) different metaphorical combinations to choose from.
Instantiating metaphorical options is inherently exploratory, a necessary part of creativity. But perhaps the most important issue, even beyond the metaphorical reference, is the clear demonstration of computable formal parameters independent of analytical judgments. The vehicle of metaphor is extremely convenient because it projects an image of conceptual process distinctly within the realm of human thought.

**Emergent Lines**

Ambiguity in the interpretation of 3-D wire frame drawings in a 2-D environment serves as the basis for [Reint-Ops](#) (Reinterpretation Operations), a proposal by Muller (1992). The program accepts a 3-D form, typically one that is topologically explicit, for example a rectangle, and decomposes the wire frame schema into separate line segments, referred to as a "line set". Visual line intersections act as break points further subdividing the form's basic composition. A set of designer controlled algorithms recompose the line segments, following various search idioms to enumerate a rich palette of differing 2-D shapes. These shapes are available for extrusion into 3-D volumes and manipulation to create formal compositions.

It is obvious that Reint-Ops formally detects emergent shapes, the forms that inherently imbed themselves in any formal composition. The software searches for a contiguous shape and if successful, presents it for the designer's evaluation. The search continues repetitively in this fashion at the designer's discretion.

A particular shape's 3-D development and massing appear restricted by the limited volume derivative options. However the program sufficiently demonstrates a clear case where computer automation fulfills the needs of design. What is missing is any particular
reference to context and while the line reinterpretations are equally expressive if done by hand on trace paper, algorithmic computation advantageously accelerates the process.

Inherently the program is a prodigious shape generator that caters to ambiguity, generality and creative exploration. The designer can move freely along a continuum by iteratively decomposing and searching a sequence of shapes each of which derives from the one prior. Productions exist distinctly detached from any contextual meaning beyond that assigned by the designer.

Relational Modeling

Following a particularly analytical direction, Gross (1990) adapts the concepts of relational databases to relational modeling. The implementation, Co, is a modeling environment composed of object relation constructs and a relational database designed to support higher level user applications. One example is Co-Draw, a prototype CAD program, another the Grid Manager, also by Gross, referred to elsewhere.

Conceptually Co exploits the precepts of parametric computation as a foundation for the relational model. Parametric reasoning establishes a concise set of input variables that in linear combination define a geometrically expanding output. This process is distinctly uni-directional from input to output. A relational view however realigns the process to reflect bi-directional influences. The output, usually considered the deterministic result of parametric input variables, recasts itself now as one of the possible inputs. Each parameter affecting the compositional relation matrix exists as a possible input variable. Depending on the problem description, any subset of variables may serve as input, leaving one or more of the remaining parameters as the derived output.
For example, a 3-D cube's dimensional characteristics reside in four parametric variables, length, width, height, and volume. A strictly parametric algorithm might take length, width, and height to determine volume. Another singular possibility is length, width, and volume to derive height. Other algorithms are possible but the ideal is a relational algorithm that allows a set of any three as the input to derive the fourth as output.

After determining the properties, described by variables, for a specific model, the properties and a complete set of relations between them create the relational order that articulates the composition's geometry. Within Co the designer is free to interactively vary the parametric variables and their relations in real time.

The particular examples given by the author seem to imply a very analytical and deterministic system. This would seem to question the applicability of relational modeling to conceptual design processes where the form generator is not quite so concrete. Relational processes obviously involve an accretion of design knowledge for the express purpose of analysis. However, recast in the context of formal composition relational modeling holds an influential position, not by generating form but rather by dynamically maintaining designer specified spatial relationships between the composition's sub-shapes. The description of Co reveals just this possibility for a representational, rather than analytical, enhancement to form generating systems. It highlights as well a relational system's unequivocal multi-level capability to integrate analysis, representation, and supposition. While parametric design enables an extensive range of form by articulating a limited set of variables, relational design implements a
wide range of internal geometric consistencies in response to external form manipulations.

Co's influence on the Grid Sketcher then is twofold. First is its demonstration of the manipulative strength in relational concepts, and second the suppositional implications inherent in relational manipulations. Following these suggestions the Grid Sketcher implements a set of variables that surreptitiously arbitrates spatial relationships between both unit forms and groups of unit forms. While these parameters do not follow the precise formal definition of a relational system, the paradigm is the Sketcher's formal determinate of adjacency relationships. In conceptual design such relational dynamics within a composition hold significant creative potential in the exploration of form.

Dynamic Form Generation

A fundamentally different approach by Terzidis (1989) emphasizes the drawing capabilities of a CAAD system. The software implementation requires two forms, the initial object and the destination object. A system defined step-wise topological transformation between the two reveals their combinatorial relationships. Transformations extend in both directions beyond the initial states and are also fully reversible.

This particular implementation emphasizes the dynamic nature of process over the static nature of a single image. Topological mapping establishes either an identity or an interpolation relationship between comparable vertices in each form. What follows then is a sequential reforming of the initial object in a dynamic and visual sequence of images.
to match the destination object. The designer designates the number of intermediate images and the speed of the transformation process.

Topological definitions play an important role in how objects react to the algorithm. For example, a volume-to-volume reformation with connectivity restraints ensures that all vertices remain attached thus effectively maintaining the form's structural integrity. Another example is face-to-face transformations without connectivity restraints. This allows individual surfaces to detach from the initial form and perhaps reattach later in the transformation. The algorithmic mapping process computes the topological and geometric shape of intermediate objects at any point along a line that represents a mathematical continuum. Surreptitiously, the line extends beyond both forms to yield conjectural images.

A possible solution to a design intent exists in the process of transformation at one or more of the intermediate steps rather than in the initial forms themselves. Issues of contextual reference depend on the choice of initial forms; possibly no reference at all, perhaps a selected set of dimensions, or in the case of precedents, a very clear contextual definition. But even in this last case the intermediate images may entice reinterpretations that vary the content of contextual reference. In any case the forms' dimensional attributes can express certain contextual information.

What this implementation does most effectively is explore the formal cross currents and influences between two forms. Initially the forms may represent whatever interests the designer, from architectural precedents to simple platonic shapes. The usefulness for design then resides in the speculative and suppositional nature of the
transformational process rather than the forms' initial definition. Process is the message of reformation.

The Physics of Form

A project by Barnes (1990) investigates the physics of solids as the functional determinants of form and order. The program, nicknamed DICE, takes as its operands two solid objects. Each object assumes a mass, a velocity, an elasticity coefficient, and a friction coefficient. Either one or both solids are set in motion and on mutual impact their dynamic response modifies in various combinations the forms' shape and positional relationships.

Specific transformations occur either as a simple change in order without a change in shape, a topological reformation of shape, or a geometrical deformation of shape. Depending upon the physical qualities assigned to each shape the three transformational modes may interact simultaneously in any combination. Object attribute values, interaction modes, and the initiation of dynamic interaction are repeatable at the designer's discretion.

Although DICE requires pre-defined forms, which presumably might represent program requirements, the system potentially generates new forms by topological deformation. There is also a certain elegance in the system's ability to create, modify and reorder forms in the same dynamic invocation. Contextual influences are minimal requiring in the original forms a close approximation of external dimensions.

For the system to effectively embrace conceptual design the designer must feel convinced that the corporeal physics of form in fact has validity as a generative influence.
It is certainly true that DICE can produce some very interesting formal arrangements, but as an ontological, or even metaphorical inquiry, why these particular attributes? One possible answer may lie in the object's vector analysis. For example, one component of a vector is direction, or in more precise architectural terms, orientation, and a deformation along a particular orientation might express the dominance of one axis over another. Further, the dynamic interaction of two vectors implies a sense of deterministic process that aligns itself with the typically architectural precept of organization.

The DICE project is particularly notable for its attention to two important issues. First, it is blatantly suppositional, a strictly exploratory environment that looks at form isolated from external influences. It explicitly favors computing as a formal rather than an analytical tool. Second, there is significant inspiration for the designer in the evocative stance proffered by the particular choice of physical attributes. As the designer explores the interactive environment the imagination wants to playfully question the purpose of such attributes. For this the system is admirably speculative, a trait closely associated with creative design.

A Computing Grid

The Tartan Worlds generative system presented by Woodbury et al. (1992) is a rather interesting implementation of computational shape grammars. A central feature is the Tartan grid, a monotonic a, b, a, b pattern that functions as the space delimiter for both the shape grammar's rules and the 2-D composition space. The grid metaphor is a visual, or pictorial, organizational schema that solves both the shape orientation and shape scaling requirements implied by the production rules. The grid is both directional
and modular, and ensures a transfer of commonality from rule definition to rule application.

Demonstrating an alternate, organizational, frame of reference for shape grammars is Tartan World's most significant contribution. Computational form generation assumes an explicit architectural content derived from an underlying formal organization while still maintaining an unaltered suppositional attitude.

Shape grammar production definitions follow the standard paradigm. The LHS (left hand side) of a production rule defines a shape within the tartan grid structure. The rule's RHS (right hand side) is a different shape also complying with the grid. The prototypical production replaces instances of the rule's LHS with an instance of the RHS. The designer graphically defines the initial shapes that originate the composition in a world design space. As 2-D graphical entities, the shapes can become quite complex while retaining their versatility.

Production rules may apply to more than one design world in a layering scheme that allows selective designation of active design world spaces. This feature's interaction with designer manipulated recursive rule applications yields a rather supple design environment.

The system is an exploratory one even under the limits imposed by the shape grammar. The grammar is neither parametric nor able to break the grid restraint to adjust its shapes. There is also a certain dichotomy within the grid metaphor. It is simultaneously both formally expressive and structurally restraining, which presents an
interesting and challenging problem. The designer must commit to the power of the grid, to its speculative nature, before fully engaging in any meaningful design.

In its conception Tartan Worlds specifically avoids the evaluation and suitability modes of implementations like, for example, Topdown. It is a generative system that requires the designer's creativeness to interpret its compositions.

Replacement as Representation

A software program written by Bonn (1989) referred to as MARCOS, takes the form of a shape grammar to define a set of replacement and attachment operations. Parts of defined forms either replace or attach to other defined forms to generate transformations. A transformation in the grammar follows a series of topological replacement operations defined by the grammar's production rules. The designer first defines a shape, the base, which is replaced by another defined shape, the generator, in a specific production rule. Compositions in the grammar follow from recursive application of one or more production rules.

3-D replacements adhere to a four dimensional matrix defined over point, edge, surface, and volume elements. The 4 X 4 matrix constitutes 16 different replacement operations. The matrix is valuable and portrays the real substance of MARCOS because of the well-defined and deterministic framework it provides for the shape grammar. However the software implements just two of the replacement operations, the volume-to-volume and the face-to-volume, as the most illustrative.

While the system manipulates shapes quite freely, the complexity in visual positioning of different topological shapes for the base and generator instills a relational
ambiguity in the productions. Additional constraints required to clarify positional questions seem deterministic and cumbersome. The program almost becomes too analytical, actually equivocating between analysis and representation.

MARCOS is recursive over the replacement and attachment operations until it creates an object that might evaluate to an acceptable solution. It is important to note that the process is virtually free of any contextual parameters and that further the system makes no attempt at evaluation.

Volume-to-volume replacements invoked as form generators using simple shapes are the most flexible and capacious tools for formal expression. The software also introduces a random variable at the designer's discretion. This seems almost trivial yet it almost immediately exerts itself as one of the systems most expressive elements. The randomness represents a natural influence that softens the rigidity of the shape grammar. As a generator of form the random variable modifies each occurrence of the generator, altering the size, location, and rotation of each additional shape. Under these persuasions the formal process possess a speculative potential and hold particular meaning for the Grid Sketcher in their demonstrations of randomization.

Graphic Layering

In another implementation, graphic ideation forms the foundation for a design program developed by Marshall (1992). The software, Sketch 3-D, presents an environment where typical CAD drawing commands operate on user specified "elevations" in both plan and section. A "cut line" defines the surface over which a drawing resides, and a composite drawing may hold many planar surfaces. All the
drawing activities in plan and section continuously display in a 3-D model resident with the plan and section views on the screen.

Pragmatically the software really presents an elegant refinement in the user interface that determines drawing surfaces. In this case a simple, direct, graphic tool selects surfaces in two specific topological orientations. Conceptually, there is an inference of layering, or visual slicing, that carries significant impact. Designers sketch in a very real layering context. The next sheet of tracing paper overlays the previous one as the surface where extractions from lower layers will eventually reside. In the computer a particular blending of lines and shapes can exist on the most recent layer in either 2-D or 3-D representations.

While the development of Sketch 3-D probably did not intend quite this emphasis, its most fruitful expansion suggests just this conceptual layering. All visual graphic systems are invariably representational independent of their other design orientations. Graphic software in particular must necessarily recognize representation and exploiting conceptual layering is clearly an advantageous use of representational facilities.

Of greater importance is the implicit suggestion that one or more descriptive formal information sets can reside within a layered composition. Further, the layering order, completely following the designer's intents and manipulations, is just as expressive as the individual forms themselves. Seen in another way, the visual slicing referred to by layering is a comprehensive tool for facilitating transformations and reinterpreting formal arrangements in design exploration. The Grid Sketcher's layering potential derives from just this conceptual motivation.
The Conjectural Grid

Organization is pervasive throughout the history of architecture and represents one of the most diverse elements in formal design. The Grid as a metaphorical system expresses arguably one of the most powerful of the formal organizing schemata. Gross (1991) rather convincingly delineates one version of the Grid as a design enhancement in an implementation referred to as the Grid Manager.

The module's functional purpose is to manipulate grids as a "layout tool" within the larger context of a CAAD program. Grid Manager allows the designer to explore solutions required to implement pragmatic design requirements. For example, building structures, wall placement, and functional space requirements. The program is not a conceptual form generator but rather a representational tool that provides both a regular geometry and a set of constraints. These deterministic attributes actively promote the process of schematic development within the context of a design's formal description.

Conceptually, the Grid Manager manipulates grids following three seminal ideas. First, grids are parametric in their dimensional delineation which sets the foundation for differing grid configurations. Second, various grids can supplement each other in cohesive and influential compositional structures. Third, the Grid sets the framework for specifying rules about structuring realizable building components within a specified set of dimensional constraints.

The designer selects grid spacings based on external design criteria for the type of objects or functional system the grid represents. Two separate grid spacings, for example, structure and circulation, may jointly occupy the same space while describing...
somewhat disjoint sets of functional components. In the Grid Manager the specific spatial relationships between grids ensure the functions fit together in the design's formal resolution. Typically, positioning rules establish object placement relative to grid lines, intersections, and internal area divisions. The designer sets the placement rules in conjunction with grid dimensions to reflect constraints on design decisions. What follows is a combinatorial exploration to reveal possible design solutions within the bounds of the constraints.

This context casts the grid as a very interesting and speculative design proposition. Several issues are notable. For example, the grid is very clearly an ordered environment capable of a rich and varied dimensional content. Dimensions, by determining both grid spacing and element positions, convey a very cogent set of contextual information. Evident as well is the grid's inherently malleability while still maintaining its supremacy of organization.

Grids also display the curious property of being at the same time both abstract and definitive. What is the designer to make of this? The grid is a cognitive expression in its deterministic geometric regularity yet unclear in its literal meaning. However, this division presents an opportunity in the implication that grids may uniquely act as independent forms. For example the designer might find a lot of creative content in a composition of interlacing grids on varying dimensional axes.

For the designer the conjectural grid represents a very fruitful and creative area of inquiry. Should the grid serve as an explicitly defined system as in the present
implementation, or perhaps just as an image or suggestion of geometry, a conceptual background for a more formal exploration?

Spatial Constraints on Form

A particularly insightful, and useful, implementation presented by Tobin (1991) recognizes the fundamental relationships between spatial parameters and their effect on form generation. Defined as knowledge of the design space, geometrical, dimensional, volumetric, and mobility variables effectively act as spatial constraints on formal designs.

The software, referred to as C.Mod, (constraint modeler) is a solid modeler that forces its forms to comply with limiting values selected by the user. For example limits on the general 3-dimensional space that contains the design's forms, minimum and maximum boundaries enclosing individual forms, dimensional descriptions of spatial and solid entities, and relations between adjacent entities. In its implementation C.Mod accepts a rather narrow definition of design knowledge. The constraint system is admittedly a knowledge base, but only insofar as its manipulations are strictly procedural and objective. Specific boundary conditions imposed on its entities are distinct from either the form's geometrical composition or its value in design. On this particular point both C.Mod and the Grid Sketcher agree.

One other important issue in C.Mod is the creative intent that motivates the spatial constraints in the construction of forms. The parallel between C.Mod and the Grid Sketcher diverges here. Virtually all of C.Mod's variables apply to relations between entities, for example, proximity relationships and mobility characteristics. Conversely, the Grid Sketcher's parametric variables yield their expressions explicitly in the
production of the forms themselves. Size, scale, and proportion are form generating
dimensions rather than form relating dimensions. While the Grid Sketcher also considers
a proximity relation, it is subservient to the dominant context of form generation.

Even though C.Mod constructs its entities through externally defined solid
modeler commands, the program is essentially a constraint implementor rather than a
form implementor. The constraints speak to the spatial concerns of the design, not
necessarily the formal. C.Mod is an invaluable demonstration of the constraint as an
additional contextual element. Such limits represent a class of information sets that are to
a degree speculative, but fundamentally emphasize representational issues over the
suppositional.

A Shape Grammar Demonstrated

Shape grammars set the foundation for the Shape Grammar Shell (SGS), an
explicit and strict form generating system developed by Santamarina (1989). The
software demonstrates an application intended to solve the standard "floor plan" problem.

Essentially, the shell codifies five explicit design "actions"; add a space, change a
space's position, replace a space with another, remove a space, and change a space's form.
Within a completely defined interactive environment, the designer first creates the shapes
of the "spaces" designated as the grammar's set of shapes. Following this, is delineation
of specific production rules to guide the shape replacement, translation, and
transformation processes. One of the author's examples described eleven production rules
defined over seven shapes to provide a complete grammar sufficient to generate
acceptable solutions.
SGS is notable for its completeness of shape grammar analysis and technical implementation. It uses both standard, non-parametric, and modified "scale sensitive" parametric grammars. The system demonstrates two important points. First, that shape grammars may serve very conceptual formal generation systems (since the shapes assume virtually any configuration or meaning) and second, that their procedural and technical implementation in a practical software package can be extremely complex and demanding.

The implication then is that perhaps a shape grammar's formal intent might exist in a less complicated and cumbersome algorithmic system while still remaining computable.

A Representational Panoply

A particularly enlightening exposition by Goldman and Zdepski (1988) on design representation illustrates the rather ubiquitous and diverse environment of computer graphics. Their discussion centers on a range of existing representational techniques rather than a specific implementation. The investigation's theme is that the means and methods of graphical representation will modify the design in ways that will reflect their graphic influences. The designer not only responds to the developing design's formal content but also to the character of the visual stimulus imposed by the mechanics of representation. Of interest here are the particular graphical techniques considered unique to computing.

Precision, or at least the inference of precision, is a tenacious, ever-present hallmark of computing. A certain confidence or sense of ruled discipline is always
evident in computational processes. The following representational types, discussed by the authors, clearly emphasize the computer's speed, precision, and computational flexibility.

"Slicing", the idea of looking at sections of a 3-D model from differing directions and locations. The "slice" represents a thickness that stands alone as an object for analysis.

"Inverts", the relationships between objects, for example mass and void, evident by reversing, varying and emphasizing color contrasts.

"Rescaling", a means of quickly varying the dimensional characteristic of a composition to elucidate varying proportional relationships.

"Serial vision", the ability to represent a composition sequentially along a path in a series of "real-time" views.

"3-D abstractions", which represent the essentials, for example form and scale, of a 2-D planar composition in 3-D.

"Surface/structure", the idea that 2-D surfaces, rather than assume their own detached character, must recognize the composition's holistic context. Only 3-D extensions can clearly illustrate a design's complete intent.

"Windowing", manipulating the external 3-D views of the environment from inside the model.

"Parts < whole", the concept of dissecting the composition into its constituent elements followed by recombinations in formal exploration. This process is particularly well suited to computer algorithmics.
"Pixelization", essentially working at the pixel level to add rendering detail and a sense of softened precision.

"Transformations", the purposeful delineation of a wire frame representation in a selected vocabulary of surfaces. The process activates the trichotomy between line, surface, and plane.

"Separations", the investigation of spatial relationships between interior and exterior by articulating the size of openings in wall surfaces.

These graphical representations are in certain contexts exploratory as well, illustrating the computer's facile capabilities in almost any design regime. Many of these techniques emphasize 3-D and are now standard in most CAAD programs. They are quite accessible and allow application programs to exploit their power and utility.

Analysis and Design Knowledge

Mitchell et al. (1990) presents an interesting system notable for both its blending of concept and expression of knowledge based design. Topdown implements a conceptual structure paralleling that of computer programming languages. Central to Topdown is the assumption that an artifact is first represented very abstract physical form. This representation undergoes further refinement by an iterative process that adds more detail at each layer until the artifact complies with the design requirements.

Topdown's programming reflects one implementation of a parametric shape grammar. Thus the shape grammar's algorithmic foundations idealize the realization of Topdown in a computerized system. The system requires the program, rather than the user, to define the grammar's initial shapes, the parametric variables, and the production

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
rules that determine final shapes in the grammar's language. The user then selects sets of predefined forms to complete a design.

While shape grammars do not inherently require knowledge for their coding, Topdown explicitly represents a knowledge based context. In particular, the example illustrated by the authors derived a variety of columns out of the vocabulary of constituent parts. The initial shape represents a vertical structure as an abstracted column, implying that the final derivation must satisfy one of a set of predetermined solutions. Consequently Topdown's solutions are particularly pragmatic, reflecting a particular knowledge of combinations. The required definitions of columns exist prior to the design problem by their explicit encoding in the shape grammar.

It is possible to extend a system like Topdown to include an ever growing set of shapes and production rules. Assumably such growth will generate almost any solution no matter how complex. Yet in all cases the solution's derivations exist in the shape grammar's knowledge based decision encoding rather than a designer's creative, and perhaps subconscious, thought processes. This suggests that the design's solution set exists prior to any interface with the computing environment. The material effect then simply tends to represent one possible solution in the set.

Topdown illustrates the fundamental contrast between knowledge based design and conceptual design where a priori knowledge is quite rarefied. The two are not completely incompatible, yet for the purposes of the ill-defined, ambiguous (and complex) conceptual design problem, finely tuned knowledge systems appear quite cumbersome.
CHAPTER 4

THE GRID SKETCHER'S ALGORITHMS

The Grid Sketcher represents an example where advanced CAAD today's technology melds with formal computing algorithms to produce a usable exploratory design tool. As an exacting algorithmic process the Sketcher aspires to certain expressive attributes. First, it is above all a prolific form generator, a potential to which its five formal growth algorithms generously speak. Second, it casts its forms in 3-D, depicting simultaneously images as solids in line, plane, and mass. Third, its formal compositions are purposefully nondeterministic and thoroughly imbued with a sense of speculative supposition. Fourth, the productions always assume an elementary stochastic character paralleling that of natural processes.

Among these design intents also resides the very significant and prudent ability to acknowledge dimensional variables as contextual parameters while still remaining a flexible form generator.

The particular disposition described by these attributes tends to replicate a design environment where productions suggest images similar to those a designer might
intuitively expect while following some particular conceptual design path. Designers work in this creative arena with information and knowledge gained through experience and research. Although intuition is not a computable talent, the computing environment can substantially facilitate the intuitive database through generation and suggestion of alternatives. As noted earlier, the Sketcher makes no pretense to possessing inherent decision making abilities or knowledge based manipulations. Its processes are strictly formal.

Generating alternative solutions within a conditioned design space is the Grid Sketcher’s primary focus. Speed of computation, manipulation of intermediate designs, and dimensions as contextual restrictions on the design space give the designer a unique ability to coerce the computing process in directions that comply with intuitive responses.

Among other intuitive issues is how the designer responds to the emotional, tactile, and artistic content of the hand driven pencil as it carves its images on paper. While it is difficult for a computer to generate a sense of feel (perhaps the mouse “feels”), a sense of emotion, understanding, and perhaps even compassion may reveal itself in the designer as computed forms begin to emerge. Certainly the designer’s need, and expectation, of minimal entropy places a great responsibility on the computing system.

Algorithmic Intent

The multitude of issues presented so far essentially define the Grid Sketcher’s descriptive intentions. They find their realization in a specific set of computable algorithms that jointly represent the Grid Sketcher’s persona. The following describe these algorithmic processes:
Shape grammar theory as a model for formal productions.
Grids as fundamental space organizing schemata.
The distributive influence of randomization.
Layering as a foundation for combinatorial processes.
Form generating (growth) algorithms.

In the following sections a more elaborate description of these processes will reveal their essential character and composition.

It is fruitful at this point to pause and consider the generic nature of an algorithmic process. Stiny and Gips (1978) provide an elegant and descriptive commentary on the essentials. First of all an algorithm is deterministic, a specific finite sequence of explicit instructions executable in some mechanical way. The instruction set accepts a finite string of sequential symbols that represent a subset of all the symbols defined for the algorithm. Output follows input as another set of sequential symbols defined for the instruction set. This algorithmic process is consistent. A particular input string will always yield the same output.

Although a single algorithm is deterministic, a collection of several may not necessarily convey the same rigidity. Two or more algorithmic processes related by some common intent, for example manipulating a number series, may generate output of similar content but different character. In the number example one output might be a form representing a geometric equation, the other a logarithmic equation. The inputs are the same, the outputs different, yet taken together in composition the two forms find various nondeterministic relations in the variety of their mutual juxtapositions. That is, the output forms prove the efficacy of their algorithmic foundations by becoming the subject matter of a corollary nondeterministic process.
Two or more algorithms may also relate in ways that maintain a continuity of determinism. In particular, if the output of one algorithm is the unaltered input of another the two appear as a single deterministic process. Algorithms also hold recursive properties in that an algorithm's output may return as its input under the guidance of a control algorithm. This last is a specific example of the more generalized notion that several algorithmic processes may function in a combinatorial environment that is itself a deterministic algorithm. There is also a particular significance imbedded in an algorithm's symbol set. Input symbol strings are variable in that they represent any one of the possibly infinite subsets of the symbol set. As an empirical proposition then such variability defines a range of input parameters that allow a very refined control of the output's character.

The Shape Grammar Model

One of the most direct interpretations of algorithmic process exists in the description of shape grammars. Specifically oriented towards design, shape grammars developed in response to the emerging context of computers as the computational effectors of algorithms (Knight 1991; Mitchell 1991, Stiny 1989). Stiny (1980) formally presented shape grammar theory in the late 1970's. In his discussions processes yielded sets of finalized objects, known as shapes, which became the language of the processes that generated them. This is analogous to a language, for example the English language, where the "language" is the set of all possible sentences formed by applying the language's grammatical rules. Just as sentences convey meaning, knowledge, and
information in their language, the terminal, or final, shapes of a shape grammar also communicate information about their context.

**Figure 5 A Shape Grammar**
Just as any language's grammatical rules operate on the words in its vocabulary, a shape grammar transforms a finite set of initial shapes as its vocabulary. The grammar's shape rules manipulate the set of initial shapes to derive a distinctly different and unique set of terminal shapes. A language's grammar also implies sentence termination as a necessary statement of completeness. Ending a sentence is somewhat arbitrary and at the discretion of the writer. Similarly, the label terminology, introduced to a shape grammar as the process terminator necessary to yield a terminal shape, applies as well at the discretion of the designer. Conceptually, a label enables the shape rule's iterative capability to continue the generative process unencumbered, while removing a label serves to terminate the process.

Shape rules specify the transformation of one shape into another. There may be many intermediate shapes in the generative process towards a terminal shape. A shape rule takes the form of an arrow with a shape, perhaps labelled, on each side. In Figure 5 the square, and the triangle within it labelled with a dot on one corner, form a shape from the initial set, called the initial shape, and is used to begin the process. The arrow implies a production function, the object, or shape, on the right being the result of the production function. The arrow replaces an instance of the shape on its left with the shape on its right by, typically, applying translating, rotating, reflecting, or scaling operations (in any combination) to the shape on the left. The grammar continues searching for a terminal shape as long as the initial shape, or any other shape on the left side of a shape rule, occurs as a subshape of any intermediate shape created on the right side of a shape rule.
The shape grammar illustrated in Figure 5 embraces just two rules, an initial shape, and a label. Both the triangle and the square are components of the set of initial shapes. The first rule creates a square of dimension equal to the triangle's base, rotates the initial shape 30 degrees counter-clockwise and then scales the shape to fit within the square. In this rudimentary grammar iteratively applying rule 1 will continue to build a triangle within a square, within a square, within a square . . . , each rotated 30 degrees, until someone makes a decision to stop the generation. Rule 2 provides the escape by removing the dot, the label, from the triangle and, since neither rules 1 nor 2, which require the dot, can apply again, the process leaves a terminal shape in the language. The terminal shape is only one of many such shapes possible in this particular grammar.

To summarize, the following requirements fully define a shape grammar:

A finite set of initial shapes  
A finite set of identifying symbols  
A finite set of shape rules (production rules as defined earlier)  
An initial shape from the set of initial shapes to seed the grammar

This framework is quite compliant and sufficiently general to embrace diverse interpretations in its application.

Essentially, a shape grammar represents an algorithmic process that takes as input a set of defined initial shapes and generates, through application of its production rules, an output composed of one or more instances of the initial shapes. Definitions for the initial shapes and production rules may take form through a graphical shape grammar interpreter or by a variable parameter set distributed over explicitly encoded production rules.
Shape generation may follow one of several schemata. For example, given two shapes, shape 2 may be a simple replacement of an exact instance of shape 1, a one step replacement of exactness without variation. Another possibility allows a multi-step process of transformation where shape 1 topologically transforms by increments into shape 2. In this case the intermediate steps, the transformation process itself, is of greater interest than the initial shapes.

A third possibility allows the production rule as a unitary entity to specify certain parametric constraints on each invocation. Shape, size, placement, scale and proportion for example, become flexible parameters controlled through a schema of user accessible variables within the production rule itself.

At a higher organizational level constraints apply to the entire set of production rules, i.e., the grammar as a holistic entity. Such constraints might require a series of production rule invocations to comply with certain user selected constraints, say for example, a generalized organizing principle.

These last two concepts hold significant potential for the Grid Sketcher. Shape attributes such as relative size, orientation, proportion, and even color and label improve the descriptive quality of the production (Ching 1979). Further, a holistic concept uniformly affecting the composition provides a strong sense of spatial continuity.

The Grid Sketcher is not an explicitly defined shape grammar but just one of many applications that derives its motivation, form generation algorithms, and intents from shape grammar concepts (Bonn 1989; Flemming 1990; Madrazo 1991; Woodbury 1991). Formal compositions are the result of "growth algorithms" that individually
embody several production rules. Parametric variables affect both the initial form's shape and the growth algorithm's manipulative context. For example, a set of \((x, y, z)\) dimensions defines the initial shape topologically for all productions as a rectangle. Another example is a spacing parameter that influences adjacencies between shape instances. Formally, the Sketcher considers its growth algorithms (explained in later sections) as computational algorithmic paradigms that mimic shape grammars. The Sketcher presents their output as subject matter for the broader nondeterministic design processes pursued by the designer.

Conditioned Space - The Grid

As pointed out earlier, one of the (few) architectural principles surviving historical banishment is that of formal organization. The Grid Sketcher abides by organizational precedent first in its fundamental expression of order and architectural format, and second as the foremost conduit for reflecting dimension as a contextual determinant. This immediately establishes a sense of control for the designer and the perception that there is an inviolable unifying principle inherent in the system's organizational structure.

Many ordering systems exist, both the traditional, and in recent decades some that are exceptionally exploitive. Ching (1979) offers an endearing summary of the more traditional spatial relationships and organizations. Radial, clustered, linear, centralized, and grid schemata are the most prevalent. The grid, because of its persistent and undeniable sense of organization while engaging an unlimited dimensional variation, holds the greatest potential as an organizational schema. As an integral component the grid is also the subject of several examples that successfully demonstrate its relation to
creative design. Both Gross (1991) and Woodbury et al. (1992) explicitly task a grid system with the responsibility of defining dimensional and spatial relationships between objects in a developing design.

For these reasons the Grid Sketcher tacitly assumes a 3-dimensional grid as an underlying organizational structure for all of its formal productions. Productions first require a 3-D definition of the bounding space, a space that sets the forms graphical limits in the x, y, and z axes and beyond which forms will not grow. These bounding volume limits may reflect contextual parameters such as building footprint, or perhaps site related constraints, or maximum building heights. Figure 6 shows four examples depicting the production space's bounding volume.

While the grid never graphically interposes itself over the production space, it is nevertheless implicit in the dimensional definition of the Sketcher's seminal rectangular form, or "growth unit". The designer assigns a particular set of x, y, and z dimensions to the rectangle that holds for a series of formal productions. The compositional forms then iteratively evolve out of individual growth units following one of five growth algorithms. Figure 7 clearly illustrates, in plan, elevation, and volume, both the grid organization and the growth unit's rectilinear character. Unit dimensions may also reflect contextual parameters, for example, scale and proportion, or perhaps even structural spacing requirements.

Growth Unit Substitution

Architectural detail is always a matter of special interest in composition and design. One of the Sketcher's most elegant capabilities recognizes this by providing the
designer the option of substitution. This event retains the standard rectilinear growth unit's dimensions but replaces the rectangle with a similarly dimensioned blocked form of some predetermined architectural character. Typically, such a block represents an

Figure 6 Four Possible Limits on Growth Space

AutoCAD drawing created at another time, independent of the Grid Sketcher.

Coordination between the design intent, or program, and developing ideas may establish a block content that enhances some particular design characteristic, element or texture.
Figure 7 The Grid Organizing Schema

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 8a illustrates an example of a vertical rectangle replaced by a column. This instance expresses a very concrete and specific real world contextual design component. (Note that even though the ten objects in both drawings reflect exactly the same production parameters, their position in the grid varies in response to the Sketcher's random distribution variable.)

Another example, Figure 8b, shows a more relaxed substitution where the production, while appearing rather distributed and loosely organized, still portrays an image of texture and structure. The substitution entices questions not only about the character and meaning of the spaces between forms, but also about the internal nature of the "space" of individual units and their common affinity.

Substitution is a relatively straightforward, yet very powerful, Sketcher function. Once the designer develops and fully explores the implications of substitution, its effects on an emerging concept and developing productions become quite pervasive. As Figure 8b and several of the graphics in later chapters illustrate, even rather minimally defined blocks can provocatively alter a drawing's speculative nature.

A Process of Natural Distribution

As an algorithmic process a randomizing variable pervades all the growth algorithms in their distribution of growth units. A random number generator is always at work reflecting an evolutionary environment where compositions exhibit a sense of natural selection. Each algorithm takes as input a set of parametric values that ensure algorithmic determinism. The ever-present randomness however tends to soften the algorithm's deterministic nature so that formal productions may vary, possibly infinitely,
Figure 8a Block Replacement Option
Figure 8b EDGES - with Block Substitution
in their composition given identical parametric inputs. This natural distribution ensures compositional fluency and variety over a range of repetitive growth algorithm applications.

Figure 9 shows an example of four compositions created by four separate invocations of the same algorithm. The parametric input values are identical, the formal compositions differ only by the algorithm's inherent variability.

Finke, Ward, and Smith (1992) demonstrate random selection as an extremely useful process to increase the creative content of object groupings. In particular the authors found, given a limited set of objects, random selection very effective in avoiding object combinations that represent the conventional. The implication for formal constructs then is that within the restriction of, for example, a simple rectilinear form, random influence tends to avoid the typical groupings that designers might first pursue in their sketches. Random variability holds the anticipation of exciting aggregations of form, some unexpected and even elegantly capricious, that portray an image of natural evolution.

Each of the Sketcher's five growth algorithms responds uniquely to the system's random variable. CORNERS, the algorithm of Figure 7, randomly selects any one of the previous growth unit's corners as the attachment point for the next unit in the series. Similarly, EDGES and FACES productions grow by accumulating forms randomly at the edges and faces of the prior unit respectively.

STACKS and SLOPES, the two remaining algorithms, exhibit a more complex response to randomization. For example, STACKS selects both the next stacking
Figure 9 The Effects of Random Distribution

position and the lateral distribution of growth elements in the stacks as a random distribution function. Further, the algorithm's randomization events share an algorithmic dependency with the set of parametric values selected by the designer. SLOPES, the most complex of the five algorithms, achieves a complete and comprehensive integration
between the randomizing distribution and the input parameters. Slope position, gradient in three axes, gradient spacing, and density reflect parallel randomization and parametric influences.

Although straightforward in concept, the randomization of process is a tremendously influential component of both creativity and formal design.

Combinatorial Layering

One fundamental precept of layering is that of a mechanism that facilitates combinatorial processes. In this context the Grid Sketcher implements layering not as an internalized algorithm but rather as an optimization of certain propitious AutoCAD functions. First, the Sketcher works in a hierarchical format where an invocation of one growth algorithm production can serve as the basis for another. Several completely disjoint productions may then aggregate to present a more comprehensive and articulate formal composition. Using the utility of AutoCAD each algorithmic invocation resides on a distinctly separate layer. In this way the Sketcher's drawings are complete AutoCAD drawing files capable of manipulation within the AutoCAD environment. AutoCAD's standard layer commands apply (as do all the regular AutoCAD drawing features).

Layering also allows the designer to overlay the Sketcher's forms on top of pre-existing AutoCAD drawings. A site plan for example may serve as a drawing base for composite overlays representing building forms.

Composite imaging is evident in the productions shown in Figures 10a and 10b. Layering schemata are 3-dimensional, biased in orientation to seemingly align with any
one of the x, y, or z axes. Under the designer's control the production space bounding
box position and dimensions set the limits for a particular layer's forms. Essentially then
by appropriately selecting the bounding box origin point in 3-space and the x, y, and z
axes limits, the designer defines modules of forms that join in adjacent (or perhaps
overlaying) compositions.

Formal Growth Algorithms

The form producing growth process begins by first selecting a 3D point inside the
bounding volume. Setting this "seedpoint" appropriately allows the designer to bias the
developing form towards a predetermined geographical location in the growth space.

Next, the designer selects one of the five growth algorithms to control the form's
cumulative generation. Each algorithm essentially implements, in the context of shape
grammars, a range of parametric production rules similar to those illustrated in Figure 11.
The first three growth algorithms develop their shapes through adjacency. The first
algorithm, CORNERS, adds succeeding growth units to one of the previous element's
corners. Growth algorithms 2 and 3, EDGES and FACES, respond similarly by adding
successive units to edges and faces respectively. The productions shown in Figures 12,
13, and 14, which supplement the following detailed discussions, reflect the three
algorithms. All three examples respond to the same set of parametric variables.
Differences between the figures reflect only the applied algorithm and the effects of the
random variable.

Algorithm 4 follows a stacking paradigm where growth units stack around a
composition of vertical axes. Algorithm 5 creates a panoply of sloping forms of varying
Figure 10a Two Combinations of Three Layers

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 10b A Combination of Four Layers

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
character determined by a supple set of variables. Figures 15a, 15b, and 16a, through 16d, also described in detail below, show a few of the possibilities inherent in these two algorithms.

A growth algorithm composes its form by iteratively applying the set of production rules, an additional growth unit each time, until completing a user specified cycle of iterations. The algorithm runs to completion leaving the designer with a composition for contemplation and further evaluation.

Corners

CORNERS, the Sketcher's first growth algorithm is straightforward and accessible. Its forms, which appear amorphous, respond only to the grid definition, the random variable, and a set of production rules. Growth unit dimensions are the only designer controlled parametric variables.

Figure 11 describes several production rules that in particular implement the CORNERS algorithm. While the right side of a production does not literally replace the shape on the left, it does reflect the left side's transformed condition after the production rule's application. A corner symbolized by a black dot is occupied and unavailable for attachment by another growth unit. The rule set of Figure 11 does not hold all the possible production rules for the CORNERS algorithm but rather reflects a typical subset. For example, left hand side shapes with two unavailable corners represent another class of rules that disallows attachment at two corners rather than one. Similarly, several other production rule classes exist relative to the number of accessible corners.
Figure 11 A Set of Production Rules
Figure 12 The CORNERS Algorithm

Using the complete set of production rules the algorithm iteratively constructs a composite form by invoking the rules, one for each growth unit, in sequence until the formal composition is complete. The choice of each growth unit's production rule is a
function of the random variable. Grid conformance follows from the continuity of the growth unit's corner to corner sequencing.

Looking at the formal productions of Figure 12 highlights the form's unique variability of suppositional interpretations. Entirely different visual messages exist in plan compared to, for example, the 3-dimensional view. Likewise the two perspective views propose other variations. Diversity increases even further with compositional layering. Several invocations of CORNERS, each following disparate bounding space limits and growth unit dimensions, extends the possibilities for exploring scale and proportion among an array of grid systems.

Edges

Just one fundamental algorithmic variation delineates EDGES from CORNERS. The random variable selects a production rule that adds the next rectangle to an available edge rather than a corner. The two algorithms are identical otherwise. Comparing illustration (a) in Figures 12 and 13 clarifies the algorithm's positional variance. While the productions shown in the other illustrations project certain similarities, there are distinct compositional differences between them. For example, EDGES enhances the mass to void ratio by essentially increasing the density of forms per unit volume.

An edge relationship also exists that reinforces and solidifies continuity between forms. The physical transition between two rectangles occurs over an edge rather than a point, implying greater accessibility between forms. EDGES also exhibits a more dynamic variation of formal arrangement as a result of number of available edges for attachment. Rectangular, prismatic forms have 12 edges but only 8 corners. In the
algorithmic growth process at most 11 edges and 7 corners are available, representing a 57 percent increase in the "jitter" factor. EDGES' forms also appear visually less static, less stable than CORNERS and particularly FACES.
Faces

Like EDGES, FACES differs again by the attachment function. Additional growth units bind to one of the previous growth unit’s faces rather than an edge or corner. FACES continues the two trends of emphasizing mass over void and reinforcing the physical continuity between forms. Virtually all voids disappear since the attachment is now a surface that effectively extends to fill adjacent intervening spaces. The face to face affinity now provides a two-dimensional doorway that allows physical movement between adjacent forms.

Taken together these phenomena compress the composition’s mass in what begins to look like an enclosed solid delineated by erratic edge definitions. The character and quality of the visual image seem rather segmented yet suggest a substantial continuity relationship. The rectangular form’s limited number of attachment surfaces, at most 5, serve to further stabilize the composition and subdue compositional jitter.

A visual exploration of comparable illustrations in Figures 12, 13, and 14 demonstrates the trends and differences among the three algorithms.

All three growth algorithms conform to a common database definition. Database integrity within the Sketcher ensures completely disjoint forms. For any singular invocation of an algorithm, no growth unit will interfere spatially with any other. Each algorithm must also accommodate the possibility of random variable disorientation. Embedded search paradigms substitute various growth units in the composition for the next attachment if the random search does not locate an available attachment on the most
Figure 14 The FACES Algorithm

recent rectangle. The randomizing process concludes to a completed composition if the search becomes too complex.
Database integrity and random search apply equally to both STACKS and SLOPES, with the exception of certain overlap states allowed to accommodate their algorithmic complexity.

Stacks

The two drawings in figures 15a and 15b are examples of STACKS. The algorithm is more complex than the three adjacency algorithms, yet portrays a less abstract and more structured image. Compositions derive from a series of parametric production rules that create an ordered series of vertically stacked growth units. The designer sets several parameters to determine the composition's stacking distribution and character. As usual, the process starts at the seedpoint located somewhere in the bounded production space. Parametric variables, as described below, determine specific input values:

- **stack height**, for which there are four options:
  - full height
  - mid height
  - low height
  - variable height

- **stack eccentricity**, represented by three options:
  - none
  - attached to vertical axis
  - maximum about the axis

- **stack spacing**, with four options:
  - overlap spacing
  - intrusion spacing
  - intermediate spacing
  - wide spacing
Figure 15a The STACKS Algorithm

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 15b The STACKS Algorithm
Stack height, sets the composite stack’s vertical dimensions. Eccentricity determines the stack’s growth unit’s lateral distribution about its vertical axis. The most structured eccentricity is none, where elements stack exactly aligned with one another. The two variable options allow the algorithm’s randomness to displace the units from the axis in the x and y directions. The spacing factor sets the distance between successive stacks which in turn controls the stack’s compositional density in the growth space. Values vary from immediate adjacency to wide spacing. Stacks begin at the same level (z value) but their x and y coordinates vary randomly.

STACK’s structured presence very deftly moves between a virtually explicit, literal definition of building structure to an almost completely fluid expression of mass and form. As the illustrations show, the stacking variables provide a robust compositional palette suitable for exploring an exceptionally diverse compendium of formal compositions. Even though grid modularity seems vague at times, algorithmic adherence to the organization follows by computing all stacking positions in increments of growth unit dimensions.

Slopes

The SLOPES algorithm is parametric as well, providing a full range of user selected variables. Figures 16a through 16b illustrate several forms derived by the algorithm. The form’s sloping character essentially represents successive strings of growth units that begin at the seedpoint and progresses towards the positive x and y axes. SLOPES is the most finely integrated of the five algorithms. The following lists the algorithm’s designer accessible parametric variables and their associated options:
Slope density, providing two possibilities:

- surface slopes
- mass slopes

Slope contour, also with two options:

- linear surface
- variable surface

Slope character, again offering two choices:

- constant slope
- variable slope

Initial slope gradient:

- slope rise
- slope run

Slope density sets the form either as a sloping mass or a sloping surface. If the designer chooses surface the contour variable determines the constancy or variability of the developing form’s contour along the edge of its formative surface. A final parameter biases the slope’s vertical character, either a form that rises at a constant slope or a form where the surface changes its slope at periodic intervals. As indicated earlier, all three parameters respond to a continuous random process that ensures a certain sense of natural evolution in the form’s development.

SLOPES generates forms that most prevalently project a character suggesting amorphous growth. Their purpose, or rather the algorithm’s purpose, is to create forms that aggressively transcend the grid structure in a deliberate, opposing juxtaposition while still retaining the grid’s modular ethic. Consequently the forms are potentially the most
Figure 16a The SLOPES Algorithm
Figure 16b The SLOPES Algorithm
Figure 16c The SLOPES Algorithm
Figure 16d The SLOPES Algorithm
suppositional of the five algorithms and offer the designer another fertile interpretive arena.

This cursory indication of the five algorithm's fundamental parameters serves only to describe their expressive attributes. A complete functional elaboration of each algorithm's implementation follows in Chapter 5.

Compositional Issues

All of the algorithms generate their forms quickly by exploiting the computing environment's inherent speed and precision. As mentioned earlier, the Grid Sketcher avoids knowledge based activities leaving contextual evaluations to the designer. The Sketcher explicitly recognizes this role by offering the designer a compendium of editing features. Most robust perhaps is the previously discussed combinatorial layering, the option to overlay one formal production with a succession of forms, each of which responds to its own user determined set of parameters. Such composite drawings facilitate the concretization of ideas and cater to the designer's intuitive need to move between the simple and complex. Figures 17 through 20 display several composite images.

As the drawings show, the Grid Sketcher's computational capabilities generate expressive form with a sense of architectural content. The wide range of system variables and algorithmic parameters, conditioned by the systems random influence, ensure almost boundless formal expression. The Grid Sketcher's intent is just this since the designer, in the beginning, also sketches in an almost boundless design space. Form generates very quickly which serves to significantly leverage the designer's pencil strokes.
Figure 17 Overlay Composite- STACKS, SLOPES, FACES
Figure 18 Overlay Composite- STACKS, CORNERS, FACES
Figure 19 Overlay Composite- STACKS, FACES
Figure 20 Overlay Composite- STACKS, FACES, SLOPES
Further, even though the designer perceives a virtually unlimited range of design options, the computing environment by necessity does not since its logical processes do not necessarily mimic human thought. So any particular implementation that portends to facilitate conceptual design process must recognize this limit and can never replicate the designer's experiential world. The computer in this context functions only as an enhancement, a tool, that must continually arbitrate the evolutionary process between structured, explicit rule based computer logic and the world of developing design rules that flow from the designer's pencil.
CHAPTER 5

IMPLEMENTATION

This chapter describes in detail the Grid Sketcher's implementation in computer software and its functional relation to AutoCAD. As indicated earlier, the Sketcher resides within the AutoCAD 3-D environment. It builds it productions as composites of forms drawn in AutoCAD's standard format and protocol. All drawings remain intact, in the normal AutoCAD configuration, and are completely accessible to AutoCAD after exiting the Sketcher.

AutoLISP, a version of standard LISP interpretively supported by AutoCAD, is the Sketcher's source programming language. LISP, a high level language widely used in artificial intelligence, incorporates most of software programming's conventional input/output, logic, and data handling features. The unusually elegant graphic capabilities in the AutoLISP version derive from its complete access to AutoCAD's drawing and database functions.

As an issue of computational process the Grid Sketcher implements three distinct sets of procedures. The first is realizing the five growth algorithms and their supporting structures. Second is automating AutoCAD's drawing features to graphically portray the
algorithm's formal effects. And third is the user interface that controls the Sketcher's drawing activities.

Unlike most of the design systems previously described in chapter 3, the Grid Sketcher does not implement a graphical user interface. Rather it follows the more prosaic, but entirely functional, menu protocol, which quite appropriately parallels that of AutoCAD. Subsequently, all the software's input variables respond to sequential menu prompts. A few of the systems investigated aspire as well to a user interface that also provides the designer with graphical definition, rather than just selection, of the algorithmic form production rules. The Grid Sketcher avoids this implementation as well only because it is possible to realize its intents in other ways.

This last issue in particular raises an important point about the relationship between design and computing. Fundamentally, software that purports to emulate, or even simply enhance, creative design, must follow one of two programming philosophies. Either it reflects within its corporeal encoding the ideas, concepts, intents and motivations unique to the design environment it wishes describe, or it must provide a coded alternative where the designer can manipulate the software to implement these characteristics. In either case, the software's cognitive presence must derive from design intentions, not programming intentions.

Such design driven encoding is possible either through definition within a graphical user interface as discussed earlier, or by design encoded software. This second option however necessarily presupposes the singularly essential position designers hold in creating the algorithms encoded in the software. In essence, the algorithms represent a
class of designs similar to what designer's produce by filtering or reinterpreting their
design knowledge through an algorithmic process. The Grid Sketcher explicitly follows
this direction by melding design knowledge, algorithmic precepts, and the technical and
functional adaptation of computing into a single holistic, efficient paradigm that
emphasizes design as its perpetrator.

Overview

After opening AutoCAD, the designer starts the Grid Sketcher by first entering
(load "gr") at the command line followed by the command gr. Just like any other
AutoCAD command, ESC/CTRL-C cancels gr, and immediately pressing <enter> or
the space bar restarts it.

To review, the Sketcher's 3-D forms derive from combinations of user controlled
variables complimented by a set of random attributes programmed in the software. The
designer sets both the 3-D (x, y, z) bounding dimensions that define the bounding box,
the volume of work space within which forms grow, and the 3-D dimensions of the
individual rectangular growth units that propagate to create forms. Figure 6 shows
examples of the bounding box. The bounding box's origin is always (0,0,0) in the UCS
coordinate system selected by the designer. All form development takes place in the
positive quadrant. Note that both the bounding box origin, dimensions, and growth unit
rectilinear definition, are completely variable within the limits of AutoCAD's WCS
(World Coordinate System). An x, y, z valued seedpoint within the growth space's limits
designates both a reference point for the grid and the initial position to begin growth unit
propagation.
One challenge presented by the Sketcher is to find usable relationships between growth unit dimensions relative to each other and the larger growth space. Typically proportion, ratio, scale, density, and proximity are motivations for selecting unit dimensions.

After setting the bounding volume and unit size, the designer chooses the seedpoint anywhere within the volume to begin the growth process, followed by one of the five growth algorithms: Corners, Edges, Faces, sTacks, or sLopes. Again, one of the Sketcher's more influential offerings is the opportunity to replace the usual rectangular growth unit by an AutoCAD defined BLOCK or .DWG file. The substitution block may be both scaled and rotated relative to the unit's growth axis. Substitution adds a layer of information suitable either for enumerating architectural detail at the micro-level or, by adjusting scale and dimension, to express more definitive large scale architectural form.

All of the algorithms' growth processes follow an iterative paradigm that sequentially adds growth units to the developing form. At the designer's discretion, growth stops either when encountering a growth space boundary, usually applicable only to the Corners, Edges, and Faces modes, or after a specified number of iterations. Termination occurs also if the Sketcher gets lost and spends too much time searching for the next growth unit's coordinates. Search time problems occur either when the space becomes too crowded or growth gets very near a boundary. A - searching - message in the command line indicates that the production may end without fully iterating all of its growth units.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Sketcher offers several finishing options at the completion of a drawing. The first allows manipulation of the drawing by zooming in, zooming out, specifying viewpoints, undoing previous overlays, erasing objects, regenerating the drawing, hiding lines, and shading, saving drawings, and erasing all objects in the drawing. The next, draw again, takes advantage of Sketcher's sticky variables and draws again using the same parameters as the previous drawing. Bypassing draw again steps to the exit option, the only graceful way to leave the Grid Sketcher. Not exiting will reset the Sketcher to the beginning where it awaits another set of variables.

Sketcher retains the most recent variable set as the default until either changed by the designer or the drawing is closed. This means that the designer may leave the Sketcher, edit the drawing in AutoCAD, and then recall the Sketcher with all the previous session's variable default values intact.

Whenever a drawing exists that holds at least one object, Sketcher will ask, at the command line, whether to overlay with the next drawing. As noted earlier this is a very supple feature. Overlaying different productions from the various growth modes will generate composite drawings rich in complexity and character. Figures 17 through 20s show several examples of overlay composites. Starting the Grid Sketcher in a drawing with previous productions provides a way to overlay forms between sketching sessions.

While the Grid Sketcher's modular forms conform to the grid spacing set by the growth unit's dimensions, by appropriately adjusting scale, modularity can fade into a sense of surface while still following the formal grid. Sketcher's formal vocabulary is almost limitless given the software's robust set of variables. The program effectively
expresses geometric emanations of form very quickly and precisely, and derives images that, presumably, the designer might miss in the typical design process. The Sketcher holds elements of surprise, the unexpected, and even caprice. By skillfully manipulating the Sketcher's variables, and pursuing a sense of curiosity and experimentation, the designer searches for images and interpretations that suggest formal solutions to design problems.

**Dimensional Variables**

The following discusses, in sequential order, the details of each of the Grid Sketcher's input variables. Each explanation lists the exact menu prompts, and where appropriate, their optional responses.

**enter a WCS 3-D point to define the bounding box origin -**

-origin point <0,0,0>:

While the bounding box's origin is always 0,0,0 in a particular UCS, this option allows placing the UCS origin anywhere in the WCS. Since the bounding box defines a restricting volume, varying the bounding box origin on successive overlays adds flexibility and discrimination in restricting growth to selected areas within the larger composition. Note that the default origin is WCS 0, 0, 0 and typical of AutoCAD prompts, becomes the accepted origin by entering a <return>. Default values for the Sketcher's other variables respond similarly.

**enter grid bounding dimensions -**
Growth algorithms propagate within these boundaries. Input must be in integer values and the software interprets them as feet in architectural units. The software presents the resulting bounding box view from above, to the right, and in front of the origin,

(viewpoint = 10, -7, 10).

The Sketcher interprets these integer values in feet as well. They establish the rectangular grid's three-dimensional structure, although the grid pattern appears unstructured in the conventional sense. The grid references the x, y, z seedpoint, explained next, not the (0, 0, 0) bounding box origin, and rectangular growth unit positions conform to this grid system. The software rejects grid spacings that exceed the bounding box dimensions.

enter a 3-D point to seed the growth process -

seedpoint:
This is an x, y, z point value, either integer or decimal, entered in the following format:

3,7,19

A seedpoint must reside somewhere within the bounding box limits. The software rejects bad format, negatives, nil values, and points that exceed defined values derived from the bounding box limits. Growth begins at the seedpoint and propagates in the three axial directions.

The Growth Modes

The following section describes the five growth modes, three of which are adjacency algorithms and two that are parametric, their options, and the BLOCK vs rectangle option.

select one of the following GROWTH MODES -
add to Corners - c
add to Edges - e
add to Faces - f
sTacks - t
sLopes - l

growth mode:

BLOCK substitution -

Block substitution, presented immediately after selecting a growth mode, replaces the Sketcher's rectangles with either a defined AutoCAD block or an AutoCAD drawing file. The block name requires no extension.
do you want to build with Sketcher's rectangles or an externally defined block? enter (b) for block or <return> to use rectangles -

Entering b brings up the following prompt asking for a block name. As usual, <return> will accept the default to rectangles.

enter a predefined block name <>:

As mentioned earlier, an external block can add important detail, and meaning, to the growth unit's definition. However, for large drawings, complex block definitions generate excessive HIDE and REGEN times, and significantly increase the drawing's database.

Choosing a block activates the following block scaling option:

select a block insert scaling option -

scale factor of 1 - 1
grid spacing - g

scaling option:

A scale factor of 1 retains the dimensional relationships of the original blocked drawing. If the grid spacing dimensions do not match the blocked dimensions, the inserted block may either underflow or overflow its allotted rectangular space. Selecting
the grid spacing option resolves the mismatch by automatically scaling the block insertion to the grid dimensions. However, this may cause block distortion in one or more axes.

The block option rejects invalid block names by issuing a prompt asking for either a valid block name or a <return> to continue.

Three Adjacency Algorithms

Corners -

Corners set the first growth unit's lower left corner at the seedpoint. Growth adds units randomly to any one of the previous unit's unused corners. Growth continues until reaching a box boundary, or a selected iteration limit. The iteration limit selection prompt looks like this:

select one of the following to end the process -

at a grid boundary - 1
after a number of iterations - 2

ending option:

Selecting option 1 switches the screen to graphics and starts the growth process. Option 2 asks for the iteration limit:

enter number of iterations:
There are no absolute bounds on the iteration limit. The only limits are those implied by the bounding space's numerical capacity and the algorithm's tenacity in finding a spot for the next growth unit. If the growth space becomes too crowded, requiring extensive search time, growth will stop. Note that all five growth algorithms use this same termination sequence.

Corners, Edges, and Faces prevent intersection of growth units, guaranteeing that growth unit volumes will not intersect one another in the search for the next attachment. Figures 21a and 21b illustrate patterns and forms in the Corners mode.

**Edges** -

As its name implies, Edges adds the next growth unit to an available edge rather than a corner. It works exactly like corners otherwise. Forms produced by Edges look similar to Corners but are denser and usually better organized. Note that, unlike Corners and Faces, Edges sorts its growth units into three color groups, each on its own layer. By discreetly turning layers on and off the algorithm also becomes a tool to investigate deconstructing and reconstructing the production in different patterns. See figures 22a and 22b for examples.

**Faces** -

Faces replaces the last growth unit with two joined face to face, producing forms even more compact and structured than Corners or Edges. Faces follows all the other algorithmic determinants seen in Corners and Edges. See figures 23a, 23b and 23c for examples.
Figure 21a A Composition in CORNERS
Figure 21b A Composition in CORNERS
Figure 22a A Composition in EDGES
Figure 22b A Composition in EDGES
Figure 23a A Composition in FACES
Figure 23b A Composition in FACES
Figure 23c A Composition in FACES

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Two Parametric Algorithms

sTacks -

sTacks builds "vertical" forms that rest at an elevation determined by the seedpoint's z value. There are three controllable stack parameters: height, eccentricity, and spacing. Stack organization grows vertically about a yellow colored axis. Axis height, and thus stack height, follows either the zaxis bounding limit or a different limit selected by the height parameter as follows:

select a stack HEIGHT option -

<table>
<thead>
<tr>
<th>Height</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full height</td>
<td>f</td>
</tr>
<tr>
<td>Mid height</td>
<td>m</td>
</tr>
<tr>
<td>Low height</td>
<td>l</td>
</tr>
<tr>
<td>Variable height</td>
<td>v</td>
</tr>
</tbody>
</table>

height:

The difference between the z axis limit and the seed point's z coordinate establishes the Full height, the maximum height of any stack. 75 % of maximum defines Mid height, 50 % sets Low height. Variable height allows the growth algorithm to randomly select stack heights that compose to a contoured texture at the form's upper surface. See Figures 24a, 24b and 24c for examples of stack heights.

Selecting an eccentricity option determines growth unit dispersion about the vertical axis:

select a stacking ECCENTRICITY -
None builds exactly stacked units. Attached to vertical axis allows the algorithm to randomly shift each growth unit along its x and y axes while still ensuring the unit remains attached to the vertical axis. Maximum about the axis extends the x and y axes' displacement to the maximum limits of the growth unit's dimensions allowing some of the growth to proceed detached from the vertical axis. The actual displacement remains a function of the randomizing process. Figures 25a and 25b illustrate stacking eccentricities.

The spacing factor sets the stacking density, the relative proximity between stacks, as follows:

- select a stack SPACING FACTOR -
  - overlap spacing - 1
  - intrusion spacing - 2
  - intermediate spacing - 3
  - wide spacing - 4

Spacing factor:

Overlap spacing spaces vertical axes at exactly the growth unit's x and y dimensions.

Selecting an eccentricity option other than none in this mode allows a significant degree of volume intersection between adjacent stacks. Intrusion spacing sets the stacks at twice
Figure 24a Height Variations in STACKS
Figure 24b Height Variations in STACKS
Figure 24c Height Variation in STACKS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 25a Eccentricity Variations in STACKS
Figure 25b Eccentricity Variations in STACKS
the growth unit's x and y dimensions, which reduces the degree of growth volume overlap between stacks. Intermediate spacing and wide spacing, set at 3 and 4 times the growth unit dimensions, precludes any volume intrusion no matter what the eccentricity option. See Figures 26a - 26d for examples. Note that in Figure 26c introducing an eccentricity factor improves the architectural image and articulation. Figure 26d varies stack heights as well so that relatively prosaic stacks of cubes begin to reveal a useable architectural content.

Deftly manipulating stacking variables produces a panoply of forms, some quite simple, others rich and interesting in their content. For example, choosing no dispersion (eccentricity), closest stack spacing, and constant height essentially builds a solid rectangle with surface divisions articulated along the grid spacing. On the other hand, as shown in Figures 27a - 27e, selecting the maximum displacement for each variable yields forms so diverse in their character that making value judgments about the meaning of their images becomes quite challenging.

sLopes -

sLopes, like stacks, is a multi-variable algorithm, however its character varies quite decidedly towards the horizontal. See Figures 16a through 16d. Since, conventionally, rise over run defines the slope, the algorithm asks for these values to establish the form's initial slope. Growth begins at the seedpoint and looks very much like stacks skewed or sloped towards the horizontal by just the value of the slope. The sloping axes, although segmented, still align along the y axis. Each growth unit's x value varies randomly about the growth axis.
Figure 26a Spacing Factor in STACKS
Figure 26b Spacing Factor in STACKS
Figure 26c Spacing Factor in STACKS
Figure 26d Spacing Factor in STACKS
full height / no eccentricity / overlap spacing

Figure 27a STACKS
variable height / max. eccentricity / wide spacing

Figure 27b STACKS
Figure 27c STACKS

variable height / max. eccentricity / wide spacing
variable height / max. eccentricity / wide spacing

Figure 27d STACKS
Figure 27e STACKS

variable height / max. eccentricity / wide spacing
Three controllable parameters help define the slope's detail and character: slope density, slope contour, and slope character. As with the other growth algorithms, sLopes’ growth unit may be a rotated and scaled substituted block.

One of the two following options determines slope density:

select a slope DENSITY option -

- Surface slopes - s
- Mass slopes - m

Surface slopes creates a form just one layer thick, which at the appropriate scale approximates to a surface. Surface slopes grow continuously only in the positive x direction away from the seed point. Note that the character of surface slopes varies considerably from that of mass slopes. Selecting surface slopes defaults to a sub-menu, the CONTOUR option shown next, not offered by mass slopes.

The mass slopes option stacks surfaces under and on top of each other in a progression where each sloping stack begins at a randomly selected x, y point. The form takes on a very compact, sloping character reminiscent of hills. Figures 28a and 28b contrast the two slope density options, holding all other variables constant. Also note the variable gradient illustrated in the two drawings.

Choosing surface slopes in response to the above slope density menu presents the following two slope contouring options:

select a slope CONTOUR option -

- Linear surface - l
Variable surface

slope contour:

Linear surface limits the x/y plane slope to a linear value forcing the form's front edge to parallel the x axis. A variable surface's sloping stacks grow in the x direction in constant increments while a geometric algorithm seeded by a random variable determines each successive stack's y value. The resulting form not only slopes in the y/z plane, but its front edge slopes across the x/y plane as well. Independent of the selected iteration limit (chosen later), both option's growth stops at the x axis bounding limit. A variable surface's growth stops as well when the next sloping stack along the surface contour exceeds the y axis bounding limit. Figures 28c illustrates the variation.

One of the following two choices determines, in general, the forms dominant sloping character:

select a slope CHARACTER option -

  Constant slope - c
  Variable slopes - v

slope character:

Constant slope limits the entire form to just the slope computed from the rise and run values entered next. Variable slopes generates random slope changes determined by not only the random variable but also the growth unit size and bounding box dimensions.
Figure 28a Slope Density in SLOPES
Figure 28b Slope Density in SLOPES
Figure 28c Slope Contour in SURFACE SLOPES
Figure 28d Slope Character in SLOPES
Figure 29 Slope Contour & Character in SLOPES
The varying slope pattern, computed at the outset, remains constant for each of the form's sloping stacks. See Figure 28d for examples of constant and variable slopes.

After setting the three slope variables, the following prompts ask for the rise and the run required to compute the initial slope value:

- enter a value for the initial slope's rise:
- enter a value for the initial slope's run:

sLopes reads the rise and run as either integers or decimals without units since they form a ratio. The initial slope value biases subsequent slope computations. For example, selecting a steep initial slope (large rise compared to run) induces the steeper slope increments that more appropriately for emphasize the vertical.

sLopes is probably the most provocative of the five growth algorithms. As figure 29 suggests, in appearance its forms have a certain structure yet remain difficult to decipher. Growth unit size and proportion have greater impact in their ability to manipulate the form's context and interpretation. Perhaps sLopes shows its most flagrant contribution in its interaction with other growth mode forms in the compositions created by overlays.

**Finishing Touches**

Once a form is complete Grid Sketcher provides several drawing manipulation tools. First, a short menu at the command line includes ten choices: zoomin (zi), zoomout (zo), viewpoints (vp), undo layer (un), erase obj (eo), regen (rg), hide (hd), shade (sh), save dwg (sv), and erase all (ea). These options are continuously available in any sequence until terminated by a <return>. Zoomin is just the AutoCAD zoom-
window function and prompts for point one and point two entered with the mouse. Zoom out returns the drawing to the Sketcher's default view. Viewpoints provides a way to look at the production from any desired 3-D point in the WCS system by entering the \(x, y, z\) values for the desired viewpoint.

The first six overlays, not including the original opened drawing, exist on their own layers, with additional overlays adding to the last (sixth) layer. Undo layer allows erasing overlays, in sequence, the most recent one first, back to the original drawing. Undo layer is quite useful since it is the only tool available to sequentially erase previous overlays without completely erasing the active drawing. It is important to note that a drawing's layers are also available as standard AutoCAD layers external to the Sketcher.

Erase obj activates the cursor pick-box for selecting and deleting individual objects in the composition. Regen (the AutoCAD regeneration function) facilitates redrawing after an undo. Hide (the AutoCAD hide function) can take a lot of time for complex drawings involving many forms. Shade (the AutoCAD shade command) can usually render complex drawings faster, and with greater visual clarity, than the hide option. Save dwg drawing prompts for an alpha character only file name, (maximum of eight characters), adds the .dwg extension, and saves the current drawing under this name without leaving the Sketcher. Erase all is a one time deletion of all the objects in the composition. The drawing space is left completely empty.

Next, the Grid Sketcher prompts, asking about drawing another form derived from the same set of parameters used for the previous production. The drawings will not be quite the same though since the Sketcher's inherent randomness is always at work. The
draw again option just provides an expeditious way of repeating the same parameters
while avoiding paging through all the option menus. Entering a <return> will step past
draw again.

Selecting draw again brings up the overlay prompt, overlay previous drawing?,
which requires a (y) or (n) answer. As mentioned earlier, the overlay is a powerful
accommodation that allows layering forms into composite drawings. There is no limit
on overlay repetitions, however drawings can become quite complex very quickly.
Forms, volumes, and shapes may intersect on successive overlays since the Sketcher's
database does not prevent growth unit conflict between growth algorithms. Whenever a
drawing holds at least one object, even the first drawing opened in AutoCAD, the
Sketcher will ask about overlays. Not selecting overlay completely erases the drawing,
including all forms from previous overlays.

Bypassing draw again reveals the exit option and the one chance to exit the
Sketcher and save the current drawing. It is worth noting again that as long as the current
AutoCAD drawing remains active all the Grid Sketcher's variables will remain intact as
well when exiting. Invoking the Grid Sketcher again (by entering gr) will display the
previous session's default parameters.

While e exits, <return> completes the cycle and returns the Grid Sketcher to the
opening dimension menus. The Sketcher remembers its parameters and offers them as
default values. This allows paging through the menus quite rapidly, changing only those
parameters of interest.

Notes and Comments
The Sketcher's drawings are complete AutoCAD .dwg files and on occasion it is extremely valuable to edit them as AutoCAD drawings outside the Sketcher. For example a promising overlay drawing may improve dramatically by moving, copying, or erasing selected elements in its forms.

The Sketcher initially seems very abstract, however with growing familiarity variables become more meaningful at the outset and productions can assume a sense of predetermination. For example in Figure 20 the forms appear rather structured implying an image of buildings while in Figure 19 the more loosely constructed image seems to convey a very urban scale. Both drawings represent a purposeful manipulation of variables to generate a desired image.

The Grid Sketcher derives its power from the AutoCAD environment, the computer's processing speed, fertile and capable growth algorithms, and an inquisitive designer. Its product is a robust and challenging set of abstract forms, not all of which elicit a positive response, but that always require significant interactive interpretation and response from the designer.
CHAPTER 6

A WORKING EXAMPLE - THE DESIGN PROJECT

Work done to support the previous chapter’s presentation represents significant experience and experimentation with the Grid Sketcher. As the examples aptly illustrate, the software’s form generating capabilities are both substantial and robust.

What remains now is the pursuit of an evaluative system to verify the grid sketcher’s efficacy within the context of the design principles presented in chapter 2. The evaluation, which follows formally in chapter 7, is grounded in a two-part experience base found first in the software development process as described throughout the previous chapters, and second by software application to a specific corporeal design project.

The Project - A Resort

This chapter presents the working example, a formal design experiment illustrating the Grid Sketcher’s use in deriving conceptual issues of form and organization and their influences on the project’s formal constructs. It is important to emphasize that the goals and pursuits of the project, an upscale and leisurely lakeside resort, narrowly address, by design, two very specific interests. Represented first and foremost is an
academic investigation of form manipulation expressly for its architectural content.

Secondly, and as important, is the pursuit of formal research to reveal possible kinetic relationships between uniquely digital machine computation and human response to discreetly non-computational conceptual design methods.

No other issues of architectural orientation are intended or sought other than as adjuncts necessary to discuss and enlighten the project's main themes.

Project Context

Functionally the resort must serve as a quiet, relaxing and completely congenial environment for those seeking an elegant and private recluse. Lodging and pastime activities must be low key, restful and pleasant. An assigned to unassigned space ratio of 60/40 establishes the resort as a facility of excellent to superb quality.

An irregularly shaped 35 acre parcel, the site's shoreline sumptuously engages the northeastern edge of Lake Las Vegas. Imbedded in picturesque foothills east of the city of Las Vegas, the lake is the focal point for a cohesive, master planned and very eloquent 320 acre resort community. A continuously varying shoreline, propitiously located access roads, and a gentle, contoured, sloping gradient that flows southward towards the lake creates a very interesting and productive parcel. Appendix I includes a diagram of the site and local context.

The project's internal environment considers a tripartite entity expressing formal architecture, physical resort amenities, and site landscape development. Project architecture must stimulate interest and expressively present itself as one of the resort's most desirable characteristics.
Initial research and assessment of resort functions generated a program for the project that consumed approximately 600,000 square feet, about one-third of the site area. The program specified six functional units:

- A developed entry space and guest greeting
- Guest rooms in two different configurations
- Food service and beverage/lounge areas
- Theaters and entertainment
- Indoor leisure activities
- Limited outdoor recreational facilities

Appendix I is more explicit than this brief description. The appendix completely describes all Resort development philosophy, environmental context, and programmatic requirements.

Initial Design Exploration

This is a very aggressive project, presenting an environment replete with design potential and opportunity for exploring a wide range of solutions. While an energetic and rich design process unfolded in the course of studio work associated with the project, this thesis embraces only the computer oriented component. As well, only a very limited subset of all the exploration through the Grid Sketcher appears here.

The following derivations and their development follows a process intended to describe a logical sequence leading to a formal solution that represents the character and form of one possible design solution.

Associated diagrams and images represent about ten percent of those generated by the Grid Sketcher in the course of investigations. Further, the process consumed twenty
hours of work, a rather lengthy stretch including an initial learning curve and numerous serendipitous diversions along the way.

Noted early in the site analysis was the circular node, a “rotary”, centrally located midpoint at the site’s northern boundary. This node quickly became a logical focal point for both project entry and locus from which to propagate project functions. Both the Site Analysis diagram in Appendix I and Figure 30 illustrate the rotary nature of the node as well as the site’s boundaries and its relation to the lake edge.

A line from the rotary extending southward to a peninsular form at the lake’s edge defined a natural site division. Contouring along this axis established it as a somewhat singular middle ground form which then implied a tripartite division of the site as shown, again, in Figure 30. Three rectangular planforms followed, setting the production space bounds for the Grid Sketcher. For reference, the western rectangle extends 500 feet by 700 feet, the central rectangle 250 feet by 1100 feet, and the eastern 450 feet by 1000 feet. While these dimensions remained constant throughout design explorations, height values varied constantly depending on the instant course of design direction.

Massings of the Grid Sketcher’s “rectangles” developed the initial set of working forms. Figure 31 shows one particular set in the series where the linear horizontal tendency in the central space extends to the vertical. Form generation followed the CORNERS algorithm.

It is also worth noting early in the investigations the extremely useful 3-D environment offered by the computer. Figure 31’s three dimensional view, an axonometric, is quickly and easily selectable by the designer. The conceptual power of
Figure 30 Project Site

site showing boundary & three growth regions (parcels)
Figure 31 Resort - Initial Derivation

form distribution using the CORNERS algorithm
3-D view manipulation derives from the elegant if intuitive observation that different views of a complex object can present a virtual disjunction of visual images. (Figures 34a and 34b, both the same composite form, are evidence of this disjoint relationship.) A designer may assume a very strong position in the contextual implications of emergent form while deftly manipulating compositions in 3-D space.

A Series Of Contrasting Forms And Concepts

While the composite form of Figure 31 is abstract, the explicit contrast developing between the central and adjacent spaces suggests that further explorations may hold value. Knowing something about the functional program also begins to influence thoughts about differentiating functions.

Circular forms introduced in Figure 32 as replacement for the "rectangles" enhance the image of distinct functions while simultaneously stimulating an interest in the disparity of architectural form. The circular forms, applied by the FACES algorithm, are dimensionally both larger and taller than the replaced rectangles. FACES also creates a more linear and regularized pattern determinant in both directions while varying the heights vertically.

Form introduction by "BLOCK" substitution enabled the circular form. Notably, as the process continues, block substitution becomes an invaluable tool for varying forms across alternative compositions.

Among the the possibilities at this juncture, continuing to enable the central element held a lot of interest as a metaphorical water/mountain reference. Two possibilities were evident: either the central form represents the tall, majestic firmly
form distribution using the FACES algorithm

Figure 32 Introducing Circular Form
formed mountains and the smaller scale rectangles the consistent, more evenly surfaced lake texture - or just the opposite. This controversy held speculative arguments for both orientations and remained unresolved until just about the end of investigations.

Possibilities of varying architectural scale led to retaining the three-dimensional relationship between the central and adjacent forms. Figure 33 shows the composition modified by FACES generating an alternative rectangular form while maintaining existing scalar relationships of scale. The architectural image is now one of a generalized, structured concept embodying variations on a theme of rectangles. Concurrently, a sense of detail and point complexity appears in the new form to emphasize its relative uniqueness.

Although it is possible to accept Figure 33's existing set of forms, modified to an optimal configuration for function, as a "formal solution", sufficient ambiguity remains to ensure that the perceptual quest for alternatives will continue.

Figure series 34a through 34f represent one of several derivations pursued responding to a purposeful exploitation of ambiguity. While the site spaces remain cohesive and relatively undisturbed, the central axis demands continued attention. The modified circular unit yields to a more complex block that joins both circles and rectangles. Contextually, the evolving forms assume an internal dialog between varying shapes that exists at a more intimate level compared to the more easily observed site oriented contrasts. Here is the first indication of layering, or intermingling of a generalized concept specifically directed towards a solution.
Figure 33 Modified Circular Form
As the design process begins to focus, accepting a narrowing perceptual scope allows the machine’s computational power and flexibility to once again provide new dimensions. Given a fixed composition, the remaining alternative views of the figure set provide opportunities to reinterpret, or validate, the formal composition.

Architectural scale, massing, rhythm, and structured contextual image assume a more concrete meaning as the perceptual “eye” moves around and about the site. While figure 34d implies a loosely associated relationship, figures 34c and 34e seem to state just the opposite. A particularly provocative image exits in figure 34b, one that blatantly demonstrates a formal character in the composition unperceived until now. For example, rhythmic relationships seem to disintegrate to an extent that questions whether the two figures are the same composition.

Figure 34f reveals in its sense of detail particular issues of scale and function. The view originates at eye level, about six feet above the surface, which realistically shows the form’s height at the environmental scale. As well, setbacks at the intermediate levels imply a useable, functional articulation of space while the apparent openings at ground level create a sense of functional building penetrations.

Taken from this series of drawings is the assumption, and validation, that elements exist within the formal composition that facilitate a functional solution. However, another clear conclusion is that if this particular set of forms completed the explorative design process, a series of manipulations external to the Grid Sketcher are necessary to final a functional design.
Figure 34a Complex Central Form
form distribution using FACES & CORNERS

Figure 34b Complex Central Form
form distribution using FACES & CORNERS

Figure 34c Complex Central Form
form distribution using FACES & CORNERS

Figure 34d Complex Central Form
form distribution using FACES & CORNERS

Figure 34e Complex Central Form
form distribution using FACES & CORNERS

Figure 34f Complex Central Form
Recognition of Function

As discussed earlier, the images in this chapter represent only a small subset of those pursued in the project. Through these extend alternative explorations a sense of functional paralleling evolved which matches the right and left parcels with private, internalized housing functions while assigning the central axis the responsibility of portraying public externalized activities. As well, there exists a very strong implication that as an architectural paradigm the two functions demand expression through a contrast of architectural form.

Continued exploration towards a refined generalized concept is evident in the Figure set 35a through 35f. The design process revealed here represents an oscillation between a continuously streaming conceptual design continuum and the realization that ambiguity must eventually yield to value judgments.

As well, the metaphorical water/mountain associations mentioned earlier found a kernel of refinement in the evolving form/function dialog. For example, one very probable interpretation holds that the central axis represents water while the peripheral forms the mountains. A centralized, public, fluid space directionally oriented towards the lake expresses an exposed and active environment similar to the exposed image typical of open bodies of water. The adjacent housing implies enclosure, in parallel with the lake's repose between its two mountainous formations. Further, the housing masses are introverted, quiet, and secluded reminiscent of mountain environments.

Finally, there exits in this interpretation a subtle metaphorical counter-point. The centralized form evidently shows a pronounced affinity for "mountainous" height while
the peripheral forms suggest a more diminutive, evenly dispersed character redolent of
the sea. Surprisingly, this dichotomous relationship weaves itself through and through
the design process presenting a recurring tension and an additional speculative element.

By now the project's complexity is at a level such that, for reasons of clarity and
simplification of process, the primary developmental emphasis will address almost
exclusively the central axial form and its functions. Very little change will occur in the
two peripheral parcels.

Figure 35a documents refinements in the central axis' form as a product of
manual adjustments. These changes represent the first experimentation involving manual
reformations of the productions external to the Grid Sketcher. For example deleting
several of the complex units serves to improve the form's balance in both scale and image
so that the composition begins to approximate a useable structure. Modifications include
removing the third level units and adjusting selected edge units to emphasize boundaries.
While the composition's density remains, there is a growing contextual implication that
the generalized form should eventually conform to a volume/space relationship.

Progress towards functionally useable spaces in this example introduces details
that force a reduction in the form's abstraction. While the general concept remains intact,
subsets of the generalized form begin to assume individualized meaning. In the Figure 35
series, ideas are apparent suggesting that within the generalized form's composition there
resides a particular formation "parts". For example, a beginning, an end or terminus, a
centralized node - or nodes, and edges. Imbedded within these abstractions exits the
potential for an emerging definition of program functions.
Form resolution however still remains the primary design objective. Implied the
generalized concept is an “originator” that should logically anchor the global site forms
and act as both an architectural focal point and a functional “node”. Figure 35a
introduces such an object at the circulation “rotary” along the site’s access road. The
vertical composition is a STACKS algorithm generated form, one that, as most of the
Grid Sketcher’s invocations, includes a BLOCK unit. Architectural variations in scale of
the individual units and stack heights clearly define a separate function that melds into
the generalized abstraction’s unifying, complementary form. As a matter of process, the
nodal object, although interesting for its contrast, still obviously entices further
exploration and refinement.

Figure 35b shows the next iteration, a unit substitution and a distributed
STACKS composition that decomposes the nodal form. Individual circular masses imply
emerging interstitial space and improve visual contrast with the central axis forms below.
As sense of function appears again in the “core” image, a characteristic inherently
provided by the STACKS algorithm. The “core” returns in the next figure as an element
to increase diversity and interest, and as a clear positor of vertical functions. Figure 35d
increase further the nodal complexity by integrating multiple stacking compositions at the
node. A provocatively complete composition in circular form between the node and the
axial composition is evident. At this point it is possible to engage programmatic function
to provide the framework and context for manually adjusting the generalized
composition.
Figure 35a Central Form with Entry Node

form distribution using FACES & STACKS
Figure 35b Central Form with Modified Entry Node

form distribution using FACES & STACKS
Form distribution using FACES & STACKS

Figure 35c Form with Multi-Node Entry Node
form distribution using FACES & STACKS

Figure 35d Form with Multi-Node Entry Node
form distribution using FACES & STACKS

Figure 35e Form with Multi-Node Entry Node
Figure 35f Form with Multi-Node Entry Node
One additional iteration in STACKS as an inquiry into the nature of the node's core provided the nodal composition shown in Figures 35e and 35f. The contrast in form and function becomes much stronger by defining two distributed stacks of large units adjacent to the narrower, taller and thinner "core" element. As well, the space between the three elements assumes greater definition.

Emergence

Continuing invocations of the Grid Sketcher produced the site composition in the series of drawings presented by Figures 36a through 36f. The six are views of the same unified composition for the purposes of exploring the emerging solution's architectural character. These images are study drawings set at a time when abstraction is beginning to yield to concretization, a refinement towards the window of emergent form - and function. The series proved very useful as inticement to return to the general concept and, after numerous iterations in the Grid Sketcher, to make certain evaluative comparisons. Indeed, the objects of the Figure 36 series represent one of perhaps three, or four, competing solutions developed over the course of investigations.

Choosing one alternative over another proved to be a continuing challenge, one that completely resided with the designer. The Grid Sketcher's forms are rich, varied and strongly suppositional, as intended during the software's development. Ultimately, the sense of emerging value, an integral design component, proved invaluable as an arbitrator of formal appropriateness and compatibility. The Grid Sketcher's blocks are extremely susceptible to definition of detail and consequently decisions about the form of blocks became an important inquiry. Imbedded in the process was a repetitive series of
comparisons between possible blocked forms, the knowledge of the software, and an internalized set of contextual ideas about an external value system.

Numerous decisions found adequate resolution only by considering a contextual value system. For example, the Figure 36 series composition presents forms of a particularly "high tech" nature. Such images are in part inherent to the Grid Sketcher's computational algorithms and enforced systemization. But the composition was ultimately successful only through a series of value decisions that found the high tech image acceptable as a contextual design element appropriate to the architectural design goals stated for the resort.

Two notable refinements apparent in Figure 36a distinguish the series. First, the complex units distributed along the central axis are more compact, and follow a uniform linear alignment. An invocation of FACES utilizing a rescaled block similar to the one previous establishes a clearer cohesiveness in the combination of complex units. Heights still vary yet the scale is more uniform while maintaining a sense of continuity among the individual "mini-nodes".

Formally the concept of contrasts within similar form finds strength in the centrally distributed mini-nodes. A regular, normal distribution exists with respect to the taller, more erratic entry node. Even though both formal groupings remain abstract, they illustrate quite powerfully the notion that each represents a distinct program function, and further that each is receptive to sequential refinement.

A second form now also exists, exactly placed at the shoreline, and in fact partially floating on the lake's surface. This form, an iteration of STACKS with again
another variety of circular form, is a pointed recognition of procession and sequence from beginning to end (itself a mild form of architectural concept). As a “terminus node” the aggregate form suggests multiple instances of the same function as a destination event. As well, the form assumes its position in both scale and position as one more articulation in the play of circular events. Once again the concept gains strength, but notably at the expense of weakened ambiguity.

By now an imminently well developed tripartite, axial central core exits, one that is acceptable as a solution within the limited context of solutions expected from the software. What remains is, as usual, the sometimes unrestrainable tendency to continue manipulating the form’s subsets and details.

A sequenced and considered perusal through the Figure 36 series of drawings reveals numerous insights into the development’s architectural character, style and sense of engagement. Nodes, circulation space, functional entities, and an architectural interplay between mass and void all are apparent from differing views. For example, Figure 36c clearly shows the spatial relationship between, and within, the nodal entry form and the centralized axially distributed mini-nodes below. As well, an interesting relationship inherently exists among the mini-nodes themselves.

At this point in the design process some very distinct ideas about functional definitions appear, and it is probably a decision branch where the Grid Sketcher’s usefulness is becoming somewhat diminished as a design tool. Yet there is more. For example, a subtle variation appears at the east parcel. CORNERS once again redistributed the rectangular units as simply a matter of recognition, and involvement.
form distribution using FACES & STACKS

Figure 36a Form Extension to Lake
form distribution using CORNERS, FACES & STACKS

Figure 36b Form Extension to Lake
form distribution using CORNERS, FACES & STACKS

Figure 36c Modified Distribution at East Parcel

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
form distribution using CORNERS, FACES & STACKS

Figure 36d Modified Distribution at East Parcel
Figure 36e Modified Distribution at East Parcel
form distribution using CORNERS, FACES & STACKS

Figure 36f Modified Distribution at East Parcel
The spatial effects of the reorder are not obvious, yet in the quest to maintain an interaction with the Grid Sketcher the implications are important.

The Conceptual Product

For the sake of completion and closure in the investigation, and to remain bound to a finite process, a limited set of refinements followed to test the interplay between the generalized concept and a functional value system. Figure set 37a through 37j is the documented evidence of that quest and represents a rather determinate and forced solution. All ten drawings represent the same composition; a cohesive, descriptive event compared to the investigation’s initial images.

While the central core’s spatial relationship is more clearly revealed, a greater sense of unity in contrasts now exits in the redefinition of modular constructs at both the east and west parcels. Figures 37f and 37j in particular illustrate the formal arrangements. Modular definition is now distinct and clearly opposed to the intermingled transition space. Further, at the first degree of concept definition, there exits a much finer, yet subtle, contrast between the parcels rectangular forms and those of the north-south axis.

The drawings presented in the Figure 37 series illustrate a distinct set of core issues in the context of this investigation’s stated objectives. First, the computing environment’s influence is blatant both in the precision of rendering, the flexibility in presenting the dynamics of perspective, and in images that reflect a rather technical, but extremely controlled, spatial content. Second, there is clearly an implication of deterministic process evident in the form’s corporeal composition. Modularity,
repetition, organization, juxtaposition, scale and proportion, all architectural phenomenon, protrude at every turn of the camera. Even though it is not readily apparent that these particles of design are the responsibility of interactive, procedural software programming, the logic of sequential events entices questions about process and the sources of the form's derivation.

Third, independent of derivational techniques, there are the more obvious questions about design precepts and how the model of events so far fits in the continuum of traditional design processes. The image left is both suggestive of a terminated process, of interest solely for it intrinsic worth as an artifact, and as a distinct counterpoint to termination. This second consideration specifically treats the existent composition as an intermediate “picture” of events, still so strongly bound by concept and abstraction that, to avoid moral infraction, further derivation is a necessity.

The Figure 37 series model is both an artifact and a process component, a fortunate condition that verifies the intent of investigations. For purposes associated with the resort project presented in this chapter, the model is an important derivation that serves to define the formal, conceptual context in which further project refinement may proceed.
Figure 37a Manually Modified Form Distribution
Figure 37b Manually Modified Form Distribution
Figure 37c Manually Modified Form Distribution
form distribution using CORNERS, FACES & STACKS

Figure 37d Manually Modified Form Distribution
form distribution using CORNERS, FACES & STACKS

Figure 37e Manually Modified Form Distribution
form distribution using CORNERS, FACES & STACKS

Figure 37f Manually Modified Form Distribution
Figure 37g Manually Modified Form Distribution
form distribution using CORNERS, FACES & STACKS

Figure 37h Manually Modified Form Distribution
Figure 37i Manually Modified Form Distribution
form distribution using CORNERS, FACES & STACKS

Figure 37j Manually Modified Form Distribution
CHAPTER 7

ANALYSIS, EVALUATION AND CONCLUSION

Applications of digital computing interleave, and subsequently modify, human existence as a matter of evolutionary progression. This thesis proposes to show that the exploratory design component of architecture is susceptible as well to computing algorithms. The Grid Sketcher is one such evolving computer application intended to mimic the processes associated with pencil and sketch paper the designer uses in early design explorations.

Analysis

Creative conceptual design, as noted earlier, is a rather elusive description of the environment where designers begin their quest for unique, effective and humanistic solutions to design problems. While it is true that in architecture a particular design problem usually arrives with its own preattached, and predetermined set of specifications, most are still void of the suggestion of solution. These circumstances leave the architect with a “program” interpretable as a loosely defined set of performance parameters to be manipulated at the discretion of the design process. Moreover, even if the program
establishes definitive solution requirements, creative processes will, at the very least, suggest stretching the rules. More probable though is a surreptitious change in the rule set as challenging solutions emerge, evolve, and mature.

Programs also carry with them a two-part package of contextual information. On the more qualitative level are facts about the surrounding physical, historical and social environment that the designer will inevitably weave into the fabric of the solution set. Part two offers a considerably more carnal, earthy array of dimensional limits that express the biases and influences of a pragmatic world. Fortunately, imbedded in the numbers is a crucial seed of dimensional context, one that algorithmic processes find richly endowed.

At the outset then the architect as designer faces three contextual design issues; first, an external information set describing the qualitative character of the environment in which solutions will develop, second, a set of quantitative program limits, and third, an arguably infinite set of solutions, many of which will populate the acceptable solution subset.

Evaluation - Procedure

An evaluation of the Grid Sketcher should begin with these three contextual design issues. At the outset it is worth reiterating that the Grid Sketcher is not a qualitative arbitrator of either evolving design processes, or solutions. Yet the architect can not deny the continuous stream of qualitative judgments explicit in an evolutionary design process. So even though the Grid Sketcher is not self-referential, one evaluative
measure of its efficacy as both a medium and a facilitator is how effectively it supports the designer's qualitative arbitrations and value judgments.

Throughout the Grid Sketcher's development the issue of dimension and algorithmic manipulation of dimensional quantities was paramount. While avoiding qualitative value, techniques for expressing quantitative ideas, and objects, are the software's core medium of contextual response. Another necessary and essential evaluative parameter then is how facile the Grid Sketcher is in responding to and presenting the rather explosive nature of dimensional variables associated with a particular program.

As a product of computing systems the Grid Sketcher exhibits, in addition to its form generating algorithms, certain traits and characteristics that naturally evolve from its digital domain. For example, it strictly enforces an organizational discipline and structure through out its algorithmic form generating process. Secondly, it is a prolific form generator enabled by a robust set of distinct, formal algorithms and their associated dimensional variables. Third, the Grid Sketcher is not only prolific, but is iteratively fast as well due in part to the inherent speed of computing hardware, and in part to the computing algorithm's internal structure. Further, within the context of architectural form, the algorithms are almost limitless in their variety of formal combinations.

At an external level, AutoCAD's extremely capable and robust domain is an equally crucial component of the Grid Sketcher's host computing environment. The Grid Sketcher's implementation finds an appealing visual realization in the numerous AutoCAD drawing features and presentation techniques.
Finally, the software finds firm grounding in a set of drawing techniques typical of, and conducive to, the design process. For example, overlays, repetition, randomization, structure, and variety, all that actively encourage the designer to think in terms of sketching and conceptual procedures.

Ultimately, if the computer is to replicate the intent of a pencil, paper, and the venue of the sketch, it must prove its worth as an effective conduit towards selection of the problem’s solution set. The software must demonstrate its facility for manipulation by the architect as an integral and necessary design tool. It must contribute unequivocally to the developing progression of concept and refinement of concept that eventually begins to express a solution. As a matter of rational expectations, the designer and the computer actually pursue jointly, and simultaneously, two distinct evolutionary tracks. One is the process of generating conceptual images, the other the selection and refinement of a particular conceptual track chosen by the architect.

Chapter 1 presented the following set of six perceptual concepts frequently accepted as characteristic of creative design processes:

- metaphor
- emergent form
- emergent value
- abstraction
- ambiguity
- generality

Each concept’s definition followed in Chapter 2. The concepts appear frequently throughout discussions about the Grid Sketcher’s theoretical foundation, development, and presentation. Subsequently, sufficient detail for each exits to justify serving the ancillary responsibility of evaluative criteria.
The six conceptual design criteria cross-referenced with the three contextual design issues highlighted previously form a useful 18 cell evaluation matrix. Such an interleaving between perceptual concepts and the set of corporeal events over which they apply presents one method of representing considered judgments about the Grid Sketcher's effectiveness in design.

Evaluation - Determinations

Figure 38 summarizes the specific 18 cell matrix used for evaluation. The general approach is to make judgments about how the Grid Sketcher supports the architect's design endeavors, activities, and responsibilities pursued in the search of a specific solution. Each of the matrix variables contains an evaluative mark of excellent (E), good (G), fair (F), or poor (P) to reflect the Grid Sketcher's value in that perceptual concept relative to its influence on each contextual design issue.

As explained in the thesis introduction, particular judgments supporting the marks derive from insights gained while developing the Grid Sketcher and those associated with creating the examples and projects contained in the body of the thesis. Each matrix cell represents an opportunity to consider certain software characteristics. A brief discussion for each will explain the particular mark assigned.

First in the series is the consideration of Metaphor for its influence on each of the three contextual issues. Metaphor/Context receives a mark of "F" By definition, a metaphorical reference occurs when an object is seen, or described in terms of another. Figure 27e intimates the compositional context of high-rise structure while Figure 19 portrays the image of neighborhood. Even though the image relationship is strong very
## PERCEPTUAL CONCEPTS

<table>
<thead>
<tr>
<th>CONTEXT DESIGN ISSUES</th>
<th>metaphor</th>
<th>emergent form</th>
<th>emergent value</th>
<th>abstraction</th>
<th>ambiguity</th>
<th>generality</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXT environmental variables</td>
<td>F</td>
<td>G</td>
<td>E</td>
<td>G</td>
<td>P</td>
<td>E</td>
</tr>
<tr>
<td>DIMENSION dimensional variables</td>
<td>E</td>
<td>E</td>
<td>F</td>
<td>E</td>
<td>G</td>
<td>E</td>
</tr>
<tr>
<td>SOLUTION solution subset</td>
<td>E</td>
<td>E</td>
<td>G</td>
<td>E</td>
<td>F</td>
<td>E</td>
</tr>
</tbody>
</table>

### RATING SYMBOLS

- **E** - EXCELLENT
- **G** - GOOD
- **F** - FAIR
- **P** - POOR

Figure 38 Evaluation Matrix
little information exists to suggest the value of either to contextual issues surrounding their interpretations. Conveniently, questions about context occur early in the discussion to illustrate that, in general, contextual environmental variables are weakly represented relative to formal dimensional variables. This implies that the remaining conceptual variables will score similarly low for Context.

Metaphor/Dimension receives an “E” clearly for the richness and variety of form, and the sequential nature of formal productions. Metaphor/Solution grades “E” as well since in all progressions involve some degree of metaphorical transformation.

Emergent form is an interesting issue, one imbedded in the formal definitions of “form”. For the present it is sufficient to accept that a normalized form derives its compositional meaning from its constituent subshapes. Within this simple definition resides the very explicit notion that for form to have meaning it must contain details that describe its purpose. As the figure series of chapter 6 reveal, formal constructs emerge with greater implied meaning as complexity grows. Consequently, Emergent Form/Context fares better with a “G” since with refinement, there tends to be a closer relationship between form and context. Emergent Form/Dimension earns an “E” because the Grid Sketcher generously provides a wide range of dimensional variation. Emergent Form/Solution also receives an “E” as a variant of the dimensional context inherent in all solution subsets, and in particular the terminal solution.

Questions of progressively emerging “value” are clearly those held closely within the thoughts of the designer. While the Grid Sketcher does not viscerally present subjective value information, it implicitly requires value decisions by the architect to
validate design solutions. An effective process of emerging value progressively assigns to formal transformations a set of contemporary societal values. A creative solution will hold within its formal image a pointed and visual allusion that recognizes one or more issues relative to present state of human existence. Emergent Value/Context grades “E” for the interrelated contextual web implied by the software.

Choices limited to relevant dimensional manipulations as a value response are less apparent. During the course of productions, numerous formal excursions found aesthetic value and meaning independent of external value issues. This loose relationship suggested A “F” for Emergent Value/Dimension. However, through development, value implications once again became more tightly woven in the concretization of definitive solutions. For this, Emergent Value/Solution marks a “G”, although the judgment is still somewhat unresolved and tenuous.

Abstractions represent concepts by clearly portraying an “image” that expresses what is most important or meaningful in a particular referential context. What follows then is an assumption that abstraction embraces a diverse and widely dispersed perceptual environment. Among the range of illustrations associated with this thesis’ investigation, it is easy to contemplate various levels of abstraction for the numerous corporeal, functional constructs.

Furthermore, abstraction finds a close relationship with all three contextual design issues, although somewhat weakly connected with Context. Virtually all images hold an inviolable, implicit requirement for abstraction as a necessary path towards completing transitional sequences. Without the continuity of an initial abstract concept, continued
refinements tend to lose meaning. The Grid Sketcher’s formal productions assume an
instant quality that exhibits both an affinity for the familiar and the immediate suggestion
that the next logical step is formation of an abstracted concept.

This sense of immediacy is pervasive and difficult to avoid, and influences most
strongly the particular issues associated with dimension. Once again, judgments about
the Grid Sketcher’s abstractive qualities are problematic and somewhat elusive. However
given that Ambiguity is one of the most influential of the perceptual concepts, the marks
assigned to Abstraction/Context, Abstraction/Dimension, and Abstraction/Solution, G", "E", "E", are singularly noteworthy.

Next in the perceptual sequence is Ambiguity, the particular ability to obscure the
obvious, to equivocate between alternatives, to seed competing ideas. An ambiguous
proposition lacks the inherent order of specific concept and is thus without abstraction.
Ideally, the creative designer promotes an ongoing sense of ambiguity parallel to the
quest for resolution in order to force changes in perceptual viewpoint, or alternately,
contextual shifts.

One of the Grid Sketcher’s very real and problematic attributes derives exactly
from difficulties understanding its contribution to ambiguous form. Because the
software’s productions contain such strong architectural content it is quite often very
difficult to disassociate any one particular set of formal images from a conceptual idea.
As well, because the Grid Sketcher provides such a robust variety of detail enhancing
manipulations, ambiguity quickly becomes obscured, or rather resolved, during the
course of casual experimentation. In other words, one of the Grid Sketcher’s most
valuable and productive attributes, its rich and robust visual presentations, is also one of its primary weaknesses.

Even when simplified, the Grid Sketcher’s forms tend to defeat rather than promote forcefully ambiguous processes. It fails to systematically introduce formal variances specifically designed to force a context shift. Consequently, in the opposing struggle between the very real need for ambiguity and the more influential, and indeed dominating, propensity for definition of concept, the Grid Sketcher fares rather weakly.

Where Ambiguity falls short, Generality tends to assert itself. Generalities take the form of concepts, exactly those that Ambiguity would obscure for the sake of expanding the range of ideas. Almost any concept is capable of generalization, and throughout the presentation of the Grid Sketcher numerous generalities expressed themselves as conceptual ideas for the sake of illustration.

Generalization is unarguably the most pervasive, if not the most influential, of the perceptual concepts. As a matter of rational process the design continuum must reside within a cohesive web of generalizations in order to adequately track ideas. Virtually all considerations of value systems, formal systems, and design systems find their initial direction as some from of generalization.

Figure series 35, 36, and 37, those associated with the resort project in Chapter 6, illustrate the initial formation of functional as well as formal concepts. Through out the resolution of the resort problem there exits a continuous pursuit for generalized concepts, one of which eventually defines the problem’s generalized “solution”. Given the generalized solution, continued problem resolution becomes one of refining the
generalization through added detail and reduction to its constituent parts. Note that most of the constituents are inherently generalizations as well.

Graphic productions offered by the Grid Sketcher inherently demand, at the outset, the formation of generalized ideas about their primal origination, formal content, and contextual meaning. For these reasons the Grid Sketcher fares very well as a tool for promoting generalization in design.

In summary, it is important to note that the solutions derived through the Grid Sketcher are neither complete formal design solutions, nor solutions to programmatical functions. Their intent is to, primarily, present a set of architectural forms generalized sufficiently to express an image similar in content to one an architect might produce to represent a conceptual solution.

The condensed evaluation of the Grid Sketcher conducted here provides a meaningful consideration of the software's attributes and finds them, generally, and with the exceptions noted, well suited for application to a creative, conceptual design process.

Conclusion

As just noted, the Grid Sketcher is generally successful and, given the experiences so far, represents an inquiry into the substance and content of design well worth the effort. A rather thorough literature search conducted in conjunction with this project revealed no digital software that integrates conceptual design and computing as intimately and directly as the Grid Sketcher.

Evidence of both process and product prevails throughout the investigations, particularly in chapter 6. The Grid Sketcher offers a maneuverable medium responsive to
the ideals of conceptual design while concurrently accommodating the inherent bounds of digital computing systems. Figure series 37a through 37j illustrates the rendering of a conceptual solution reasonably compatible with the creative intents of the conceptual sketch.

Typical of similar projects, there are numerous avenues of investigation and intention worthy of further interest. Three topics solicit the greatest appeal for their natural follow-on to the investigation so far. Numerous other tangential issues present themselves as well, however such diversions, even though interesting and speculative, exceed the bounds of this thesis.

Of the three issues here, the first presents enhancements to the Grid Sketcher. Coded in AutoLISP, the software consumes about 4000 lines of code. (A short sample is shown in Appendix IV.) Much of this represents the user interface which works well to quickly access the Grid Sketcher's varied components. Like all software, the Grid Sketcher succumbed to "creeping elegance" a programming black hole where the finer art of software design tends to overcome functional purposes. So far though, such diversions contribute successfully to the Software's utility and organizational framework. Further programming should then address directly the Grid Sketcher's declared purpose.

Two other areas are important. First is the grid system, the software's singular conceptual organization. While the architectural concept of "rectangular grid" does not represent a specific programming object, or algorithm, it forms a widely dispersed but cohesive network within which the software's algorithms function. One of the most powerful enhancements then to the Grid Sketcher is an additional network, or networks,
defining the necessary conceptual environment for alternative organizations. For example, Ching (1979) elucidates several, including radial and nodal. The organizations Ching describes are pervasive and well known throughout the design community, and particularly in Architecture.

Adding just one additional conceptual organization to the Grid Sketcher would enhance its productions and interest exponentially. Existing growth algorithms are now not only available to another fundamental architectural organization, but also entice speculative interest in how the two organizations might interact with each other. Organizational systems offer rich opportunities for the Grid Sketcher.

Secondly most important are the specific algorithms imbedded in the software. The CORNERS, EDGES, and FACES production algorithms are basic and hold within their conceptual roots ideas fundamental to the incremental, sequential nature of computer processes. As the various illustrations show, these algorithms are quite robust and functional. STACKS and SLOPES both serve to increase variability and improve the supple implications inherent in the software. STACKS in particular turns out to be quite elegant while SLOPES remains somewhat unresolved and perhaps peculiar. SLOPES then entices further investigation, perhaps in adjusting the range and content of its dimensional variables to align more closely with an architectural image.

However the greatest potential resides in additional production algorithms, particularly those that represent formal architectural concepts. For example, a multiple mode rhythmic variation algorithm based, say, in a mathematical series, or perhaps one that is proportional in response to 3-dimensional layering. Actually a subset of both these
precepts exists already within the Grid Sketcher’s range of dimensional variables. Yet algorithms that explicitly formalize these concepts hold an expressive and rewarding implications in the context of computing.

There are two other interests external to the software worth consideration. First is the verification process, presented here very briefly and condensed, which solicits a wider audience for review. The eighteen cell evaluation matrix works well, however a broader range of experience and informed opinion about the Grid Sketcher’s interests is important. An expanded review would serve as an idea generator that would tend to adjust the Grid Sketcher’s alignment with the intent’s of conceptual design.

Finally, there resides in the Grid Sketcher the adjacent, but distinct, quality of demonstrative and procedural purpose. For this the grid Sketcher may hold potential, not as a design implementor, but simply as a vehicle to pursue questions of process. Computing is unavoidable and consequently requires continued significant inquiry into the relationships between design and the digital domain.
APPENDIX I

PROJECT DESCRIPTION

This appendix exhibits two documents written early in thesis development. They represent work done in preparation for the design project and offer background information that will assist and enhance understanding for the project presented in chapter 6.

As is typical of most formal design projects in architecture, a defined program serves as the generator for program analysis. The Program Document presents a rather complete description of the Sailor's Club Resort project at Lake Las Vegas. Significant information about the project's environmental context, uses and functions, and amenities provide a clear look at the project's purpose.

The second document, a Outline specification, enhances information provided in the Program Document by listing details about the project's construction components. Such data serves to stimulate thought about systems, materials, and functional components of building systems and construction techniques.

Both documents represent invaluable research and are integral to the thesis' purpose of design exploration and development.
Program Document

for
The Sailor's Club Resort
at
Lake Las Vegas, Nevada

Prepared by
Brian M. Gardner
for
AAE - 773L
Architectural Design and Final Project
Fall Semester, 1994

Dr. Hugh Burgess, Instructor

College of Architecture, Construction Management and Planning
University of Nevada, Las Vegas
Las Vegas, Nevada
PROGRAM DOCUMENT

Introduction

The product of this design exercise is a resort complex for patrons seeking leisure and exclusion in a quiet and restful surroundings. This program provides a comprehensive description of various physical requirements and contextual influences that mutually derive the design solution.

Site Geography

Lake Las Vegas is a recreational facility privately designed for residential and resort development. The lake, approximately 30 minutes east of downtown Las Vegas lies in the green belt between Las Vegas and Lake Mead. At approximately 320 acres, its orientation is generally East to West following a valley defined by low hills on each side. South Shore is subdivided for residential homes, townhouses, and a variety of leisure amenities artfully sited among the landscaped features of several golf courses. North Shore presents a distinctly different character in its expressive orientation towards resort activities. Six parcels, at approximately 35 acres each, nestled among golf course fairways, represent separate and unique resort sites for further development. (See the Lake Las Vegas project diagram) A major arterial connector, residential home sites, and golf courses bound the resort sites on their northern edge.

An earthen levee restrains the lake on its eastern edge and establishes an elevation advantage overlooking Lake Mead to the East. The particular site for this project is the most eastern of the six resort sites and rests very eloquently at the levee’s northern edge. The site slopes nominally 100 vertically feet towards the lake shore providing panoramic views across the lake to the western mountains, the hills of South Shore, and East towards Lake Mead.

One of the site's most interesting features is its undulating, almost amorphous shoreline which articulates four rather private cove-like formations along the lake edge. As well, the most western shore lies along a bay protected by adjacent land formations. In consort with the shoreline, the levee, the pronounced slope from the arterial towards the shore, and the confined western bay serve to establish the requisite opportunity for seclusion within the parcel.

Natural desert tundra, flora and rock formations surround Lake Las Vegas and extend profusely over the site. Desert winds, typically from the Southwest between April and November, become a cooling element as they blow from the lake inland across the shore. Cooler winter winds from the Northwest dissipate somewhat as they cross the...
lake's northern hills. Both Winter and Summer sun paths cross the site unimpeded by the hills to the South.

These geographical conditions, in particular the lake and a southern protected orientation, serve as one of the more amenable and eloquent desert settings in the Las Vegas Valley.

Climate

Las Vegas, named after an ancient vernacular for "The Meadows", grew out of the alluvial remnants of prehistoric Lake Bonneville. Virtually surrounded by a panoply of picturesque mountains the Las Vegas valley extends over an area in excess of 600 square miles, most of which exists as natural desert. An intricate wash system pervades the valley flowing from the higher western mountain ranges eastward across the valley to eventually converge at Lake Mead. The Valley is classic desert, an environment filled with attributes, both congenial and emotive, that uniquely define a wonderful opportunity for leisure pastimes.

Average daytime temperatures range between 45 degrees in the Winter to 90 degrees in the Summer. Spring and Fall weather conditions are optimum, considered by many local residents as ideal, while most winters see a few days in the 30 degree range. Later Summer temperatures hover typically around 100 degrees cooling to the mid 80s at night. This Summer heat pattern gives Las Vegas a reputation as one of the nation's Summer hot spots particularly when the temperature hits 115 as it occasionally does.

The summer heat responds favorably to the ameliorating effects of both the predominately southwestern summer breezes and the desert's low humidity, typically 10 to 15 percent. Although extremely windy at times the summer evening breezes are quite nice. Winter winds, usually associated with frontal weather systems, blow from the Northwest and can be quite cold. These harsh winds are the single most wintry weather phenomenon in the Valley.

Humidity remains uniformly in the teens because of the minimal rainfall, typically about 4 inches per year, and the prevailing dry desert air mass. Humidity levels rise temporarily however after thunderstorm activity. The associated rainfall fills the air with the best of the desert's native scents and aromas. These particular days are exceptionally enjoyable yet perversely carry with them the threat of flash floods. Because of the deserts hard packed clay-like under soil, the downpour from torrential thunderstorms runs off over the desert's surface. Violent and aggressive thunderstorms in the western hills can quickly exceed the natural wash formation's drainage capability and produce uncontrollable flooding. Typically the Valley experiences the brunt of two to three flash flooding events each year.
Lake Las Vegas, even though somewhat sheltered by its northern and southern mountains, remains integral to the Las Vegas Valley geography. Climatic effects are essentially the same for both, which means that the Sailor's Club Resort site will enjoy an amiable and pleasant environment.

Site Access

An excellent road system connects Lake Las Vegas with the city, suburbs, and extensions of Las Vegas. U.S. 95 and Boulder Highway, the primary southeastern connectors to the Valley meet with Lake Mead Boulevard in Henderson. Lake Mead Blvd. extends northeast into the Sunrise Mountain foothills towards the Lake Mead Recreational Area. Lake Las Vegas Drive exits off Lake Mead Blvd. as the primary arterial, winding its way through scenic and colorful hills to the main entrance at the Lake's west end. This 14 mile drive from the Las Vegas “Strip” provides guests with a splendid view of Las Vegas and its desert setting while journeying to the resort.

An interesting matrix of residential streets links South Shore with the entry road. Presently a single major arterial street crosses the river to North Shore, ascends the foothills to the North, and then turns East towards the Lake's levee. This residential parkway serves as the project's primary access and a most pleasant and scenic visual entry to the site.

Problem Statement

Function - The Resort must serve as a quiescent, relaxing and completely congenial environment for those seeking an elegant and private recluse. Lodging and pastime activities must be low key, restful, and pleasant.

Form - The Resort's environment is conceived as a tripartite entity expressing formal architecture, physical resort amenities, and site landscape development. This environment must eminently support the project's Function. The building's architecture must stimulate interest and expressively present itself as one of the resorts most desirable characteristics. Site attributes and the landscape environment must follow the same philosophy.

Economy - The Resort targets a specific clientele and funding, for both the initial project and continued operation, will reflect the needs and desires as stated in the Function. Specifically, the assigned to unassigned space ratio of 60/40 establishes the resort as a facility of excellent to superb quality.

Time - The Resort must maintain a sense of permanence and predictability over five to seven year time increments.
Program

Functional space definitions for the Sailor's Club Leisure Resort distribute over six categories:

1. Entry space and administration facilities
2. Guest rooms in three different configurations
3. Food service and beverage/lounge areas
4. Theaters and Entertainment
5. Indoor leisure activities
6. Outdoor recreational activities

The following tabulations lists specific unit spaces and their square footage requirements. Unassigned Space reflects a 60/40 ratio of assigned to unassigned spaces.

1. Entry Space and Administration Facilities

<table>
<thead>
<tr>
<th>1.1 Foyer and Reception</th>
<th>4000 sq.ft.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 Registration - Reception Desk and Cashier</td>
<td>500</td>
</tr>
</tbody>
</table>
| 1.3 Offices
  - Managers (2) @ 250 sq.ft | 500 |
  - Desk Clerks | 200 |
  - Receptionist | 150 |
| 1.4 Sales and Reservations | 1000 |
| 1.5 Accounting
  - Cashier | 200 |
  - Auditor | 150 |
  - Accounts | 300 |
| 1.6 Personnel Office/Human Resources | 400 |
| 1.7 Small Conference Room - Staff | 500 |
| 1.8 Bell Captain and Luggage Storage | 300 |
| 1.9 Activities Desk | 500 |
| 1.10 Security | 300 |
| 1.11 Engineering | 200 |
| 1.12 Telephone Exchange (PBX) | 200 |
| 1.13 Computers | 250 |
| 1.14 Mailroom | 200 |
| 1.15 Storage - General | 500 |
| 1.16 Staff Restrooms and Lounge Area | 250 |
| 1.17 Public Restrooms | 400 |
| 1.18 Total Assigned Space | 11000 sq.ft. |
1.19 Unassigned Space 7300 sq ft.

Total Category 1 Space Required 18,300 sq.ft.

2. Guest Rooms

2.1 Standard Size
   400 15' x 30' @ 450 sq.ft. 180000 sq.ft.

2.2 Business Class
   100 20' x 30' @ 600 sq.ft. 60000

2.3 Suites
   100 25' x 40' @ 1000 sq.ft. 100000

2.4 Housekeeping 3000

2.5 Furniture Storage 1000

2.6 Workshop and Maintenance 2000

2.7 Staff Administration 500

2.8 Staff Dining and Lounge 300

2.9 Laundry Facility 500

2.10 Receiving Area and Supply Storage 400

2.11 Linen Storage w/Service Carts 300

2.12 Locker Rooms and Uniform Storage 300

2.13 Total Assigned Space 349000 sq.ft.

2.14 Unassigned Space 232600 sq.ft.

Total Category 2 Space Required 581,600 sq.ft.

3. Food Service and Beverage/Lounge Areas

3.1 Four Restaurants - 150 Person capacity
   @ 2500 sq.ft. each 10000 sq.ft.

3.2 Kitchens and Food Preparation 5000

3.3 Food Storage Areas (Fresh and Staples) 5000

3.4 Two Cafes/Coffee Bars @ 1500 sq.ft. each 3000

3.5 Food Preparation for Cafes 1500

3.6 Two Cocktail Lounges @ 1800 sq.ft. each 3600

3.7 Two Sports Bars @ 1500 sq.ft. each 3000

3.8 Service, Receiving and Supply Storage 1500

3.9 Laundry Facility 400

3.10 Staff Lounge and Restrooms 300

3.11 Staff Uniform Issue 300

3.12 Employees Dining Room 1000

3.13 Total Assigned Space 33600 sq.ft.
3.14 Unassigned Space 22400 sq.ft.

Total Category 3 Space Required 56,000 sq.ft.

4. Theaters and Entertainment

4.1 Live Performance Theater 9000 sq.ft
   Stage 5000
   Support 5000
4.2 Small Movie Theater 3000
4.3 Two Small Lounge Stages @ 1000 sq.ft. each 2000
4.4 Dance Hall/Ballroom 4000
4.5 Eight Small Conference Rooms @ 900 sq.ft. each 7200
4.6 Total Assigned space 35200 sq.ft.
4.7 Unassigned Space 23400 sq.ft.

Total Category 4 Space Required 58,600 sq.ft

5. Indoor Leisure Activities

5.1 Reading Room 1000 sq.ft.
5.2 Adjoining Library 2000
5.3 Billiards Parlor 3000
5.4 Four Card Rooms @ 1000 sq.ft. each 4000
5.5 High Tech Video Arcade 1500
5.6 Two Physical Exercise/Workout Rooms @ 1800 sq.ft. each 3600
5.7 General Store, Gift Shop, and Newsstand 3500
5.8 Men's Salon 2000
5.9 Women's Salon/Boutique 2500
5.10 Two Saunas @ 250 sq.ft. each 500
5.11 Total Assigned Space 23100 sq.ft.
5.12 Unassigned Space 15300 sq.ft.

Total Category 5 Space Required 38,300 sq.ft.

6. Outdoor Recreational Activities

6.1 Three Swimming Pools @ 3000 sq.ft. each 9000 sq.ft.
6.2 Eight Outdoor Spas @ 150 sq.ft. each 1200
6.3 Jogging Paths
6.4 Walking Gardens
<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Space Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>Four Tennis Courts @ 1500 sq.ft. each</td>
<td>6000</td>
</tr>
<tr>
<td>6.6</td>
<td>Boat Docks for Small Sailboats</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Boat Docks for Jet Skis</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>Beach Picnic Areas</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>Barbecue Areas</td>
<td></td>
</tr>
<tr>
<td>6.10</td>
<td>Outdoor Presentation Theater</td>
<td>8000</td>
</tr>
<tr>
<td>6.11</td>
<td>Bicycle Riding</td>
<td></td>
</tr>
<tr>
<td>6.12</td>
<td>Putting Greens</td>
<td></td>
</tr>
<tr>
<td>6.13</td>
<td>Sand Volleyball Courts</td>
<td>4000</td>
</tr>
</tbody>
</table>

**Total Category 6 Space Required - As Site Space Permits**

**Total Enclosed Space Estimate for the Project -**

<table>
<thead>
<tr>
<th>Space Type</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned Space</td>
<td>451900 sq.ft.</td>
</tr>
<tr>
<td>Unassigned Space</td>
<td>301000 sq.ft.</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>752,900 sq.ft.</strong></td>
</tr>
</tbody>
</table>
OUTLINE SPECIFICATION
for
THE SAILOR'S CLUB
Resort at Lake Las Vegas

Prepared by Brian Gardner
for AAE-761
Advanced Construction Documents/Specifications
Fall-1994
Instructor- Reed Settle
College of Architecture, Construction Management, and Planning
University of Nevada, Las Vegas
DIVISION 1 - GENERAL REQUIREMENTS

01010 - Summary of work
01021 - Cash allowances
01025 - Measurement and payment
01030 - Alternatives/alternatives
01035 - Modification procedures
01041 - Project coordination
01042 - Mechanical and electrical coordination
01045 - Cutting and patching
01050 - Field engineering
01060 - Regulatory requirements
  A. - Building code requirements -
01080 - Identification systems
01091 - Reference standards
01092 - Abbreviations
01093 - Symbols
01094 - Definitions
01210 - Preconstruction conferences
01220 - Progress meetings
01245 - Installation meetings
01310 - Progress schedules
01320 - Progress reports
01330 - Survey and layout data
01340 - Shop drawings, product data, and samples
01360 - Quality control submittals
01380 - Construction photographs
01410 - Testing laboratory services
01420 - Inspection services
01425 - Field samples
01430 - Mock-ups
01440 - Contractor's quality control
01445 - Manufacturer's field services
01505 - Mobilization
01510 - Temporary utilities
01520 - Temporary construction
01525 - Construction aids
01530 - Barriers and enclosures
01540 - Security
01550 - Access roads and parking areas
01560 - Temporary controls
01570 - Traffic regulation
01580 - Project identification and signs
01590 - Field offices and sheds
01610 - Delivery, storage, and handling
01620 - Installation standards
01630 - Product options and substitutions
01655 - Starting of systems
01660 - Testing, adjusting, and balancing of systems
01670 - Systems demonstrations
01710 - Final cleaning
01720 - Project record documents
01730 - Operation and maintenance data
01740 - Warrants and bonds
01750 - Spare parts and maintenance materials
01760 - Warranty inspections
01800 - Maintenance

DIVISION 2 - SITEWORK

02012 - Standard penetration tests
   A. - See soils report
02110 - Site clearing
02210 - Grading
02220 - Excavating, backfilling, and compacting
02230 - Base courses
02270 - Slope protection and erosion control
02280 - Soil treatment
02510 - Asphaltic concrete paving
   A. - Parking and traffic areas
02515 - Unit pavers
   A. - Tile and stone pavers in designated local pathways and recreation areas
02520 - Portland cement concrete paving
   A. - Colored and struck concrete paving in designated facility linking pathways and major outdoor recreation areas
02580 - Pavement marking
02605 - Utility structures
02610 - Pipe and fittings
02640 - Valves and cocks
02645 - Hydrants
02665 - Water systems
02675 - Disinfecting of water distribution systems
02710 - Subdrainage systems
02720 - Storm sewerage
02730 - Sanitary Sewerage
02785 - Electric power transmission
02790 - Communication transmission

Outline Specification for The Sailor's Club resort at Lake Las Vegas
Prepared by Brian Gardner for AAE-761 - Advanced Construction Documents/Specifications - Fall, 1994

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
A. - Telephone, business and entertainment systems

02810 - Irrigation systems
   A. - Installed irrigation systems for landscape elements - see Landscape Plan

02820 - Fountains
   A. - Fountains with pools and water features in designated recreation areas - see Site Plan

02830 - Fences and gates
   A. - Perimeter access and control - fences and entrances

02840 - Walk, road, and parking appurtenances

02860 - Playfield equipment and structures
   A. - Prepared areas for sand volley-ball courts, beaches, and putting greens - see Site Plan

02870 - Site and street furnishings

02890 - Footbridges
   A. - Integrated with designated water features

02920 - Soil preparation
   A. - See Landscape Plan

02930 - Lawns and grasses

02950 - Trees, plants, and ground covers

02970 - Landscape maintenance

02980 - Landscape accessories

DIVISION 3 - CONCRETE

03110 - Structural cast-in-place concrete formwork
   A. - foundations, below grade space enclosures, retaining walls

03210 - Reinforcing steel

03220 - Welded wire fabric

03230 - Stressing tendons

03250 - Concrete accessories

03310 - Structural concrete
   A. - Below grade walls, foundation footings and slabs

03345 - Concrete finishing

03350 - Concrete finishes
   A. - Textured finishes for pathways and pool decks in outdoor recreation areas

03550 - Concrete toppings
   A. - Concrete floor toppings in multi-level lodging structures

03600 - Grout

DIVISION 4 - MASONRY

04100 - Mortar and masonry grout

04150 - Masonry accessories

Outline Specification for The Sailor's Club resort at Lake Las Vegas
Prepared by Brian Gardner for AAE-761 - Advanced Construction Documents/Specifications - Fall, 1994

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
04230 - Reinforced unit masonry
   A. - Exposed aggregate CMU masonry construction for designated low rise structures and lodging units

04460 - Limestone
   A. - Composite limestone veneer over selected CMU construction

04470 - Sandstone
   A. - Lodging units - sandstone veneer over metal frame

04475 - Slate
   A. - Slate veneer over exterior selected CMU construction

DIVISION 5 - METALS

05010 - Metal materials
05050 - Metal fastening
05120 - Structural steel
   A. - Structural steel construction for hotel tower
05210 - Steel joists
   A. - Custom steel joist roofing support systems
05310 - Steel deck
   A. - Hotel tower flooring system
05450 - Metal support systems
05510 - Metal stairs
   A. - Lodging unit exterior stairs
05520 - Handrails and railings
   A. - All stair assemblies
05530 - Gratings
05550 - Stair treads and nosings
05584 - Heating/cooling unit enclosures
05810 - Expansion joint cover assemblies

DIVISION 6 - WOOD AND PLASTICS

06050 - Fasteners and adhesives
06110 - Wood framing
   A. - Lodging unit interior partitions, party walls, and roof framing
06220 - Millwork
   A. - Interior and exterior doors, windows, and interior trim
06310 - Preservative treatment
06320 - Fire retardent treatment
06410 - Custom casework
   A. - Kitchen and bath cabinets

DIVISION 7 - THERMAL AND MOISTURE PROTECTION

07130 - Bentonite waterproofing
A. - Below grade exterior concrete walls surrounding commercial service core spaces

07190 - Vapor retarders
   A. - Concrete foundations and lodging unit slabs

07210 - Building insulation
   A. - Foamed-in-place insulation in all building walls
      1. - Exterior walls adjacent to habitable spaces
      2. - Roof over habitable spaces
   B. - Batt insulation
      1. - Sono-batt insulation in bathroom walls adjacent to living areas

07255 - Cementitious fireproofing
   A. - Fireproofing for steel structural elements

07270 - Firestopping
   A. - At all penetrations through floors and walls

07410 - Manufactured roof and wall panels
   A. - Metal roofing panels for all buildings - commercial buildings, hotel, theater, and restaurants
   B. - Lodging units - concrete topping over metal panels

07572 - Pedestrian traffic coatings
07576 - Vehicular traffic coatings
07620 - Sheet metal finishing and trim
07630 - Sheet metal roofing specialties
07650 - Flexible flashing
07910 - Joint fillers and gaskets
07920 - Sealants and calkings

DIVISION 8 - DOORS AND WINDOWS

08120 - Aluminum doors and frames
   A. - Exterior doors - storefront doors on commercial buildings

08210 - Wood doors
   A. - Pre-finished doors for lodging units - exterior and interior

08305 - Access doors
   A. - Mechanical rooms and service cores

08320 - Security doors
   A. - Administration, communications, and security areas

08325 - Cold storage doors
   A. - Food preparation and storage spaces

08520 - Aluminum windows
   designated activity and housing units

08710 - Door hardware
   A. - Schlage, commercial grade throughout, ADA approved lever handles

08770 - Door and window accessories

08810 - Glass
   A. - Dual glazed throughout
B. - Low E on South, East, and West orientations
08850 - Glazing accessories

DIVISION 9 - FINISHES

09215 - Veneer plaster
09220 - Portland cement plaster
   A. - All lodging unit exterior walls - courtyard side
09250 - Gypsum board
   A. - All interior partition walls
09310 - Ceramic tile
   A. - Lodging units
      1. - Entry
      2. - Kitchen
      3. - Baths
   B. - Commercial
      1. - Restaurants and lounges - selected areas
      2. - Store and sales spaces - selected areas
      3. - Selected circulation spaces
09340 - Paving tile
   A. - Lodging units
      1. - Atriums
      2. - Entries
      3. - Courtyards
   B. - Commercial
      1. - Selected exterior circulation and connecting routes
      2. - Selected exterior activity areas
09450 - Stone facing
   A. - Lodging unit atrium structures
   B. - Commercial building trim and finish
09510 - Acoustical ceilings
   A. - Theater and lounge spaces
09520 - Acoustical wall treatment
   A. - Theater and lounge spaces
09530 - Acoustical insulation and barriers
   A. - Lodging unit party walls
09560 - Wood strip flooring
   A. - Dance floor surfaces in lounge areas
09590 - Resilient wood flooring systems
09682 - Carpet cushion
09685 - Sheet carpet
   A. - Lodging units
   B. - Administration office spaces
   C. - Restaurant dining areas
   D. - Selected lounge areas
E. - Selected store sales areas

09910 - Exterior painting
   A. - Trim at designated locations

09920 - Interior painting

09970 - Wallcovering
   A. - Lodging units - baths, sleeping and living areas
   B. - Commercial
      1. - Resort entry and reception spaces
      2. - Restaurant and lounge public areas

DIVISION 10 - SPECIALTIES

10115 - Markerboards
   A. - Meeting rooms - business commercial and resort administration

10120 - Tackboards
   A. - General use in administrative spaces

10160 - Metal toilet compartments

10210 - Metal wall louvers
   A. - Food service areas

10250 - Service wall units

10260 - Wall and corner guards
   A. - Lodging unit interior wall corners
   B. - Restaurant, lounge, and retail space exposed surfaces

10305 - Manufactured fireplaces
   A. - Selected restaurant and lounge areas

10410 - Directories
   A. - Resort reception areas
   B. - Hotel complex
   C. - Entries to public activity and recreation areas

10430 - Exterior signs
   A. - Resort identification at primary access roads
   B. - Decorative facility identification signs

10505 - Metal lockers
   A. - Employee storage

10522 - Fire extinguishers, cabinets and accessories
   A. - Recessed with glass panel and key lock

10532 - Walkway covers
   A. - Shade structures in designated activity areas

10538 - Canopies

10675 - Metal storage shelving
   A. - General storage in service facilities

10750 - Telephone specialties
   A. - Interior and exterior telephone installations - see Site Plan

10810 - Toilet accessories
   A. - Public toilet facilities

Outline Specification for The Sailor’s Club resort at Lake Las Vegas
Prepared by Brian Gardner for AAE-761 - Advanced Construction Documents/Specifications - Fall, 1994

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
10820 - Residential bath accessories
   A. - Kohler ceramic sink, shower/tub and bath fixtures throughout
   B. - Upgraded fixture hardware
10900 - Wardrobe and closet specialties

DIVISION 11 - EQUIPMENT

11014 - Window washing systems
11016 - Floor and wall cleaning equipment
11018 - Housekeeping carts
11026 - Safes
   A. - Administration security and accounting
11028 - Safe deposit boxes
   A. - Guest safe deposit facilities
11052 - Book theft protection equipment
   A. - General library facility
11062 - Stage curtains
   A. - Theater and lounge facilities
11064 - Rigging systems and controls
   A. - Theater and lounges
11080 - Registration equipment
   A. - Resort registration facilities
11090 - Checkroom equipment
   A. - Theater, lounge, and selected activity areas
11102 - Barber and beauty shop equipment
   A. - Personal care facilities
11104 - Cash registers and checking equipment
   A. - Registration, retail areas, and lounges
11112 - Washers and extractors
11116 - Drying and conditioning equipment
11132 - Projection screens
   A. - Meeting and commercial conference rooms
11134 - Projectors
   A. - Meeting and commercial conference rooms
11156 - Key and card control units
   A. - Guest parking facilities
11162 - Dock lifts
11165 - Dock bumpers
   A. - Resort facility service and supply entrances
11405 - Food storage equipment
   A. - Kitchen and food preparation facilities
11410 - Food preparation equipment
   A. - Kitchen and food preparation facilities
11415 - Food delivery carts and conveyors
   A. - Food service facilities for restaurants and lounges
11420 - Food cooking equipment
   A. - Kitchens
   B. - Lounge snack facilities
11425 - Hood and ventilation equipment
   A. - Kitchen and food preparation facilities
11430 - Food dispensing equipment
11435 - Ice machines
   A. - Lodging units and lounges
11440 - Cleaning and disposal equipment
11445 - Bar and soda fountain equipment
   A. - Selected indoor and outdoor activity areas and lounges
11492 - Exercise equipment
   A. - Health club exercise facilities
11720 - Examination and treatment equipment
   A. - Resort clinic
11730 - Patient care equipment
   A. - Resort Clinic

DIVISION 12 - FURNISHINGS

12050 - Fabrics
12120 - Wall decorations
   A. - Resort registration areas, lodging facilities
12140 - Sculpture
   A. - Outdoor lounge and activity areas
12540 - Curtains
   A. - Draperies and light proof curtains in lodging units
12620 - Furniture
   A. - General furniture required for lodging units and offices - see interior design drawings
12650 - Furniture accessories
12690 - Floor mats and frames
12740 - Booths and tables
   A. - Restaurants and lounges
12810 - Interior plants
   A. - Restaurants, lounge areas
12820 - Interior plants
12825 - Interior landscape accessories
12830 - Interior plant maintenance

DIVISION 13 - SPECIAL CONSTRUCTION

13052 - Saunas
   A. - Designated activity areas
13152 - Swimming pools
A. - Lodging unit courtyards
13170 - Tubs and pools
  A. - Main public activity complex
13815 - Environmental control systems
  A. - Resort reception, administration facilities
  B. - Hotel structure
13820 - Communication systems
  A. - Inter-resort telephone and business communications
    1. - Administration telephone system
    2. - Guest telephone system
    3. - FAX and computing facilities
13825 - Security systems
  A. - General resort access and perimeter control
13835 - Elevator monitoring and control systems
  A. - Hotel structure
13845 - Alarm and detection systems
  A. - Administration spaces
13850 - Door control systems
  A. - Administration spaces
  B. - Hotel and lodging unit rooms and access
  C. - Commercial facilities
13900 - Fire suppression and supervisory systems
  A. - Hotel, lodging and commercial public facilities

DIVISION 14 - CONVEYING SYSTEMS

14120 - Electric dumbwaiters
  A. - Food service between kitchen and restaurants
14210 - Electric traction elevators
  A. - Hotel structure

DIVISION 15 - MECHANICAL

15060 - Pipes and pipe fittings
15100 - Valves
15120 - Piping specialties
15130 - Gages
15140 - Supports and anchors
15150 - Meters
15160 - Pumps
15170 - Motors
15175 - Tanks
15190 - Mechanical identification
15240 - Mechanical sound, vibration, and seismic control
15260 - Piping insulation

Outline Specification for The Sailor's Club resort at Lake Las Vegas
Prepared by Brian Gardner for AAE-761 - Advanced Construction Documents/Specifications - Fall, 1994

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
15280 - Equipment insulation
15290 - Ductwork insulation
15310 - Fire protection piping
15320 - Fire pumps
15330 - Wet pipe sprinkler systems
15375 - Standpipe and hose systems
15410 - Plumbing piping
15430 - Plumbing specialties
15440 - Plumbing fixtures
15450 - Plumbing equipment
15475 - Pool and fountain equipment
15510 - Hydronic piping
15515 - Hydronic specialties
15530 - Refrigerant piping
15535 - Refrigerant specialties
15540 - HVAC pumps
15555 - Boilers
15570 - Boiler accessories
15575 - Breechings, chimneys, and stacks
15580 - Feedwater equipment
15590 - Fuel handling equipment
15610 - Furnaces
15655 - Refrigeration compressors
15670 - Condensing units
15680 - Water chillers
15710 - Cooling towers
15730 - Liquid coolers
15740 - Condensers
15855 - Air handling units with coils
15860 - Centrifugal fans
15885 - Air cleaning devices
15890 - Ductwork
15910 - Ductwork accessories
15920 - Sound attenuators
15930 - Air terminal units
15940 - Air outlets and inlets
15970 - Control systems
15980 - Instrumentation
15985 - Sequence of operation
15991 - Mechanical equipment testing, adjusting, and balancing
15992 - Piping system testing, adjusting and balancing
15993 - Air systems testing, adjusting, and balancing
DIVISION 16 - ELECTRICAL

16110 - Raceways
16120 - Wires and cables
16130 - Boxes
16140 - Wiring devices
16150 - Manufactured wiring systems
16160 - Cabinets and enclosures
16190 - Supporting devices
16195 - Electrical identification
16410 - Power factor correction
16415 - Voltage regulators
16420 - Service entrance
16425 - Switchboards
16430 - Metering
16435 - Converters
16440 - Disconnect switches
16445 - Peak load controllers
16450 - Secondary grounding
16460 - Transformers
16465 - Bus duct
16470 - Panel boards
16475 - Overcurrent protective devices
16480 - Motor control
16485 - Contactors
16490 - Switches
16501 - Lamps
16502 - Luminaire accessories
16510 - Interior luminaires
16520 - Exterior luminaires
16535 - Emergency lighting
16545 - Underwater lighting
16580 - Theatrical lighting
16720 - Alarm and detection equipment
16740 - Voice and data systems
16770 - Public address and music systems
16780 - Television systems
16910 - Electrical systems control
16915 - Lighting control systems
16920 - Environmental systems control
16930 - Building systems control
16940 - Instrumentation
16960 - Electrical system testing
16970 - Electrical system startup/commissioning
16980 - Demonstration of electrical equipment
APPENDIX II

PROJECT PRESENTATION

Represented in this appendix are several sheets showing examples of the final drawings done for the Sailor's Club design project. By referring to these illustrations the reader may further compare details of the final project solution with those investigated in chapter 6.
THE SAILOR'S CLUB
A SECLUDED RESORT AT NORTH SHORE
LAKE LAS VEGAS
LAS VEGAS, NEVADA

KEY PLAN
1. RESORT ENTRANCE PARKWAY
2. ENTRY ROTARY
3. PORT COCHERE AND VALET
4. GUEST REGISTRATION
5. RESORT ADMINISTRATION
6. SPA, GYMNASIUM, AND EXERCISE FACILITY
7. SUB-LEVEL SERVICE CORRIDOR ACCESS
8. HIGH RISE HOTEL
9. GENERAL PARKING
10. ACTIVITY AND SHOPPING PROMENADE
11. SHORE COURTYARD
12. CIRCULATION CORRIDOR
13. THEATER
14. RESTAURANT
15. SHOPS
16. RESTAURANT ON THE LAKE
17. DINING, OBSERVATION AND LOUNGE
18. LOUNGE SIDES AT THE WAVE
19. ENTRANCE COURT TO HOUSES QUADS
20. HOUSES QUAD COMMON COURTYARD
21. 54-LEVEL GUEST QUARTERS & ATIUM SPACE
22. TYPICAL AT EACH HOUSES QUAD
23. BEACH RECREATION AREA

VITALITY MAP

LAKESIDE
LAKESIDE RIDGE
HENDERSON

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPENDIX III

GRID SKETCHER’S USER MANUAL

Included here is a summary user’s manual for the Grid Sketcher. The document represents a concise synopsis of the software’s user interface, its algorithms, and of its interface with AutoCAD’s 3-D drawing environment. As a practical matter, this guide provides sufficient information for a user familiar with AutoCAD to understand both the intent and outcome of the Grid Sketcher, and to use the software as a design tool.
WELCOME -

This program, referred to as the "Sketcher", works within the AutoCAD 3-D environment. It builds forms, or productions, within a grid-like topology, as composites of rectilinear objects drawn in AutoCAD's standard format and protocol. Once activating AutoCAD the Grid Sketcher is first loaded by entering (load "gr") at the command line and then called by entering gr. Just like any other AutoCAD command, CTRL-C cancels gr, and immediately pressing <enter> or the space bar restarts it.

OVERVIEW -

The Sketcher's 3-D forms derive from combinations of user controlled variables complimented by a set of random attributes programmed in the software. The designer sets both the 3-D (x, y, z) bounding dimensions that define the bounding box, the volume of work space within which forms grow, and the 3-D dimensions of the individual rectangular growth units that propagate to create forms. Figure 6 illustrates examples of the bounding box. The bounding box's origin is always (0,0,0) in the UCS coordinate system selected by the designer. All form development takes place in the positive quadrant relative to the bounding box origin. Note however that both the bounding box origin and dimensions, and growth unit rectilinear definition, are completely variable within the limits of the WCS.

One challenge presented by the Sketcher is to find usable relationships between growth unit dimensions relative to both each other and the larger growth space.
dimensions. Typically proportion, ratio, scale, density, and position are motivations for selecting unit size.

After setting the bounding volume and unit size, the designer chooses a point, the seed point, anywhere within the volume, to begin the growth process. The Sketcher presently implements five growth algorithms. The first, Corners, progresses sequentially by adding another growth unit to a randomly selected corner of the previous growth unit. The second, Edges, adds the next growth unit to a randomly selected edge while the third, Faces, adds the next rectangle to any one of the previous rectangle's unused faces. Figures 12, 13, and 14 show examples of the three growth modes.

The fourth growth algorithm, sTacks, builds one or more stacks of growth units about a vertical axis. sTacks builds a more structured and rational image than the first three algorithms. Several selected options modify stacking attributes and characteristics. After selecting the seed point, grid spacing and the algorithm's random selection determine each succeeding stack's position in the grid's x-y plane. Randomness also affects dispersion of the growth units about the vertical axis. See figures 15a and 15b for examples.

sLopes, the fifth growth algorithm, builds sloping forms, all of which slope in a positive direction, along the x and y axes, from the seedpoint. Like sTacks, selected options set slope variables while random selection determines certain dispersion characteristics. Figures 16a through 16d illustrate the algorithm.

One of the Sketcher's more influential offerings is the opportunity to replace the usual rectangular growth unit by an AutoCAD defined BLOCK. Doing so adds a layer of information suitable either for enumerating architectural detail at the micro-level or, by adjusting scale and dimension, to express more definitively large-scale architectural form.

All of the algorithms' growth processes follow an iterative paradigm, derived from shape grammars, that sequentially adds growth units to the developing form. At the user's discretion, growth stops either when encountering a growth space boundary,
usually applicable only to the Corners, Edges, and Faces modes, or after a specified number of iterations. Termination also occurs if the Sketcher gets lost and spends too much time searching for the next growth unit's coordinates. Search time problems occur either when the space becomes too crowded or growth gets very near to a boundary. A searching - message in the command line indicates that the production may end without fully iterating all of its growth units.

The Sketcher offers several finishing options at the completion of a drawing. The first allows manipulation of the drawing by zooming in, zooming out, specifying viewpoints, undoing previous overlays, regenerating the drawing, hiding lines, and shading, and saving drawings. The next, draw again, takes advantage of Sketcher's sticky variables and draws again using the same parameters as the previous drawing. Bypassing draw again steps to the exit option, the only graceful way leave the Grid Sketcher. Not exiting will reset the Sketcher to the beginning where it awaits another set of variables. Sketcher retains the previous variable set as the default set.

Whenever a drawing exists that holds at least one object, Sketcher will ask, at the command line, whether or not to overlay with the next drawing. This feature illustrates another one of the Sketcher's very valuable, and supple, capabilities. Overlaying different productions from the various growth modes will generate composite drawings rich in complexity and character. Figures 10a and 10b show several examples of overlays. Starting the Grid Sketcher in a drawing with previous productions provides a way to overlay forms between sketching sessions.

The Grid Sketcher's modular forms conform to the grid spacing set by the growth unit's dimensions. Yet, as some of the examples show, by appropriately adjusting scale, modularity can fade into a sense of surface while still following the formal grid. Sketcher's formal vocabulary is almost limitless given the software's robust set of variables. The program's purpose is to effectively express geometric emanations of form very quickly and precisely, and derive images that, presumably, the designer might miss.
in the typical design process. The Sketcher holds elements of surprise, the unexpected, and even caprice. By skillfully manipulating the Sketcher's variables, and pursuing a sense of curiosity and experimentation, the designer may find interpretations and images in the forms that suggest formal solutions to design problems.

DIMENSIONAL VARIABLES -

The following discusses, in sequential order, the details of each of the Grid Sketcher's input variables.

**enter a WCS 3-D point to define the bounding box origin** -
*origin point* <0,0,0>:

While the bounding box's origin is always 0,0,0 in a particular UCS, this option allows placing the UCS origin anywhere in the WCS's positive quadrant. Since the bounding box defines a restricting volume, varying the bounding box origin, on successive overlays adds flexibility and discrimination in restricting growth to selected areas within the larger composition.

**enter grid bounding dimensions** -

\[ x \text{ axis:} \]
\[ y \text{ axis:} \]
\[ z \text{ axis:} \]

Growth algorithms propagate within these boundaries. The Sketcher expects integer values and interprets them as feet in architectural units. The software presents the resulting bounding box view from above, to the right, and in front of the origin, (viewpoint = 10, -7, 10).

**set grid spacing in feet** -

\[ x \text{ axis:} \]

Grid Sketcher -
y axis:
z axis:

The Sketcher interprets these integer values as feet, too. They establish the rectangular grid's three-dimensional structure although the grid pattern appears unstructured in the conventional sense. The grid references the seed point, not the (0, 0, 0) bounding box origin, and rectangular growth unit positions conform to the grid system. The software rejects grid spacings that exceed the bounding box dimensions.

**enter a 3-D point to seed the growth process -**

**seedpoint:**
Enter this x, y, z point value, either integer or decimal, in the following format:

3,7,19

Choose the seed point anywhere within the bounding box limits. The software rejects bad format, negatives, nil values, and points that exceed defined values derived from the bounding box limits. Growth begins at the seedpoint and propagates in the three axial directions.

**GROWTH ALGORITHMS -**
The following section describes the five growth modes, including their options, and the BLOCK vs rectangle option.

**select one of the following GROWTH MODES -**

- add to Corners - c
- add to Edges - e
- add to Faces - f
- sTacks - t
- sLopes - l

**growth mode:**

---

Grid Sketcher -
**BLOCK substitution**

Block substitution, presented immediately after selecting a growth mode, replaces the Sketcher's rectangles with either a defined AutoCAD block or an AutoCAD drawing file. The block name requires no extension.

**do you want to build with Sketcher's rectangles or an externally defined block? enter (b) for block or <return> to use rectangles**

Entering b brings up the following prompt asking for a block name. A <return> will default to rectangles.

**enter a predefined block name <>:**

As mentioned earlier, an external block can add important detail, and meaning, to the growth unit's definition. However, for large drawings, complex block definitions generate excessive HIDE and REGEN times, and significantly increase the drawing's database.

Choosing a block activates the following block scaling option:

**select a block insert scaling option**

- scale factor of 1
- grid spacing

**scaling option:**

A scale factor of 1 retains the dimensional relationships of the original blocked drawing. If the grid spacing dimensions do not match the blocked dimensions, the inserted block may either underflow or overflow its allotted rectangular space. Selecting the grid spacing option resolves the mismatch by automatically scaling the block
insertion to the grid dimensions. However, this causes block distortion in one or more axes.

The block option rejects invalid block names by issuing a prompt asking for either a valid block name or a <return> to continue.

**Corners**

Corners set the first growth unit's lower left corner at the seedpoint. Growth adds units randomly to any one of the previous unit's unused corners. Growth continues until reaching a box boundary, or a selected iteration limit. The iteration limit selection looks like this:

```
select one of the following to end the process -
    at a grid boundary - 1
    after a number of iterations - 2
```

ending option:

Selecting option (1) switches the screen to graphics and starts the growth process. Option (2) asks for the iteration limit:

**enter number of iterations:**

There are no absolute bounds on the iteration limit. The only limits are those implied by the bounding space's numerical capacity and the algorithm's tenacity in finding a spot for the next growth unit. If the growth space becomes too crowded, requiring extensive search time, growth will stop. Note that all five growth algorithms use this same termination sequence.

Corners, Edges, and Faces prevent intersection of growth units, guaranteeing that growth unit volumes will not intersect one another in the search for the next attachment. Figures 4 and 5 illustrate patterns and forms in the Corners mode.
**Edges**

As its name implies, Edges adds the next growth unit to an available edge rather than a corner. It works exactly like corners otherwise. Forms produced by Edges look similar to Corners but are denser and usually better organized. Note that, unlike Corners and Faces, Edges sorts its growth units into three color groups, each on its own layer. By discreetly turning layers on and off the algorithm also becomes a tool to investigate deconstructing and reconstructing the production in different patterns. See figures 6 and 7 for examples.

**Faces**

Faces replaces the last growth unit with two joined face to face, producing forms even more compact and structured than Corners or Edges. Faces follows all the other algorithmic determinants seen in Corners and Edges. See figures 8 and 9 for examples.

**Stacks**

Stacks builds "vertical" forms that rest at an elevation determined by the seed point's z value. There are three controllable stack parameters: height, eccentricity, and spacing. Stack organization grows vertically about a yellow colored axis. Axis height, and thus stack height, follows either the zaxis bounding limit or a different limit selected by the height parameter as follows:

**select a stack HEIGHT option**

- Full height - \( f \)
- Mid height - \( m \)
- Low height - \( l \)
- Variable height - \( v \)

height:
The difference between the z axis limit and the seed point's z coordinate establishes the 
*Full height*, the maximum height of any stack (Fig. 10). 75% of maximum defines *Mid height*, 50% sets *Low height* (Fig. 11). *Variable height* allows the growth algorithm to randomly select stack heights which compose to a contoured texture at the form's upper surface (Fig. 12).

Selecting an eccentricity option determines growth unit dispersion about the vertical axis:

**select a stacking ECCENTRICITY** -

- None - n
- Attached to vertical axis - a
- Maximum about the axis - m

**eccentricity**:

*None* builds exactly stacked units (Fig. 10). *Attached to vertical axis* allows the algorithm to randomly shift each growth unit along its x and y axes while still ensuring the unit remains attached to the vertical axis (Fig 11). *Maximum about the axis* extends the x and y axes displacement to the full limits of the growth unit's dimensions allowing some of the growth to proceed detached from the vertical axis (Fig. 12).

The spacing factor sets the stacking density, the relative proximity between stacks, as follows:

**select a stack SPACING FACTOR** -

- overlap spacing - 1
- intrusion spacing - 2
- intermediate spacing - 3
- wide spacing - 4

**spacing factor**:

*Overlap spacing* spaces vertical axes at exactly the growth unit's x and y dimensions.

---

**Grid Sketcher** - 9

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Selecting an eccentricity option other than *none* in this mode allows a significant degree of volume intersection between adjacent stacks. *Intrusion spacing* sets the stacks at twice the growth unit's x and y dimensions, which reduces the degree of growth volume overlap between stacks (Fig 11). *Intermediate spacing* (Fig. 12) and *wide spacing*, set at 3 and 4 times the growth unit dimensions, precludes any volume intrusion no matter what the eccentricity option.

Deftly manipulating stacking variables produces a panoply of forms, some quite simple, others rich and interesting in their content. For example, choosing no dispersion, closest stack spacing, and constant height essentially builds a solid rectangle with surface divisions articulated at the grid spacing. See figure. On the other hand, selecting the maximum displacement for each variable yields forms so diverse in their character that making value judgments about the meaning of their images becomes quite challenging.

**Slopes**

Slopes, like stacks, is a multi-variable algorithm, however its character is decidedly more horizontal. See Figures 13, 14 and 15. Since, conventionally, rise over run defines the slope, the algorithm asks for these values to establish the form's initial slope. Growth begins at the seed point and looks very much like stacks skewed or sloping towards the horizontal by just the value of the slope. The sloping axes, although segmented, still align along the y axis. Each growth unit's x value varies randomly about the growth axis.

The three controllable parameters help define the slope's detail and character: slope density, slope contour, and slope character. One of the two following options determines slope density:

**select a slope DENSITY option**

| Surface slopes | - s |
| Mass slopes    | - m |
slope density:

*Surface slopes* creates a form just one layer thick, which at the appropriate scale approximates to a surface (Figs. 13 and 15). Surface slopes grow continuously only in the positive x direction away from the seed point. Note that the character of surface slopes varies considerably from that of mass slopes. Selecting surface slopes brings up a sub-menu, the CONTOUR option shown next, not offered by mass slopes.

The *mass slopes* option stacks surfaces under and on top of each other in a progression where each sloping stack begins at a randomly selected x, y point (Fig. 14). The form takes on a very compact, sloping character reminiscent of hills.

Choosing *surface slopes* in response to the above slope density menu presents the following two slope contouring options:

**select a slope CONTOUR option** -  
- Linear surface - *l*  
- Variable surface - *v*

**slope contour:**

*Linear surface* limits the x/y plane slope to a linear value forcing the form’s front edge to parallel the x axis (Fig. 13). A *variable surface’s* sloping stacks grow in the x direction in constant increments while a geometric algorithm seeded by a random variable determines each successive stack’s y value (Figs. 14 and 15). The resulting form not only slopes in the y/z plane, but its front edge slopes across the x/y plane as well. Independent of the selected iteration limit (chosen later), both option’s growth stops at the x axis bounding limit. A variable surface’s growth stops as well when the next sloping stack along the surface contour exceeds the y axis bounding limit.

One of the following two choices determines, in general, the forms dominant sloping character:
select a slope CHARACTER option -

Constant slope - c
Variable slopes - v

slope character:

Constant slope limits the entire form to just the slope computed from the rise and run values entered next (Fig. 13). Variable slopes generates random slope changes determined by not only the random variable but also the growth unit size and bounding box dimensions (Figs. 14 and 15). The varying slope pattern, computed at the outset, remains constant for each of the form's sloping stacks.

After setting the three slope variables, the following prompts ask for the rise and the run required to compute the initial slope value:

enter a value for the initial slope's rise:
enter a value for the initial slope's run:

Slopes reads the rise and run as either integers or decimals without units and since they form a ratio. The initial slope biases subsequent slope value computations. For example, selecting a steep initial slope (large rise compared to run) induces the steeper slope increments more appropriate for emphasizing the vertical.

Slopes is probably the most provocative of the five growth algorithms. In appearance its forms have a certain structure yet remain difficult to decipher. Growth unit size and proportion have greater impact in their ability to manipulate the form's context and interpretation. Perhaps slopes shows its most flagrant contribution in its interaction with other growth mode forms in the compositions created by overlays.

FINISHING TOUCHES -
Once a form is complete Grid Sketcher provides several drawing manipulation tools. First, a short menu at the command line includes four choices: zoomin (zi), zoomout (zo), viewpoints (vv), undo layer (un), erase obj (eo), regen (rg), hide (hd), shade (sh), save dwg (sv), and erase all (ea). These options are continuously available in any sequence until terminated by a <return>. Zoomin is just the AutoCAD zoom-window function and prompts for point one and point two entered with the mouse. Zoomout returns the drawing to the Sketcher's default view. Viewpoints provides a way to look at the production from any desired 3-D point in the WCS system by entering the x, y, z values for the desired viewpoint.

The first six overlays, not including the original opened drawing, exist on their own layers, with additional overlays adding to the last (sixth) layer. Undo layer allows erasing overlays, in sequence, the last one first, back to the original drawing. Undo overlay is quite useful since it is the only tool available to sequentially erase previous overlays without completely erasing the active drawing. Erase obj activates the cursor to look at the production from any desired 3-D point in the WCS system by entering the x, y, z values for the desired viewpoint.

Regen (the AutoCAD regeneration function) facilitates redrawing after an undo. Hide (the AutoCAD hide function) can take a lot of time for complex drawings involving many forms. Shade, the AutoCAD shade command, can usually render complex drawings faster, and with greater visual clarity, than the hide option. Save dwg prompts for an alpha character only file name, (maximum of eight characters), adds the .dwg extension, and saves the current drawing under this name without leaving the Sketcher. Erase all is a one time deletion of all objects in the composition. The drawing space ids left completely empty.

Next, the Grid Sketcher prompts, asking about drawing another form derived from the same set of parameters used for the previous production. The drawings will not be quite the same though since the Sketcher's inherent randomness is always at work. The
The draw again option just provides an expeditious way of repeating the same parameters while avoiding paging through all the option menus. Entering a <return> will step past draw again.

Selecting draw again brings up the overlay prompt, which requires a (y) or (n) answer. As mentioned earlier, overlays are powerful accommodations that allow layering forms into composite drawings. There is no limit on overlay repetitions, however drawings can become quite complex very quickly. Forms, volumes, and shapes may intersect on successive overlays since the Sketcher’s database does not prevent growth unit conflict between growth algorithms. Whenever a drawing holds at least one object Sketcher will ask about overlays. Not selecting overlay completely erases the drawing, including all forms from previous overlays.

Bypassing draw again reveals the exit option, the one chance to exit the Sketcher as a normal procedure. Note that as long as the current AutoCAD drawing remains open all of the Grid Sketcher’s variables will remain intact as well. Invoking the Grid Sketcher again (by entering gr ) will display the previous grid sketcher session’s default parameters.

While e exits, <return> completes the cycle and returns the Grid Sketcher to the opening dimension menus. The Sketcher remembers its parameters and offers them as default values. This allows paging through the menus quite rapidly, changing only those parameters of interest.

NOTES -

The Sketcher’s drawings are complete AutoCAD .dwg files and on occasion it is extremely valuable to edit them as AutoCAD drawings outside the Sketcher. For example a promising overlay drawing may improve dramatically by moving, copying, or erasing selected elements in its forms.
The Sketcher initially seems very abstract, however with growing familiarity variables become more meaningful at the outset and productions can assume a sense of predetermination. For example in Figure 20 the forms appear rather structured implying an image of buildings while in Figure 19 the more loosely constructed image seems to convey a very suburban scale. Both drawings represent a purposeful manipulation of variables to generate a desired image.

The Grid Sketcher derives its power from the AutoCAD environment, the computer's processing speed, fertile and capable growth algorithms, and an inquisitive designer. Its product is a robust and challenging set of abstract forms, not all of which elicit a positive response, but that always require significant interactive interpretation and response from the designer.
APPENDIX IV

SAMPLE AUTOLISP CODE

The Grid Sketcher's coding in AutoLISP represents about 4000 lines of code contained in 50 pages of text. For the sake of completeness in the thesis, this appendix presents a small portion of the code to illustrate both AutoLISP coding technique, and as verification of the code as stated in the thesis.

Since the code follows a protocol of modularity, most of the software's functions are encapsulated in sub-routines representing modules "called" by a control program. One such sub-routine is the sLopes algorithm, the one chosen for illustration on the following pages.

275
(defun slopes()  
  (setq count 1)  
  (end nil)  
  (endctr 1)  
  (xcoords '(0))  
  (constx (car stpt))  
  (consty (cadr stpt))  
  (nxtypt (cadr stpt))  
  (slope (/ rise run))  
  (gslope (/ (float zgs) (float ygs)))  
  (searchflagx1 nil)  
  (searchflagx2 nil)  
  (searchflagy1 nil)  
  (searchflagy2 nil)  
  (endssearch nil)  
)  

(setq slopenumber (fix (/ (/ yaxis ygs) 4)))  
(if (= slopenumber 0)  
  (progn  
    (setq end t)  
    (prompt "the bounding box is too narrow for the slope")  
    (prompt "and grid spacing - please increase the y axis")  
    (prompt "dimension - enter <return> to continue -")  
    (getstring)  
  )  
)  
(setq slopecounter slopenumber)  
(if (> slopenumber 8)  
  (setq slopenumber 8)  
)  
(while (> slopecounter 0)  
  (cond  
    ((= slopecounter 1)  
      (setq slope1 slope)  
      (setq rand 0)  
      (while (< rand 3)  
        (counter)  
        (setq slope1ctr (ranum))  
      )  
    )  
    ((= slopecounter 2)  
      (nextslope)  
      (setq slope2 newslp)  
      (setq slope2ctr slpctr)  
    )  
    ((= slopecounter 3)  
      (nextslope)  
      (setq slope3 newslp)  
    )  
  )  
)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(setq slope3ctr slpctr)

((= slopecounter 4)
  (nextslope)
  (setq slope4 newslp)
  (setq slope4ctr slpctr)
)

((= slopecounter 5)
  (nextslope)
  (setq slope5 newslp)
  (setq slope5ctr slpctr)
)

((= slopecounter 6)
  (nextslope)
  (setq slope6 newslp)
  (setq slope6ctr slpctr)
)

((= slopecounter 7)
  (nextslope)
  (setq slope7 newslp)
  (setq slope7ctr slpctr)
)

((= slopecounter 8)
  (nextslope)
  (setq slope8 newslp)
  (setq slope8ctr slpctr)
)

(t nil)

):cond

(setq slopecounter (- slopecounter 1))

);end while slopecounter

(setq slopecounter 1)

(setq slopeflag t)

(setq Inslpctr 0)

(while (= end nil)
  (if (and (= slopeflag t) (= constslopeflag nil))
      (cond
        (((= slopecounter 1)
           (setq slopeunitcnt slope1ctr)
           (setq slope slope1)
           (setq slopeflag nil)
         )
        (((= slopecounter 2)
           (setq slopeunitcnt slope2ctr)
           (setq slope slope2)
           (setq slopeflag nil)
         )
        (((= slopecounter 3)
           (setq slopeunitcnt slope3ctr)
           (setq slope slope3)
           (setq slopeflag nil)
         )
        (((= slopecounter 4)
           (setq slopeunitcnt slope4ctr)
         )
      ))
  )
)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(setq slope slope4)
(setq slopeflag nil)
)
((= slopecounter 5)
  (setq slopeunitcnt slope5ctr)
  (setq slope slope5)
  (setq slopeflag nil)
)
((= slopecounter 6)
  (setq slopeunitcnt slope6ctr)
  (setq slope slope6)
  (setq slopeflag nil)
)
((= slopecounter 7)
  (setq slopeunitcnt slope7ctr)
  (setq slope slope7)
  (setq slopeflag nil)
)
((= slopecounter 8)
  (setq slopeunitcnt slope8ctr)
  (setq slope slope8)
  (setq slopeflag nil)
)
(t nil)
);cond
);end if slopeflag
(if (= constslopeflag nil)
  (progn
    (setq slopeunitcnt (- slopeunitcnt 1))
    (if (and (= slopeunitcnt 0) (< slopecounter slopenumber))
      (progn
        (setq slopeflag t)
        (setq slopecounter (+ slopecounter 1))
      )
    )
  )
)
(prompt "ndrawing shape # - ")
(print count)
(setq axispt (list constx ypt zpt))
(command "thickness" zgs)
(command "color" "yellow")
(command "circle" axispt (* xgs 0.05))
(command "color" "bylayer")
(drawshapes)
(dbase)
(setq count (1+ count))
(setq maxcy 1
  stop 1
)
(counter)
(if (= (ranum) 0)
  (setq rand 10)
)
(setq xpt (- constx (/ xgs rand)))
(cond
  ((= slope gslope)
   (setq ypt (+ ypt (float ygs)))
   (setq zpt (+ zpt (float zgs))))
  ((> slope gslope)
   (setq ypt (+ ypt (* (/ gslope slope) (float ygs))))
   (setq zpt (+ zpt (float zgs))))
  ((< slope gslope)
   (setq ypt (+ ypt (float ygs)))
   (setq zpt (+ zpt (* (/ slope gslope) (float zgs)))))
)
(setq next t)
(if (= slpsfcflag t)
  (if (or (= (setq bndryval (slpbndry)) nil) (= (chkpt) t))
    (cond
      ((and (= quirval "1") (= bndryval nil))
       (setq end t)
       (setq next nil)
       (setq slpsfcflag nil)
     )
      (setq ygridvar ygs)
      (setq Inslpctr Inslpnmbr)
      (setq ygridvar (• ygridvar Inslprct))
      (setq ypt (+ nxtypt ygridvar))
      (setq nxtypt ypt)
      (setq Inslpctr (- Inslpctr 1))
    )
  )
  (progn
    (setq ygridvar (* ygridvar inspct))
    (if (< insipvar 5)
      (setq ypt (+ nxtypt ygridvar))
      (setq ypt (- nxtypt ygridvar))
    )
  )
  (setq nslpctr (- nslpctr 1))
)
(if (= insfcflag t)
    (setq ypt consty)
)
(setq xpt (+ constx (* xgs 2)))
(setq constx xpt)
(setq slopeflag t)
(setq slopecounter 1)
(setq zpt (caddr seedpt))
(if (> xpt (- xaxis xgs))
    (setq end t)
)
(if (> ypt (- yaxis ygs))
    (setq end t)
)

; end slpsfcflag
(while (and (= next t) (= slpsfcflag nil))
  (if (or (/= (setq bndiyval (slpbndry)) nil) (= (chkpt) t))
      (cond
        ((and (= quitval "1") (= bndryval nil))
          (setq end t)
          (setq next nil)
          (setq nextslpt nil)
        )
        ((= quitval "2")
          (setq rand 0)
          (while (or (= rand 0) (= rand 9))
            (= rand 9).
            (counter)
            (ranum)
          )
          (if (or (= rand 1) (= rand 2))
            (progn
              (setq xpt (+ constx (* xgs 2)))
              (setq constx xpt)
              (setq ypt consty)
            )
          )
          (if (or (= rand 3) (= rand 4))
            (progn
              (setq xpt (- constx (* xgs 2)))
              (setq constx xpt)
              (setq ypt consty)
            )
          )
          (if (or (= rand 5) (= rand 6))
            (progn
              (setq ypt (+ consty (* ygs 2)))
              (setq consty ypt)
              (setq xpt constx)
            )
          )
      )
    )
  )

21
(if (or (= rand 7) (= rand 8))
  (progn
    (setq ypt (- consty (* ygs 2)))
    (setq consty ypt)
    (setq xpt constx)
  )
)
(setq zpt (caddr seedpt))
(setq nextslpt t)
(setq slopflag t)
(setq slopecounter 1)
):quitval 2
):cond
(progn
  (setq next nil)
  (setq nextslpt nil)
)
):if
(if (and (= nextslpt t) (= (setq bndryval (slpbndry)) nil))
  (progn
    (setq xpt constx)
    (setq ypt consty)
    (cond
      ((= bndryval 1)
        (setq xpt (+ xpt (* xgs 2)))
        (while (= (chkpt) t)
          (setq xpt (+ xpt (* xgs 2)))
          (setq constx xpt)
          (setq consty ypt)
        )
      )
      ((= bndryval 2)
        (setq xpt (- xpt (* xgs 2)))
        (while (= (chkpt) t)
          (setq xpt (- xpt (* xgs 2)))
          (setq constx xpt)
          (setq consty ypt)
        )
      )
      ((= bndryval 3)
        (setq ypt (+ ypt (* ygs 2)))
        (while (= (chkpt) t)
          (setq ypt (+ ypt (* ygs 2)))
          (setq consty ypt)
          (setq constx xpt)
        )
      )
      ((= bndryval 4)
        (setq ypt (- ypt (* ygs 2)))
        (while (= (chkpt) t)
          (setq ypt (- ypt (* ygs 2)))
          (setq consty ypt)
          (setq constx xpt)
        )
      )
    )
  )
)
; cond
  ; progn
  ; if
  (maxcycle 0)
  ; end while - next
(if (= quitval "2")
  (stkslpndcycle)
)
(if (= endt)
 (dbase)
)
(setq stpt (list xpt ypt zpt))
(delay 0)
) ; end while - end
) ; end SLOPES

;----------
; main function - GR
;----------

(defun c:GR()
(setq continue t
  bnrflag t
  overlaydwg nil)
(command
  "layer" "m" "overlay1" "c" "cyan" ===
  "layer" "m" "overlay2" "c" "cyan" ===
  "layer" "m" "overlay3" "c" "cyan" ===
  "layer" "m" "overlay4" "c" "cyan" ===
  "layer" "m" "overlay5" "c" "cyan" ===
  "layer" "m" "overlay6" "c" "cyan" ===
  "shadedge" "1"
)
(textpage)
(while (= continue t)
(command
  "osnap" ===
  "cmdecho" "0"
  "blipmode" "off"
  "snap" "off"
)
(textpage)
(setvar "orthomode" 0)
(command "units" 4 4 1 2 0 "n")
(command "elevation" 0)
(if (= bnrflag t)
(progn
 (textpage)
 (prompt "\n\n")
 (prompt "\n-----------------------------------")
)
BIBLIOGRAPHY


Omura, George. 1990. The ABC's of AutoLISP. San Francisco: SYBEX.


Terzidis, Constantinos A. 1989a. DAEDALUS, the architectural transformation system. Master Thesis. The Ohio State University.


VITA

Graduate College
University of Nevada, Las Vegas

Brian M. Gardner

Local Address:
1065 East Flamingo Road, Suite 606
Las Vegas, Nevada 89119

Degrees:
Bachelor of Science, Aeronautical Operations, 1971
San Jose State University

Master of Business Administration, 1981
San Jose State University

Master of Science, Computer Science Engineering, 1984
San Jose State University

Thesis Title: The Grid Sketcher - An AutoCAD Based Tool for Conceptual Design Processes

Thesis Examination Committee:
Chairperson, Dr. Hugh Burgess, D. Arch.
Committee Member, Dr. Zouheir Hashem, Ph. D.
Committee Member, Mark Hoversten, MFA
Graduate Faculty Member, Dr. Robert Tracy, Ph. D.