THE TOPDOWN SYSTEM AND ITS USE IN TEACHING
An Exploration of Structured, Knowledge-Based Design
William J. Mitchell
Harvard Graduate School of Design
Robin S. Liggett
UCLA Graduate School of Architecture and Urban Planning
Milton Tan
Harvard Graduate School of Design
September 1988

ABSTRACT
The Topdown System is a shell for use in developing simple (but we believe non-trivial) knowledge-based CAD systems. It provides a data structure, graphics capabilities, a sophisticated user interface, and programming tools for rapid construction of knowledge bases. Implementation is for Macintosh, Macintosh II, IBM PC/AT, PS/2, and Sun workstations.

The basic idea is that of top-down design – beginning with a very abstract representation, and elaborating that, in step-by-step fashion, into a complete and detailed representation. The basic operations are real-time parametric variation of designs (using the mouse and slide bar) and substitution of objects. Essentially, then, a knowledge-base in Topdown implements a kind of parametric shape grammar.

The main applications of Topdown are in introductory teaching of CAD, and (since it provides a very quick and easy way for a user to develop detailed geometric models) to provide a uniform front-end for a variety of different applications. The shell, and some example knowledge-bases, are publicly available.

This paper discusses the principles of the Topdown Shell, the implementation of knowledge bases within it, and a variety of practical design applications.

THEORETICAL BACKGROUND: TOP-DOWN KNOWLEDGE-BASED DESIGN

Conventionally, computer-aided design and drafting systems (2D drafting systems, 3D modeling systems, paint systems, etc.) work essentially in bottom-up fashion. They provide a range of graphic primitives (vectors, arcs, splines, etc.) and facilities for deploying instances of these to build up complex drawings. The menus of these systems typically show the available primitives and the “tools” (operators) provided for inserting, deleting, transforming, and combining them. This approach is effective when an existing drawing is to be input to the database, or when a drawing stored in the database is to be edited, but it provides little more help than a pencil when you want to construct from scratch a drawing of some complex object such as a human figure, an automobile, or a classical column: you must depend on your own skill and knowledge of what the pieces are, and how to shape them and put them together. If you already know how to draw something then a CAD or graphics system will enable you to do so efficiently, but if you do not know how to begin, or how to develop and refine a drawing, then the efficiency that you gain is of little practical consequence.

By contrast, experienced expert graphic artists and designers usually work in top-down fashion – beginning with a very schematic sketch of the whole object, then refining this, in step-by-step fashion, till the requisite level of precision and completeness is reached. For example, a figure drawing might begin as a “stick figure” schema, then be developed to show general massing, and finally be resolved down to the detailed features. Similarly, an architectural drawing might begin as a “front” showing just a skeleton of construction lines, then be developed into a single-line floor plan, then a double-line floor plan, and finally a fully developed and detailed drawing.

251
"How to" drawing manuals often explicitly demonstrate this top-down, stepwise refinement process. For example, the famous sketchbooks of Villard de Honnecourt, Leonardo da Vinci and Albrecht Dürer all show numerous examples of schemata for various types of objects, together with procedures for elaborating these into fully-developed drawings (figure 1). The design method of the École des Beaux Arts, as formalized in the textbooks of J. N. L. Durand, begins with a parti and proceeds by stepwise refinement (figure 2). In our own time, books of tips for amateur artists (as found on sale in any artists' supply store) are usually set out in much the same way.

Figure 1: A page from Dürer's notebooks, illustrating his method of top-down, step-by-step refinement of stick-figure schemata into detailed and accurate figure drawing.
Figure 2: Durand's method of top-down, step-by-step refinement of a schematic parti into a detailed floor plan.
A theoretical foundation for this view of representational graphics has been provided by *Art and Illusion* – a classic work by the eminent art historian E. H. Gombrich [Gombrich 1960]. In this work, Gombrich argues convincingly that artists proceed not by naive, direct, observation and recording of nature, but through a cycle of "schemata and correction". An artist has knowledge of schemata for various objects, and of how to refine and develop these, and always draws on this knowledge in constructing images. Observations from nature lead to modification and correction of these schemata, but some established schema is always there as a "default" starting point.

All this will sound familiar to anyone with a knowledge of recent developments in computation. Top-down stepwise refinement has become a standard methodology for program development and "outline processors" for top-down development of text documents are coming into widespread use; these operate on different types of artifacts, but the basic ideas are very closely analogous to that of top-down drawing development. In the field of knowledge engineering, structures such as frames, which embody the idea of a predefined schema, are of increasing importance. The success of these techniques encourages confidence in the success of analogous approaches to graphics.

**THE TOPDOWN SYSTEM**

The Topdown system is a shell for quickly and inexpensively implementing fairly simple (but we believe non-trivial) CAD systems based on the idea of top-down knowledge-based design. Prototype versions were developed for the Macintosh and IBM PC/AT computers. Current versions are implemented on higher-powered machines with better graphics capabilities: the Macintosh II and the IBM PS/2 series. Shortly we expect to add a Unix version for Sun workstations. All versions are coded in Pascal. They are publicly available, and we encourage their dissemination and use.

There are three levels of Topdown: the "user" level, the "programmer" level, and the "substructure" level. We shall consider these in turn.

**THE USER LEVEL**

A designer working at the "user" level sees the screen shown in figure 3. (It is essentially the same in all implementations.) There are two graphic windows – the peek window and the poke window – a control bar, dialogue boxes which appear as required; and the usual icons and pulldown menus.

![Image of the Topdown screen with peek and poke windows](image)

Figure 3: The Topdown screen, with its peek and poke windows.
The peek window always shows the current state of the design – depicted as a two-dimension composition of lines and color-filled polygons. (Current versions of Topdown are all two-dimensional, but we expect to implement three-dimensional versions in the future.) There are the usual pan, zoom, and other basic display functions. The peek window is strictly for viewing only: the user cannot directly select or operate upon graphic objects displayed within it.

All design interaction is accomplished via the poke window. The designer only needs to know two basic moves: substitution of a more detailed representation for a less detailed representation, and parametric variation of dimensions, proportions, colors, and so on. Substitution is accomplished by pointing (with the mouse-controlled cursor) in the poke window to the part which is to be replaced, clicking to see replacement alternatives displayed in situ, then clicking OK to select one. Figure 4, for example, shows the screen as it appears during selection of a particular type of column capital and substitution of the choice within a schema for a complete column.

![Diagram of Tuscan/Doric Abacus](image)

**Figure 4:** Selection and substitution of an element.

Parametric variation of dimensions is accomplished by pointing at a dimension line in the poke window, then resizing (in real time) with the slide bar that is displayed on the control bar (see figure 5). The control bar displays the name of the selected variable, its current value, and other relevant information.

Essentially, then, the user manipulates a (somewhat simplified) form of parametric shape grammar. All interaction is via the mouse (the keyboard need never be used) and is in real time. Users soon learn to employ Topdown for extremely rapid exploration of design alternatives – so rapid, in fact, that spectators usually cannot follow the process. Users conceive of this process not as step-by-step editing of a mostly static drawing, but as continuous reshaping of a very fluid object – much like throwing a clay pot on a wheel.

**THE PROGRAMMER LEVEL**

A programmer working at the "programmer" level of Topdown sees an extension of the Pascal programming language with a range of special functions and procedures. This is used to encode knowledge of what the parts of an artifact are, the properties of these parts, and how they can be put together in larger subsystems (figure 6 shows how parts of classical columns may be organized in a top-down hierarchy). In other words, the programmer specifies the vocabulary and syntax of the graphic language that is to be manipulated by the user.
Figure 7: An example of code expressing some drawing rules.
Although extension of Pascal is a realistic, practical approach to providing a programming capability (given currently available workstations and development environments), this does have some obvious limitations, and it is likely that the programmer level of Topdown will undergo some evolution in the future. In particular, it could benefit from adoption of a more wholeheartedly declarative style, and from use of graphic rather than symbolic expression.

THE SUBSTRUCTURE LEVEL

The "substructure" level of Topdown is written in Pascal, making use of windowing and graphic capabilities of the various hardware and operating system environments for which it is implemented. Thus it varies from implementation to implementation – but programmers and user need never be concerned with this. It handles the data structure, the screen, and user interaction, and provides necessary graphics functions.

The current Macintosh II version is implemented with Lighthouse Pascal, and the PS/2 version is implemented with Microsoft Pascal and Microsoft Windows.

EXAMPLES OF KNOWLEDGE BASES

Figure 8 illustrates some output from a typical knowledge base implemented in Topdown. This knowledge base (by Milton Tan) encodes knowledge of classical columns: essentially it is a computer-based parallel of the orders. A user can employ it to design a convincing classical column, (Doric, Ionic, Corinthian, Tuscan or Composite), in full detail, in a matter of minutes. Figure 9 shows some of the substitution rules implemented in this knowledge base, and figures 10a and 10b summarize the relationship of abstract and detailed designs.

Figure 8: Some classical column designs produced using a "parallel of the orders" knowledge base.
Figure 9: Some of the substitution rules in the "parallel of the orders" knowledge base.

Figure 10a: Steps in the process of reducing a detailed column to an abstraction.
Figure 10b: Steps in the process of refining an abstract capital schema into a detailed design.

Figures 11a and 11b show another example. In this case, the knowledge base (by Bojana Botanac) encodes knowledge of how to draw trees. Wide variations in size and proportion, branching structure, and foliage texture are possible, and an image of almost any imaginable type of tree can be produced in just a few minutes.

Figure 11a: Images of trees produced using a Topdown knowledge base.
PRACTICAL APPLICATIONS

The most important practical application of Topdown will probably be as an adjunct to conventional CAD systems. We expect that extensive libraries of knowledge bases will develop, and that these will be used by designers to make quick, accurate sketch designs of architectural elements and subsystems of various types. These sketch designs will then be transferred to conventional CAD systems for further editing and detailing, in the usual way.

Another application is as a uniform front-end to a wide variety of analysis programs. Designs for beams, arches, trusses, windows, ducts and other types of elements can quickly be developed using Topdown, then subjected to analysis. This has a significant practical advantage: users do not have to learn a new set of input conventions every time they want to use a new analysis program.

USE IN TEACHING

Knowledge of how to put things together (kitchen layouts, window details, roof trusses, accurate figure drawings, renderings of automobiles, etc.) traditionally transmitted by means of handbooks (such as Graphic Standards), by one-on-one demonstration in the design studio, and through apprenticeship in practice. Topdown provides a new way to record such knowledge, and to disseminate it widely and inexpensively. Furthermore, it supports an active, engaged mode of learning by doing—rapidly trying out many ideas and seeing where they lead. We contend that this can be much more effective than passive reading of texts, listening to lectures, and carrying out a few laborious, limited, manual exercises.
At the level of knowledge-base construction, Topdown provides a medium for teaching the art of formalizing knowledge and expressing it in machine-processable form— that is, computer programming. At Harvard and UCLA we have successfully used Topdown as an environment for student exercises in our introductory programming courses. (This is a direct extension of the approach outlined in our text The Art of Computer Graphics Programming [Mitchell, Liggett & Kvan 1987].) We find that students can immediately write simple but architecturally interesting Topdown programs without difficulty, and that they can program very elaborate and interesting applications within a semester. Topdown frees them from worrying about low-level issues of database organization and graphic interfaces, and allows them to concentrate their attention directly on the fundamental architectural and graphic logic of the situation. This, we contend, is precisely where architecture students (as opposed to professional computer technologists) should concentrate their programming efforts.

By providing a consistent user interface and programming environment across a variety of workstations, Topdown reduces the intellectual overhead of introducing computer applications to students, and allows useful coursework to accumulate and be disseminated.

CONCLUSION

Topdown is based on the idea that much architectural knowledge is knowledge of the sizes and shapes of things, and of how to put them together. It provides ways to encode and record such knowledge, and to use it effectively in design processes. Unlike most knowledge-based systems it is compact, inexpensive, and fast. It is intended for widespread practical use, by people without much technical background, on inexpensive personal computers and workstations.

It should be understood as an initial, relatively crude and simplistic prototype for more sophisticated systems of the future. The substitution mechanism provides only a limited first approximation to the capabilities of a shape grammar, the strict hierarchical structure that Topdown imposes on designs is too rigid, the programming language is not as concise and elegant as it might be, and the limitation to two-dimensional design is a serious one. But Topdown has clearly demonstrated the potential of structured knowledge-based design systems, and it will certainly serve us as a very useful testing and teaching tool until we are able to produce something better.

REFERENCES

Gombrich, E. H.

Art and Illusion - a study in the psychology of pictorial representation.

Mitchell, W.J., Liggett, R.L. & Kvan, T.

Mitchell, W.J., Liggett, R.L. & Tan, M.
The Topdown Manual.