Grids in Design and CAD

Mark D. Gross
University of Colorado at Boulder

The grid is a useful device for expressing design rules about the placement of elements in a layout. By expressing position rules for elements in relation to a grid, a designer can organize decisions in a layout design problem systematically. Grids and placement rules offer a discipline that can help a designer work effectively to lay out complex designs, and it can also facilitate group design work. Unfortunately, computer supported drawing systems often cannot support this way of working because they lack a sufficiently rich implementation of grids. The Grid Manager module of the CoDraw program shows enhancements useful for architectural Computer Assisted Design. These enhancements would enable effective ways of using the computer as a design tool.

1. Grids as Tools for Design.

The grid, one of the oldest architectural design tools, is a useful device for controlling the position of building elements. Grids have been and continue to be used in all manner of layout tasks from urban design to building construction (see figure 1) . A grid can help a designer control the positions of built and space elements, making the layout task more systematic. By determining positions of different building elements in relation to a grid or to a set of grids, the designer can specify design rules that describe a typology of physical forms. Many interesting architectural 'form families' can be described this way. The grid-based coordination of layout design can also support a team of designers where each designer is responsible for deploying a different subsystem. In laying out plans for new towns and cities, the use of grids permits the designers at the urban scale to make decisions, yet allow relative freedom at the block and lot scale for individual developers and house designers.

Figure 1. Historical uses of grids in design
a) Roman town grids; b) Jefferson's drawings for the University of Virginia

Most Computer-Assisted Drafting (CAD) programs offer a simple grid capability, where a designer can overlay a grid on a drawing, and can snap points and other graphic elements to the grid. Unfortunately most CAD programs fail to take full advantage of the grid as a design tool. Often the designer is limited to square grids and grid gravity is either "on" or "off" for all elements.

We have developed the CoDraw Grid Manager to explore how a drawing program might better support the use of grids in layout design. In CoDraw, grids are first class graphics objects and as many of them may be used in a design as needed. Grid parameters include two sequence variables that specify the grid's horizontal and vertical spacing units. A grid may be limited in extent, or it may fill the design work area. Grids may be selected and moved about the work area, and they may be grouped into aggregate grid configurations.

The concept of *element class* is essential for the applications of grids discussed here. The CoDraw program uses an object-oriented scheme to organize its database of elements. Every element belongs to a class which defines its generic properties, for example shape, color, and material. Class definitions are structured in a hierarchy, each level providing more specific definition for levels below. This scheme can be used in various ways. For example, the designer could define classes by color, e.g. "blue things," "red things." Another, perhaps more useful, application defines each building subsystem

(concrete foundation, structural steel, partition walls) as a class, and within each class defines different component types as subclasses. Then we can express generic placement rules for each class and subclass. For example, structural steel elements may be programmed to limit placement to relate to a certain grid, with different particular relations for I-beams, angle-iron, and C-section steel. Using this organization of element classes, CoDraw can be programmed to enforce design rules expressed in terms of grid relations.

We begin with examples of how grids can be used to express layout rules for architectural design. Then we introduce the CoDraw Grid Manager, and describes how this program supports the use of grids to express layout rules. Finally we discuss this approach to programming layout rules in a CAD program, comparing it with other representations for rules about shape and form in architectural design. Unlike shape grammars, for example, this approach is not generative. The drawing environment can be programmed with layout rules; within these rules the designer works freely. The rules are programmed interactively; should they prove too limiting the designer can change them.

2. Grids in Layout Design.

To understand the CAD support we want, let's look at how grids can be used as a layout tool. Three main concepts will emerge: (1) a variety of kinds of grids are used, from the simple square grid, to rectangular and tartan grids. (2) grids can be grouped and used together, and (3) rules about element placement can be expressed in relation to a grid or grids.

In layout design a grid is most often used as an underlay to a drawing, to organize the positions of elements. The grid-size is chosen carefully. It is usually related to the dimensions of the spaces to be laid out or the components to be placed. For example, in laying out wood framing members in a stick-built dwelling, a 16" or 24" grid is useful because in that system 16" or 24" is the on-center spacing between studs and joists, and other components in the construction system are compatibly sized.

2.1 Layout rules govern placement on a grid.

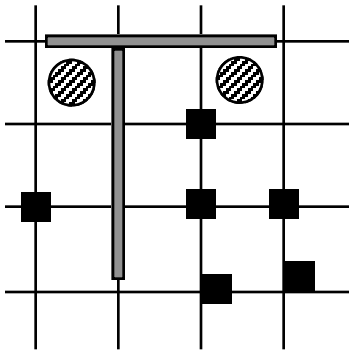

Figure 2. Different element-grid relations

To use a grid as a design tool, the architect must determine rules for placing elements relative to the grid. The simplest and most obvious placement rule is that elements center on grid crossings. However, other rules can be formulated: elements center only on one of their dimensions; elements center in grid squares; or their edges align with grid lines. For example, figure 2 shows different position relations for elements on a simple square grid.

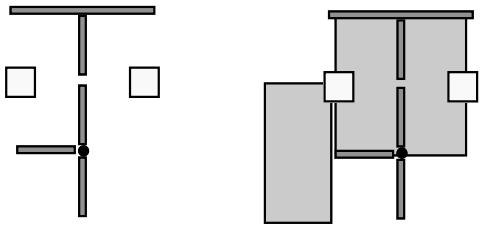

Figure 3. Various relations between an element class and a grid.

Figure 3 shows what happens when different grid positions are assigned to different types or classes of elements. In this example, wall centerlines run along grid lines; concrete columns are offset on grid crossings, and space boundaries (shown in gray) fall along grid lines.

2.2 Subdivided and superimposed grids.

Figure 4. Grids can be subdivided and superimposed.

Often it is useful to work with one grid at a large scale, and a subdivision of that grid at a smaller scale. The two grids are superimposed and registered (figure 4). For example, in the 2x4 stick building system, in addition to the 16" grid, a larger 48" (4') grid is useful for positioning larger elements such as gypsum board and plywood panels. A smaller 4" grid can also be used to place light switches, electric outlets, and other hardware.

2.3 Rectangular grids.

Figure 5. A rectangular grid is a basis for post and beam construction.

Grids need not be square. More often than not the landscape, building system, or the directionality of the design itself suggests a rectangular grid. A common use of a rectangular grid is to position members of a directional structural system, for example the post and beam construction in figure 5.

2.4 Interface conditions where grids meet.

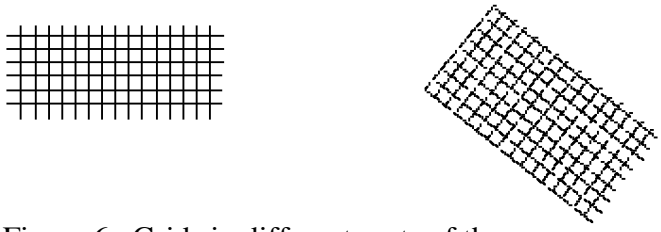

Figure 6. Grids in different parts of the building meet.

Complex designs often involve different grids in different parts of a building (figure 6). When two or more grids are used, the designer must consider the interface condition where they meet. In some cases special interface elements and rules are used. For example, a special, round column might be employed to make and mark the transition between two grids at different orientations (figure 7).

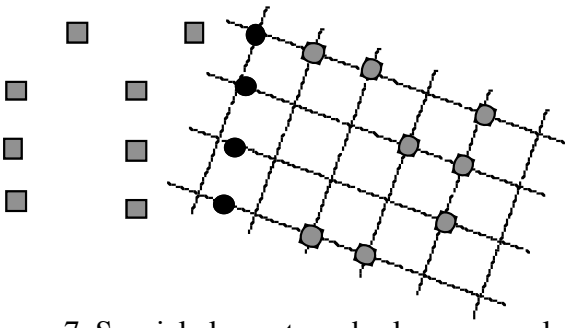

Figure 7. Special elements and rules may apply at interface conditions.

2.5 Several related grids.

It is often useful to work with several related grids when placing different elements in a layout. We can say that each building subsystem defines a class of elements, and we can use a different grid for each different class of element. For example, (figure 8) we can restrict placement of concrete columns to the crossings of one grid, and program partition walls to take their places on the lines of another, offset, grid. A similar effect was obtained in figure 2, where each element class was assigned a different grid relation. In this case, the offset relation between these superimposed grids represents an important design decision.

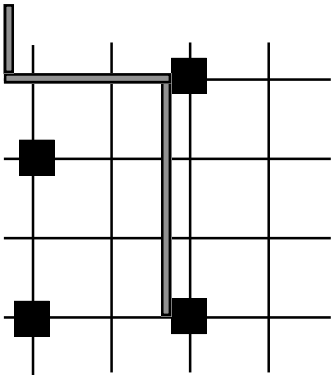

Figure 8. Element classes center on different grids.

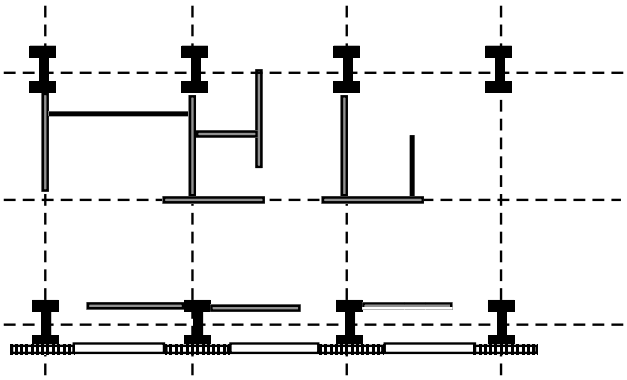

Figure 9. Different grids are used for structural and partition systems.

In another example, the major structural columns, beams, and bearing walls are placed on a large, master grid; interior partitions on a second grid that subdivides the master (figure 9), and curtain-wall or skin elements on a third, related, grid. Although the different systems are manipulated separately in the design process, perhaps by different designers, the coordination of the grids allows decisions to be made relatively independently.

In large projects, the layout design of different building subsystems and services (structural steel, partitions, water, electricity, HVAC) may be assigned to different experts from different firms or different work groups. It is important that each expert be able to proceed without constantly checking with other members of the team. By setting up an initial set of agreements or rules that govern the placement of elements of each subsystem, the designers can proceed relatively independently. The initial selection of grids and assignment of subsystem elements to certain grid relations represents this set of agreements. Once the team agrees to work within these rules, interference conflicts will be limited to a finite and predictable set of locations and conditions.

The initial steps of choosing grids and setting rules about the relations of grids and subsystem elements are crucial to the successful application of this method. Some testing of the grids and rules can be valuable at the early stages, to check that the rules permit certain desired configurations. Although simple in concept, the application of grid techniques in large design projects requires some experience.

2.6 A grid establishes relations between elements.

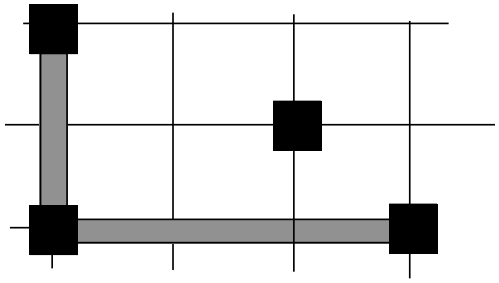

Figure 10. The grid is a device for controlling the joining conditions of elements.

By programming different element classes to take different positions relative to a grid, the designer indirectly controls the relationships of elements with respect to one other. Rather than specifying assembly rules that describe how elements are to join, elements are related to a common grid. In figure 10, square columns are centered on grid crossings and walls are centered only along their lengths, giving each element class a direct relationship to the grid, which indirectly defines the position relations between columns and walls.

2.7 Tartan or band grids.

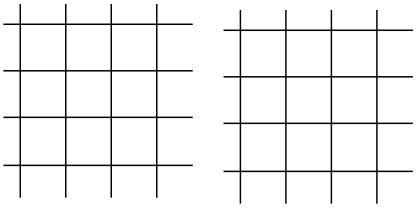

Figure 11. A tartan grid can be combined with a grid marking band centerlines.

Grids need not be always unitary; an alternating sequence of dimensional units can be used, fto form a tartan or band grid (figure 11a,b). A tartan grid can be superimposed on a simpler grid that marks the band centerlines (figure 11c). A rule can be expressed that requires or prohibits the placement of an element class in a band of the tartan grid, for example "partition walls must be located only in the 10 cm bands of a 10-20cm tartan grid."

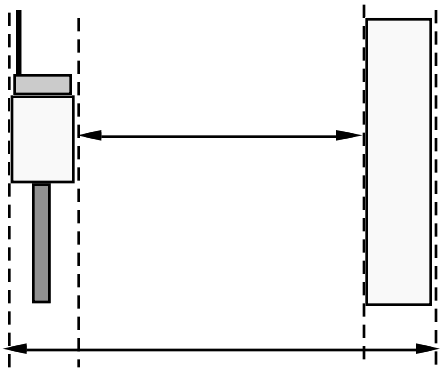

Figure 12. Tartan grids allow for variation in size of built elements.

Elements can be restricted to center on the centerline grid, and limited in dimension to stay within the tartan bands (figure 12). Specifically, their edge coordinates would be constrained to lie within the same band, or range of values. By expressing a rule about element dimension, the actual selection of components can be delayed and alternatives evaluated, so long as the components eventually chosen fit within the tartan band.

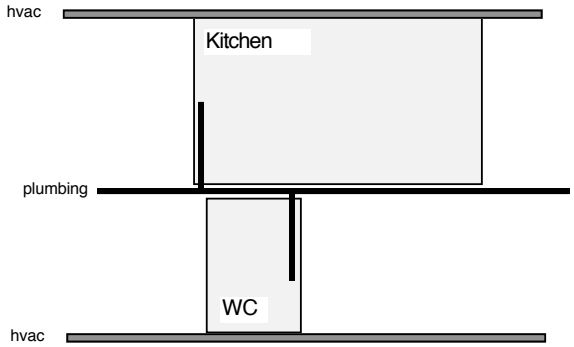

Figure 13. Each band can house a different service.

Similarly, building services such as electricity, plumbing, and ventilation can be routed in restricted zones. This is shown in figure 13. The tartan grid is an important part of a specific design methodology for dwelling design (Habraken et al. 1976) (Kroll 1987) and it is also the basis of description in the Dutch building code standards.

2.8 Exceptions and field deployment.

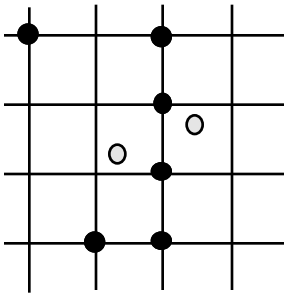

Figure 14. The white columns are exceptions to the class position relation.

When the designer establishes a position relation between a grid and an element class, it is understood to mean that this is the way elements of this class are to be placed. That is, every occurrence of the element on the grid must take the specified position relation. However, the designer can override the grid relationship defined in the class to make a particular element an exception. For example, the two white columns in figure 14 are exceptions to the class

relation, which allows columns only on grid crossings.

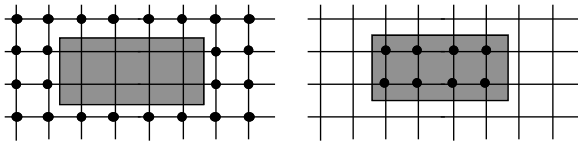

Figure 15. a) columns on grid except inside rectangle; b) columns on grid only inside rectangle

Normally, a grid-element relation means that if an element is placed on the grid, it must take its proper position. Another way to treat an element-grid placement relation is that for every occurrence of the grid condition, an instance of the element should be found. Thus, the rule "columns at grid crossings" would produce a field of columns, limited only by the extent of the grid. This treatment can be useful, combined with the ability to restrict, or bound, the deployment of the grid to certain regions. For example, figure 15 shows two bounding relations: columns on all grid crossings inside the rectangle (b) and columns on grid centers except inside the rectangle (a).

2.9 A simple example.

Let's look at a simple example of the use of grids in schematic building design. The first step is the design of a basic grid for layout. The decisions to be made are the choice of dimensions of the grid units. The criteria for making these decisions are primarily programmatic —the use dimensions of spaces to be made in the building, and technical —the dimensions of components in the building system that is to be employed.

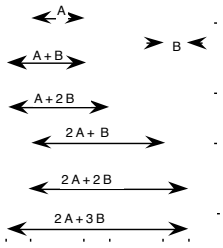

Figure 16. Use dimensions suggested by a tartan grid.

Figure 16 shows the different dimensions that a tartan grid provides. Comparing these dimensions with the use dimensions required for the functional program can give the designer a good idea of how well the grid will work. For example, A is 5', and B is 2', then the grid will suggest room widths of 5', 7', 9', 12', 14', ..., a fairly good match for a housing design project. Of course, the actual space available between walls will be diminished by thickness of the walls. An experienced designer or design firm may well have a standard grid or set of grids for basic layout design.

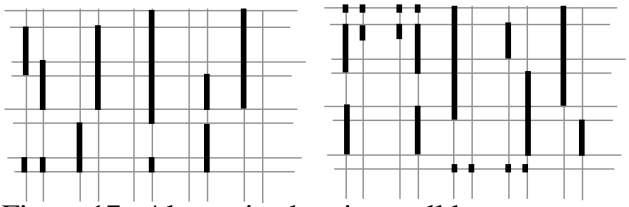

Figure 17. Alternative bearing wall layouts.

Once a basic grid for layout has been designed, a next step may be to experiment with the placement of bearing walls. Adopting a rule that locates bearing walls only on vertical grid lines, and limiting bearing wall dimensions to grid modules, the designer can rapidly explore the range of options that this system permits. Although at first these restrictions might seem to overly constrain the design, in fact a reasonable variation can be achieved. Figure 17 shows studies for two bearing wall layouts.

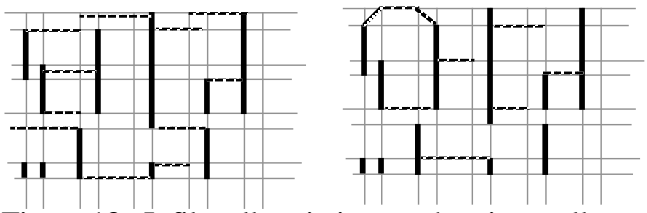

Figure 18. Infil wall variations on bearing wall alternative 'a'.

The next step in the design might be the location of infil walls. Each alternative placement of bearing walls will offer several variations in the placement of infil walls. Figure 18 shows infil wall variations.

The role for the grid in designing is to support, not to make, design decisions. By limiting the placement of elements to certain places, the grid

simplifies decision-making, allowing the designer to work with and compare a relatively small number of alternatives. However, the designer must ensure that the grid and placement permit a sufficiently rich range of variation. If it doesn't, the designer must redesign the grid, or relax the placement rules. Also, the designer must ultimately do the designing, determining where to place each element to realize a functional program and other design criteria. The grid is simply a tool that supports and organizes the decision-making.

2.10 Summary of grid uses.

We have reviewed some elementary uses of grids in layout design. For many architects, these applications will be familiar; however, most drawing programs cannot support them. From this brief review, we take a list of features that we would like to see supported by architectural drawing software. We would like to make grids of various proportions and dimensions: rectangular grids, tartan grids, and grids with bounded extent (e.g. a grid inside a room for laying out furniture). We would like to make grid aggregates, or configurations of several grids. We would like to define relationships between grids and classes of elements, so that different element classes can be programmed to take different positions relative to a grid or grids.

3. CoDraw's Grid Manager.

In the previous section we reviewed several ways that designers use grids to support layout tasks, in particular, by defining grid-relative rules for placement of different elements. Traditionally these methods are carried out by hand, with the designer remembering and enforcing rules about element placement. This section describes a program that supports the use of grids to express design rules. It aids the layout task by enforcing placement rules that the designer makes.

Grid snap behavior, or "grid gravity" is a feature in almost every drawing and drafting program. By setting a sufficiently fine grid spacing, a designer can ensure that elements placed into the design align their edges and that line segments latch, or connect. Without grid gravity, it can be difficult to draw accurately with a mouse or digitizer pen. The advantage of extending the support for grids and grid gravity is that designers can use it to specify position relations, or rules, among design elements. The drawing program will retain and enforce these rules as the designer edits the design.

CoDraw is a constraint-based drawing program, which manages and maintains spatial relations between elements that the designer selects (Gross 1990). For example, CoDraw maintains alignments, tangencies, and other geometric relations. Other algebraic relations, such as area and proportion relations, are also maintained by CoDraw. Constraints involving grids and grid-relations are handled specially by CoDraw's Grid Manager module. CoDraw is a graphic application that uses the experimental constraint programming language "Co", which is written in Common Lisp on the Macintosh.

3.1 Grids as design elements.

In CoDraw, grids are regular design elements, no different than other elements in the drawing. Every individual grid in a design belongs to a class that defines its spacing units and perhaps other properties. For example, the class 60-20-BAND-GRID in Figure 19 defines a tartan grid with alternating 60 and 20 unit bands. All grid classes are members of the larger class named 'grid'. Figure 19 shows GRID and its subclasses as defined when the Grid Manager is first invoked.

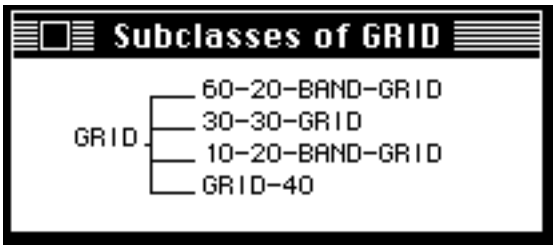

Figure 19. Class GRID and initial subclasses.

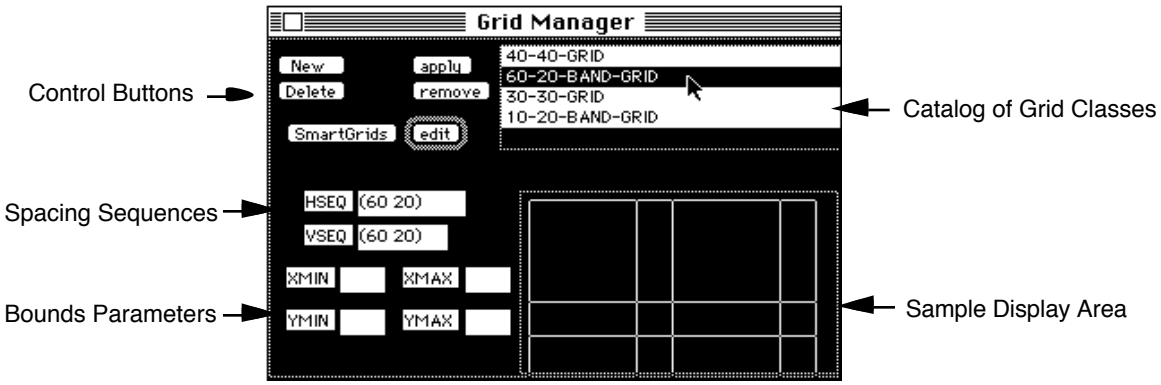

Figure 20. The Grid Manager window.

The Grid Manager window (figure 20) shows a catalog of already-defined grid classes. Clicking on a grid class name in the catalog causes the grid to be displayed in the sample display area. The Spacing Sequence parameters control the number of bands and the horizontal (HSEQ) and vertical (VSEQ) spacing units of the grid: in this case (60 20) defines alternating bands of 60 and 20 units. Values typed in here change the definition of the grid class. The Bounds Parameters control the extent of the grid: if no values are entered, then instances of the grid will fill any Work Sheet that they are placed into.

The Control Buttons in the upper left part of the window are used to manage the grid classes and to apply specific instances of grids to the design. The 'new' button defines a new grid subclass based on the currently selected class in the catalog; by entering different values for the Spacing Sequence parameters and the Bounds parameters, the new grid class can be modified or specialized. The 'apply' button makes an instance of the currently selected grid class in the Work Sheet, and the 'remove' button deletes instances of the selected grid class in the Work Sheet. The 'SmartGrids' button produces another window where the designer can specify the relation between the grid and an element or an element class.

A grid entered into the Work Sheet is treated as any other element. It can be selected, dragged about, colored, sized, copied, and deleted. (To avoid selecting grids unintentionally, a modifier key must be held down while selecting a grid with the mouse.) The Work Sheet can contain several grids simultaneously, grids can be grouped into configurations, and spatial relations between grids can be established.

3.2 Relations between grids and other elements.

Using the SmartGrids feature of the CoDraw Grid Manager, we can program the position relation between an element class and a grid. We must specify whether the element is to be centered, offset, or adjacent, registered along an edge on the grid, or whether the element is to be limited to certain grid bands or zones.

Whenever an element of that class is moved into a portion of the Work Sheet where the grid is deployed, it will take its position according to the programmed relation. Most often a class of elements is associated with a class of grids. However, the relation can be programmed to operate between a specific element and a specific grid, a class of elements and a specific grid, or a specific element and a class of grids. For example, the class of 'round columns' can be programmed to center on all instances of 'grid-40', or only on the instance of 'grid-40' in the upper right hand corner of the Work Sheet. Likewise, a specific 'round column' can be programmed to take its place in relation to all instances of 'grid-40' or to only a specific instance of that grid.

3.3 Implementation.

The Grid Manager is embedded in CoDraw and takes advantage of CoDraw's prototype inheritance scheme; it also uses CoDraw's graphics. The organization of CoDraw's elements into a graph of prototypes and individuals that inherit constraints enables the assignment of grid behavior to different element classes.

At present, grid relations are implemented separately from CoDraw's general constraint management routines, which implement multi-directional value propagation and simple algebra on the constraint net. Grid relations could be expressed as algebraic expressions and managed along with other algebraic constraints. However the algebraic constraint manager cannot handle the discontinuity and the multiple values that grid constraints require. These grid-relations should be incorporated into CoDraw's general constraint management scheme.

When an element or element class is assigned a grid-relation, both the grid and the relation are stored with the element or element class, in special 'snap-to-grid' variable and a special relation named 'grid-relation'. When the element is placed, sized, or moved in the Work Sheet, if the element is over the grid, then the 'grid-relation' is used to calculate a rectified position.

The arithmetic for grid-relation behavior is simple. Think of a grid as of a set of modules, in both horizontal and vertical dimensions; each module contains one set of bands. Figure 21 illustrates the concept of module in one dimension of a simple tartan grid.

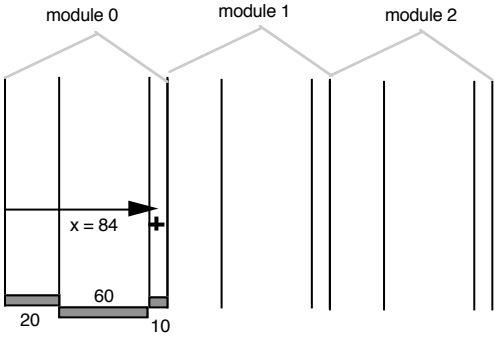

Figure 21. Horizontal modules of a tartan grid.

The function 'totals' converts the sequence of spacing units (HSeq) to a sequence of running totals (HCoord) giving coordinate values:

```
HCoord <- totals ( HSeq )
; (20 80 90) <- totals (20 60 10)
```

The width of the module is the last HCoord value, the sum of spacings in the module:

```
X_module_width <- last (HCoord)
; 90 <- last (20 80 90)
```

Then the function floor (modulo) is used to calculate the module number of the coordinate:

```
(module_number, remainder)
      <- floor (x, X_module_width)
```

and the remainder, or offset into the module is simultaneously computed. In this example, if x = 84, then module_number will be 0 and remainder will be 84. Finally, we use the Nearest function to find the coordinate of the nearest grid line in the module.

```
Nearest (remainder, HCoord)
; 80 = Nearest (84, (20 80 90))
```

4. Discussion.

4.1 Grids and placement rules as constraints.

Using grids and placement rules, a designer can program a CAD system to enforce desired spatial relationships among building components and spaces. These spatial relationships are constraints on the design and they represent decisions that the designer makes about how to organize the building. The constraints do not prescribe or generate particular forms; rather they circumscribe or bound a space of alternative arrangements without specifying a solution. The constraints structure the manipulations that the designer can make by restricting the placement of pieces. Within these self-imposed constraints, the designer explores alternative layout designs. The constraints provide a means to program placement rules for elements and spaces, using grids as a basis for positioning.

To use the tool effectively requires a discipline and an understanding of this design method. The use of grids as positioning devices for layout design has been discussed by N. John Habraken, in a number of publications (Habraken et al. 1976), (Habraken 1980), (Habraken and Gross 1988) and through a series of "thematic design" workshops at MIT. Once the approach is understood, the tool can be an effective way to organize spatial decision-making.

Grids, as a kind of constraint on the placement of elements in a layout, are a way of embedding knowledge in the design environment. By programming the behavior of element classes into the layout editor, the designer no longer must check the design against the placement rules.

Another, rather different, way to represent knowledge formally about layouts is a shape grammar. A shape grammar is a generative system, in which regularities of a family of shapes are expressed as of a set of production rules (Flemming 1987). The production rules generate a constrained set of possible shape arrangements. The set can be generated by exercising all legal sequences of the rules,which may be infinite. Our grid-relationships also constrain a set of physical arrangements. However, unlike a grammar, they do not suggest an order of form-generation.

The Grid Manager program was built as a module of CoDraw, a constraint-based drawing program that maintains design relations that the designer asserts. CoDraw's goal is to demonstrate a flexible and interactive graphically oriented constraint-based construction kit, within which designers can define and work within their own rules, or constraints. Other constraint based drawing kits have also been developed (e.g. Nelson 1985) and constraint-based programming environments are an active area of research related to Computer Assisted Design (see for example, (Sapossnek 1989; Murtagh and Shimura 1990)). Interactive grid snap has been proposed as a means to achieve some of the functionality of a more general constraint-based design system with less computational overhead (Bier and Stone 1986) and this "snap-dragging" approach has also been extended to three-dimensions (Bier 1990). However implemented, grid-snap is surely a kind of spatial constraint that a CAD program can be programmed to understand, and it is useful in layout design.

4.2 Grid based design rules can mediate group work.

The use of grids and placement rules is particularly suited as a means to mediate group design work. Layout of different subsystems can be divided among members of a design team and each team member can work relatively independently. Placement rules allow each designer to know where to put elements, and where to expect other designers to place elements of different subsystems. Each designer on the team can explore alternative layouts, knowing that when the time comes to integrate the subsystem designs, the grid-based positioning rules ensure that interference problems are limited and controlled. If plumbing elements are always found in band "alpha" and HVAC ducting in band "beta," then plumbing and HVAC can only interfere in an intersection between the two bands. Knowing the location, we can define interface conditions for the limited combinations that occur. The publications of the S.A.R. (Habraken et al. 1976), the OBOM, and the Dutch housing design standards (NEN 2883, 1981) outline these methods in detail. They argue that by regularizing the positioning of elements of different subsystems according to a grid, the designer can render the building quicker to build, easier to maintain, and more flexible for remodeling.

4.3 Conclusion.

Grids can help a designer organize the placement and dimension of building elements and spaces. By associating different grid positionings for each class of element, the designer can control relationships among elements. Some architects may not favor this method of designing —□r some may favor it for some projects but not others. It is a formal approach, in which the designer makes and records systematic placement rules. It thus places a value on explicit formulation of design rules. Design of the configuration of grids, and

specification of placement rules can become an important preliminary phase of the layout process. The advantage is that once the designer makes decisions about placement rules, the system helps organize the layout.

CoDraw's Grid Manager program supports the use of grids as a design tool. In CoDraw, grids are regular elements of the design just as any other graphical element. The Grid Manager enables the designer to first define and then work within simple positioning rules with respect to grids. Within these self-imposed constraints on element positions, a designer can freely design, exploring alternate layouts.

References.

Bier, E. 1990. Snap-dragging in three dimensions. Computer Graphics 24 (2) : 193-204.

Bier, E. A., and M. C. Stone. 1986. Snap-Dragging. Computer Graphics (SIGGRAPH 86) 20 (4) : 233-240.

Flemming, U. 1987. The role of shape grammars in the analysis and creation of designs. In *The Computability of Design*. Edited by Y. Kalay. New York: Wiley.

Gross, M. 1990. Relational Modeling. In *The Electronic Design Studio*. Edited by M. McCullough, W. Mitchell and P. Purcell. Cambridge, MA: MIT Press.

Habraken, N. J. 1980. The Grunsfeld Variations. Cambridge, MA: MIT School of Architecture and Planning.

Habraken, N.J., J.T. Boekholt, A.P. Thijssen, and P. Dinjens. 1976. *Variations - The Systematic Design of Supports*. Cambridge, MA: MIT Press.

Habraken, N. John, and M. D. Gross. 1988. Concept Design Games. Design Studies 9 (3)

Kroll, Lucien. 1987. *An Architecture of Complexity*. Cambridge, MA: MIT Press.

Murtagh, N., and M. Shimura. 1990. Parametric Engineering Design using Constraint Based Reasoning. AAAI-90. AAAI Press/Addison Wesley (Boston, MA)

NEN 2883 Project Group Modular Coordination. 1981. PLANS and DETAILS according to NEN 2883 - Manuals on the application of NEN 2883. TH Delft.

Nelson, G. 1985. Juno—A Constraint-based Graphics System. Computer Graphics 19 (3):235-243.

Sapossnek, M. 1989. Research on Constraint-Based Design Systems. Proc. 4th Intl. Conf. Applications of AI in Engineering. (Cambridge, England)