Human-Computer Interaction -- INTERACT'03
M. Rauterberg et al. (Eds.)
Published by IOS Press, (c) IFIP, 2003, pp. 860-863

A Design System based on Architectural Representations.

Sviataslau Pranovich, Jarke J. van Wijk
Technische Universiteit Eindhoven, Netherlands
s.pranovich@tue.nl; vanwijk@win.tue.nl

Abstract: A wide variety of drawing packages is available for architectural design. However, most of these systems are oriented to the production of final technical drawings, and only few support the early phase of design. In this paper, we present a new approach for a design system to support the early design phase. The method is based on a framework from architecture on the meaning of 2D drawings in architectural design. In such drawings, not only standard graphics elements like contours, but also elements like grids and axes of symmetry play an important role. These are encountered also in standard drawing systems, but often only as tools. We propose to consider such elements as graphical objects. Relations between these objects can be defined to capture higher-level information on the structure of the design. These relations are used for the propagation of the geometric transformations on objects. Additionally, we offer a natural user interface for the designer, which enables them to explore design space effectively and efficiently.

Keywords: user interface, design support

1 Introduction

Current drawing systems have reached a high level of sophistication, and are suited for the production of the final technical drawings in the final phases of the design process. However, they do not offer support for the early phase of the design process when concept formation is important. Such drawing systems require designers to specify many details in the drawing, while the designer does not care about them in this stage of the design process. Moreover, detailed drawing restricts the design creativity, whereas a system that supports early design should support and stimulate the generation of new ideas.

In order to develop a system that supports architects in an architectural fashion, it is necessary to look at the way architects work in the early phase of design. Research in the use of drawings by architects (Achten, 1997) leads to the view that architects use well-defined forms of graphic representations (graphic units) to depict their design intentions. Examples of these graphic units are contour, grid, circulation scheme, schematic subdivision, zone, etc. We use the results of Achten as a basis for our approach and propose to use these graphic units as the basic building blocks.

Current drawing systems also offer this functionality, but in the form of tools: to mirror objects, to align objects with respect to each other, to align them to a grid, etc. These tools have a limited scope: for instance, the fact that two objects are mirrored copies of each other is not stored explicitly; after a mirroring tool has been applied, higher-level knowledge (relations between objects) disappears.

In our approach we propose to represent tools as geometric objects (presented in the form of graphic units) themselves: for instance, to introduce a symmetry axis as an object that has a graphic representation, can be manipulated, and influences other objects. Graphic units in this case allow establishing permanent relations between them, which is not possible when tools are used instead.

2 Background

Several directions have been pursued to support designers in the early phase of design. One direction is to attempt to bring a drawing tool closer to a sketching tool, another direction is to offer more support for conceptual information, for instance on relations between objects.

Sketching tools like SmartSketch provide *beautification* (SmartSketch, 2003). The designer can sketch free hand, the systems attempts to recognize common graphic elements from this input. The Pegasus system introduces *predictive drawing*

that predicts the user's next drawing operation based on the existing drawing (Igarashi, 1998). But in general, systems supporting freehand sketching with beautification techniques still suffer from a lot of limitations (Plimmer et al, 2002).

The other direction for design support aims at enabling the user to enter and use higher-level information. As a first step, many drawing systems offer tools and aids, such as mirroring, alignment, grids, gravity, and snap-dragging. With these tools users can establish relationships between objects, but unfortunately most of the systems forget these relationships after the positioning operation is complete (Gleicher, 1992). Another well-known support aid is grouping. Objects are merged, possibly recursively, and can be manipulated as a group.

Constraint techniques make a powerful addition to the interaction techniques available in graphical editors. Many other systems have been developed that provide constraint-based solutions for graphical applications, such as Garnet, Unidraw, ArtKit, and Inventor (Gleicher, 1993). In fact, the success of constraint-based approaches to drawing has been limited by difficulty in creating constraints, solving them, and presenting them to users.

In our work we do not improve existing approaches for design support, but we offer a new approach, which is based on the designer's view on drawings.

3 Approach

3.1 Overview

Research in the use of drawings by architects has led to the framework of Generic Representations: Architects use well-defined forms of graphic representations to depict their design intentions. These forms have been identified and described as graphic units. A *graphic unit* is a set of graphic elements that are organized in a specific way and that have a generally agreed upon meaning for the designer. Some examples are: contour, grid, functional symbols, circulation scheme, zone, etc. Graphic units can be considered as a medium to express the ideas in an architectural design (Achten, 1997). We propose to use them as building blocks for a design system, i.e. to treat them as geometrical objects that can be edited and manipulated.

A set of related graphic units defines a *generic representation* (Achten, 1997), therefore relations between graphic units in our approach play an important role. The user defines a design in terms of graphic units and relations between them (Pranovich et al, 2002a). We use the idea of data-flow, for the propagation of manipulations on graphic units. Manipulations are propagated through a graph, where the nodes are graphic units and edges are relations between them. In summary, we aim at enabling the designer to use graphics units with an architectural meaning, such that these graphic units have intuitive behaviour and respond to the user actions in meaningful and predictive way.

3.2 Interaction elements

Every graphic unit has a special meaning for the architect. Below we give a description of the main graphic units in our system and their features.

Contour: the most encountered graphic unit in the drawing of the architect. It is the basic unit to construct the design. A contour is visualized as a polyline.

Grid: the alignment frame for the elements, which structures the design. About 70-75% of grids that the architect uses in a design process are orthogonal. We have defined grids as two sets of not necessarily orthogonal lines. Each set is called a grid component and is visualized as a set of parallel lines. By combination of grids more complex grids, such as the tartan grid can be defined.

Axial system: presents a notion of symmetry between objects in the drawing of the architects. In our design system the user can mirror an instance of a selected graphic unit by creating an axial system. An axial system keeps the symmetry between twin graphic units by transferring mirrored geometrical transformations between twins graphic units. An axial system is visualized as a dashed line.

Zone: Zones structure the design. It presents a general characteristic for the set of objects, which geometrically belong to some area. For instance, to define the space that is related to water (kitchen, toilet, etc.) architects use a wet zone. A zone is visualized as a semitransparent filled polygon. All objects that are covered by a zone are related to this zone.

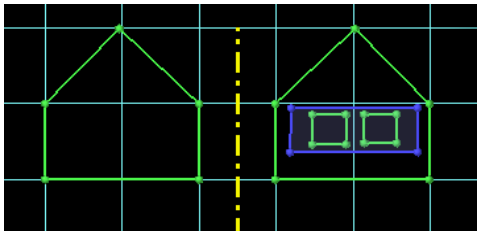

Figure 1: Axial system, grid, zone, and contours.

Image: for inspiration in the design process architects often use images. The designer can place different images on his workspace in order to trace

elements and to draw on top of them, or he can place them in the drawing as illustrations.

A user can create, edit, and delete graphic units: instances of predefined types of graphic units. Each graphic unit has a visual representation, and can be geometrically transformed. For this we developed geometric transformation tool, the KITE manipulator (Pranovich et al, 2002b). Standard geometric transformations (translation, rotation, scaling), and also skewing are supported.

3.3 Geometrical engine

Separate decisions can be made how the transformations are dealt with when they are propagated along the graph of graphic units. The user can define which types of transformations have to be passed and/or applied for each graphic unit and relation.

For the manipulations of connected graphic units we use a special geometrical engine. The engine is based on the propagation of geometrical transformations between anchor points using the relations between them, where the anchor points define the origins for the local transformations of associated graphic units.

If an anchor point is associated to a graphic unit, then it has properties, that tell which types of transformations have to be passed to this graphic unit (e.g. is the graphic unit scalable, translatable, etc.). The user can define relations between anchor points along which transformations are propagated. Relations have also associated properties that record which types of transformations must be passed or blocked. Every graphic unit has at least one anchor point associated .

The propagation of a transformation starts at a point selected by the user. Next the transformation is propagated through the anchor points and relations, affecting the graphic units that are associated to the anchor points. Anchor points and relations can be added and changed by direct manipulation.

The geometrical engine uses the graph of anchor points as a transformable skeleton for the propagation of manipulations between graphic units.

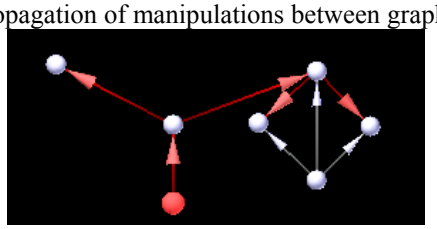

Figure 2. The propagation of a transformation in the graph of anchor points.

The variety in properties of relations and anchor points assist to achieve a large range of geometrical functionality in the manipulations of graphic units. In other words, the designer can specify the structure of his design explicitly, which enables him to explore different realizations efficiently.

4 User interface

The system offers many options to define the relations between graphic units. But having too many options sometimes is not productive since the user has to manage a complicated user interface (relations and anchor points sometimes become intricate). In order to solve this problem we provide an extra user interface (skin) on top of the interface of the geometrical engine. This extra user interface uses a natural interaction technique for architects and hides relations, anchor points, and their properties from the user. The system creates and deletes relations and anchor points without the need for explicit actions of the user. As a starting point for this user interface we use a well-known design metaphor, which is called "paper and scissors": the designer is experimenting by means of constructing and assembling different objects from paper, placing them on top of each other and manipulating them.

In order to apply this concept in our system we make one assumption: Every graphic unit has a depth that can be changed by the user. Using the depth and layout geometry the system extracts information about relations between graphic units. We look at an example (see Figure 3): we put one sheet of paper *B* on top of the other *A*.

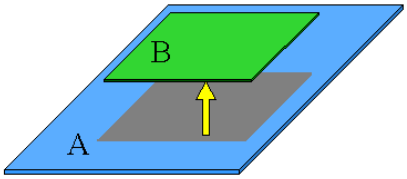

Figure 3: Example with two sheets of paper.

If we manipulate sheet A, then sheet B will be affected also, because the user implicitly implies a relation $A \rightarrow B$ between two sheets of paper. The system is capable to reconstruct (establish and delete) such relations between graphic units using following criterion: If graphic units have an overlapping area, the relation between them is created (from the bottom one to the top one).

Moreover, the idea of auxiliary local origins (or anchor points) is implemented in the new interface as a pin (the equivalent of the paper-pin). Using pins the user can connect graphic units and block propagation of particular transformations. The user can modify the pin by switching on/off the blocking of transformations.

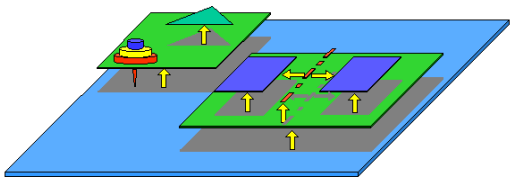

Figure 4: The arrows visualize the relations between graphic units that are extracted by the system. The pin, which blocks all transformations, visualized as a nail-head pyramid, where each nail-head has own colour and blocks a particular transformation.

The user also can use a clip (the equivalent of a paper-clip), which simply is an equivalent of a bi-directional relation between graphic units.

5 Results

We have implemented a prototype of the system and tested it with a few architects (see Figure 5).

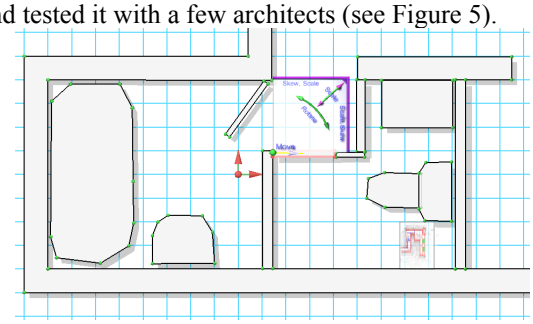

Figure 5: A design example made by an architect on the preliminary design phase.

Despite that the current implementation of the prototype is not perfect yet, architects find it useful and inspiring for the creation of new ideas during the design process.

The system is suitable especially for the preliminary design phase; the phase that follows a sketch design phase in the early design stage.

Furthermore, architects like that objects can be manipulated effortlessly (what is provided by the functionality of the geometrical engine, see Figure 6), and the familiarity of the interaction style.

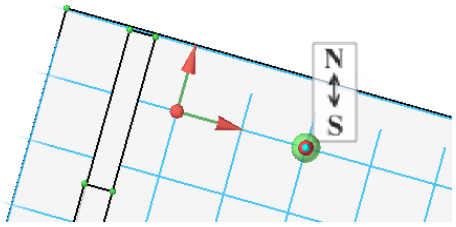

Figure 6: The Pin is used to stick the "North-South" sign to a grid and to freeze its orientation.

Summarising, the designer in our system can specify the structure of his design implicitly, focusing more on the design process rather than on the interaction with the system.

6 Conclusion

In this paper we introduced a new approach for a design system for architects, based on a framework of architect drawings.

We have implemented the prototype of the system and discussed it with architects. The system supports and stimulates the generation of new ideas, which is very important on the early stages of design.

At the moment we are exploring directions for improving our system. New functionalities can be added, such as new types of graphic units. It is not so difficult to offer many options and much flexibility, the main challenge however is to tune the interaction techniques and settle the visual metaphor on top of the system in such a way that the user can define what he wants in an intuitive way.

References

Achten, H.H. (1997), Generic representations - Knowledge representation for architectural design. In: *Journal of Architectural Management 13.*

Gleicher M. (1992), Integrating Constraints and Direct Manipulation. *Proceedings of the 1992 Symposium on Interactive 3D graphics,* p 171-174.

Gleicher M. (1993), A Graphics Toolkit Based on Differential Constraints. *{ACM} Symposium on User Interface Software and Technology,* p. 109-120.

Igarashi T., Matsuoka S., Kawachiya S., Tanaka H. (1998) Pegasus: A Drawing System for Rapid Geometric Design. *CHI'98*, pp.24-25.

Plimmer B., Apperley M. (2002), Computer-Aided Sketching to Capture Preliminary Design, *Third Australasian User Interface Conference (AUIC2002).*

Pranovich, S., Achten, H., J. J. van Wijk (2002a), Towards an architectural design system based on generic representations*, Artificial Intelligence in design'02,* p. 153-164.

Pranovich, S., J. J. van Wijk, C.W.A.M. van Overveld (2002b), The Kite geometry manipulator. *Extended abstracts CHI2002*, vol. 4, issue 1, p. 764-765.

Smartsketch (2003),web-site,http://www.smartsketch.com.