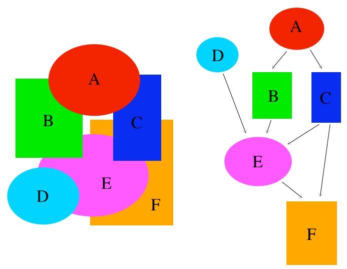
Representation of Interwoven Surfaces in $2\frac{1}{2}$D Drawing

Keith Wiley, Lance Williams, Department of Computer Science, University of New Mexico

Introduction

Existing drawing programs represent drawings as a set of surfaces that reside in a series of distinct layers. These layers impose a directed acyclic graph on the surfaces such that no two surfaces can interweave one another (right).



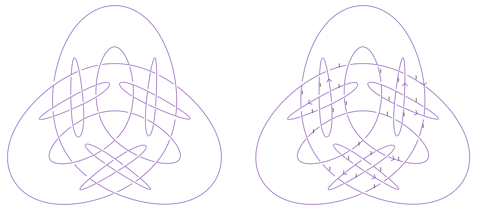
The main problem exhibited by existing drawing programs is that they use a very restrictive representation for overlapping surfaces. Our research uses a more general representation as the basis for a more powerful drawing tool, called *Druid*.

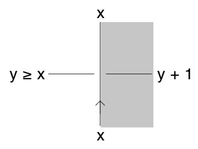
Crucial user interactions include:

- Create a new boundary
- Delete a boundary
- Smoothly reshape a boundary
- Drag a surface (drag all of its boundaries)
- Add or remove spline control points
- Increase or decrease spline degree (less crucial, but helpful)
- Flip a boundary's sign of occlusion
- Flip the layering of two overlapping surface regions

Labeling Scheme

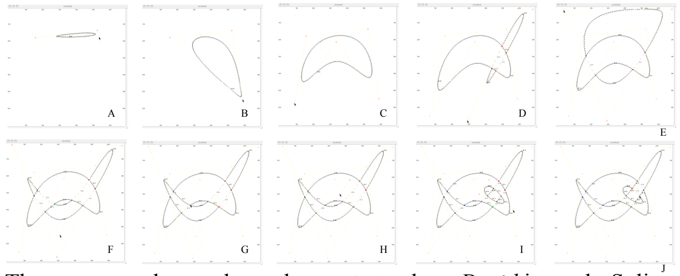
Our system, *Druid*, represents a scene of surfaces as a *labeled knot diagram* [1]. A knot diagram is a projection of a set of closed curves onto a plane and indicates which curve is above whereever two curves intersect, (below left). We extend ordinary knot diagrams to include a *sign of occlusion* for every boundary and a *depth index* for every boundary segment (below middle).





The labeling scheme is a set of local constraints on the relative depths of the four boundary segments that meet at an intersection (above right). The sign of occlusion of a boundary is denoted by an arrow, and designates a bounded surface to its right.

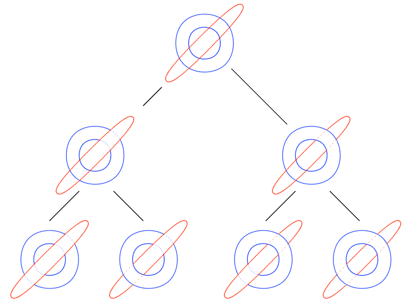
Demonstration of *Druid*



The sequence shown above demonstrates how *Druid* is used. Spline control points are defined in either a clockwise order to create solids (*A, B, C, D, E, F*) or in a counter-clockwise order to create holes (H, *I*). Intersections are clicked to flip overlapping regions (*G, J*).

Finding a Legal Labeling

Given an unlabeled drawing, there exists a set of possible labelings. Each labeling can be reached by traversing a labeling tree from root to leaf (right).



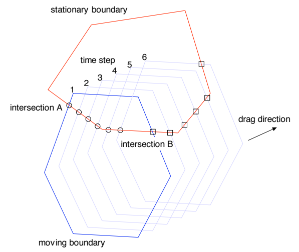
Legal labelings are found by an iterative deepening tree search.

Labeling Preserving Interactions

It is important to preserve the labeling during changes whenever possible.

Doing this precludes the naive method of deletion and rediscovery of intersections, which would necessitate relabeling for every interaction.

For some interactions, *e.g.*, drags and reshapes which do not alter the topology, the labeling can be preserved by projecting intersections along the paths they follow on the boundaries (right).



Interactions Requiring Relabeling

However, some interactions change the topology, requiring a search for a new legal labeling:

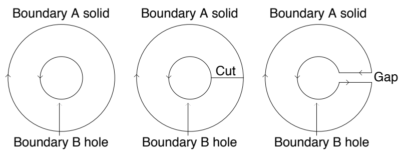
- Drags or reshapes that create new intersections
- Drags or reshapes that delete existing intersections
- Drags or reshapes that change the order of intersections
- Change of crossing state of intersection
- Change of sign occlusion of boundary

Druid strives to find the *minimum difference labeling* because this generally reflects the user's intent. The primary challenge of our research has been devising an algorithm that can find the minimum difference labeling acceptably fast. This can be difficult because the search space can be extremely large relative to the complexity of the drawing.

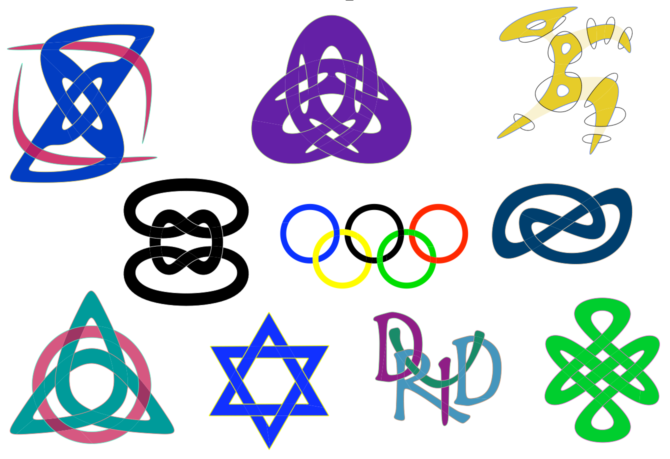
Boundary Grouping with Cuts

Boundary groups are not required for *Druid* to legally label a drawing. However, groups can provide a basis for translation of a surface with multiple boundaries and for eliminating ambiguities about which surfaces boundaries belong to.

We find and maintain groups by searching for *cuts*. A cut is a scissor cut through a surface connecting two boundaries that are part of that surface. A cut converts two boundaries of a surface into a single boundary (above). The discovery of a cut is used to create a boundary group.



Examples



Conclusions

Druid uses a novel surface representation which makes it possible to represent a more general class of drawings than is possible with existing programs. Because its user interactions are based on the natural affordances [2] of this more general class, *Druid* minimizes the effort required to produce drawings.

References

[1] Lance R. Williams. *Perceptual Completion of Occluded Surfaces*. Ph.D. dissertation. Univ. of Massachusetts at Amherst, 1994.

[2] Donald A. Norman. *The Design of Everyday Things*. Basic Books. 2002.