Druid

Representation of Interwoven Surfaces in 2½D Drawing

Keith Wiley
Thesis advisor: Lance R. Williams

University of New Mexico
Department of Computer Science
Albuquerque, NM 87131 USA
Interwoven 2½D Scenes
Introduction

Existing drawing programs:

- Use distinct layers
- Impose a DAG
- Do not permit interwoven surfaces

Our program, **Druid**, does not suffer from these limitations.
Existing Drawing Programs

Noninterwoven layers

Boolean combinations of boundaries, i.e., holes.

Do not span the full space of $2\frac{1}{2}D$ scenes.
Knots vs. Interwoven Surfaces
Interwoven Surfaces in Conventional Drawing Programs

1. Spoofs

2. Painting planarized graphs, e.g., Adobe Illustrator

3. Local DAG manipulation, e.g., MediaChance Real-Draw
Spoofs

A layered arrangement that produces the illusion of interwoven surfaces

(1) Copy from right annulus
(2) Paste
(3) Precisely position
(4) The spoof is brittle. If either annulus is moved, the spoof breaks.

Tedious to construct

Tedious to maintain
Adobe Illustrator Method

- Convert drawing to planar graph
- Paint faces of the graph independently
**MediaChance Real-Draw Pro-3**

*Push-back tool*: The user can push the top layer down (figures left)

- Insufficient for transparent surfaces
- Cannot represent self-overlapping surfaces (figure below)

The right annulus is pushed down
Feasability is not the sole issue. Convenience and naturalness are also issues.

Affordances: The set of interactions that a physical object suggests for itself (Norman ‘02).

Unlike conventional drawing programs, Druid’s affordances are isomorphic to those of idealized physical surfaces.

The user’s experience is of interacting with surfaces, not pictures of surfaces.

Druid’s Representation

Knot-diagram:
A projection of closed curves indicating which curve is above where two cross

Labeled knot-diagram (Williams ‘94):
Sign of occlusion for every boundary (arrows)
Depth index for every boundary segment

Williams, L. R., Perceptual Completion of Occluded Surfaces, PhD dissertation, Univ. of Massachusetts at Amherst, Amherst, MA, 1994.
Labeling Scheme

Imposes local constraints on the four boundary segment depths at a crossing

$x, y$: boundary segment depths

Legal labeling: A labeling in which every crossing satisfies the labeling scheme (Williams ‘94)

Williams, L. R., Perceptual Completion of Occluded Surfaces, PhD dissertation, Univ. of Massachusetts at Amherst, Amherst, MA, 1994.
Labeling Scheme Justification

\[ y \geq x \]

\[ y + 1 \]
The Crossing-Flip Interaction
Drawing Program Interactions

- Create & delete boundaries
- Reshape & drag boundaries
- Crossing flip (Invert two surfaces’ relative depths in an area of overlap)
- Sign-of-occlusion flip
Effects of Interactions on the Labeling

Requiring relabeling (topological change)
- Creation & deletion of crossings
- Reordering of crossings around boundaries
- Crossing-state flips
- Sign-of-occlusion flips

Not requiring relabeling (no topological change)
- Reshaping or dragging boundaries without causing topological changes
Important to preserve crossing-states

Naive destruction/rediscovery of crossings would lose crossing-states

*Druid* projects crossings as they move around boundaries
Demonstration of *Druid*

*Druid* knows to move both boundaries at once.

*Druid* relabels when the interlock breaks.
Finding a Legal Labeling

Labeling space: Possible labelings for a labeled knot-diagram. Labeling space size: $2^C$

Druid maintains a legal labeling automatically.
Druid searches the labeling space for the minimum-difference labeling.

Labeling is currently in state B.
User clicks the blue-circle marked crossing.
C and D are possible solutions, C is minimum difference from B.
The Labeling Search

- Branch-and-bound
- Constraint propagation
- Iterative deepening
- Timeouts
Branch-and-bound

- **Search goal:** *minimum difference labeling*

- Node expansion can never decrease the accumulated labeling difference

- Minimum difference legal solution gives the bound

- Search is truncated when the accumulated current difference exceeds the bound
Constraint Propagation (Waltz ‘75)

- Orders the search so that legal solutions are found earlier

- Legal solutions define bounds

- Constraint propagation works in concert with branch-and-bound to increase search efficiency

Iterative Deepening

- Branch-and-bound works best if good solutions are found earlier.

- In good solutions, changes are localized to the *area of interest*.

- Search is restarted with increasing *search horizons*. 
The search can take too long

Two timeouts:

- **Very short timeout (0.1 sec):** If a solution has been found during the search
- **Longer timeout (5.0 sec):** If no solution has been found yet
Measuring Drawing Complexity

- Total number of crossings
- Maximum depth
Experiments: Two Labeling Methods

- Randomized labeling
- Incremental labeling
Number of crossings: linear in the number of surfaces

Max depth: constant
Test 1: Labeling Time vs. # Crossings

Running time vs. # Crossings

Running Time vs. # Crossings (Incremental Labeling)

Running time vs. # Crossings (log Y axis)
Number of crossings: quadratic in the number of surfaces

Max depth: linear in the number of surfaces
Test 2: Labeling Time vs. # Crossings

Running time vs. Number of Crossings

Running Time vs. Number of Crossings (Incremental Labeling)

Running time vs. Number of Crossings (log Y axis)
Test 2: Labeling Time vs. Max Depth

Running time vs. Max Depth

![Graph 1](image1)

![Graph 2](image2)

![Graph 3](image3)

Running time vs. Max Depth (log Y axis)
Boundary Grouping with Cuts

- Some surfaces have multiple boundaries
- This can cause problems
- A cut between two different boundaries reduces the number of boundaries by one

Cuts are a geometric device. Needn’t be horizontal or straight.
Cut Labeling Schemes

Using cuts requires four new labeling schemes

1. A boundary crosses a cut with the boundary above
2. A boundary crosses a cut with the boundary below
3. A cut crosses a cut
4. A cut ends and attaches to a boundary, i.e., a T-junction

Cuts denoted with a double line (top row) and a gap (bottom row)
Finding Legal Cuts

A successful cut: Last crossing (e) is legal.

An unsuccessful cut: Last crossing (d) is illegal.
Conversion of a labeled knot-diagram to an image with solid fills

Requires full depth ordering of all surfaces covering each region

*Druid* uses the *episcotister model*  
(Metelli ‘74)

---

Slice

- A slice connects a location on a boundary to a point within the bounded surface.
- Similar to a cut.

Slices are a geometric device. Needn’t be horizontal or straight.
Red is above green, which is above blue
Druid Examples
A Problem with the Search

- Search space size: 2 for C crossings
- A drawing can have hundreds of crossings.
- The search takes too long for complex drawings.
- Thus, *Druid* as described in (Wiley and Williams ‘06a) was limited.

Druid fails to label this flip in under 120 seconds in 50% of tests.

Druid takes 35 seconds on average to perform one of these flips (and fails in 2% of tests).
Discovered a property of 2½D scenes, the crossing-state equivalence class (CSEC) rule

Use this property to improve performance
\textit{Area of overlap}: The maximum contiguous area where two surfaces overlap, e.g., the shaded area for surfaces 1 and 2.

\textit{Corner}: A crossing where a traversal of an area of overlap’s border switches boundaries, e.g., the \textcolor{blue}{blue diamonds} for the shaded area.
Crossing-State Equivalence Class (CSEC)

The corners of an area of overlap

Unique shapes/colors indicate CSECs
Finding CSECs on Labeled Figures

Intend to use CSECs to improve performance

But *Druid* must **find** the CSECs before they can be used

How long does this take? Does it cancel the benefit of using CSECs in the first place?
Finding CSECs on Labeled Figures

Experiment: Across a spectrum of CSEC sizes, measure the time required to find all CSECs.

In this experiment there is only one CSEC.
Running time to find CSECs for these figures is polynomial in the number of crossings.

Note: The actual time is very low (.3 secs for 52 crossings)
Crossing-State Equivalence Class Rule

All members of a crossing-state equivalence class must be in the same state.

e.g., for surfaces 2 and 3 all corners of the green circle CSEC must be in the same state, i.e., either 2 is above 3 or vs/va.
Two Relabeling Methods


2. *Druid* (NEW): Maintains the CSECs without a search. Deduces resulting segment depth changes directly (Wiley and Williams ‘06b).


Results: A Small CSEC Flip

- Size 4, indicated with circles
- Running times on 1.6GHz G5 PowerMac
- \textit{Druid (NEW)} performs 85 times faster than \textit{Druid (OLD)}

Min, mean, max with respect to a crossing-flip performed independently on each corner
Results: A Large CSEC Flip

- Size 16, indicated with circles
- *Druid (OLD)* cannot relabel in a reasonable time
- *Druid (NEW)* performs 967 times faster
- Note: *Druid (OLD)* failed 50% of the time

Min, mean, max with respect to a crossing-flip performed independently on each corner
Results: A Complex Figure

- 256 crossings, 64 CSECs
- **Druid (OLD)** cannot relabel this small CSEC flip in a reasonable time
- **Druid (NEW)** relabels in .02 seconds, 1900 times faster
- Note: **Druid (OLD)** failed 2% of the time
CSEC Flip Performance

Flipped CSEC size: linear in the total number of crossings
CSEC Flip Performance

Running time vs. CSEC size

1. Flipped CSEC size (also total number of crossings)

Performance is polynomial w.r.t. CSEC size

2. CSEC Flip Performance (Druid (NEW)) (log Y axis)

3. CSEC Flip Performance (Druid (NEW))

Performance is exponential w.r.t. CSEC size

4. Search Performance (Druid (OLD)) (log Y axis)

5. Search Performance (Druid (OLD))
Future Work

- Labeling with CSECs
- Locking and kinematic interactions
- Occluding contours and pita surfaces
Future Work: Labeling with CSECs

- CSECs have a profound effect on the search space size.
- e.g., this drawing has 40 crossings but only 7 CSECs, an improvement by a factor of \(2^{33}\) or 8.5 billion.

<table>
<thead>
<tr>
<th>CSECs Used</th>
<th>Crossing-state search space size</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>(2^{40}) (for 40 crossings)</td>
</tr>
<tr>
<td>Yes</td>
<td>(2^7) (for 7 CSECs)</td>
</tr>
</tbody>
</table>
Currently, can only find CSECs on legally labeled figures.

Cannot use CSECs to *label*, only to *relabel*.

Labeling must search the naive search space $2^C$, not the improved search space $2^E$.

Having the CSECs for an unlabeled figure would greatly assist the labeling search.
Future Work: Locking Interactions

Original

Broken

Locked
Locking and Kinematic Interactions

Original

Broken

Locked
Future Work: Occluding Contours and Pita Surfaces

An *occluding contour* is the projection of a fold.
Occluding Contours: Examples

Occluding contours enable construction of cylinders and Mobius strips.
Occluding Contours: Pita Surfaces

Occluding contours enable construction of *pita* surfaces.

---

pita surface

pita *containment*
Occluding Contour Labeling Schemes

Y-junctions

Occluding contour to boundary crossings

Occluding contour to cut crossings

Cusp V-junctions
Conclusions

- Developed *Druid*, a system for constructing interwoven 2½D scenes
- Use of branch-and-bound search to relabel gives the user the experience of interacting directly with idealized physical surfaces
- Search hinders *Druid*’s scalability
- Discovered a topological property of 2½D scenes, the crossing-state equivalence class rule
- Exploitation of this property can alleviate the need to search in some situations, and can dramatically reduce the search space in remaining situations
- Vastly extended the complexity of drawings that users of *Druid* can construct
Min. Acceptable Mouse Delta

Sequential dummy mouse locations (letters)

Sequential actual mouse locations (numbers)

After processing dummy loc a, actual loc remains 1.
After processing dummy loc b, actual loc has become 2.
After processing dummy loc c, actual loc remains 2.
After processing dummy loc d, actual loc has become 3.
After processing dummy loc e, actual loc has become 4.
After processing dummy loc f, actual loc remains 4.
After processing dummy loc f, actual loc 4 is within the min acceptable delta. The catch up phase is complete. Actual loc 4 is processed directly.

Actual loc starts at 0, and moves to 1. First dummy loc is created at a.

Minimum acceptable mouse delta

Projection rays used to calculate dummy mouse locations

Dummy mouse projections
CSEC Flip Performance

Flipped CSEC size:
constant (green)

Running time vs.
Total Number of Crossings

Red plot is the same plot shown on the previous slide
(seconds to perform the red CSEC flip)
Depth Sort vs. *Druid*

**Depth Sort:**
- Uses cuts to remove cycles and create a DAG.
- Renders by sorting polygons in 3D from back to front.

**Druid:**
- Uses cuts to group boundaries *not* to remove cycles.
- Makes weaker assumptions to render than required by depth sort – does not require DAG.


Scanline Algorithms vs. *Druid*

**Scanline algorithms:**
- Raster-based
- Method for rendering vector objects

**Druid:**
- Vector-based
- Relies on graphical API to render vector objects


Hidden Surface Removal vs. Druid

**Hidden surface removal:**
- Assumes opaque surfaces bounding solid objects

**Druid:**
- Assumes transparent fronto-parallel surfaces
- Opaque surfaces are a special case
