VideHoc: A Visualizer for Homogeneous Coordinates

Robert R. Lewis
bobl@cs.ubc.ca

University of British Columbia
Department of Computer Science

7 March, 1995

1 Introduction

VideHoc is an interactive graphical program that visualizes two-dimensional homogeneous coordinates. Users manipulate data in one of four views and all views are dynamically updated to reflect the change.

The program is a re-implementation of the program *Etch* developed by Snoeyink [snoe88]. Hanrahan [hanr84] describes a non-graphical tool with a similar purpose. XYZ (Nievergelt, et al. [niev91]) is a similar and in many ways more powerful tool, but without *VideHoc*'s intrinsic support for homogeneous coordinates.

VideHoc is intended to serve two roles: as an instructive tool for classes in computational geometry and as a research framework for testing new algorithms in computational geometry, particularly those exploiting homogeneous duality.

2 A *VideHoc* Overview

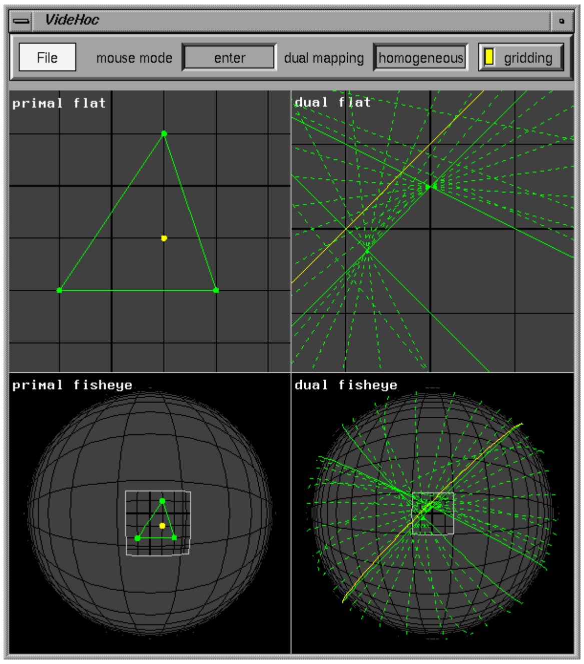

Figure 1: A typical *VideHoc* session

Figure 1 shows a typical *VideHoc* session. The canvas, or drawing area, shows the data the user has entered as elements: points, lines, chains (of line segments), and wedges (swept angles). The canvas can show several views: primal flat (Cartesian), primal fisheye (projection of the entire Cartesian plane onto a hemisphere), dual flat, and dual fisheye, either individually or all at the same time. When all views are shown, the same data appears in all of them: any change made to one window is reflected immediately in the other three.

The control bar above the canvas lets the user see and change *VideHoc*'s current state, including:

mouse mode This is what the mouse is currently being used for: **enter** for entering points and lines, **view** for altering the what the canvas is displaying, or **select** for selecting points and lines for use in more complicated commands.

dual mapping This defines the kind of dual mapping that takes place: (classical) homogeneous, Edelsbrunner, or Brown.

gridding Whether or not points and lines (the two points defining them, actually) entered are to be constrained to lie on integer grid points.

All canvas input is done with the mouse. The left button is for single clicks: enter a point, pan to a new center, select a nearby element, etc. The middle button is for "down-drag-up" operations: area-related or two-point commands such as: enter a line, zoom in or out around the center, or select points within a given rectangle. The right button brings up a pop-up menu of commands relevant to the mouse mode.

3 Commands

Non-trivial commands the user can invoke from the **select** mouse mode include:

sort ... brings up a dialog box that allows the user to sort the selected points by x, y, or their selection order. The points are labelled with their resulting sort order, which is maintained until something (like adding or deleting a point) occurs that could possibly change the order.

join if two lines are selected, adds their intersection point or, if two points are selected, adds the line that passes through them. The result is dynamic, so that if either or both of the two lines or points are moved later, the result gets moved accordingly.

chain constructs a line segment going through all selected points (in selection order). This chain is dynamic: it will change if any of the points along it get moved and will vanish if any of the points get deleted.

convex hull constructs the convex hull of the selected points. This hull is dynamic: it may change if any of the points that define it get moved so as to redefine the hull and it will go away if any of the points that it was defined over get deleted.

4 Implementation Lessons

Implementing *VideHoc* has been instructive in itself. The distinction between an algorithms's conceptual form and its robust implementation was abundantly clear on several occasions. It was challenging, for example, to make the Graham convex hull algorithm work with both for points at infinity and for multiple points with the same coordinates[1]!

[1] Since *VideHoc* allows gridding, this situation was easy to create.

5 Conclusions

The goals of *VideHoc* are to support computational geometry education and research. The purpose of this communication is, therefore, to release it and elicit a response from the computational geometry community.

VideHoc is far from complete. Possible improvements include on-line help, commands to save and restore a user's configuration, hard copy output (including, optionally, color), support for oriented projective geometry (as specified in Stolfi [stol91]), an "undo" command, an application language for constructing (and debugging) algorithms interactively, support for 3 (or more?) dimensions (possibly making use of stereo viewing), and ports of *VideHoc* to more generally-available platforms.

6 Availability

A Silicon Graphics executable version of *VideHoc* is available via ftp from the node `ftp.cs.ubc.ca` in the directory `/pub/local/bobl/VideHoc`. It is also available as a download via the UBC Imager Web (URL: `http://www.cs.ubc.ca/nest/imager/imager.html`).

References

[hanr84] Pat Hanrahan. "A Homogeneous Geometry Calculator". 3-D Technical Memo 7, NYIT, 1984.

[niev91] Jürg Nievergelt, Peter Schorn, Michele de Lorenzi, Christoph Ammann, and Adrian Brüngger. "XYZ: A project in experimental geometric computation". *Computational Geometry — Methods, Algorithms and Applications: Proc. Internat. Workshop Comput. Geom. CG '91*, Vol. 553 of *Lecture Notes in Computer Science*, pp. 171–186. Springer-Verlag, 1991.

[snoe88] Jack Snoeyink. "Etch: A Drawing Program That Illustrates Geometric Duality". unpublished article, Xerox PARC, Palo Alto, CA, 1988.

[stol91] Jorge Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations.* Academic Press, 1991.