GemmaCAD 2D – Constraint-based Interactive Drawing System

Franc Gerencer
franc.gerencer@uni-mb.si

Laboratory for Geometric Modelling and Multimedia Algorithms
Institute of Computer Science
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor / Slovenia

Abstract

In the paper, some details on implementation of our new constraint-based drawing system are highlighted. The system enables users to design in a plane in an elegant declarative way by employing dependencies among geometric elements like dimensions of distances and angles, instead of calculating absolute positions. The user specifies what to draw, and not how to draw it. In the background, the constructive constraint solver is employed to establish the geometry from given relations (constraints). The main step of the algorithm is pre-processing, which first transforms various geometric elements into corresponding control points and lines only, and all types of geometric constraints into the constraints of only five different types. After this, redundant constraints of distances and angles are added, and finally, the transformed constraint problem is solved in a simple way by local propagation. A wide variety of well-constrained problems with a few exceptions can be handled, over-constrained scenes and input data contradictory to some well-known mathematical theorems are detected, and the algorithm is proved successful in many under-constrained cases as well.

Keywords: CAD, constraint-based design, geometric constraints, geometric modelling.

1. INTRODUCTION

Conventional geometric modellers and drawing systems “think” in a procedural way. A computer program only substitutes a drawing board, but it does not offer to a designer any support by time-consuming and error-prone calculations of coordinates and dimensions. A computer is able to draw the line segment from the coordinate origin to the point (1, 1), but it does not “know” how to draw the same line segment if its length (

```
ERROR: rangecheck
OFFENDING COMMAND: .pdfshow

STACK:

{--show-- }
(H)
[0 ]
(H)
```