DecoSketch – Towards Calligraphic Approaches to Interior Design

Tiago Brito, Manuel J. Fonseca, Joaquim A Jorge
Departamento de Engenharia Informática
IST/UTL, LISBON, Portugal
http://immi.inesc-id.pt
tmtb@mega.ist.utl.pt,mjf@inesc.pt, jorgej@acm.org

Abstract. *Computer-Aided Design tools have long played an important role in architecture design. However, we need to go beyond direct manipulation to devise new tools that will expedite the interior design and decoration. Indeed, conventional CAD systems, while providing ever increasing functionality, do not provide equal support to the drafting and drawing tasks. This makes even the simplest drawings a complicated endeavor. Draftspeople struggle with different concepts that those learnt from their earlier days in school and have to think long and hard to translate familiar sequences of operations to commands which require navigating a dense jungle of menus.*

The term calligraphic interfaces was coined by us to designate a large family of applications organized around the drawing paradigm, using a digital stylus and a tablet-and-screen combination as seen most recently in Tablet PCs®. Using these, users can enter drawings in a natural manner, largely evocative of drafting techniques that were perfected for pencil-and-paper media. This paper presents a simple calligraphic interface to explore interior design literally from the ground up.

The Decosketch application is a modeling and visualization tool structured around 2 ½D architectural plants. Its purpose is to help architects or customers easily creating and navigating through house designs starting from the floorplan and moving to their three-dimensional representation. Moreover, both 2D and 3D representations can be independently edited, providing a natural interface that tries to adhere to well-known representations and idioms used by architects when drafting using pencil and paper.

Introduction

We are aware that many recent approaches have been brought to bear on this problem. Furthermore, the area of pen-based interfaces is one of the oldest researched in computer science, with pioneer works such as Ivan Sutherland's Sketchpad, dating back to 1964, paving the way to more recent approaches, most notably Gross and Do's Electronic Cocktail Napkin [Gross and Do 1996] and Pierre Leclerc's Esquisse [Juchmes and Leclerc 2004]. More recently, commercial products such as SketchUp [sketchup 2003] and Outline 3D [ouline 2005] have made significant inroads in terms both of usability and expressiveness. Even there have been significant advances in recent years, we feel that much remains to be explored.

Our work presents two main contributions. One is the direct input of architectural features using gestures and visual languages. Another is directly sketching on top of 3D drawings to add architectural fixtures, such as doors, windows and objects. Finally, our system allows users to add objects such as tables, desks, lamps, shades and sofas using a combination of gesturing and geometric-driven search in a database.

System Architecture

Figure 1 describes the architecture of our system. Next in this section we detail the functions and operation of the main functional blocks. As we can see, the system contains four main building blocks. The Graphics/Interaction engine supports stroke input and graphical output. The Primitive recognizer converts strokes to input primitives. The database contains modeler objects, while the modeling core assists users in creating layouts.

The Graphics/Interaction engine handles all interaction to/from the user. It contains two major building blocks. The first is the windowing interface and input processor. This supports context switching between major modes by using a task bar and interaction with each major mode, by creating virtual objects which describe a layout. Also this subsystem is responsible for collecting user strokes either by means of a tablet/stylus combination or via a mouse. This is the main input modality for the user to enter data to the rest of the application. The second module manages the virtual world as created by the user. This is accomplished via OpenGL and OpenInventor libraries for displaying and modeling three-dimensional entities but this module also handles interactive events and allows sketching on three-dimensional scenes. This module allows other interactive events received from the window manager to support navigation, moving the camera, selecting objects, etc.

The Primitive Recognizer comprises four modules which handle the different stages required to construct an architectural object or feature. First, the graphics recognition library (CALI) receives user strokes, which it then classifies as graphical tokens. Then it uses a visual grammar to parse sets of tokens as architectural features such as doors and windows. Finally, it identifies rooms or divisions from the set of walls created.

The Database module serves mostly as a repository for objects used in decoration. In the current version of our system, objects are retrieved from this database using direct manipulation. However, we envisage using more sophisticated approaches to specify objects by context and sketched hints rather than direct manipulation. Objects are stored in VRML format to allow simple importing and exporting of designs and objects.

The Modeling core contains two sub-modules. One is responsible for creating a scene graph containing both the two-dimensional and corresponding three-dimensional views of the plant. This module is responsible not only for handling visualization but also to ensure that the two different representations (plant and three dimensional model) are kept consistent after changes or modifications. That is, changes to the two-dimensional plant get reflected on the three-dimensional view

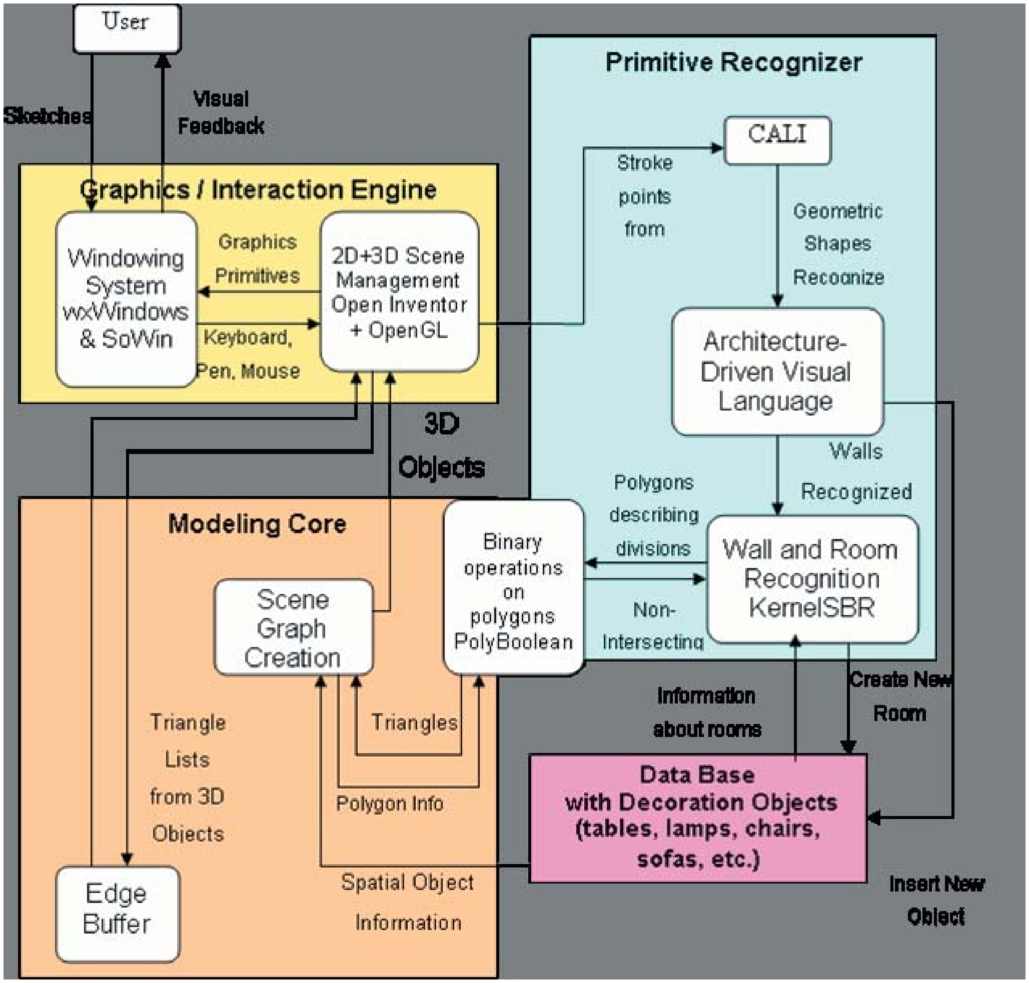
User
Sketches
Visual Feedback
Graphics / Interaction Engine
Windowing System wxWindows & SoWin
Graphics Primitives
Keyboard, Pen, Mouse
2D+3D Scene Management Open Inventor + OpenGL
3D Objects
Primitive Recognizer
CALI
Stroke points from
Geometric Shapes Recognize
Architecture-Driven Visual Language
Walls Recognized
Polygons describing divisions
Non-Intersecting
Wall and Room Recognition KernelSBR
Modeling Core
Binary operations on polygons PolyBoolean
Scene Graph Creation
Triangle Lists from 3D Objects
Triangles
Polygon Info
Spatial Object Information
Edge Buffer
Information about rooms
Create New Room
Data Base with Decoration Objects (tables, lamps, chairs, sofas, etc.)
Insert New Object

Figure 1: System Architecture

and vice-versa. Furthermore, the non-photorealistic renderer is responsible for managing non-traditional representations of the virtual world suitable for interactive illustration.

From 2D to 3D

One interesting feature of our system is that it allows creating a 2 ½ D representation of rooms after a valid layout has been introduced, using recognized wall and room information. Our method to identify walls and rooms uses a generic shape recognizer to parse gestures from user input, called CALI [Fonseca 2000]. This package recognizes individual geometric shapes, such as triangles, circles, lines, open/closed arc circles, ellipses, dashed/dotted lines, etc. We then use spatial relations (above, to-the-left-of, parallel-to, etc) to identify symbol groupings. Walls/Rooms and other symbols are identified separately by a specialized visual language, which uses connectivity information and a set of visual rules. Each rule uses an expectation list that is compared with the object identified by the sketch recognizer. Thus, each time the CALI identifies a sketch (as a circle, rectangle, etc.) the result is stored in a list that will be analyzed by the visual parser, to see if they satisfy any of the rules from the grammar. We call this a visual grammar, since productions take into account special relations in addition to syntactic categories of constituents. One such example is the window primitive which is input as two disjoint quasi-parallel line segments that individually cross the same wall at adjacent points, as illustrated in Figure 2. Objects identified by the grammar rules (walls, doors, windows, etc.) are then stored in a database of architectural elements. After identifying all the architectural elements of the plant, walls receive a special treatment to decompose and assemble them into polygons which define rooms or divisions. Most processing is done via a previously developed library, KernelSBR [Ferreira 2003]. Methods in this library compute polygons (forming divisions) from a set of line segments defining these by considering their intersections and adjacencies. We also use the functionality in another library [Polyboolean 2004] to merge and detect polygons that represent divisions.

Extrusion from two dimensions to 2 ½ dimensions is done on the basis of plant divisions, not on individual walls. For each division we compute its "inner polygon", considering wall thickness and then apply a similar method to extrude windows and doors. Figure 4 shows an example room in 3D, including a sofa, inflated from 2D. We use NPR techniques to highlight edges, via a modified version of Buchanan and Sousa's edge buffer

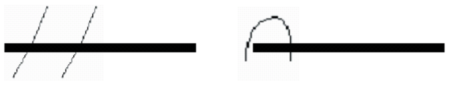

Figure 2 – Symbols describing windows and doors, respectively.

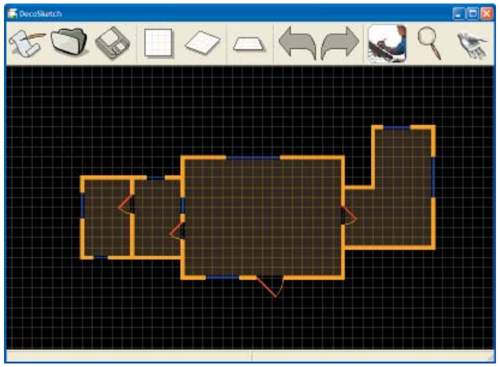

Figure 3. A screen shot of our interface

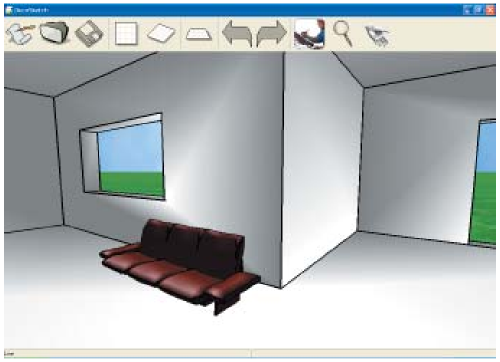

Figure 4: Room Layout in 3D

algorithm [BS 2000]. It is possible to add/remove windows, doors, etc. from the current layout interactively by sketching directly on the three dimensional representations of walls.

The great advantage of our approach lies in that we can almost entirely without menus enter room information and layout, in a marked departure from previous approaches.

Evaluation

We evaluated our application by asking a group of users to enter drawings and convert them to three dimensions using our program. The test group included ten people with diverse backgrounds ranging flow elementary school to university students with ages between 10 and 48 years. After a brief explanation (five minutes) of our system and its principles of operation, participants were asked to draw (using a tablet/stylus combo, placed on an horizontal surface), following by answering a questionnaire. Users were able to create simple layouts at an average rate of 48s per division, including drawing of doors, corridors and windows. We also tested the ability of users to create 2 ½ D layouts from plants as well as their skills at editing and navigating the three dimensional representations. While we noticed that navigation tasks were the least familiar to users, they still were able to perform edits on three-dimensional representations of walls and rooms (creating windows and doors), which were accomplished on 56 seconds on average. Users were also able to retrieve objects from a database (using direct manipulation and placing them inside a room) under 50 seconds on average, even though this is one of the least developed components of our interface.

Using geometric search to add interior objects

As mentioned earlier, our approach uses geometric search to add objects to a scene. Currently this approach is still in its infancy, but we believe that there is much promise in this method. Objects are added by sketching on walls, floor plans or other objects. Currently only a generic insert object gesture is recognized, but we plan to add more elements to the visual vocabulary to distinguish between tables, sofas, chairs, etc. We have developed a geometric search technique that supports querying by sketch as described in [Fonseca 2004] and [Fonseca 2005].

Conclusions

Calligraphic interfaces seem to represent a workable alternative for applications which are oriented to drawing and sketching spatial representations of objects, since their structure and affordances bring users closer to traditional paper-and-pencil methods. As can be seen from our test results, this translates to short learning times and rapid sketching of layouts. While the current implementation is still limited in functionality, we feel that it has the optional to scale up to realistic settings while retaining most of its simplicity. Indeed, while the implemented functionality only allows crating and editing rooms, doors, windows and include furniture from a limited database, studies are under way to enlarge the command vocabulary to allow users to change dimensions in doors/windows, enter circular windows and also to enter objects outside the floorplan with a minimum of additional gestures. Furthermore, we plan to improve the representation of objects via enhanced NPR techniques to bring the drawings closer to those created by architects and illustrators.

Acknowledgements

This work was supported in part by Portuguese Science Foundation Grant POSI/EIA/59938/2004 – Augmented Decoration: Combining Augmented Reality and 3D Object Retrieval for Interior Decoration.

References

[Alias] Maya modeling package http://www.alias-wavefront.com/
[AutoDesk1] AutoCad www.autocad.com
[AutoDesk2] AutoDesk http://www.discreet.com/
[BS 2000] John W. Buchanan and Mario C. Sousa. The edge buffer: A data structure for easy sil-

houette rendering. NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering, 2000, http://citeseer.ist.psu.edu/buchanan00edge.html

[Coin3D] http://www.coin3d.org, http://doc.coin3d.org/Coin/

[Ferreira 2003] Alfredo Ferreira, Manuel J. Fonseca, Joaquim A. Jorge, Polygon Detection from a Set of Lines, Proceedings of 12th Encontro Português de Computação Gráfica, Porto, Portugal, October 2003

[Fonseca 2000] Manuel J. Fonseca, http://immi.inesc.pt/projects/cali/

[Fonseca 2000b] Manuel J. Fonseca and Joaquim A. Jorge, Using Fuzzy Logic to Recognize Geometric Shapes Interactively, Proceedings of the 9th Int. Conference on Fuzzy Systems (FUZZ-IEEE 2000), San Antonio, USA, May 2000

[Fonseca 2004] Manuel Fonseca, Bruno Barroso, Pedro Ribeiro, e Joaquim A. Jorge, Retrieving Vector Graphics Using Sketches", SmartGraphics 2004 Proceedings, Springer LNCS vol. 3031, pp 66-76, 22-24 May 2004, Banff, Canada.

[Fonseca 2005], Manuel Fonseca, Alfredo Ferreira Jr. e J Jorge, Content-Based Retrieval of Technical Drawings", International Journal of Computer Applications in Technology (IJCAT) on "Models and methods for representing and processing shape semantics", March 2005

[Gross and Do 96] Mark Gross, Ellen Do, Demonstrating the Electronic Cocktail Napkin: a paper-like interface for early design, Proceedings of the International Conference on Human Factors in Computing Systems (CHI'96), 1996, pp 5-6.

[Jorge 2003] Joaquim A. Jorge, Manuel J. Fonseca and Alfredo Ferreira, Polygon Detection from a Set of Lines, Proceedings of 12º Encontro Português de Computação Gráfica (12th EPCG), pages 159-162, Porto, Portugal, Oct 2003 http://immi.inesc.pt/publication.php?publication_id=63

[Juchmes and Leclerc 2004] A Multi-Agent System for the Interpretation of Architectural Sketches, Roland Juchmes and Pierre Leclercq, Eurographics Workshop on Sketch-Based Interfaces and Modeling, Grenoble, September 2004.

[JW 1994] Josie Wernecke, The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor, Rel. 2, 1994,
http://www-evasion.imag.fr/Membres/Francois.Faure/doc/inventorMentor/sgi_html/

[MFC] Microsoft Foundation Classes http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_mfc_class_library_reference_introduction.asp

[Molich and Nielsen 1990] Rolf Molich, Jakob Nielsen, Improving a human-computer dialogue, 1990

[outline 2005] Outline 3D version 4.5, http://www.outline3d.com/ (Accessed in June 2005)

[PolyBoolean 2004] http://www.complex-a5.ru/polyboolean/index.html

[SGI-IV] OpenInventot SceneGraph management package http://oss.sgi.com/projects/inventor/

[SketchUp 2003] http://www.sketchup.com (accessed in March 2005)

[SoWin] http://doc.coin3d.org/SoWin/

[WNDS 1999] Mason Woo, Jackie Neider, Tom Davis, e Dave Shreiner; OpenGL Programming Guide, Third Edition, 1999, http://fly.cc.fer.hr/~unreal/theredbook/

[wxWindows] www.wxwindows.org