Modern architecture of light-weight CAD

Vladimir N. Malukh, Alexey G. Nickitin
ProPro Group,
Novosibirsk, Russia
vmalukh@propro.ru, nick@complex-a5.ru

Abstract

Paper describes the design of open architecture for CAD systems and implementation of particular application based on this design. The purpose of described project is to provide CAD developers with open, scaleable set of software components, based on common architecture allowing to implement geometry modeling and drafting applications. In this paper is given description of projects logical hierarchy, external references implementation, wire-frame, surface and solid entities representation, application-specific data incapsulation into the projects. At the end of the paper is given short description of bCAD 4 software, developed using present design.

Keywords: *GraphiCon'2005, CAD, CAD Architecture, bCAD.*

1. INTRODUCTION

Light-weight CAD systems are quite popular solution in small, price-sensitive businesses. Traditionally these CAD were simple 2D drafting tools. Rapid development of hardware performance, that we see recent years forces light CAD developers to incorporate into their system features, that were belong to high-end systems. Though, 2D roots of lightweight CAD are putting many obstacles in the way to achieve decent functionality and easy-of-use. In this paper we describe design of modern architecture, goal of which is to avoiding these problems and provide CAD developers with scaleable development platform with good performance.

2. PROJECTS STRUCTURE

General idea of geometry model structure is to make it simple to understand and maintain. Therefore it was decided to make is as much as possible similar to common file system, used in OS.

2.1 Hierarchy and layering

First lets talk about logical structure of the projects. Traditionally there are two basic approaches, widely used in CAD - layering and tree-based hierarchy. Both of them have pro and contras. While layering is simple to understand and good about maintaining visibility (printability, sorting objects and so on) properties, it is not sufficient enough to maintain assemblies, which three-hierarchy is good for. So far in our design we go for combination of both. From one side geometry entities and service objects are stored in hierarchy three in the same manner as files are stored in file system folders. From another side project data provides a list of traditional named layers with set of individual properties which control their visibility and accessibility. Each entity can rely to any number of layers simultaneously. Relying to multiple layers simultaneously allows to sort entities by combination of properties.

2.2 Clones and external references

Any project hierarchy also can be included into another project as a *Clone* linked to the *Block* . Parent project can contain multiple links to one instance of the block. So far, amending block internals causes amending all related clones automatically. Similarity of entities hierarchy to the file systems simplifies implementation of external references. Blocks can be organized either by internal storage of the project or by *External Reference* to the projects, stored in the separate files.

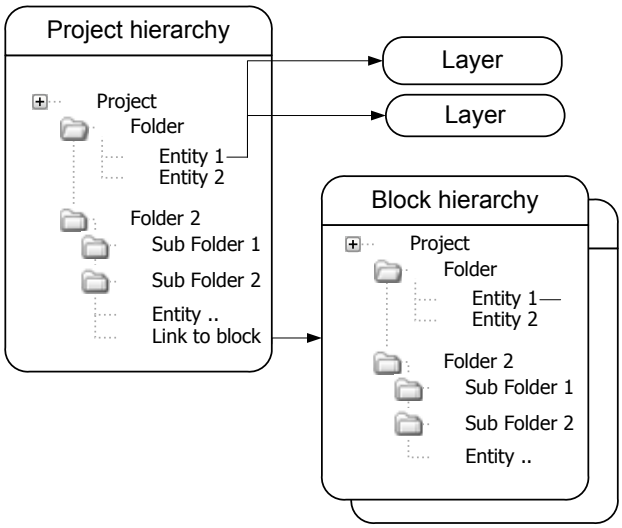

Figure 1: Logical structure of projects.

3. GEOMETRY REPRESENTATION

3.1 Wire-frame representation

All the geometry entities, wire-frame, surfaces and solid bodies are based on the universal core feature, called composite contour. Composite contour is a set of linear, elliptical and spline [1] segments, connected sequentially by their nodes (Fig. 2). Each node contains also information about type of coupling – filleted or chamfered and reciprocal direction of consequent segment – orthogonal, smooth or arbitrary (Fig. 3). Contour also can have ***closed*** and ***flat*** attributes. All the standard wire-frame entities, such as rectangle, circle, right polygon are just subset of composite contour with appropriate number and shape of segments. This approach simplifies implementation of system core and allows to provide universal editing context for any kind of entities.

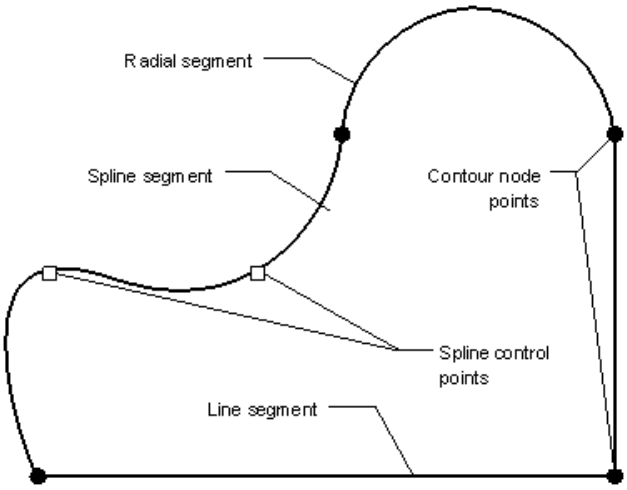

Figure 2: Composite contour**.**

There is full set of boolean operation, defined for closed flat contours, which allows to describe so called regions – an areas with number of contours describing boundaries and second set describing holes. This is used for combined objects such a filled and hatched areas.

3.2 Surfaces and solids

Unlike of simples 3D systems, where surfaces represented by triangle or NURBS approximation [2] geometry of surface entities in presented design is represented by two features – skeleton and approximation. Skeleton is number of composite contours, defining the context of surface construction. For example simplest surface of revolution is defined by two contours: generatrix curve and linear axis. In presented design there were four basic types of surfaces implemented:

- *Surface of revolution*, defined by boundary contour and axis of revolution
- *Lofted surface*, defined by path curve and set of cross-section shapes
- *Flat region*, defined by boundaries and hole contours and thickness
- *Net surface*, defined by number of longitudinal and transversal sections.

Approximation of surface is sort of skin, covering skeleton. So far both – triangular and NURBS representation can be used, depending on functional requirements. Separation of skeleton and approximation allows to keep precise geometry data and vary approximation level, for example for visualization purposes depending on available computer performance, by simplifying approximation for real-time or, opposite, make it more detailed for photorealistic images. Also skeletons data is enough to be stored to project files, while approximation is generated only for work session.

Separating skeleton and approximation approach allows also usage of third part geometry engines, such as standard solid-modeling engines

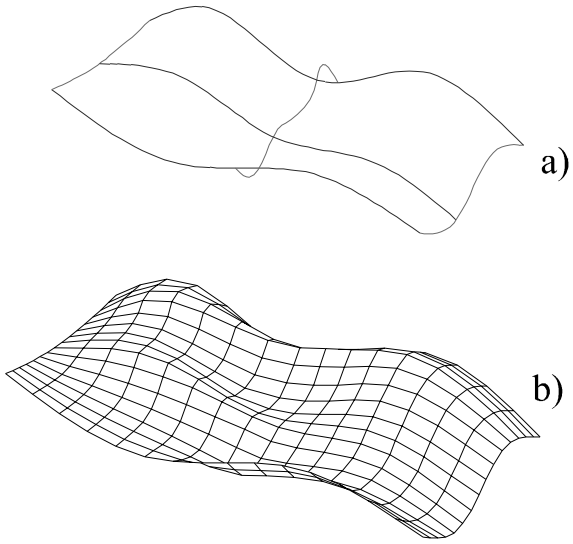

Figure 3: Surface elements: skeleton (a) and polygonal approximation (b)**.**

The solid bodies are represented as a closed space surrounded by set of surfaces with common boundaries – classic BREP (Boundary REPresentation). There is full set of boolean operations defined for solid bodies. The result of boolean operations is stored in traditional CSG-tree (Constructive Solid Geometry).

Thus, geometry entities representation is providing access to geometry data any time in pretty universal way, in the same time keeping context of entities construction for future use. Any element of combined objects, such as surface or solid body can be used for definition of other entities. All entities by default are edited just by amending appropriate composite contours, which are used as a skeleton objects. Though developers are allowed to implement their own editors, which would amend geometry in specific way.

4. USER-DEFINED DATA

To allow the system to store into the project special data, defined by particular applications each entity can have multiple *data containers* attached. Each such container have uniquely generated ID, which is used to specify what data is used for and which application is bale to access this data. There are several types of containers data:

- pure binary storage
- text storage
- data as a arrays of 3D vertices coordinates. Therefore when any of CAD built-in transformations or deformations, such as move, rotate, mirror, scale, bend or twist are applied to that object, these vertices are being transformed in corresponding way.

5. SAMPLE OF IMPLEMENTATION

System is designed as a composition of set of independent libraries. In purpose to have ability to develop future functionality

there were declared a number of system and application extensions. In present realization there are several such extensions implemented, such as import-export, data access and visualization, application programming interface and so on. System extension are implemented as standard DLLs, which allow to build and distribute flexible configurations, depending on functionality requirements, of cure in case system-extension interface still unchanged.

Basic core library contain mechanism of automatic extensions compatibility checking, based on comparison of extension unique ID and core build version number. User interface is based on idea of independent editors-viewers. Project structure can be displayed simultaneously in number of different independent manners, for example traditional WYSIWIG geometry editor, to tree-control based project browser. Application developers are allowed to implement their own type of viewer, electronic table for example.

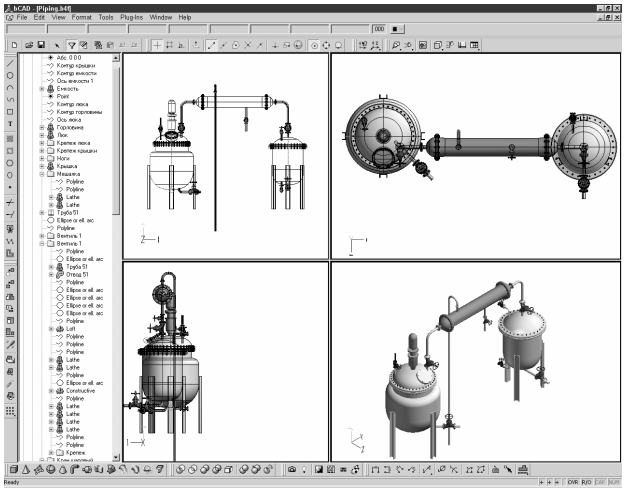

Figure 4. Implementation of presented architecture in bCAD version 4. Note various view-ports, displaying geometry and hierarchy.

6. CONCLUSION

Described architecture is implemented in prototype of bCAD version 4. It proves the basic ideas of building light-weight CAD system with robust performance, yet flexible extendible core functionality and API, suitable to develop end-user solutions. You can obtain detailed description of architecture, API specifications and information about current state of the project on **http://www.propro.ru/bcad4**

7. REFERENCES

[1] P. Germain-Lacour, Mathematiques et CAO. Russian edition, Mir, 1989.

[2] G. Farin, Curves and Surfaces for Computer-Aided Geometric Design, Academic Press, 1997.

[3] E.Yares, Cracked foundation: how hidden flaws undermine the acceptance of advanced software technology, isiCAD-2004, Novosibirsk.

[4] V. Malukh, Intelligent capabilities of small CAD systems, isiCAD-2004, Novosibirsk.

About the authors

Vladimir Malukh is technical director of the ProPro Group company and research fellow of Institute Informatics systems of Russian Academy of science. His contact email is vmalukh@propro.ru.

Alexey Nickitin is director of Complex 5 company and research fellow of Institute Informatics systems of Russian Academy of science. His contact email is nick@complex-a5.ru .